# Certificate in C# Programming

## Creating Web Applications in C#

By Randal Root

# Lesson 02

In this lesson, we will see how and why **HTML5 and CSS 3 are used in ASP.NET web pages**.

This lesson corresponds to the 4th chapter of the course Textbook,

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

*“Part 2: Developing ASP.NET Applications*

*The second part of this book delves into the heart of ASP.NET programming and introduces its event-based model.* ***In Chapter 4, you’ll take a look around the Visual Studio design environment and learn a few fundamentals about web forms, events, and HTML5****. In Chapters 5 and 6, you learn how to program a web page’s user interface through a layer of objects called server controls.*

*Next you’ll explore two more essentials of ASP.NET programming. Chapter 7 presents techniques for handling errors. Chapter 8 describes strategies for state management. Taken together, the chapters in this part contain all the core concepts you need to design web pages and create a basic ASP.NET website.” (MacDonald, 2012, p.xxxv)*

# The Difference Between Websites and Web Projects

As was mentioned in the last lesson, there are **two** ASP.NET development **options** in Visual Studio (VS) to choose from. The first is the **Website** option and the second, and (*Currently*) recommended is the Web **Projects** option.

“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.**” (MSDN, <http://msdn.microsoft.com/en-us/library/dd547590(v=vs.110).aspx> )

With Visual Studio **2013** (VS 2013) Microsoft announced that they **recommend** using **Project** based websites. They also changed to their ASP.NET template options. In VS 2017, the current options look like this (Figure 1):

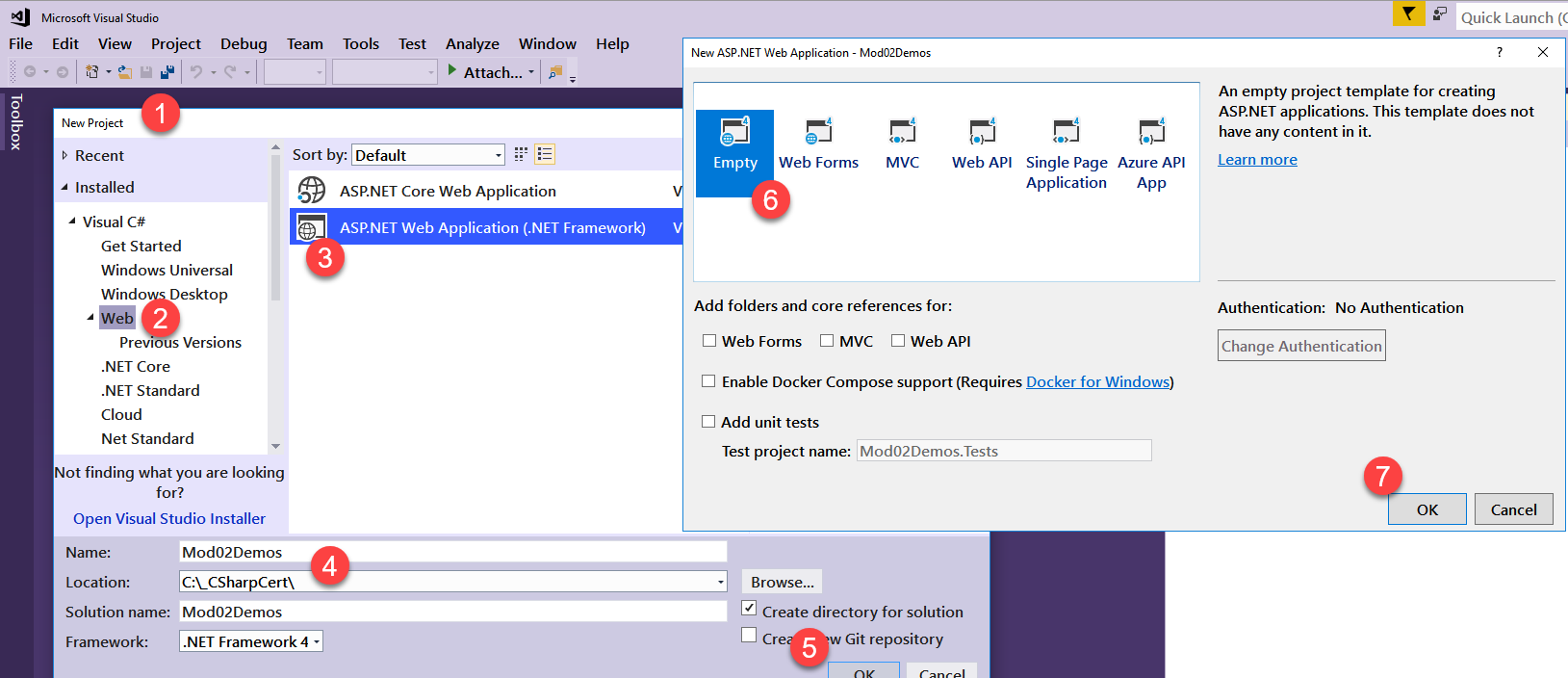


Figure 1. Project Options in VS 2017

At the time the **Author** wrote the book, Microsoft was still recommending Web Sites over Web Projects, but now they have changed their mind. So, when you read the author’s chapter, keep this in mind!

“All the examples that are shown in **this book use the projectless website model**. However, you’re free to create web projects if you fit into one of the scenarios just described. You still write the same code to power your web pages. It’s really just a matter of taste.” (MacDonald, 2012, p. 85)

NOTE: Going forward, I will be using Visual Studio 2017 Project-Based Web Applications. This way you will be able to contrast what is in the book with my examples.

Another difference between the Web Site and Project-based application is the dialog window when adding a new ASP.NET page to the solution. For the Project based application, the dialog looks like this (Figure 2)…

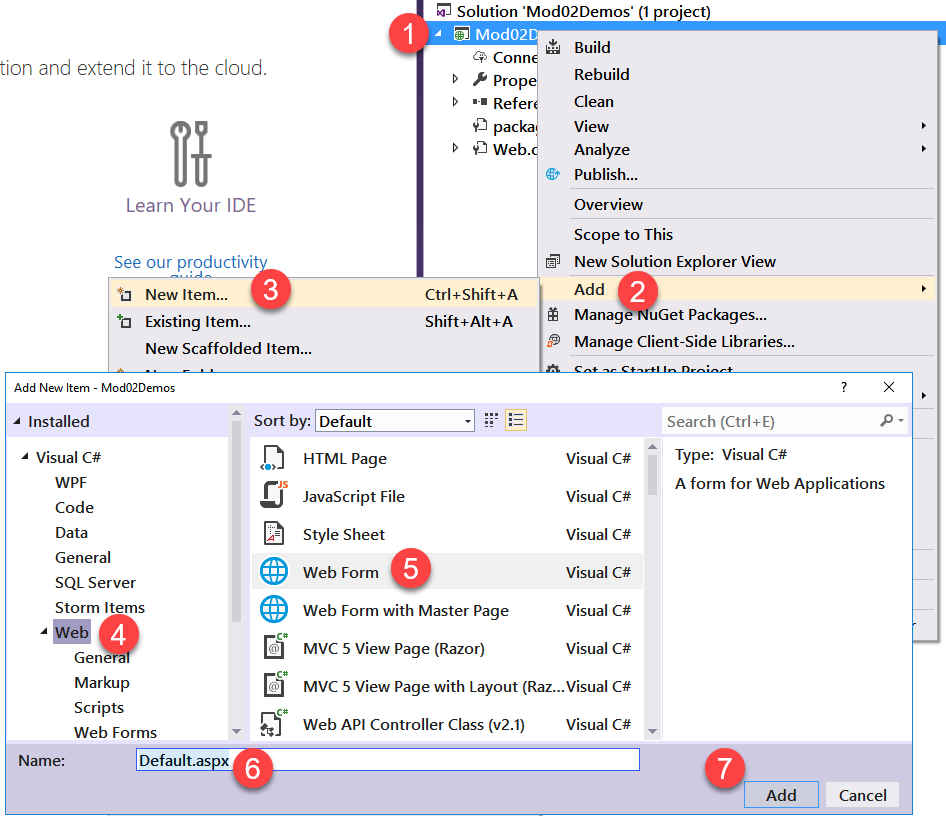


Figure 2. Adding a new ASP.NET Page with the Web Project option in VS 2017

# Web Forms

Web Forms were the original way to create ASP.NET web pages. ASP.NET **WebForms usually include a code-behind page**. However, if you do not want this, **you can remove the code that references the code behind page (Example 1)**, found at the top of the web page, and then remove the referenced .cs files from the project.

#### Example 01: The code that references the code-behind file

<%@ Page Language="C#" **AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default"** %>

# The Multi-Targeting of ASP.NET versions

There are several versions of ASP.NET. You can choose to work with one version per project. Since we are using VS 2017, our new projects are set to .NET Framework 4.6 or 4.6.1 by default (Figure 3).

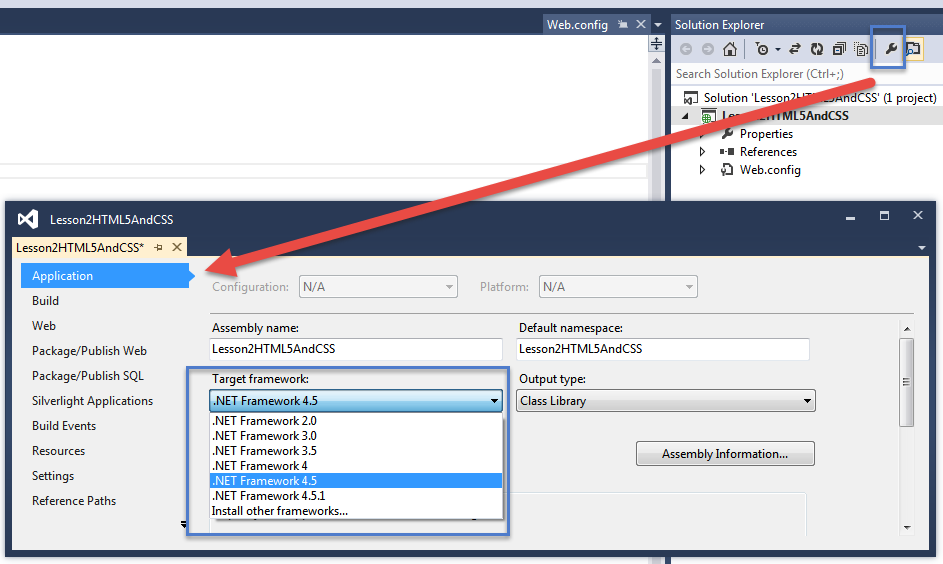


Figure 3. Setting the Framework in an ASP.NET project

“all the samples that are included with this book target .NET 4.5.” (MacDonald, 2012, p. 89)

Note: If we come across issues where we need to change the target framework to 4.5, we will do so, but until then, leave it at the default!

# The Web.config File

In general, changing properties in the Project’s property dialog window will **automatically add configurations** to the website’s **Web.config file** (Example 2). This XML file will be necessary to our web applications, so we will discuss it in detail as we continue through the course.

#### Example 2: The Standard Web.Config file

<?xml version="1.0"?>

<!--

For more information on how to configure your ASP.NET application, please visit

http://go.microsoft.com/fwlink/?LinkId=169433

-->

<configuration>

<system.web>

<compilation debug="true" targetFramework="4.5" />

<httpRuntime targetFramework="4.5" />

</system.web>

</configuration>

You **can add code directly to this file** using any text editor. This is an XML file, so you use XML code for the settings. Make sure to **include notes** if you add setting manually (Example 3).

#### Example 3: A Modified Web.Config file

<?xml version="1.0"?>

<!--

For more information on how to configure your ASP.NET application, please visit

http://go.microsoft.com/fwlink/?LinkId=169433

-->

<configuration>

**<appSettings >**

**<!--**

**Note: A student found that entering this code made his Razor code work!**

**I cannot reproduce this though, RRoot, 1/1/2020.**

**-->**

**<add key="webPages:Version" value="2.0" />**

**</appSettings >**

<system.web>

<compilation debug="true" targetFramework="4.5" />

<httpRuntime targetFramework="4.5" />

</system.web>

</configuration>

# Designing a Web Page

If you are used to working with Windows Forms applications, you may be surprised to find that **web forms are designed quite differently**. One of the significant differences is that you cannot rely on **drag-and-drop feature**s to position your controls.

“**You’ll find that some controls can’t be resized**. Instead, they grow or shrink to fit the amount of content in them. For **example**, the size of a **Label** control depends on how much text you enter in it. On the other hand, you can adjust the size of a Button or a TextBox control by clicking and dragging in the design environment.” (MacDonald, 2012, p. 89)

While this is **true**, it is **not the whole truth**. It turns out that you can change this behavior if you **change the controls style**! Here is an example:

#### **Demo01.aspx**: Positioning Controls

<!DOCTYPE html>

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

<head runat="server">

<title>Demo01</title>

</head>

<body>

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

<div>

<p>Try Dragging the Label 1 and Textbox 1 controls using the Split View feature! <p />

<asp:Label ID="Label1" runat="server" Text="Label 1"></asp:Label>

<br />

<asp:Label ID="Label2" runat="server" Text="Label 2" Width="200px"></asp:Label>

<br />

<asp:TextBox ID="TextBox1" runat="server" Text="Textbox1"></asp:TextBox>

<br />

<asp:TextBox ID="TextBox2" runat="server" Text="Textbox2" Width="200px"></asp:TextBox>

<br />

</div>

</form>

</body>

</html>

### Grid Mode

Microsoft includes an option called **Grid mode** that allows you to position controls on the design surface. Although this mode was popular early on it was **recommended that you not use it now**. Instead, Microsoft **Recommends using cascading style sheets** to adjust the look and feel of your web page.

“If you do decide to use **absolute positioning, the best idea is to apply it to a container, such as the <div> element. The <div> element represents a box** …*” (MacDonald 2012, p. 93)*

#### Demo02.aspx: Adding an Inline Style to a div

<!DOCTYPE html>

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

<head runat="server">

<title>Demo02</title>

</head>

<body>

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

<div **style** = "POSITION: absolute;

left: 50px;

top: 50px;

background-color: silver;

width: 200px;

height: 50px;">

</div>

</form>

</body>

</html>

### Using CSS

A **best option** is to add a CSS style block to your page as shown here.

#### Demo03.aspx: Adding an Internal style for a div

<!DOCTYPE html>

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

<head runat="server">

<title>Demo03</title>

<style type="text/css">

.MyFormDiv {

POSITION: absolute;

left: 50px;

top: 50px;

background-color: silver;

width: 200px;

height: 50px;

}

</style>

</head>

<body class="newStyle1">

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

<div class="MyFormDiv" >

</div>

</form>

</body>

</html>

### CSS3

Cascading Style Sheets (CSS) have been used in web pages for well over a decade. In the last couple of years, CSS received a major upgrade in the form of CSS3. You can find a lot of information about CSS on the **official CSS web page** (<http://www.w3.org/Style/CSS/> ). However, this **may not be the best place to get started**. Instead, **start with** simple web-based tutorials such as the one on **the W3Schools website**. (<http://www.w3schools.com/css/css3_intro.asp>).

Of course, the best way to learn a new language is to try it out. Here is an example:

#### Demo04.aspx: Adding CSS3 styles to a div

<!DOCTYPE html>

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

<head runat="server">

<title>Demo04</title>

<style type="text/css">

.MyFormDiv {

POSITION: absolute;

left: 50px;

top: 50px;

background-color: silver;

width: 250px;

height: 50px;

**/\*New in CSS3\*/**

**border: 2px solid;**

**border-radius: 25px;**

}

</style>

</head>

<body class="newStyle1">

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

<div class="MyFormDiv">

&nbsp;<p>Try moving this around in the Split View</p>

</div>

</form>

</body>

</html>

## Useful Designing Features

### Multiple Browsers

Since VS 2013 you can launch your pages in more than one web browser at a time. In addition, these pages are linked to the Brower Refresh button in VS (Figure 4).

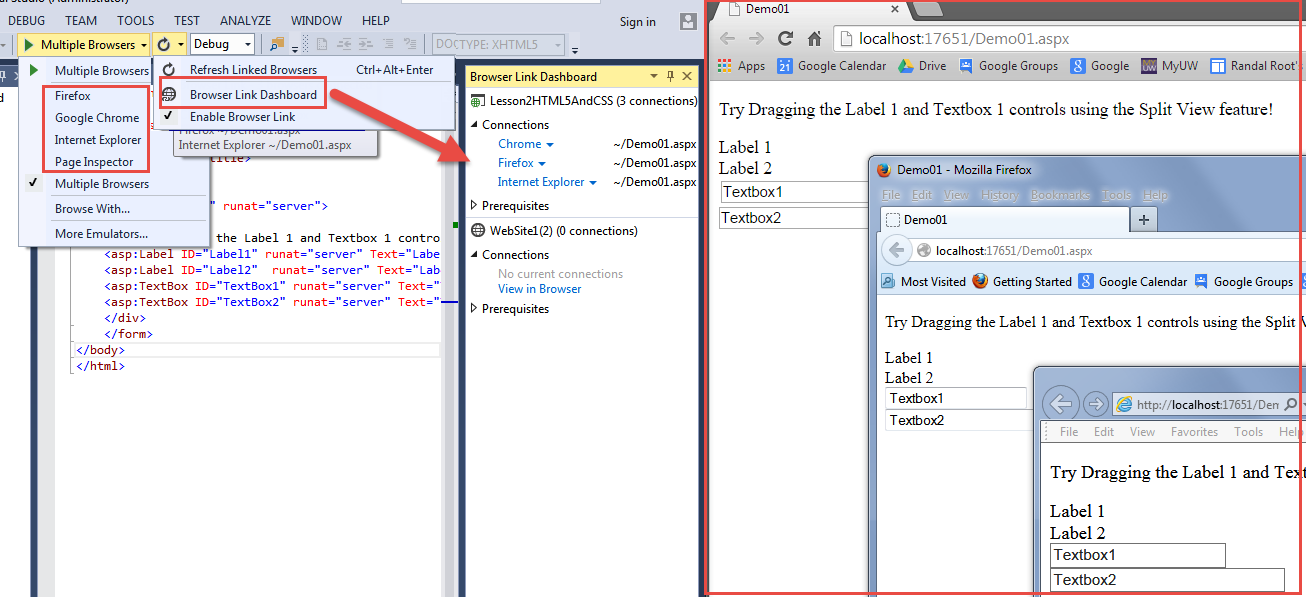


Figure 4. Using the Browser Link Dashboard

<http://www.asp.net/visual-studio/overview/2013/visual-studio-2013-web-editor-features-browser-link>

### Page Inspector

Page Inspector was a tool that allows you to see the browsers’ output code right from Visual Studio. Now you must use each web browser itself to do so using the developer tools.

### CSS Features

Sadly, VS 2015 **removed the CSS style editing tool**, so VS IntelliSense is all there is (Figure 5).

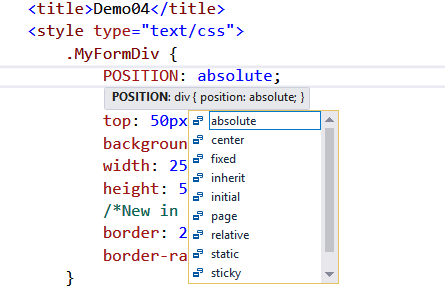


Figure 5. Using VS Intellisense

Most Web Browsers have built-in tools to help you develop and design your web pages. While it is still not as easy as "drag and drop," you can change the style code and immediately see your changes (Figure 6).

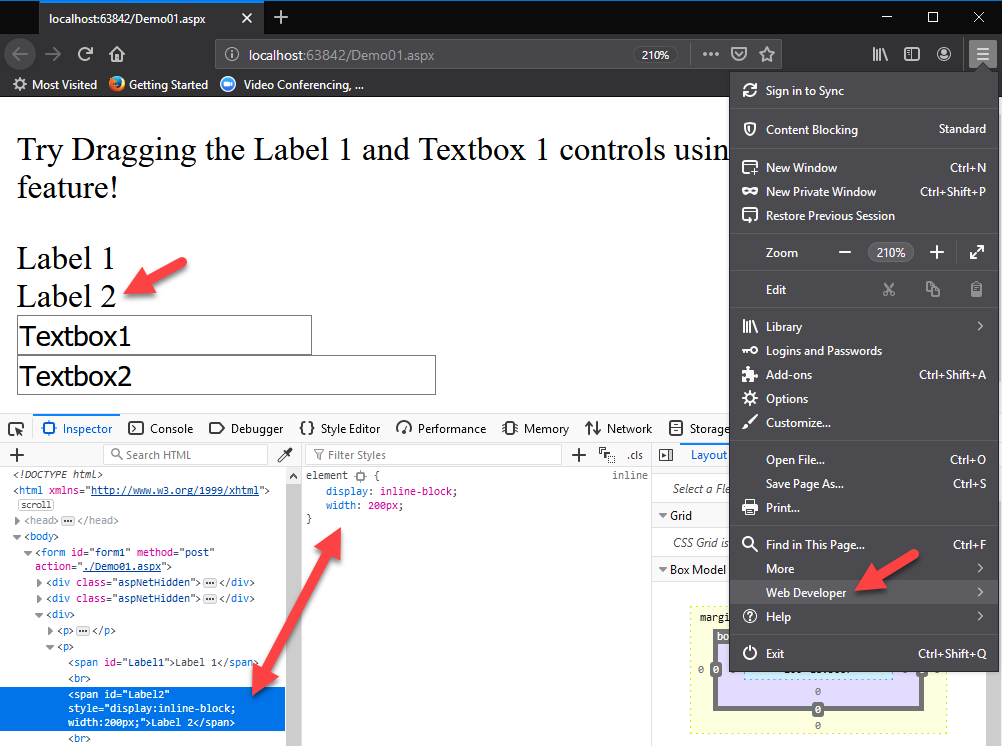


Figure 6. Using the Inspector Development Tool in Firefox.

## Using HTML5 in an ASP.NET page

HTML5 is the standard for creating new web pages. Most browsers developers have been slow to add support for all of its features, but that is changing, and more HTML5 elements are supported with each released version.

“Note **In this book**, the web page markup listings omit the doctype (because it’s just one more detail chewing up extra pages). If you download the sample code, you’ll find that **all pages use the modern HTML5 doctype**.

Here’s a dirty secret: **browsers process HTML5 pages and XHTML pages in exactly the same way**. That means they don’t complain if they encounter an XHTML document with sloppy syntax (they still try to figure it out) or one that uses newer HTML5 elements (such as <video>). This is one of the reasons that HTML5 won the web standard war: it reflects the way web browsers really work. To learn more about the HTML5 standard, you can read HTML5: The Missing Manual (O’Reilly, 2011) or view an HTML5 tutorial at www.w3schools.com/html5/html5\_intro.asp.” *(MacDonald 2012, p. 98)*

#### Demo05: Using HTML5 elements

<!DOCTYPE html>

<html>

<head runat="server">

<meta charset="utf-8" />

<title></title>

</head>

<body>

<div id="header">

<h1>This is the **standard** way to create an HTML header element</h1>

</div>

**<header>**

**<h2>This header is a HTML5 header element</h2>**

**</header>**

**<video** width="320" height="240" controls="controls">

<source src="Videos/SampleVideoForHTML5.mp4" type="video/mp4" />

Your browser does not support the video tag.

**</video>**

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

<div>

</div>

</form>

<div id="footer">

<p>This is the standard way to create an HTML footer element</p>

</div>

**<footer>**

<p>And this is a HTML5 footer element</p>

**</footer>**

</body>

</html>

## Using C# in an ASP.NET page

ASP.NET applications are created using several languages. These include HTML, CSS, XML, JavaScript, and others. Of course, you can also access the core functionality of the .NET framework using any of the .NET languages such as VB.NET and C#.

**To use C# in a page, you must include a Directive at the top of the page** indicating the language to be used by ASP.NET. (This is something only used in the Olde Web Forms pages and not in the modern MVC pages we are moving toward.)

You should add the directive before you start adding any C# code. Otherwise, IntelliSense will not react as you create your statements and methods (Figure 7).



Figure 7. Adding C# code to an ASP.NET page

Using C# code and the .NET framework, you can add code to access resources on the server that a web browser could not typically access! Here is an example of a web page displaying data from a file on the web server (Figure 8).

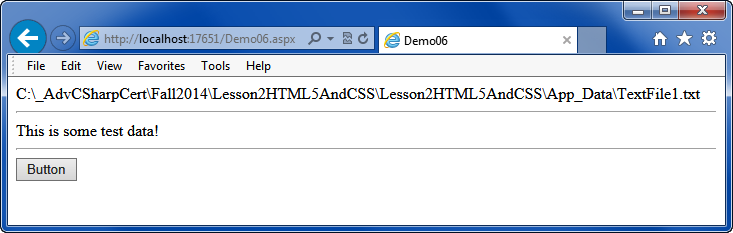


Figure 8. Display data from a Web Server’s text file

**Note:** The path to the file is displayed in this example only for testing during development. The “App\_Data” subfolder is a common feature of many ASP.NET applications.

#### Demo6: Using C# in an ASP.NET application

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

<!DOCTYPE html>

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

<head runat="server">

<title>Demo06</title>

<script runat="server">

protected void Button1\_Click(object sender, EventArgs e)

{

**//NOTE: You must create the app data folder and text file for this to work!**

string strPhyicalPath = MapPath("~/**App\_Data**/TextFile1.txt");

Response.Write(strPhyicalPath);

System.IO.StreamReader objSR;

objSR = new System.IO.StreamReader(strPhyicalPath);

string strData = objSR.ReadToEnd();

objSR.Close();

**//Show the data on the page for testing!**

Response.Write("<hr />" + strData + "<hr />");

}

</script>

</head>

<body>

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

<div>

<asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1\_Click" />

</div>

</form>

</body>

</html>

# LAB 01: Creating an ASP.NET using CSS 3

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

1. **Create** an **Empty** Web Project called **Mod02Labs** in the following folder **C:\\_CSharpCert**\Mod02Labs”.
2. **Add** a new ASP.NET page called **Lab01**.
3. **Remove** the **Code-Behind** files.
4. **Browse** the page and **verify** that it still works.
5. Add controls and C# code to the page that will display the Name and Password when the button is clicked (Figure 9)!

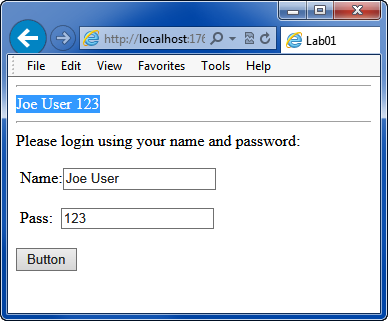


Figure 9. The Lab01 page.

6) Add CSS3 code to make the page look like the image below (Figure 10). Hint: Check out the CCS3 tutorial on the W3Schools website <https://www.w3schools.com/css/css3_borders.asp>.

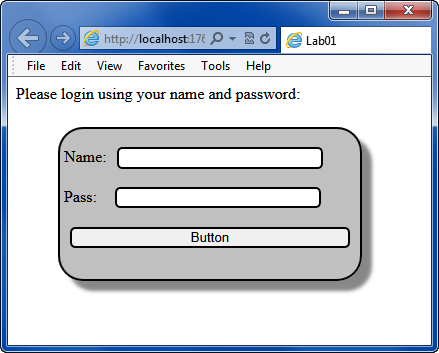


Figure 10. The Lab01 webpage with Style!

## Working with data files

So far, we have seen how to get input from a user, but we have not dealt with what to do with that input. One of the simplest things we can do is to store it in a text file.

We use the classes of the System.IO of the namespace for working with files from an ASP.NET page, just as we did with Console and Windowed applications.

*"File and stream I/O (input/output) refers to the transfer of data either to or from a storage medium. In the .NET Framework, the System.IO namespaces contain types that enable reading and writing, both synchronously and asynchronously, on data streams and files. These namespaces also contain types that perform compression and decompression on files and types that enable communication through pipes and serial ports." (*[*https://docs.microsoft.com/en-us/dotnet/standard/io/*](https://docs.microsoft.com/en-us/dotnet/standard/io/)*, 2019)*

Here is a simple example the creates and writes data to a text file:

protected void Button1\_Click(object sender, EventArgs e)

{

string strPhyicalPath = MapPath("~/App\_Data/TextFile.txt") ;

Response.Write(strPhyicalPath );

System.IO.StreamReader objSR;

objSR = new System.IO.StreamReader(strPhyicalPath);

string strData = objSR.ReadToEnd();

objSR.Close();

Response.Write("<hr>" + strData);

}

Note the use of the MapPath method. The tilde (~) symbol indicates that the starting point is the root of the web application.

This method is useful since we cannot easily predict the physical paths once our application is hosted on the Internet.

*"Server.MapPath function is used whenever physical location of file or folder is needed. MapPath expects one parameter, a virtual or relative path to file or folder, and returns corresponding physical absolute path on the server. Although we can use absolute paths directly, that's not adjustable and is considered as hard to maintain." (*[*http://www.beansoftware.com/ASP.NET-Tutorials/using-server.mappath.aspx*](http://www.beansoftware.com/ASP.NET-Tutorials/using-server.mappath.aspx)*, 2019)*