**ASP.NET Tutorial**

ASP.NET is a development framework for building web pages and web sites with HTML, CSS, JavaScript and server scripting.

ASP.NET supports three different development models:  
Web Pages, MVC (Model View Controller), and Web Forms:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [Web Pages](http://www.w3schools.com/aspnet/webpages_intro.asp) Single Pages Model |  | [MVC](http://www.w3schools.com/aspnet/mvc_intro.asp) Model View Controller |  | [Web Forms](http://www.w3schools.com/aspnet/aspnet_intro.asp) Event Driven Model |
| Simplest ASP.NET model.  Similar to PHP and classic ASP.  Built-in templates and helpers for database, video, graphics, social media and more. |  | MVC separates web applications into 3 different components:  Models for data Views for display Controllers for input |  | The traditional ASP.NET event driven development model:  Web pages with added server controls, server events, and server code. |

**Web Pages Tutorial**

**If you are new to ASP.NET, Web Pages is the perfect place to start.**

Web Pages is the easiest development model for developing ASP.NET web sites.

In our Web Pages tutorial you will learn how to combine HTML, CSS, JavaScript and server code, using the latest Razor server markup syntax with Visual Basic or C#.

You will also learn how to extend your web pages with programmable Web Helpers, including database, video, graphics, social networking and much more.

**MVC Tutorial**

MVC is a model for building web applications using a MVC (Model View Controller) design.

**If you want a lighter alternative to traditional ASP.NET, MVC is the place to start.**

In our MVC tutorial you will learn how to build web applications using a lightweight development model, integrated with all existing ASP.NET features, such as Master Pages, Security, and Authentication.

**Web Forms Tutorial**

Web Forms is the traditional ASP.NET model, based on event driven Web Forms and post backs.

Over the years, developers have used ASP.NET Web Forms to create many of the largest websites in the world.

**If you want to study the design model used by many web developers over the last 10 years, this is the place to start.**

**Who Are These Tutorials For?**

The tutorials above are for anyone who wants to learn to build websites on the Microsoft ASP.NET platform, from hobby-related sites, to new, modern, and fully commercial webs.

Even if you are new to web programming, you should be able to follow the tutorials, but it will help if you have a basic understanding of HTML and CSS.

It will also help if you have a basic understanding of scripting languages like JavaScript or VB (Visual Basic).

Do you have a preference for VB over C#? Do you want to learn both languages? The good news are: In the W3Schools tutorials most code examples are in both languages.

If you are a professional web developer with prior ASP.NET experience, you can still learn a lot, since these tutorials covers a lot of new ASP.NET concepts, like HTML5, CSS3, JQuery, and more.

# ASP.NET

## Classic ASP - Active Server Pages

Active Server Pages (ASP), also known as Classic ASP, was introduced in 1998 as Microsoft's first server side scripting engine.

ASP is a technology that enables scripts in web pages to be executed by an Internet server.

ASP pages have the file extension .asp, and are normally written in VBScript.

If you want to learn Classic ASP, [visit our Classic ASP Tutorial](http://www.w3schools.com/asp/default.asp).

## ASP.NET

ASP.NET is a new ASP generation. It is not compatible with Classic ASP, but ASP.NET may include Classic ASP.

ASP.NET pages are compiled, which makes them faster than Classic ASP.

ASP.NET has better language support, a large set of user controls, XML-based components, and integrated user authentication.

ASP.NET pages have the extension .aspx, and are normally written in VB (Visual Basic) or C# (C sharp).

User controls in ASP.NET can be written in different languages, including C++ and Java.

When a browser requests an ASP.NET file, the ASP.NET engine reads the file, compiles and executes the scripts in the file, and returns the result to the browser as plain HTML.

## ASP.NET Razor

Razor is a new and simple markup syntax for embedding server code into ASP.NET web pages, much like Classic ASP.

Razor has the power of traditional ASP.NET, but is easier to use and easier to learn.

## ASP.NET Programming Languages

This tutorial covers the following programming languages:

* Visual Basic (VB.NET)
* C# (Pronounced C sharp)

## ASP.NET Server Technologies

This tutorial covers the following server technologies:

* Web Pages (with Razor syntax)
* MVC (Model View Controller)
* Web Forms (traditional ASP.NET)

## ASP.NET Development Tools

ASP.NET supports the following development tools:

* WebMatrix
* Visual Web Developer
* Visual Studio

This tutorial uses WebMatrix for Web Pages, and Visual Web Developer for MVC and Web Forms.

## ASP.NET File Extensions

* Classic ASP files have the file extension .asp
* ASP.NET files have the file extension .aspx
* ASP.NET files with Razor C# syntax have the file extension .cshtml
* ASP.NET files with Razor VB syntax have the file extension .vbhtml

**ASP.NET Web Pages - Tutorial**

ASP.NET is a development framework for building web pages and web sites with HTML, CSS, JavaScript and server scripting.

ASP.NET supports three different development methods:  
Web Pages, MVC (Model View Controller), and Web Forms.

**THIS TUTORIAL COVERS WEB PAGES**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Web Pages |  | MVC |  | Web Forms |

**Where to Start?**

Many developers like to start learning a new technology by looking at working examples.

If you want to take a look at a working Web Pages example, follow the [ASP.NET Web Pages Demo](http://www.w3schools.com/website/webpages_demo.asp).

**Easy Learning with "Run Example"**

Our "Run Example" tool makes it easy to learn Web Pages.

It runs examples and displays the ASP.NET code and the HTML output simultaneously.

Click on the "Run Example" button to see how it works:

**Web Pages Example**

<html>  
<body>  
     <h1>Hello Web Pages</h1>  
     <p>The time is @DateTime.Now</p>  
</body>  
</html>

[Run Example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_001)

**What is Web Pages?**

Web Pages is one of the 3 programming models for creating ASP.NET web sites and web applications.

The other two programming models are Web Forms and MVC (Model, View, Controller).

Web Pages is the simplest programming model for developing ASP.NET web pages. It provides an easy way to combine HTML, CSS, JavaScript and server code:

* Easy to learn, understand, and use
* Built around single web pages
* Similar to PHP and Classic ASP
* Server scripting with Visual Basic or C#
* Full HTML, CSS, and JavaScript control

Web Pages is easy extendable with programmable Web Helpers, including database, video, graphics, social networking and much more.

**Web Pages Tutorial**

If you are new to ASP.NET, Web Pages is the perfect place to start.

In our Web Pages tutorial you will learn how to combine HTML, CSS, JavaScript and server code, using the latest Razor server markup syntax with Visual Basic or C# .

You will also learn how to extend your web pages with programmable Web Helpers.

**Web Pages Examples**

Learn by examples!

Because ASP.NET code is executed on the server, you cannot view the code in your browser. You will only see the output as plain HTML.

At W3Schools every example displays the hidden ASP.NET code. This makes it easier for you to understand how it works.

[Web Pages Examples](http://www.w3schools.com/aspnet/webpages_examples.asp)

**Web Pages References**

At the end of this tutorial you will find a complete set of ASP.NET references with objects, components, properties and methods.

[Web Pages References](http://www.w3schools.com/aspnet/webpages_ref_classes.asp)

**We Have Used WebMatrix**

In this tutorial, we have used WebMatrix.

WebMatrix is a simple but powerful free ASP.NET development tool from Microsoft, tailor made for Web Pages.

WebMatrix contains:

* Web Pages examples and templates
* A web server language (Razor using VB or C#)
* A web server (IIS Express)
* A database server (SQL Server Compact)
* A full web development framework (ASP.NET)

With WebMatrix you can start from scratch with an empty web site and a blank page, or build on open source applications from a "Web Application Gallery". Both PHP and ASP.NET applications are available, such as Umbraco, DotNetNuke, Drupal, Joomla, WordPress and many more. WebMatrix also has built-in tools for security, search engine optimization, and web publishing.

The skills and code you develop with WebMatrix can seamlessly be transformed to fully professional ASP.NET applications.

If you want to try WebMatrix, follow the link below to install:

<http://www.microsoft.com/web/gallery/install.aspx?appid=WebMatrix>

**ASP.NET Web Pages - Adding Razor Code**

[« Previous](http://www.w3schools.com/aspnet/webpages_intro.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_layout.asp)

In this tutorial we will use Razor markup with C# and Visual Basic code

**What is Razor?**

* Razor is a markup syntax for adding server-based code to web pages
* Razor has the power of traditional ASP.NET markup, but is easier to learn, and easier to use
* Razor is a server side markup syntax much like ASP and PHP
* Razor supports C# and Visual Basic programming languages

**Adding Razor Code**

Remember the web page from previous chapter:

<!DOCTYPE html>  
  
<html lang="en">  
<head>  
   <meta charset="utf-8" />  
    <title>Web Pages Demo</title>  
</head>  
<body>  
    <h1>Hello Web Pages</h1>  
</body>  
</html>

Now add some Razor code to the example:

**Example**

<!DOCTYPE html>  
  
<html lang="en">  
<head>  
     <meta charset="utf-8" />  
     <title>Web Pages Demo</title>  
</head>  
<body>  
     <h1>Hello Web Pages</h1>   
     <p>The time is @DateTime.Now</p>  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_001)

The page contains ordinary HTML markup, with one addition: the @ marked Razor code.

The Razor code does all the work of determining the current time on the server and display it. (You can specify formatting options, or just display the default)

**Main Razor Syntax Rules for C#**

* Razor code blocks are enclosed in @{ ... }
* Inline expressions (variables and functions) start with @
* Code statements end with semicolon
* Variables are declared with the var keyword
* Strings are enclosed with quotation marks
* C# code is case sensitive
* C# files have the extension .cshtml

**C# Example**

<!-- Single statement block -->  
@{ var myMessage = "Hello World"; }  
  
<!-- Inline expression or variable -->  
<p>The value of myMessage is: @myMessage</p>   
  
<!-- Multi-statement block -->  
@{  
var greeting = "Welcome to our site!";  
var weekDay = DateTime.Now.DayOfWeek;  
var greetingMessage = greeting + " Today is: " + weekDay;  
}  
<p>The greeting is: @greetingMessage</p>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_001)

**Main Razor Syntax Rules for VB**

* Razor code blocks are enclosed in @Code ... End Code
* Inline expressions (variables and functions) start with @
* Variables are declared with the Dim keyword
* Strings are enclosed with quotation marks
* VB code is not case sensitive
* VB files have the extension .vbhtml

**Example**

<!-- Single statement block  -->   
@Code dim myMessage = "Hello World" End Code  
   
<!-- Inline expression or variable -->   
<p>The value of myMessage is: @myMessage</p>   
   
<!-- Multi-statement block -->   
@Code  
dim greeting = "Welcome to our site!"   
dim weekDay = DateTime.Now.DayOfWeek   
dim greetingMessage = greeting & " Today is: " & weekDay  
End Code   
  
<p>The greeting is: @greetingMessage</p>

**ASP.NET Web Pages - Page Layout**

[« Previous](http://www.w3schools.com/aspnet/webpages_razor.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_folders.asp)

With Web Pages it is easy to create a web site with a consistent layout.

**A Consistent Look**

On the Internet you will discover many web sites with a consistent look and feel:

* Every page have the same header
* Every page have the same footer
* Every page have the same style and layout

With Web Pages this can be done very efficiently. You can have reusable blocks of content (content blocks), like headers and footers, in separate files.

You can also define a consistent layout for all your pages, using a layout template (layout file).

**Content Blocks**

Many websites have content that is displayed on every page (like headers and footers).

With Web Pages you can use the **@RenderPage()** method to import content from separate files.

Content block (from another file) can be imported anywhere in a web page, and can contain text, markup, and code, just like any regular web page.

Using common headers and footers as an example, this saves you a lot of work. You don't have to write the same content in every page, and when you change the header or footer files, the content is updated in all your pages.

This is how it looks in code:

**Example**

<html>  
<body>  
@RenderPage("header.cshtml")  
<h1>Hello Web Pages</h1>   
<p>This is a paragraph</p>  
@RenderPage("footer.cshtml")  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_002)

**Using a Layout Page**

In the previous section, you saw that including the same content in many web pages is easy.

Another approach to creating a consistent look is to use a layout page. A layout page contains the structure, but not the content, of a web page. When a web page (content page) is linked to a layout page, it will be displayed according to the layout page (template).

The layout page is just like a normal web page, except from a call to the **@RenderBody()** method where the content page will be included.

Each content page must start with a **Layout directive**.

This is how it looks in code:

**Layout Page:**

<html>  
<body>  
<p>This is header text</p>  
@RenderBody()  
<p>&copy; 2014 W3Schools. All rights reserved.</p>  
</body>  
</html>

**Any Web Page:**

@{Layout="Layout.cshtml";}  
  
<h1>Welcome to W3Schools</h1>  
  
<p>  
Lorem ipsum dolor sit amet, consectetur adipisicing elit,sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laborisnisi ut aliquip ex ea commodo consequat.  
</p>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_011)

**D.R.Y. - Don't Repeat Yourself**

With two ASP.NET tools, Content Blocks and Layout Pages, you can give your web applications a consistent look.

These tools also save you a lot of work, since you don't have to repeat the same information on all pages. Centralizing markup, style, and code makes web applications much more manageable and easier to maintain.

**Preventing Files from Being Browsed**

With ASP.NET, files with a name that starts with an underscore cannot be browsed from the web.

If you want to prevent your content blocks or layout files from being viewed by your users, rename the files to:

\_header.cshtml

\_footer.cshtml

\_Layout.cshtml

**Hiding Sensitive Information**

With ASP.NET, the common way to hide sensitive information (database passwords, email passwords, etc.) is to keep the information in a separate file named "\_AppStart".

**\_AppStart.cshtml**

@{  
WebMail.SmtpServer = "mailserver.example.com";  
WebMail.EnableSsl = true;  
WebMail.UserName = "username@example.com";  
WebMail.Password = "your-password";  
WebMail.From = "your-name-here@example.com";  
}

**ASP.NET Web Pages - Folders**

[« Previous](http://www.w3schools.com/aspnet/webpages_layout.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_global.asp)

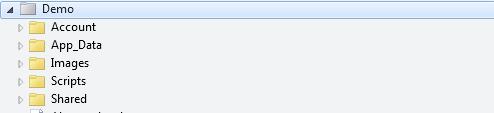
This chapter is about folders and folder paths.

In this chapter you will learn:

* About Logical and Physical folder structures
* About Virtual and Physical names
* About web URLs and Paths

**Logical Folder Structure**

Below is a typical folder structure for an ASP.NET web pages web:



* The "Account" folder contains logon and security files
* The "App\_Data" folder contains databases and data files
* The "Images" folder contains images
* The "Scripts" folder contains browser scripts
* The "Shared" folder contains common files (like layout and style files)

**Physical Folder Structure**

The physical structure for the "Images" folder at the website above might look like this on a computer:

C:\Johnny\Documents\MyWebSites\Demo\Images

**Virtual and Physical Names**

From the example above:

The virtual name of a web picture might be "Images/pic31.jpg".

But the physical name is "C:\Johnny\Documents\MyWebSites\Demo\Images\pic31.jpg"

**URLs and Paths**

URLs are used to access files from the web: <http://www.w3schools.com/html/html5_intro.asp>

The URL corresponds to a physical file on a server: C:\MyWebSites\w3schools\html\html5\_intro.asp

A virtual path is shorthand to represent physical paths. If you use virtual paths, you can move your pages to a different domain (or server) without having to update the paths.

|  |  |
| --- | --- |
| URL | http://www.w3schools.com/html/html5\_intro.asp |
| Server name | w3schools |
| Virtual path | /html/html5\_intro.asp |
| Physical path | C:\MyWebSites\w3schools\html\html5\_intro.asp |

The root on a disk drive is written like C:\, but the root on a web site is  / (forward slash).

The virtual path of a web folder is (almost) never the same as the physical folder.

In your code you will, reference both the physical path and the virtual path, depending on what you are coding.

ASP.NET has 3 tools for working with folder paths: the ~ operator, the Server.MapPath method, and the Href method.

**The ~ Operator**

To specify the virtual root in programming code, use the ~ operator.

If you use the ~ operator, instead of a path, you can move your website to a different folder or location without changing any code:

var myImagesFolder = "~/images";  
var myStyleSheet = "~/styles/StyleSheet.css";

**The Server.MapPath Method**

The Server.MapPath method converts a virtual path (/default.cshtml) to a physical path that the server can understand (C:\Johnny\MyWebSited\Demo\default.cshtml).

You will use this method when you need to open data files located on the server (data files can only be accessed with a full physical path):

var pathName = "~/dataFile.txt";  
var fileName = Server.MapPath(pathName);

You will learn more about reading from (and writing to) data files on the server in the next chapter of this tutorial.

**The Href Method**

The Href method converts a path used in the code to a path that the browser can understand (the browser cannot understand the ~ operator).

You use the Href method to create paths to resources like image files, and CSS files.

You will often use this method in HTML <a>, <img>, and <link> elements:

@{var myStyleSheet = "~/Shared/Site.css";}  
  
<!-- This creates a link to the CSS file. -->  
<link rel="stylesheet" type="text/css" href="@Href(myStyleSheet)" />  
  
<!-- Same as : -->  
<link rel="stylesheet" type="text/css" href="/Shared/Site.css" />

The Href method is a method of the WebPage Object.

# ASP.NET Web Pages - Global Pages

[« Previous](http://www.w3schools.com/aspnet/webpages_folders.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_forms.asp)

This chapter is about the global pages AppStart and PageStart.

## Before Web Startup: \_AppStart

Most server side code are written inside individual web pages. For example, if a web page contains an input form, the web page typically contains server code for reading the data.

However, by creating a page named \_AppStart in the root of your site, you can have startup code executed before the site starts. If this page exists, ASP.NET runs it the first time any page in the site is requested.

Typical use for \_AppStart is startup code and initialization of global values like counters and global names.

**Note 1:** \_AppStart should have the same file extension as your web pages, like: \_AppStart.cshtml.

**Note 2:** \_AppStart has an underscore prefix. Because of this, the files cannot be browsed directly.

## Before Every Page: \_PageStart

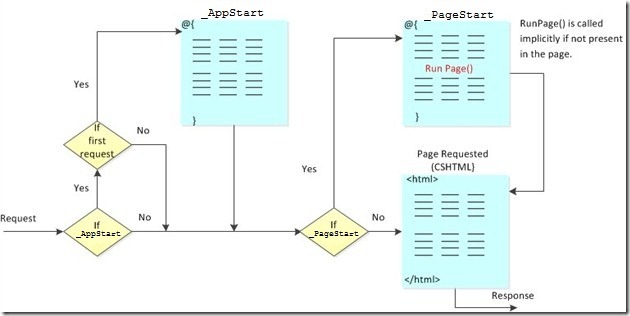
Just like \_AppStart runs before your site starts, you can write code that runs before any page in each folder.

For each folder in your web, you can add a file named \_PageStart.

Typical use for \_PageStart is setting the layout page for all pages in a folder, or checking that a user is logged in before running a page.

## How Does it Work?

The following diagram shows how it works:



When a request comes in, ASP.NET checks whether \_AppStart exists. If so, and this is the first request to the site, \_AppStart runs.

Then ASP.NET checks whether \_PageStart exists. If so, \_PageStart runs, before the requested page.

If you include a call to RunPage() inside \_PageStart you specify where you want the requested page to run. If not, the \_PageStart runs before the requested page.

# ASP.NET Web Pages - HTML Forms

A form is a section of an HTML document where you put input controls (text boxes, check boxes, radio buttons, and pull-down lists)

## Creating an HTML Input Page

## Razor Example

<html>  
<body>   
@{  
if (IsPost) {   
string companyname = Request["CompanyName"];   
string contactname = Request["ContactName"];   
<p>You entered: <br />  
Company Name: @companyname <br />  
Contact Name: @contactname </p>  
}  
else  
{  
<form method="post" action="">  
Company Name:<br />  
<input type="text" name="CompanyName" value="" /><br />  
Contact Name:<br />  
<input type="text" name="ContactName" value="" /><br /><br />  
<input type="submit" value="Submit" class="submit" />  
</form>  
}  
}   
</body>   
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_009)

## Razor Example - Displaying Images

Suppose you have 3 images in your image folder, and you want to display images dynamically by the users choice.

This is easily done by a little Razor code.

If you have an image called "Photo1.jpg" in your images folder on your web site, you can display the image using an HTML <img> element like this:

<img src="images/Photo1.jpg" alt="Sample" />

The example below shows how to display a selected picture which the user selects from a drop-down list:

## Razor Example

@{  
var imagePath="";   
if (Request["Choice"] != null)  
   {imagePath="images/" + Request["Choice"];}   
}   
<!DOCTYPE html>   
<html>   
<body>   
<h1>Display Images</h1>   
<form method="post" action="">   
I want to see:   
<select name="Choice">   
  <option value="Photo1.jpg">Photo 1</option>   
  <option value="Photo2.jpg">Photo 2</option>   
  <option value="Photo3.jpg">Photo 3</option>   
</select>   
<input type="submit" value="Submit" />   
@if (imagePath != "")  
{  
<p>  
<img src="@imagePath" alt="Sample" />  
</p>  
}    
</form>   
</body>   
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_010)

## Example explained

The server creates a variable called **imagePath**.

The HTML page has a **drop-down list** (a <select> element) named **Choice**. It lets the user select a friendly name (like **Photo 1**), and passes a file name (like **Photo1.jpg**) when the page is submitted to the web server.

The Razor code reads the value of Choice by **Request["Choice"]**. If it has a value the code constructs a path to the image (images/Photo1.jpg, and stores it in the variable **imagePath**.

In the HTML page there is an <img> element to display the image. The src attribute is set to the value of the imagePath variable when the page displays.

The <img> element is in an if block to prevent trying to display an image with no name (like the first time the page is displayed.

**ASP.NET Web Pages - Objects**

**The Page Object**

You have already seen some Page Object methods in use:

@RenderPage("header.cshtml")  
  
@RenderBody()

In the previous chapter you saw two Page Object properties being used (IsPost, and Request):

If (IsPost) {  
  
if (Request["Choice"] != null) {

**Some Page Object Methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| href | Builds a URL using the specified parameters |
| RenderBody() | Renders the portion of a content page that is not within a named section (In layout pages) |
| RenderPage(*page*) | Renders the content of one page within another page |
| RenderSection(*section*) | Renders the content of a named section (In layout pages) |
| Write(*object*) | Writes the object as an HTML-encoded string |
| WriteLiteral | Writes an object without HTML-encoding it first. |

**Some Page Object Properties**

|  |  |
| --- | --- |
| **Property** | **Description** |
| IsPost | Returns true if the HTTP data transfer method used by the client is a POST request |
| Layout | Gets or sets the path of a layout page |
| Page | Provides property-like access to data shared between pages and layout pages |
| Request | Gets the HttpRequest object for the current HTTP request |
| Server | Gets the HttpServerUtility object that provides web-page processing methods |

**The Page Property (of the Page Object)**

The Page property of the Page Object, provides property-like access to data shared between pages and layout pages.

You can use (add) your own properties to the Page property:

* Page.Title
* Page.Version
* Page.anythingyoulike

The pages property is very helpful. For instance, it makes it possible to set the page title in content files, and use it in the layout file:

**Home.cshtml**

@{  
Layout="~/Shared/Layout.cshtml";  
Page.Title="Home Page"  
}  
  
<h1>Welcome to W3Schools</h1>   
  
<h2>Web Site Main Ingredients</h2>  
  
<p>A Home Page (Default.cshtml)</p>  
<p>A Layout File (Layout.cshtml)</p>  
<p>A Style Sheet (Site.css)</p>

**Layout.cshtml**

<!DOCTYPE html>  
<html>  
<head>  
<title>@Page.Title</title>  
</head>  
<body>  
@RenderBody()  
</body>  
</html>

# ASP.NET Web Pages - Files

## Working with Text Files

Sometimes you will want to access data stored in text files.

Text files used to store data is often called flat files.

Common flat file formats are .txt, .xml, and .csv (comma-delimited values).

**In this chapter you will learn:**

* How to read and display data from a text file

## Add a Text File Manually

In the example to follow, you will need a text file to work with.

On your web site, if you don't have an App\_Data folder, create one.

In the App\_Data folder, create a new file named Persons.txt.

Add the following content to the file:

## Persons.txt

George,Lucas  
Steven,Spielberg  
Alfred,Hitchcock

## Displaying Data from a Text File

The example below shows how to display data from a text file:

## Example

@{  
var dataFile = Server.MapPath("~/App\_Data/Persons.txt");  
Array userData = File.ReadAllLines(dataFile);  
}  
  
<!DOCTYPE html>  
<html>  
<body>  
  
<h1>Reading Data from a File</h1>  
@foreach (string dataLine in userData)   
{  
  foreach (string dataItem in dataLine.Split(','))   
  {@dataItem <text>&nbsp;</text>}  
  <br />  
}  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_019)

## Example explained

**Server.MapPath** finds the exact text file path.

**File.ReadAllLines** opens the text file and reads all lines from the file into an array.

For each **dataItem** in each **dataline** of the array the data is displayed.

## Displaying Data from an Excel File

With Microsoft Excel, you can save a spreadsheet as a comma separated text file (.csv file). When you do so, each row in the spreadsheet is saved as a text line, and each data column is separated by a comma.

You can use the example above to read an Excel .csv file (just change the file name to the name of the Excel file).

**ASP.NET Web Pages - Databases**

**What We Will Do**

In this chapter we will:

* Create a web page to list data from a database

**Displaying Data from Database**

With Web Pages, you can easily display data from a database.

You can connect to an existing database, or create a new database from scratch.

In this example we will connect to an existing SQL Server Compact database.

If you want to learn how to create a database for your web, please go to the chapter [Web Database](http://www.w3schools.com/website/web_database.asp).

**Adding a Customers Page**

In the "DemoWebPages" folder, create a new CSHTML file named "Products.cshtml".

Replace the code in the file with the code from the example below:

**Products.cshtml**

@{  
var db = Database.Open("SmallBakery");   
var selectQueryString = "SELECT \* FROM Product ORDER BY Name";   
}  
<html>   
<body>   
<h1>Small Bakery Products</h1>   
<table>   
<tr>  
<th>Id</th>   
<th>Product</th>   
<th>Description</th>   
<th>Price</th>   
</tr>  
@foreach(var row in db.Query(selectQueryString))  
{  
<tr>   
<td>@row.Id</td>   
<td>@row.Name</td>   
<td>@row.Description</td>   
<td align="right">@row.Price</td>   
</tr>   
}  
</table>   
</body>   
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_003)

**Example Explained**

The Database.Open(*name*) method will connect to a database in two steps:

First, it searches the application's App\_Data folder for a database that matches the *name* parameter without the file-name extension.

If no file is found, it looks for a "connection string" in the application's Web.config file.

(A connection string contains information about how to connect to a database. It can include a file path, or the name of an SQL database, with full user name and password)

This two-step search makes it possible to test the application with a local database, and run the application on a web host using a connection string.

**ASP.NET Web Pages - Helpers**

Web Helpers greatly simplifies web development and common programming tasks.

**ASP.NET Helpers**

ASP.NET helpers are components that can be accessed by single lines of Razor code.

You can build your own helpers using Razor syntax stored as .cshtml files, or use built-in ASP.NET helpers.

You will learn how to use Razor helpers in the next chapters of this tutorial.

Below is a short description of some useful Razor helpers:

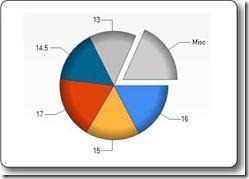
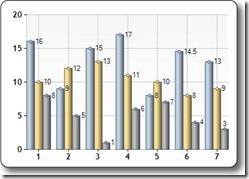
**The WebGrid Helper**

The WebGrid helper simplifies the way to display data:

* Automatically sets up an HTML table to display data
* Supports different options for formatting
* Supports paging (First, next, previous, last) through data
* Supports sorting by clicking on column headings

**The Chart Helper**

The "Chart Helper" can display chart images of different types with many formatting options and labels.



The Chart helper can display data from arrays , from databases, or from files.

**The WebMail Helper**

The WebMail helper provides functions for sending email messages using SMTP (Simple Mail Transfer Protocol).

**The WebImage Helper**

The WebImage helper provides functionality to manage images in a web page.

Keywords: flip, rotate, resize, watermark.

**Third Party Helpers**

With Razor you can take advantage of built-in or third party helpers to simplify the use of email, databases, multimedia, and social networks as well as many other issues like navigation and web security.

**Installing Helpers**

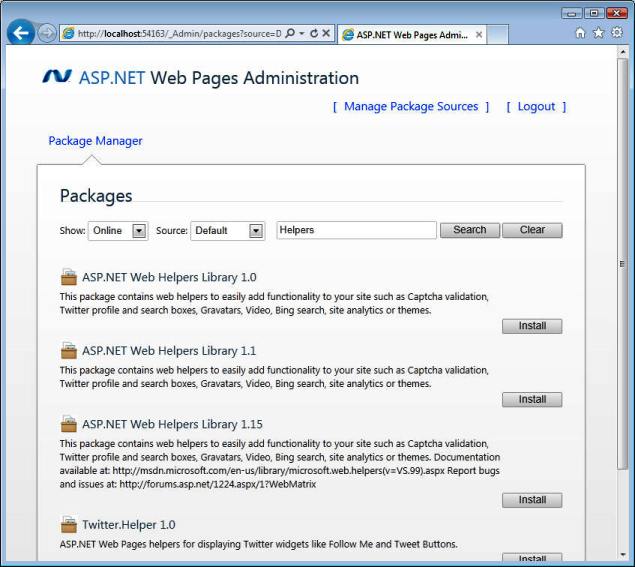
Some helpers are already included with WebMatrix, but you can install others as well.

In the [W3Schools Helper Reference](http://www.w3schools.com/aspnet/webpages_ref_helpers.asp) you can find a quick reference for included helpers and other helpers that you can install as part of a package called the ASP.NET Web Helpers Library.

If you have a web site created in WebMatrix, use the following procedure to install helpers:

1. In WebMatrix, open the **Site** workspace.
2. Click on **Web Pages Administration**.
3. Login to Web Pages Administration using a password \*.
4. Search for helpers using the **search field**.
5. Click **install** to install your desired helpers.

(\* the first time you use Web Pages Administration, it will prompt you to create a password)



**ASP.NET Web Pages - The WebGrid Helper**

**Doing the HTML Yourself**

In a previous chapter, you displayed database data by using razor code, and doing the HTML markup yourself:

**Database Example**

@{  
var db = Database.Open("SmallBakery");   
var selectQueryString = "SELECT \* FROM Product ORDER BY Name";   
}  
<html>   
<body>   
<h1>Small Bakery Products</h1>   
<table>   
<tr>  
<th>Id</th>   
<th>Product</th>   
<th>Description</th>   
<th>Price</th>   
</tr>  
@foreach(var row in db.Query(selectQueryString))  
{  
<tr>   
<td>@row.Id</td>   
<td>@row.Name</td>   
<td>@row.Description</td>   
<td style="text-align:right">@row.Price</td>   
</tr>   
}  
</table>   
</body>   
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_003)

**Using The WebGrid Helper**

Using the WebGrid helper is an easier way to display data.

The WebGrid helper:

* Automatically sets up an HTML table to display data
* Supports different options for formatting
* Supports paging through data
* Supports Sorting by clicking on column headings

**WebGrid Example**

@{   
var db = Database.Open("SmallBakery") ;   
var selectQueryString = "SELECT \* FROM Product ORDER BY Name";   
var data = db.Query(selectQueryString);   
var grid = new WebGrid(data);   
}  
<html>   
<head>   
<title>Displaying Data Using the WebGrid Helper</title>   
</head>   
<body>   
<h1>Small Bakery Products</h1>   
<div id="grid">   
@grid.GetHtml()  
</div>   
</body>   
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_004)

# ASP.NET Web Pages - The Chart Helper

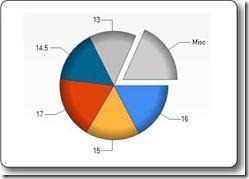
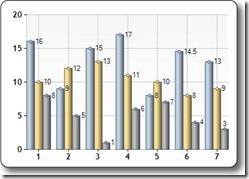
## The Chart Helper

In the previous chapters, you learned how to use an ASP.NET "Helper".

You learned how to display data in a grid using the "WebGrid Helper".

This chapter explains how to display data in graphical form, using the "Chart Helper".

The "Chart Helper" can create chart images of different types with many formatting options and labels. It can create standard charts like area charts, bar charts, column charts, line charts, and pie charts, along with more specialized charts like stock charts.



The data you display in a chart can be from an array, from a database, or from data in a file.

## Chart From an Array

The example below shows the code needed to display a chart from an array of values:

## Example

@{   
var myChart = new Chart(width: 600, height: 400)   
   .AddTitle("Employees")   
   .AddSeries(chartType: "column",  
      xValue: new[] {  "Peter", "Andrew", "Julie", "Mary", "Dave" },   
      yValues: new[] { "2", "6", "4", "5", "3" })   
   .Write();  
}

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_005)

- **new Chart** creates a new chart object and sets its width and height

- the **AddTitle** method specifies the chart title

- the **AddSeries** method adds data to the chart

- the **chartType** parameter defines the type of chart

- the **xValue** parameter defines x-axis names

- the **yValues** parameter defines the y-axis values

- the **Write()** method displays the chart

## Chart From Database Data

You can run a database query and then use data from the results to create a chart:

## Example

@{   
var db = Database.Open("SmallBakery");   
var dbdata = db.Query("SELECT Name, Price FROM Product");   
var myChart = new Chart(width: 600, height: 400)   
   .AddTitle("Product Sales")   
   .DataBindTable(dataSource: dbdata, xField: "Name")  
   .Write();  
}

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_006)

- **var db = Database.Open** opens the database (and assigns the database object to the variable db)

- **var dbdata = db.Query** runs a database query and stores the result in dbdata

- **new Chart** creates a chart new object and sets its width and height

- the **AddTitle** method specifies the chart title

- the **DataBindTable** method binds the data source to the chart

- the **Write()** method displays the chart

An alternative to using the DataBindTable method is to use AddSeries (See previous example). DataBindTable is easier to use, but AddSeries is more flexible because you can specify the chart and data more explicitly:

## Example

@{   
var db = Database.Open("SmallBakery");   
var dbdata = db.Query("SELECT Name, Price FROM Product");   
var myChart = new Chart(width: 600, height: 400)   
   .AddTitle("Product Sales")   
   .AddSeries(chartType:"Pie",  
      xValue: dbdata, xField: "Name",  
      yValues: dbdata, yFields: "Price")  
   .Write();  
}

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_007)

## Chart From XML Data

The third option for charting is to use an XML file as the data for the chart:

## Example

@using System.Data;  
  
@{  
var dataSet = new DataSet();  
dataSet.ReadXmlSchema(Server.MapPath("data.xsd"));  
dataSet.ReadXml(Server.MapPath("data.xml"));  
var dataView = new DataView(dataSet.Tables[0]);  
var myChart = new Chart(width: 600, height: 400)  
   .AddTitle("Sales Per Employee")  
   .AddSeries("Default", chartType: "Pie",  
      xValue: dataView, xField: "Name",  
      yValues: dataView, yFields: "Sales")  
   .Write();}  
}

# ASP.NET Web Pages - The WebMail Helper

## The WebMail Helper

The WebMail Helper makes it easy to send an email from a web application using SMTP (Simple Mail transfer Protocol).

## Scenario: Email Support

To demonstrate the use of email, we will create an input page for support, let the user submit the page to another page, and send an email about the support problem.

## First: Edit Your AppStart Page

If you have built the Demo application in this tutorial, you already have a page called \_AppStart.cshtml with the following content:

## \_AppStart.cshtml

@{  
WebSecurity.InitializeDatabaseConnection("Users", "UserProfile", "UserId", "Email", true);  
}

To initiate the WebMail helper, add the the following WebMail properties to your AppStart page:

## \_AppStart.cshtml

@{  
WebSecurity.InitializeDatabaseConnection("Users", "UserProfile", "UserId", "Email", true);  
WebMail.SmtpServer = "smtp.example.com";  
WebMail.SmtpPort = 25;  
WebMail.EnableSsl = false;  
WebMail.UserName = "support@example.com";  
WebMail.Password = "password-goes-here";  
WebMail.From = "john@example.com";  
}

Properties explained:

**SmtpServer:** The name the SMTP server that will be used to send the emails.

**SmtpPort:** The port the server will use to send SMTP transactions (emails).

**EnableSsl:** True, if the server should use SSL (Secure Socket Layer) encryption.

**UserName:** The name of the SMTP email account used to send the email.

**Password:** The password of the SMTP email account.

**From:** The email to appear in the from address (often the same as UserName).

## Second: Create an Email Input Page

Then create an input page, and name it Email\_Input:

## Email\_Input.cshtml

<!DOCTYPE html>   
<html>   
<body>   
<h1>Request for Assistance</h1>   
  
<form method="post" action="EmailSend.cshtml">   
<label>Username:</label>  
<input type="text name="customerEmail" />  
<label>Details about the problem:</label>   
<textarea name="customerRequest" cols="45" rows="4"></textarea>   
<p><input type="submit" value="Submit" /></p>   
</form>   
  
</body>   
</html>

The purpose of the input page is to collect information, then submit the data to a new page that can send the information as an email.

## Third: Create An Email Send Page

Then create the page that will be used to send the email, and name it Email\_Send:

## Email\_Send.cshtml

@{ // Read input  
var customerEmail = Request["customerEmail"];  
var customerRequest = Request["customerRequest"];  
try  
{  
// Send email   
WebMail.Send(to:"someone@example.com", subject: "Help request from - " + customerEmail, body: customerRequest );   
}  
catch (Exception ex )  
{  
<text>@ex</text>   
}  
}

For more information about sending emails from a ASP.NET Web Pages application, please see the: [WebMail Object Reference](http://www.w3schools.com/aspnet/webpages_ref_webmail.asp).

[« Previous](http://www.w3schools.com/aspnet/webpages_chart.asp)

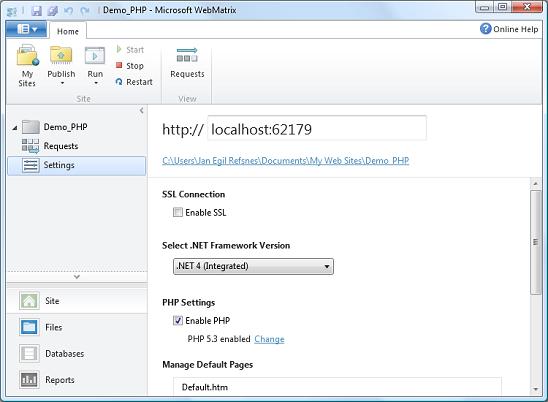
# ASP.NET Web Pages - PHP

## WebMatrix Supports PHP

At first look, WebMatrix only supports Microsoft technologies. This is not true. In WebMatrix you can write full PHP applications with MySQL and all.

## Create a PHP Site

In WebMatrix, create an empty site named "Demo\_PHP", enable PHP (see picture below), create a new empty page of the type PHP, name it "index.php", and you have created your first PHP site.



## Create a PHP Page

Put the following code inside the "index.php" file:

## index.php

<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
phpinfo();  
?>  
  
</body>  
</html>

Run the file and see PHP at work.

# ASP.NET Web Pages - Publishing the Website

Learn how to publish a Web Pages application without using WebMatrix.

## Publish Your Application Without Using WebMatrix

An ASP.NET Web Pages application can be published to a remote server by using the Publish commands in WebMatrix (or Visual Studio).

This function copies all your application files, cshtml pages, images, and all the required DLL files for Web Pages, for Razor, for Helpers, and for SQL Server Compact (if a database is used).

Sometimes you don't want to use this option. Maybe your hosting provider only supports FTP? Maybe you already have a web site based on classic ASP? Maybe you want to copy the files yourself? Maybe you want to use Front Page, Expression Web, or some other publishing software?

**Will you get a problem? Yes, you will. But you can solve it.**

To perform a web copy, you have to know how to include the right files, what DDL files to copy, and where store them.

Follow these steps:

## 1. Use the Latest Version of ASP.NET

Before you continue, make sure your hosting computer runs the latest version of ASP.NET (4.0 or 4.5).

## 2. Copy the Web Folders

Copy your website (all folders and content) from your development computer to an application folder on your remote hosting computer (server).

|  |  |
| --- | --- |
| **Note** | If your application contains data, **don't copy the data** (see point 4 below). |

## 3. The DLL Files

Make sure the bin folder, on your remote hosting computer, contains the same dll files as on your development computer.

After copying the bin folder, it should contain files like this:

 Microsoft.Web.Infrastructure.dll  
NuGet.Core.dll  
System.Web.Helpers.dll  
System.Web.Razor.dll  
System.Web.WebPages.Administration.dll  
System.Web.WebPages.Deployment.dll  
System.Web.WebPages.dll  
System.Web.WebPages.Razor.dll  
WebMatrix.Data.dll  
WebMatrix.WebData

## 4. Copy Your Data

If your application contains data or a database. For instance an SQL Server Compact database (a .sdf file in App\_Data folder), consider the following:

Do you want to publish your test data to the remote server?

Most likely not.

If you have test data on your development computer, it may overwrite production data on your remote hosting computer.

If you have to copy an SQL database (.sdf file), perhaps you should delete everything in the database, and then copy the empty .sdf file from your development computer to the server.

**THAT'S IT. GOOD LUCK !**

# ASP.NET Web Pages - Examples in C# and

Learn ASP.NET Web Pages by C# and Visual Basic examples.

|  |  |
| --- | --- |
| **Examples in C#** | **Examples in VB** |
| **Basic Web Pages**  [Display Date and Time](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_001) [Reusable Header and Footer](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_002) [Basic HTML Form](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_009)  [Examples Explained](http://www.w3schools.com/aspnet/webpages_intro.asp) | **Basic Web Pages**  [Display Date and Time](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_001) [Reusable Header and Footer](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_002) [Basic HTML Form](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_009)  [Examples Explained](http://www.w3schools.com/aspnet/webpages_intro.asp) |
| **Basic C#**  [For Loop](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_005) [For Each Loop](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_006) [While Loop](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_007) [Array](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_008)  [If Condition](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_010) [If Else Condition](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_011) [Else If Condition](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_012) [Switch Condition](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_013)  [Examples Explained](http://www.w3schools.com/aspnet/razor_intro.asp) | **Basic VB**  [For Loop](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_005) [For Each Loop](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_006) [While Loop](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_007) [Array](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_008)  [If Condition](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_010) [If Else Condition](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_011) [Else If Condition](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_012) [Select Condition](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_013)  [Examples Explained](http://www.w3schools.com/aspnet/razor_intro.asp) |
| **Working with Databases**  [Display Database Data](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_003) [Display Data with WebGrid](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_004)  [Examples Explained](http://www.w3schools.com/aspnet/webpages_database.asp) | **Working with Databases**  [Display Database Data](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_003) [Display Data with WebGrid](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_004)  [Examples Explained](http://www.w3schools.com/aspnet/webpages_database.asp) |
| **Using the Chart Helper**  [Display a Bar Chart from an Array](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_005) [Display a Bar Chart from a Database](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_006) [Display a Pie Chart from a Database](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_007) [Display a Pie Chart from an XML File](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_008)  [Examples Explained](http://www.w3schools.com/aspnet/webpages_chart.asp) | **Using the Chart Helper**  [Display a Bar Chart from an Array](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_005) [Display a Bar Chart from a Database](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_006) [Display a Pie Chart from a Database](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_007) [Display a Pie Chart from an XML File](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_008)  [Examples Explained](http://www.w3schools.com/aspnet/webpages_chart.asp) |

# ASP.NET Web Pages - Classes

## ASP.NET Classes Reference

|  |  |
| --- | --- |
| **Method** | **Description** |
| AsBool(), AsBool(true|false) | Converts a string value to a Boolean value (true/false). Returns false or the specified value if the string does not represent true/false. |
| AsDateTime(), AsDateTime(value) | Converts a string value to date/time. Returns DateTime. MinValue or the specified value if the string does not represent a date/time. |
| AsDecimal(), AsDecimal(value) | Converts a string value to a decimal value. Returns 0.0 or the specified value if the string does not represent a decimal value. |
| AsFloat(), AsFloat(value) | Converts a string value to a float. Returns 0.0 or the specified value if the string does not represent a decimal value. |
| AsInt(), AsInt(value) | Converts a string value to an integer. Returns 0 or the specified value if the string does not represent an integer. |
| Href(path [, param1 [, param2]]) | Creates a browser-compatible URL from a local file path, with optional additional path parts. |
| Html.Raw(value) | Renders value as HTML markup instead of rendering it as HTML-encoded output. |
| IsBool(), IsDateTime(), IsDecimal(), IsFloat(), IsInt() | Returns true if the value can be converted from a string to the specified type. |
| IsEmpty() | Returns true if the object or variable has no value. |
| IsPost | Returns true if the request is a POST. (Initial requests are usually a GET.) |
| Layout | Specifies the path of a layout page to apply to this page. |
| PageData[key], PageData[index], Page | Contains data shared between the page, layout pages, and partial pages in the current request. You can use the dynamic Page property to access the same data, as in the following example: |
| RenderBody() | (Layout pages) Renders the content of a content page that is not in any named sections. |
| RenderPage(path, values)  RenderPage(path[, param1 [, param2]]) | Renders a content page using the specified path and optional extra data. You can get the values of the extra parameters from PageData by position (example 1) or key (example 2). |
| RenderSection(sectionName [, required = true|false]) | (Layout pages) Renders a content section that has a name. Set required to false to make a section optional. |
| Request.Cookies[key] | Gets or sets the value of an HTTP cookie. |
| Request.Files[key] | Gets the files that were uploaded in the current request. |
| Request.Form[key] | Gets data that was posted in a form (as strings). Request[key] checks both the Request.Form and the Request.QueryString collections. |
| Request.QueryString[key] | Gets data that was specified in the URL query string. Request[key] checks both the Request.Form and the Request.QueryString collections. |
| Request.Unvalidated(key)  Request.Unvalidated().QueryString|Form|Cookies|Headers[key] | Selectively disables request validation for a form element, query-string value, cookie, or header value. Request validation is enabled by default and prevents users from posting markup or other potentially dangerous content. |
| Response.AddHeader(name, value) | Adds an HTTP server header to the response. |
| Response.OutputCache(seconds [, sliding] [, varyByParams]) | Caches the page output for a specified time. Optionally set sliding to reset the timeout on each page access and varyByParams to cache different versions of the page for each different query string in the page request. |
| Response.Redirect(path) | Redirects the browser request to a new location. |
| Response.SetStatus(httpStatusCode) | Sets the HTTP status code sent to the browser. |
| Response.WriteBinary(data [, mimetype]) | Writes the contents of data to the response with an optional MIME type. |
| Response.WriteFile(file) | Writes the contents of a file to the response. |
| @section(sectionName) { content } | (Layout pages) Defines a content section that has a name. |
| Server.HtmlDecode(htmlText) | Decodes a string that is HTML encoded. |
| Server.HtmlEncode(text) | Encodes a string for rendering in HTML markup. |
| Server.MapPath(virtualPath) | Returns the server physical path for the specified virtual path. |
| Server.UrlDecode(urlText) | Decodes text from a URL. |
| Server.UrlEncode(text) | Encodes text to put in a URL. |
| Session[key] | Gets or sets a value that exists until the user closes the browser. |
| ToString() | Displays a string representation of the object's value. |
| UrlData[index] | Gets additional data from the URL (for example, /MyPage/ExtraData). |

# ASP.NET Web Pages - WebSecurity Object

## Description

The **WebSecurity Object** provides security and authentication for ASP.NET Web Pages applications.

With the WebSecurity object you can create user accounts, login and logout users, reset or change passwords, and more.

## WebSecurity Object Reference - Properties

|  |  |
| --- | --- |
| **Properties** | **Description** |
| [CurrentUserId](http://www.w3schools.com/aspnet/prop_websecurity_currentuserid.asp) | Gets the ID for the current user |
| [CurrentUserName](http://www.w3schools.com/aspnet/prop_websecurity_currentusername.asp) | Gets the name of the current user |
| [HasUserId](http://www.w3schools.com/aspnet/prop_websecurity_hasuserid.asp) | Returns true if the current has a user ID |
| [IsAuthenticated](http://www.w3schools.com/aspnet/prop_websecurity_isauthenticated.asp) | Returns true if the current user is logged in |

## WebSecurity Object Reference - Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| [ChangePassword()](http://www.w3schools.com/aspnet/met_websecurity_changepassword.asp) | Changes the password for a user |
| [ConfirmAccount()](http://www.w3schools.com/aspnet/met_websecurity_confirmaccount.asp) | Confirms an account using a confirmation token |
| [CreateAccount()](http://www.w3schools.com/aspnet/met_websecurity_createaccount.asp) | Creates a new user account |
| [CreateUserAndAccount()](http://www.w3schools.com/aspnet/met_websecurity_createuserandaccount.asp) | Creates a new user account |
| [GeneratePasswordResetToken()](http://www.w3schools.com/aspnet/met_websecurity_generatepasswordresettoken.asp) | Generates a token that can be sent to as user by email |
| [GetCreateDate()](http://www.w3schools.com/aspnet/met_websecurity_getcreatedate.asp) | Gets the time the specified membership was created |
| [GetPasswordChangeDate()](http://www.w3schools.com/aspnet/met_websecurity_getpasswordchangedate.asp) | Gets the date and time when password was changed |
| [GetUserId()](http://www.w3schools.com/aspnet/met_websecurity_getuserid.asp) | Gets a user ID from a user name |
| [InitializeDatabaseConnection()](http://www.w3schools.com/aspnet/met_websecurity_initializedatabaseconnection.asp) | Initializes the WebSecurity system (database) |
| [IsConfirmed()](http://www.w3schools.com/aspnet/met_websecurity_isconfirmed.asp) | Checks if a user is confirmed |
| [IsCurrentUser()](http://www.w3schools.com/aspnet/met_websecurity_iscurrentuser.asp) | Checks if the current user matches a user name |
| [Login()](http://www.w3schools.com/aspnet/met_websecurity_login.asp) | Logs the user in by setting a token in the cookie |
| [Logout()](http://www.w3schools.com/aspnet/met_websecurity_logout.asp) | Logs the user out by removing the token cookie |
| [RequireAuthenticatedUser()](http://www.w3schools.com/aspnet/met_websecurity_requireauthenticateduser.asp) | Exits the page if the user is not an authenticated user |
| [RequireRoles()](http://www.w3schools.com/aspnet/met_websecurity_requireroles.asp) | Exits the page if the user is not a part of the specified roles |
| [RequireUser()](http://www.w3schools.com/aspnet/met_websecurity_requireuser.asp) | Exits the page if the user is not the specified user |
| [ResetPassword()](http://www.w3schools.com/aspnet/met_websecurity_resetpassword.asp) | Changes a user's password using a token |
| [UserExists()](http://www.w3schools.com/aspnet/met_websecurity_userexists.asp) | Checks if a given user exists |

## Technical Data

|  |  |
| --- | --- |
| **Name** | **Value** |
| Class | WebMatrix.WebData.WebSecurity |
| Namespace | WebMatrix.WebData |
| Assembly | WebMatrix.WebData.dll |

## Initializing the WebSecurity Database

You must create or initialize an WebSecurity database before you can use the WebSecurity object in your code.

In the root of your web, create a page (or edit the page ) named **\_AppStart.cshtml**.

Put the following code inside the file:

## \_AppStart.cshtml

@{  
WebSecurity.InitializeDatabaseConnection("Users", "UserProfile", "UserId", "Email", true);  
}

The code above will run each time the web site (application) starts. It initializes the WebSecurity database.

**"Users"** is the name of the WebSecurity database (Users.sdf).

**"UserProfile"** is the name of the database table that contains the user profile information.

**"UserId"** is the name of the column that contains the user IDs (primary key).

**"Email"** is the name of the column that contains user names.

The last parameter **true** is a boolean value indicating that the user profile and membership tables should be created automatically if they don't exist, otherwise **false**.

|  |  |
| --- | --- |
| **Note** | Although **true** indicates automatic creation of the database **tables,** the database itself will not be created automatically. It must exist. |

## The WebSecurity Database

The **UserProfile** table contains one record for each user, with a user ID (primary key) and the user's name (email):

|  |  |
| --- | --- |
| **UserId** | **Email** |
| 1 | john@johnson.net |
| 2 | peter@peterson.com |
| 3 | lars@larson.eut |

The **Membership table** will contain membership information about when the user was created and if (and when) the membership was confirmed.

Much like this (some columns are not shown):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Id** | **Create Date** | **Confirmation Token** | **Is Confirmed** | **Last Password Failure** | **Password** | **Password Change** |
| 1 | 12.04.2012 16:12:17 | NULL | True | NULL | AFNQhWfy.... | 12.04.2012 16:12:17 |

Note: If you want to see all columns and all content, open the database with WebMatrix and look inside each table.

## Simple Membership Configuration

You might get errors using the WebSecurity object, if your site is not configured to use the ASP.NET Web Pages membership system **SimpleMembership**.

This can occur if a hosting provider's server is configured differently than your local server. To fix this, add the following element to the site's Web.config file:

<appSettings>   
<add key="enableSimpleMembership" value="true" />   
</appSettings>

# ASP.NET Web Pages - Database Object

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_websecurity.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_ref_webmail.asp)

## ASP.NET Database Object Reference

|  |  |
| --- | --- |
| **Method** | **Description** |
| Database.Execute(SQLstatement [, parameters]) | Executes SQLstatement (with optional parameters) such as INSERT, DELETE, or UPDATE and returns a count of affected records. |
| Database.GetLastInsertId() | Returns the identity column from the most recently inserted row. |
| Database.Open(filename)  Database.Open(connectionStringName) | Opens either the specified database file or the database specified using a named connection string from the Web.config file. |
| Database.OpenConnectionString(connectionString) | Opens a database using the connection string. (This contrasts with Database.Open, which uses a connection string name.) |
| Database.Query(SQLstatement[, parameters]) | Queries the database using SQLstatement (optionally passing parameters) and returns the results as a collection. |
| Database.QuerySingle(SQLstatement [, parameters] |  |

**ASP.NET Web Pages - WebMail Object**

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_database.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_ref_helpers.asp)

With the WebMail object you can easily send emails from a web page.

**Description**

The **WebMail Object** provides email for ASP.NET Web Pages using SMTP (Simple Mail Transfer Protocol).

**Example**

See an example in the chapter: [Web Pages Email](http://www.w3schools.com/aspnet/webpages_email.asp).

**WebMail Object Reference - Properties**

|  |  |
| --- | --- |
| **Properties** | **Description** |
| SmtpServer | The name the SMTP server that will send the emails |
| SmtpPort | The port the server will use to send SMTP emails |
| EnableSsl | True, if the server should use SSL encryption |
| UserName | The name of the SMTP account used to send the email |
| Password | The password of the SMTP account |
| From | The email to appear in the from address |

**WebMail Object Reference - Methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| Send() | Sends an email message to an SMTP server for delivery |

The Send() method has the following parameters:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| to | String | The Email recipients (separated by semicolon) |
| subject | String | The subject line |
| body | String | The body of the message |

And the following optional parameters:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| from | String | The email of the sender |
| cc | String | The cc emails (separated by semicolon) |
| filesToAttach | Collection | Filenames |
| isBodyHtml | Boolean | True if the email body is in HTML |
| additionalHeaders | Collection | Additional headers |

**Technical Data**

|  |  |
| --- | --- |
| **Name** | **Value** |
| Class | System.Web.Helpers.WebMail |
| Namespace | System.Web.Helpers |
| Assembly | System.Web.Helpers.dll |

**Initializing the WebMail Helper**

To use the WebMail helper, you need access to an SMTP server. SMTP is the "output" part of email. If you use a web host, you probably already know the name of the SMTP server. If you work in a corporate network, your IT department can give you the name. If you are working at home, you might be able to use your ordinary email provider.

 In order to send an email you will need:

* The name of the SMTP server
* The port number (most often 25)
* An email user name
* An email password

In the root of your web, create a page (or edit the page ) named **\_AppStart.cshtml**.

Put the following code inside the file:

**\_AppStart.cshtml**

@{  
WebMail.SmtpServer = "smtp.example.com";  
WebMail.SmtpPort = 25;  
WebMail.EnableSsl = false;  
WebMail.UserName = "support@example.com";  
WebMail.Password = "password";  
WebMail.From = "john@example.com"  
}

The code above will run each time the web site (application) starts. It feeds your **WebMail Object** with initial values.

Please substitute:

**smtp.example.com** with the name the SMTP server that will be used to send the emails.

**25** with the port number the server will use to send SMTP transactions (emails).

**false** with true, if the server should use SSL (Secure Socket Layer) encryption.

**support@example.com** with the name of the SMTP email account used to send emails.

**password** with the password of the SMTP email account.

**john@example** with the email to appear in the from address.

|  |  |
| --- | --- |
| **Note** | You don't **have to** initiate the WebMail object in your AppStart file, but you must set these properties before you call the **WebMail.Send()** method. |

# ASP.NET Web Pages - More Helpers

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_webmail.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/razor_intro.asp)

## ASP.NET Helpers - Objects References

## Analytics Object Reference (Google)

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Analytics.GetGoogleHtml(webPropertyId) | Renders the Google Analytics JavaScript code for the specified ID. |
| Analytics.GetStatCounterHtml(project, security) | Renders the StatCounter Analytics JavaScript code for the specified project. |
| Analytics.GetYahooHtml(account) | Renders the Yahoo Analytics JavaScript code for the specified account. |

## Bing Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Bing.SearchBox([boxWidth]) | Passes a search to Bing. To specify the site to search and a title for the search box, you can set the Bing.SiteUrl and Bing.SiteTitle properties. Normally you set these properties in the \_AppStart page. |
| Bing.AdvancedSearchBox([, boxWidth] [, resultWidth] [, resultHeight]   [, themeColor] [, locale]) | Displays Bing search results in the page with optional formatting. To specify the site to search and a title for the search box, you can set the Bing.SiteUrl and Bing.SiteTitle properties. Normally you set these properties in the \_AppStart page. |

## Chart Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
|  |  |
| Chart(width, height [, template] [, templatePath]) | Initializes a chart. |
| Chart.AddLegend([title] [, name]) | Adds a legend to a chart. |
| Chart.AddSeries([name] [, chartType] [, chartArea]   [, axisLabel] [, legend] [, markerStep] [, xValue]   [, xField] [, yValues] [, yFields] [, options]) | Adds a series of values to the chart. |

## Crypto Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Crypto.Hash(string [, algorithm])  Crypto.Hash(bytes [, algorithm]) | Returns a hash for the specified data. The default algorithm is sha256. |

## Facebook Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Facebook.LikeButton(href [, buttonLayout] [, showFaces] [, width] [, height] [, action] [, font] [, colorScheme] [, refLabel]) | Lets Facebook users make a connection to pages. |

## FileUpload Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| FileUpload.GetHtml([initialNumberOfFiles] [, allowMoreFilesToBeAdded]   [, includeFormTag] [, addText] [, uploadText]) | Renders UI for uploading files. |

## GamerCard Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| GamerCard.GetHtml(gamerTag) | Renders the specified Xbox gamer tag. |

## Gravatar Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Gravatar.GetHtml(email [, imageSize] [, defaultImage] [, rating]   [, imageExtension] [, attributes]) | Renders the Gravatar image for the specified email address. |

## Json Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Json.Encode(object) | Converts a data object to a string in the JavaScript Object Notation (JSON) format. |
| Json.Decode(string) | Converts a JSON-encoded input string to a data object that you can iterate over or insert into a database. |

## LinkShare Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| LinkShare.GetHtml(pageTitle [, pageLinkBack] [, twitterUserName]   [, additionalTweetText] [, linkSites]) | Renders social networking links using the specified title and optional URL. |

## ModelState Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| ModelStateDictionary.AddError(key, errorMessage) | Associates an error message with a form field. Use the ModelState helper to access this member. |
| ModelStateDictionary.AddFormError(errorMessage) | Associates an error message with a form. Use the ModelState helper to access this member. |
| ModelStateDictionary.IsValid | Returns true if there are no validation errors. Use the ModelState helper to access this member. |

## ObjectInfo Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| ObjectInfo.Print(value [, depth] [, enumerationLength]) | Renders the properties and values of an object and any child objects. |

## Recaptcha Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Recaptcha.GetHtml([, publicKey] [, theme] [, language] [, tabIndex]) | Renders the reCAPTCHA verification test. |
| ReCaptcha.PublicKey  ReCaptcha.PrivateKey | Sets public and private keys for the reCAPTCHA service. Normally you set these properties in the \_AppStart page. |
| ReCaptcha.Validate([, privateKey]) | Returns the result of the reCAPTCHA test. |
|  |  |

## ServerInfo Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| ServerInfo.GetHtml() | Renders status information about ASP.NET Web Pages. |

## Twitter Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Twitter.Profile(twitterUserName) | Renders a Twitter stream for the specified user. |
| Twitter.Search(searchQuery) | Renders a Twitter stream for the specified search text. |

## Video Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Video.Flash(filename [, width, height]) | Renders a Flash video player for the specified file with optional width and height. |
| Video.MediaPlayer(filename [, width, height]) | Renders a Windows Media player for the specified file with optional width and height. |
| Video.Silverlight(filename, width, height) | Renders a Silverlight player for the specified .xap file with required width and height. |

## WebCache Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| WebCache.Get(key) | Returns the object specified by key, or null if the object is not found. |
| WebCache.Remove(key) | Removes the object specified by key from the cache. |
| WebCache.Set(key, value [, minutesToCache] [, slidingExpiration]) | Puts value into the cache under the name specified by key. |

## WebGrid Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| WebGrid(data) | Creates a new WebGrid object using data from a query. |
| WebGrid.GetHtml() | Renders markup to display data in an HTML table. |
| WebGrid.Pager() | Renders a pager for the WebGrid object. |

## WebImage Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| WebImage(path) | Loads an image from the specified path. |
| WebImage.AddImagesWatermark(image) | Adds the specified image as a watermark. |
| WebImage.AddTextWatermark(text) | Adds the specified text to the image. |
| WebImage.FlipHorizontal()  WebImage.FlipVertical() | Flips the image horizontally or vertically. |
| WebImage.GetImageFromRequest() | Loads an image when an image is posted to a page during a file upload. |
| WebImage.Resize(width, height) | Resizes the image. |
| WebImage.RotateLeft()  WebImage.RotateRight() | Rotates the image to the left or the right. |
| WebImage.Save(path [, imageFormat]) | Saves the image to the specified path. |

**ASP.NET Razor - Markup**

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_helpers.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/razor_syntax.asp)

Razor is not a programming language. It's a server side markup language.

**What is Razor?**

Razor is a markup syntax that lets you embed server-based code (Visual Basic and C#) into web pages.

Server-based code can create dynamic web content on the fly, while a web page is written to the browser. When a web page is called, the server executes the server-based code inside the page before it returns the page to the browser. By running on the server, the code can perform complex tasks, like accessing databases.

Razor is based on ASP.NET, and designed for creating web applications. It has the power of traditional ASP.NET markup, but it is easier to use, and easier to learn.

**Razor Syntax**

Razor uses a syntax very similar to PHP and Classic ASP.

Razor:

<ul>  
@for (int i = 0; i < 10; i++) {  
<li>@i</li>  
}  
</ul>

PHP:

<ul>  
<?php   
for ($i = 0; $i < 10; $i++) {  
echo("<li>$i</li>");  
}   
?>  
</ul>

Web Forms (and Classic ASP):

<ul>  
<% for (int i = 0; i < 10; i++) { %>  
<li><% =i %></li>  
<% } %>  
</ul>

**Razor Helpers**

ASP.NET helpers are components that can be accessed by single lines of Razor code.

You can build your own helpers using Razor syntax, or use built-in ASP.NET helpers.

Below is a short description of some useful Razor helpers:

* Web Grid
* Web Graphics
* Google Analytics
* Facebook Integration
* Twitter Integration
* Sending Email
* Validation

**Razor Programming Languages**

Razor supports both C# (C sharp) and VB (Visual Basic).

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_helpers.asp)

[Next C](http://www.w3schools.com/aspnet/razor_syntax.asp)

**ASP.NET Razor - C# and VB Code Syntax**

[« Previous](http://www.w3schools.com/aspnet/razor_intro.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/razor_cs_variables.asp)

Razor supports both C# (C sharp) and VB (Visual Basic).

**Main Razor Syntax Rules for C#**

* Razor code blocks are enclosed in @{ ... }
* Inline expressions (variables and functions) start with @
* Code statements end with semicolon
* Variables are declared with the var keyword
* Strings are enclosed with quotation marks
* C# code is case sensitive
* C# files have the extension .cshtml

**C# Example**

<!-- Single statement block -->  
@{ var myMessage = "Hello World"; }  
  
<!-- Inline expression or variable -->  
<p>The value of myMessage is: @myMessage</p>   
  
<!-- Multi-statement block -->  
@{  
var greeting = "Welcome to our site!";  
var weekDay = DateTime.Now.DayOfWeek;  
var greetingMessage = greeting + " Here in Huston it is: " + weekDay;  
}  
<p>The greeting is: @greetingMessage</p>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_001)

**Main Razor Syntax Rules for VB**

* Razor code blocks are enclosed in @Code ... End Code
* Inline expressions (variables and functions) start with @
* Variables are declared with the Dim keyword
* Strings are enclosed with quotation marks
* VB code is not case sensitive
* VB files have the extension .vbhtml

**Example**

<!-- Single statement block  -->   
@Code dim myMessage = "Hello World" End Code  
   
<!-- Inline expression or variable -->   
<p>The value of myMessage is: @myMessage</p>   
   
<!-- Multi-statement block -->   
@Code  
dim greeting = "Welcome to our site!"   
dim weekDay = DateTime.Now.DayOfWeek   
dim greetingMessage = greeting & " Here in Huston it is: " & weekDay  
End Code   
  
<p>The greeting is: @greetingMessage</p>

[Run example »](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_001)

**How Does it Work?**

Razor is a simple programming syntax for embedding server code in web pages.

Razor syntax is based on the ASP.NET framework, the part of the Microsoft.NET Framework that's specifically designed for creating web applications.

The Razor syntax gives you all the power of ASP.NET, but is using a simplified syntax that's easier to learn if you're a beginner, and makes you more productive if you're an expert.

Razor web pages can be described as HTML pages with two kinds of content: HTML content and Razor code.

When the server reads the page, it runs the Razor code first, before it sends the HTML page to the browser. The code that is executed on the server can perform tasks that cannot be done in the browser, for example accessing a server database. Server code can create dynamic HTML content on the fly, before it is sent to the browser. Seen from the browser, the HTML generated by server code is no different than static HTML content.

ASP.NET web pages with Razor syntax have the special file extension cshtml (Razor using C#) or vbhtml (Razor using VB).

**Working With Objects**

Server coding often involves objects.  
  
The "Date" object is a typical built-in ASP.NET object, but objects can also be self-defined, a web page, a text box, a file, a database record, etc.  
  
Objects may have methods they can perform. A database record might have a "Save" method, an image object might have a "Rotate" method, an email object might have a "Send" method, and so on.  
  
Objects also have properties that describe their characteristics. A database record might have a FirstName and a LastName property (amongst others).

The ASP.NET Date object has a Now property (written as Date.Now), and the Now property has a Day property (written as Date.Now.Day). The example below shows how to access some properties of the Date object:

**Example**

<table border="1">  
<tr>  
<th width="100px">Name</th>  
<td width="100px">Value</td>  
</tr>  
<tr>  
<td>Day</td><td>@DateTime.Now.Day</td>  
</tr>  
<tr>  
<td>Hour</td><td>@DateTime.Now.Hour</td>  
</tr>  
<tr>  
<td>Minute</td><td>@DateTime.Now.Minute</td>  
</tr>  
<tr>  
<td>Second</td><td>@DateTime.Now.Second</td>  
</tr>  
</td>  
</table>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_002)

**If and Else Conditions**

An important feature of dynamic web pages is that you can determine what to do based on conditions.

The common way to do this is with the if ... else statements:

**Example**

@{  
var txt = "";  
if(DateTime.Now.Hour > 12)  
  {txt = "Good Evening";}  
else  
  {txt = "Good Morning";}  
}  
<html>  
<body>  
<p>The message is @txt</p>  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_003)

**Reading User Input**

Another important feature of dynamic web pages is that you can read user input.

Input is read by the Request[] function, and posting (input) is tested by the IsPost condition:

**Example**

@{

var totalMessage = "";  
if(IsPost)  
    {  
    var num1 = Request["text1"];  
    var num2 = Request["text2"];  
    var total = num1.AsInt() + num2.AsInt();  
    totalMessage = "Total = " + total;  
    }  
}  
<html>  
<body style="background-color: beige; font-family: Verdana, Arial;">  
<form action="" method="post">  
<p><label for="text1">First Number:</label><br>  
<input type="text" name="text1" /></p>  
<p><label for="text2">Second Number:</label><br>  
<input type="text" name="text2" /></p>  
<p><input type="submit" value=" Add " /></p>  
</form>  
<p>@totalMessage</p>  
</body>  
</html>

# ASP.NET Razor - C# Variables

[« Previous](http://www.w3schools.com/aspnet/razor_syntax.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/razor_cs_loops.asp)

Variables are named entities used to store data.

## Variables

Variables are used to store data.

The name of a variable must begin with an alphabetic character and cannot contain whitespace or reserved characters.   
  
A variable can be of a specific type, indicating the kind of data it stores. String variables store string values ("Welcome to W3Schools"), integer variables store number values (103), date variables store date values, etc.  
  
Variables are declared using the var keyword, or by using the type (if you want to declare the type), but ASP.NET can usually determine data types automatically.

## Examples

// Using the var keyword:  
var greeting = "Welcome to W3Schools";  
var counter = 103;  
var today = DateTime.Today;  
  
// Using data types:  
string greeting = "Welcome to W3Schools";  
int counter = 103;  
DateTime today = DateTime.Today;

## Data Types

Below is a list of  common data types:

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Examples** |
| int | Integer (whole numbers) | 103, 12, 5168 |
| float | Floating-point number | 3.14, 3.4e38 |
| decimal | Decimal number (higher precision) | 1037.196543 |
| bool | Boolean | true, false |
| string | String | "Hello W3Schools", "John" |

## Operators

An operator tells ASP.NET what kind of command to perform in an expression.

 The C# language supports many operators. Below is a list of common operators:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Assigns a value to a variable. | i=6 |
| + - \* / | Adds a value or variable. Subtracts a value or variable. Multiplies a value or variable. Divides a value or variable. | i=5+5 i=5-5 i=5\*5 i=5/5 |
| += -= | Increments a variable. Decrements a variable. | i += 1 i -= 1 |
| == | Equality. Returns true if values are equal. | if (i==10) |
| != | Inequality. Returns true if values are not equal. | if (i!=10) |
| < > <= >= | Less than. Greater than. Less than or equal. Greater than or equal. | if (i<10) if (i>10) if (i<=10) if (i>=10) |
| + | Adding strings (concatenation). | "w3" + "schools" |
| . | Dot. Separate objects and methods. | DateTime.Hour |
| () | Parenthesis. Groups values. | (i+5) |
| () | Parenthesis. Passes parameters. | x=Add(i,5) |
| [] | Brackets. Accesses values in arrays or collections. | name[3] |
| ! | Not. Reverses true or false. | if (!ready) |
| && || | Logical AND. Logical OR. | if (ready && clear) if (ready || clear) |

## Converting Data Types

Converting from one data type to another is sometimes useful.  
  
The most common example is to convert string input to another type, such as an integer or a date.

As a rule, user input comes as strings, even if the user entered a number. Therefore, numeric input values must be converted to numbers before they can be used in calculations.

Below is a list of common conversion methods:

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| AsInt() IsInt() | Converts a string to an integer. | if (myString.IsInt())   {myInt=myString.AsInt();} |
| AsFloat() IsFloat() | Converts a string to a floating-point number. | if (myString.IsFloat())   {myFloat=myString.AsFloat();} |
| AsDecimal() IsDecimal() | Converts a string to a decimal number. | if (myString.IsDecimal())   {myDec=myString.AsDecimal();} |
| AsDateTime() IsDateTime() | Converts a string to an ASP.NET DateTime type. | myString="10/10/2012"; myDate=myString.AsDateTime(); |
| AsBool() IsBool() | Converts a string to a Boolean. | myString="True"; myBool=myString.AsBool(); |
| ToString() | Converts any data type to a string. | myInt=1234; myString=myInt.ToString(); |

[« Previous](http://www.w3schools.com/aspnet/razor_syntax.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/razor_cs_loops.asp)

# ASP.NET Razor - C# Loops and Arrays

[« Previous](http://www.w3schools.com/aspnet/razor_cs_variables.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/razor_cs_logic.asp)

Statements can be executed repeatedly in loops.

## For Loops

If you need to run the same statements repeatedly, you can program a loop.

If you know how many times you want to loop, you can use a **for loop**. This kind of loop is especially useful for counting up or counting down:

## Example

<html>  
<body>  
@for(var i = 10; i < 21; i++)  
    {<p>Line @i</p>}  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_005)

## For Each Loops

If you work with a collection or an array, you often use a **for each loop**.

A collection is a group of similar objects, and the for each loop lets you carry out a task on each item. The for each loop walks through a collection until it is finished.

The example below walks through the ASP.NET Request.ServerVariables collection.

## Example

<html>  
<body>  
<ul>  
@foreach (var x in Request.ServerVariables)  
    {<li>@x</li>}  
</ul>  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_006)

## While Loops

The **while loop** is a general purpose loop.

A while loop begins with the while keyword, followed by parentheses, where you specify how long the loop continues, then a block to repeat.

While loops typically add to, or subtract from, a variable used for counting.

In the example below, the += operator adds 1 to the variable i, each time the loop runs.

## Example

<html>  
<body>  
@{  
var i = 0;  
while (i < 5)  
    {  
    i += 1;  
    <p>Line @i</p>  
    }  
}  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_007)

## Arrays

An array is useful when you want to store similar variables but don't want to create a separate variable for each of them:

## Example

@{

string[] members = {"Jani", "Hege", "Kai", "Jim"};  
int i = Array.IndexOf(members, "Kai")+1;  
int len = members.Length;  
string x = members[2-1];  
}  
<html>  
<body>  
<h3>Members</h3>  
@foreach (var person in members)  
{  
<p>@person</p>  
}  
<p>The number of names in Members are @len</p>  
<p>The person at position 2 is @x</p>  
<p>Kai is now in position @i</p>  
</body>  
</html>

**ASP.NET Razor - C# Logic Conditions**

[« Previous](http://www.w3schools.com/aspnet/razor_cs_loops.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/razor_vb_variables.asp)

Programming Logic: Execute code based on conditions.

**The If Condition**

C# lets you execute code based on conditions.

To test a condition you use an **if statement**. The if statement returns true or false, based on your test:

* The if statement starts a code block
* The condition is written inside parenthesis
* The code inside the braces is executed if the test is true

**Example**

@{var price=50;}  
<html>  
<body>  
@if (price>30)  
    {  
    <p>The price is too high.</p>  
    }  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_010)

**The Else Condition**

An if statement can include an **else condition**.

The else condition defines the code to be executed if the condition is false.

**Example**

@{var price=20;}  
<html>  
<body>  
@if (price>30)  
  {  
  <p>The price is too high.</p>  
  }  
else  
  {  
  <p>The price is OK.</p>  
  }   
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_011)

**Note:** In the example above, if the first condition is true, it will be executed. The else condition covers "everything else".

**The Else If Condition**

Multiple conditions can be tested with an **else if condition**:

**Example**

@{var price=25;}  
<html>  
<body>  
@if (price>=30)  
  {  
  <p>The price is high.</p>  
  }  
else if (price>20 && price<30)   
  {  
  <p>The price is OK.</p>  
  }  
else  
  {  
  <p>The price is low.</p>  
  }      
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_012)

In the example above, if the first condition is true, it will be executed.

If not, then if the next condition is true, this condition will be executed.

You can have any number of else if conditions.

If none of the if and else if conditions are true, the last else block (without a condition) covers "everything else".

**Switch Conditions**

A **switch block** can be used to test a number of individual conditions:

**Example**

@{  
var weekday=DateTime.Now.DayOfWeek;  
var day=weekday.ToString();  
var message="";  
}  
<html>  
<body>  
@switch(day)  
{  
case "Monday":  
    message="This is the first weekday.";  
    break;  
case "Thursday":  
    message="Only one day before weekend.";  
    break;  
case "Friday":  
    message="Tomorrow is weekend!";  
    break;  
default:  
    message="Today is " + day;  
    break;  
}  
<p>@message</p>  
</body>  
</html>

# ASP.NET Razor - VB Variables

[« Previous](http://www.w3schools.com/aspnet/razor_cs_logic.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/razor_vb_loops.asp)

Variables are named entities used to store data.

## Variables

Variables are used to store data.

The name of a variable must begin with an alphabetic character and cannot contain whitespace or reserved characters.   
  
A variable can be of a specific type, indicating the kind of data it stores. String variables store string values ("Welcome to W3Schools"), integer variables store number values (103), date variables store date values, etc.  
  
Variables are declared using the Dim keyword, or by using the type (if you want to declare the type), but ASP.NET can usually determine data types automatically.

## Examples

// Using the Dim keyword:  
Dim greeting = "Welcome to W3Schools"  
Dim counter = 103  
Dim today = DateTime.Today  
  
// Using data types:  
Dim greeting As String = "Welcome to W3Schools"  
Dim counter As Integer = 103  
Dim today As DateTime = DateTime.Today

## Data Types

Below is a list of  common data types:

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Examples** |
| integer | Integer (whole numbers) | 103, 12, 5168 |
| double | 64 bit floating-point number | 3.14, 3.4e38 |
| decimal | Decimal number (higher precision) | 1037.196543 |
| boolean | Boolean | true, false |
| string | String | "Hello W3Schools", "John" |

## Operators

An operator tells ASP.NET what kind of command to perform in an expression.

 The VB language supports many operators. Below is a list of common operators:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Assigns a value to a variable. | i=6 |
| + - \* / | Adds a value or variable. Subtracts a value or variable. Multiplies a value or variable. Divides a value or variable. | i=5+5 i=5-5 i=5\*5 i=5/5 |
| += -= | Increments a variable. Decrements a variable. | i += 1 i -= 1 |
| = | Equality. Returns true if values are equal. | if i=10 |
| <> | Inequality. Returns true if values are not equal. | if <>10 |
| < > <= >= | Less than. Greater than. Less than or equal. Greater than or equal. | if i<10 if i>10 if i<=10 if i>=10 |
| & | Adding strings (concatenation). | "w3" & "schools" |
| . | Dot. Separate objects and methods. | DateTime.Hour |
| () | Parenthesis. Groups values. | (i+5) |
| () | Parenthesis. Passes parameters. | x=Add(i,5) |
| () | Parenthesis. Accesses values in arrays or collections. | name(3) |
| Not | Not. Reverses true or false. | if Not ready |
| And OR | Logical AND. Logical OR. | if ready And clear if ready Or clear |
| AndAlso orElse | Extended Logical AND. Extended Logical OR. | if ready AndAlso clear if ready OrElse clear |

## Converting Data Types

Converting from one data type to another is sometimes useful.  
  
The most common example is to convert string input to another type, such as an integer or a date.

As a rule, user input comes as strings, even if the user entered a number. Therefore, numeric input values must be converted to numbers before they can be used in calculations.

Below is a list of common conversion methods:

|  |  |  |
| --- | --- | --- |
| **Method** | **Decryptions** | **Example** |
| AsInt() IsInt() | Converts a string to an integer. | if myString.IsInt() then    myInt=myString.AsInt() end if |
| AsFloat() IsFloat() | Converts a string to a floating-point number. | if myString.IsFloat() then    myFloat=myString.AsFloat() end if |
| AsDecimal() IsDecimal() | Converts a string to a decimal number. | if myString.IsDecimal() then    myDec=myString.AsDecimal() end if |
| AsDateTime() IsDateTime() | Converts a string to an ASP.NET DateTime type. | myString="10/10/2012" myDate=myString.AsDateTime() |
| AsBool() IsBool() | Converts a string to a Boolean. | myString="True" myBool=myString.AsBool() |
| ToString() | Converts any data type to a string. | myInt=1234 myString=myInt.ToString() |

[« Previous](http://www.w3schools.com/aspnet/razor_cs_logic.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/razor_vb_loops.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Razor - VB Loops and Arrays

[« Previous](http://www.w3schools.com/aspnet/razor_vb_variables.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/razor_vb_logic.asp)

Statements can be executed repeatedly in loops.

## For Loops

If you need to run the same statements repeatedly, you can program a loop.

If you know how many times you want to loop, you can use a **for loop**. This kind of loop is especially useful for counting up or counting down:

## Example

<html>  
<body>  
@For i=10 To 21  
    @<p>Line #@i</p>  
Next i  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_005)

## For Each Loops

If you work with a collection or an array, you often use a **for each loop**.

A collection is a group of similar objects, and the for each loop lets you carry out a task on each item. The for each loop walks through a collection until it is finished.

The example below walks through the ASP.NET Request.ServerVariables collection.

## Example

<html>  
<body>  
<ul>  
@For Each x In Request.ServerVariables  
    @<li>@x</li>  
Next x  
</ul>  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_006)

## While Loops

The **while loop** is a general purpose loop.

A while loop begins with the while keyword, followed by parentheses, where you specify how long the loop continues, then a block to repeat.

While loops typically add to, or subtract from, a variable used for counting.

In the example below, the += operator adds 1 to the variable i, each time the loop runs.

## Example

<html>  
<body>  
@Code  
Dim i=0  
Do While i<5  
    i += 1  
    @<p>Line #@i</p>  
Loop  
End Code  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_007)

## Arrays

An array is useful when you want to store similar variables but don't want to create a separate variable for each of them:

## Example

@Code  
Dim members As String()={"Jani","Hege","Kai","Jim"}  
i=Array.IndexOf(members,"Kai")+1  
len=members.Length  
x=members(2-1)  
end Code  
<html>  
<body>  
<h3>Members</h3>  
@For Each person In members  
   @<p>@person</p>  
Next person  
<p>The number of names in Members are @len</p>  
<p>The person at position 2 is @x</p>  
<p>Kai is now in position @i</p>  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_008)

**ASP.NET Razor - VB Logic Conditions**

[« Previous](http://www.w3schools.com/aspnet/razor_vb_loops.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_intro.asp)

Programming Logic: Execute code based on conditions.

**The If Condition**

VB lets you execute code based on conditions.

To test a condition you use the **if statement**. The if statement returns true or false, based on your test:

* The if statement starts a code block
* The condition is written between if and then
* The code between if ... then and end if is executed if the test is true

**Example**

@Code  
Dim price=50  
End Code  
<html>  
<body>  
@If price>30 Then  
    @<p>The price is too high.</p>  
End If  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_010)

**The Else Condition**

An if statement can include an **else condition**.

The else condition defines the code to be executed if the condition is false.

**Example**

@Code  
Dim price=20  
End Code  
<html>  
<body>  
@if price>30 then  
    @<p>The price is too high.</p>  
Else  
    @<p>The price is OK.</p>  
End If   
</body>  
</htmlV>

[Run example »](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_011)

**Note:** In the example above, if the first condition is true, it will be executed. The else condition covers "everything else".

**The ElseIf Condition**

Multiple conditions can be tested with an **else if condition**:

**Example**

@Code  
Dim price=25  
End Code  
<html>  
<body>  
@If price>=30 Then  
    @<p>The price is high.</p>  
ElseIf price>20 And price<30   
    @<p>The price is OK.</p>  
Else  
    @<p>The price is low.</p>  
End If      
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_012)

In the example above, if the first condition is true, it will be executed.

If not, then if the next condition is true, this condition will be executed.

You can have any number of else if conditions.

If none of the if or else if conditions are true, the last else block (without a condition) covers "everything else".

**Select Conditions**

A **select block** can be used to test a number of individual conditions:

**Example**

@Code  
Dim weekday=DateTime.Now.DayOfWeek  
Dim day=weekday.ToString()  
Dim message=""  
End Code  
<html>  
<body>  
@Select Case day  
Case "Monday"  
    message="This is the first weekday."  
Case "Thursday"  
    message="Only one day before weekend."  
Case "Friday"  
    message="Tomorrow is weekend!"  
Case Else  
    message="Today is " & day  
End Select  
<p>@message</p>  
</body>  
</html>

[Run example »](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_013)

"Select Case" is followed by the test value (day). Each individual test condition has a case value, and any number of code lines. If the test value matches the case value, the code lines are executed.

A select block can have a default case (Case Else) for "everything else" that runs if none of the other cases are true

## ASP.NET MVC

**ASP.NET MVC Tutorial**

[« Previous](http://www.w3schools.com/aspnet/razor_vb_logic.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_app.asp)



ASP.NET is a development framework for building web pages and web sites with HTML, CSS, JavaScript and server scripting.

ASP.NET supports three different development models:  
Web Pages, MVC (Model View Controller), and Web Forms.

**THIS TUTORIAL COVERS MVC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Web Pages |  | MVC |  | Web Forms |

**The MVC Programming Model**

MVC is one of three ASP.NET programming models.

MVC is a framework for building web applications using a MVC (Model View Controller) design:

* The Model represents the application core (for instance a list of database records).
* The View displays the data (the database records).
* The Controller handles the input (to the database records).

The MVC model also provides full control over HTML, CSS, and JavaScript.

|  |  |
| --- | --- |
| MVC | The MVC model defines web  applications with 3 logic layers:    The business layer (Model logic)  The display layer (View logic)  The input control (Controller logic) |

**The Model** is the part of the application that handles the logic for the application data.  
Often model objects retrieve data (and store data) from a database.

**The View** is the parts of the application that handles the display of the data.  
Most often the views are created from the model data.

**The Controller** is the part of the application that handles user interaction.  
Typically controllers read data from a view, control user input, and send input data to the model.

The MVC separation helps you manage complex applications, because you can focus on one aspect a time. For example, you can focus on the view without depending on the business logic. It also makes it easier to test an application.

The MVC separation also simplifies group development. Different developers can work on the view, the controller logic, and the business logic in parallel.

**Web Forms vs MVC**

The MVC programming model is a lighter alternative to traditional ASP.NET (Web Forms). It is a lightweight, highly testable framework, integrated with all existing ASP.NET features, such as Master Pages, Security, and Authentication.

**Visual Studio Express 2012/2010**

Visual Studio Express is a free version of Microsoft Visual Studio.

Visual Studio Express is a development tool tailor made for MVC (and Web Forms).

Visual Studio Express contains:

* MVC and Web Forms
* Drag-and-drop web controls and web components
* A web server language (Razor using VB or C#)
* A web server (IIS Express)
* A database server (SQL Server Compact)
* A full web development framework (ASP.NET)

If you install Visual Studio Express, you will get more benefits from this tutorial.

If you want to install Visual Studio Express, click on one of these links:

[Visual Web Developer 2012](http://www.microsoft.com/web/handlers/webpi.ashx?command=getinstallerredirect&appid=VWDOrVs11AzurePack) (If you have Windows 7 or Windows 8)

[Visual Web Developer 2010](http://www.microsoft.com/web/gallery/install.aspx?appid=VWDorVS2010SP1Pack) (If you have Windows Vista or XP)

|  |  |
| --- | --- |
| **Note** | After you have installed Visual Studio Express the first time, it pays to run the installation one more time, to install fixes and service packs. Just click on the link once more. |

**ASP.NET MVC References**

At the end of this tutorial you will find a complete ASP.NET MVC reference.

[« Previous](http://www.w3schools.com/aspnet/razor_vb_logic.asp)

**ASP.NET MVC - Internet Application**

[« Previous](http://www.w3schools.com/aspnet/mvc_intro.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_folders.asp)

To learn ASP.NET MVC, we will Build an Internet Application

Part I: Creating the Application

**What We Will Build**

We will build an Internet application that supports adding, editing, deleting, and listing of information stored in a database.

**What We Will Do**

Visual Web Developer offers different templates for building web applications.

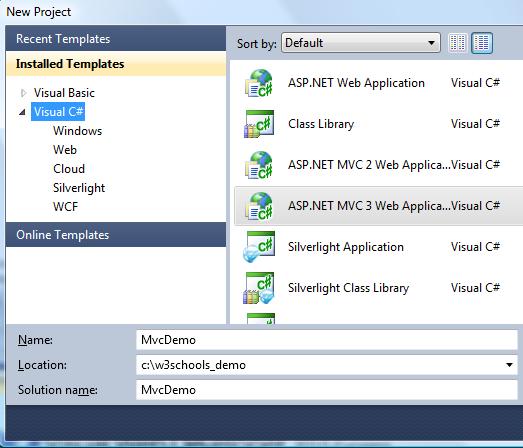
We will use **Visual Web Developer** to create an empty MVC Internet application with **HTML5 markup**.

When the empty Internet application is created, we will gradually add code to the application until it is fully finished. We will use **C#** as the programming language, and the newest **Razor** server code markup.

Along the way we will explain the content, the code, and all the components of the application.

**Creating the Web Application**

If you have Visual Web Developer installed, start Visual Web Developer and select **New Project**. Otherwise just read and learn.



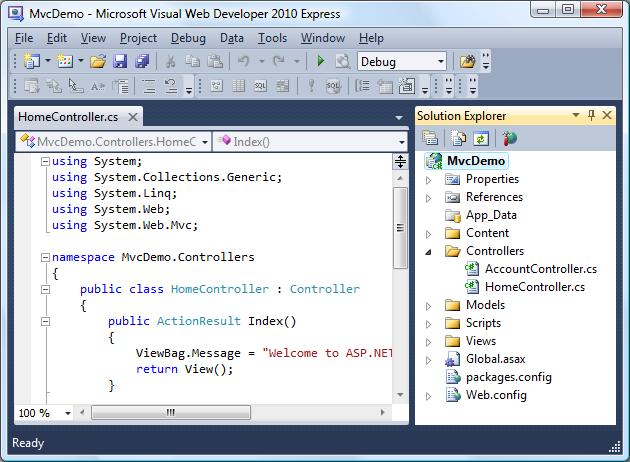
In the New Project dialog box:

* Open the **Visual C#** templates
* Select the template **ASP.NET MVC 3 Web Application**
* Set the project name to **MvcDemo**
* Set the disk location to something like **c:\w3schools\_demo**
* Click **OK**

When the New Project Dialog Box opens:

* Select the **Internet Application** template
* Select the **Razor Engine**
* Select **HTML5 Markup**
* Click **OK**

Visual Studio Express will create a project much like this:



We will explore the content of the files and folders in the next chapter of this tutorial.

# ASP.NET MVC - Application Folders

[« Previous](http://www.w3schools.com/aspnet/mvc_app.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_layout.asp)

To learn ASP.NET MVC, we are Building an Internet application

Part II: Exploring the Application Folders

## MVC Folders

A typical ASP.NET MVC web application has the following folder content:

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Solution |  | **Application information**  Properties References  **Application folders**  App\_Data Folder Content Folder Controllers Folder Models Folder Scripts Folder Views Folder  **Configuration files**  Global.asax packages.config Web.config | |

The folder names are equal in all MVC applications. The MVC framework is based on default naming. Controllers are in the Controllers folder, Views are in the Views folder, and Models are in the Models folder. You don't have to use the folder names in your application code.

Standard naming reduces the amount of code, and makes it easier for developers to understand MVC projects.

Below is a brief summary of the content of each folder:

## The App\_Data Folder

The **App\_Data** folder is for storing application data.

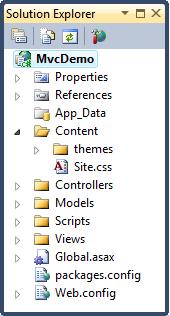
We will add an SQL database to the App\_Data folder, later in this tutorial.

## The Content Folder

The **Content** folder is used for static files like style sheets (css files), icons and images.

Visual Web Developer automatically adds a **themes** folder to the Content folder. The themes folder is filled with jQuery styles and pictures. In this project you can delete the themes folder.

Visual Web Developer also adds a standard style sheet file to the project: the file **Site.css** in the content folder. The style sheet file is the file to edit when you want to change the style of the application.



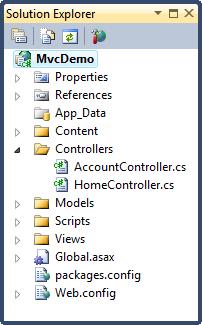
We will edit the style sheet file (Site.css) file in the next chapter of this tutorial.

## The Controllers Folder

The Controllers folder contains the controller classes responsible for handling user input and responses.

MVC requires the name of all controller files to end with "Controller".

Visual Web Developer has created a Home controller (for the Home and the About page) and an Account controller (for Login pages):



We will create more controllers later in this tutorial.

## The Models Folder

The Models folder contains the classes that represent the application models. Models hold and manipulate application data.

We will create models (classes) in a later chapter of this tutorial.

## The Views Folder

The Views folder stores the HTML files related to the display of the application (the user interfaces).

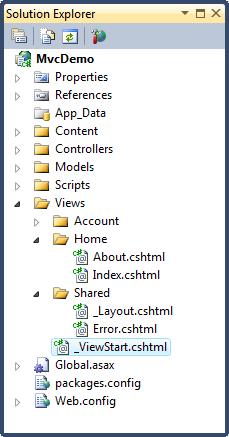
The Views folder contains one folder for each controller.

Visual Web Developer has created an Account folder, a Home folder, and a Shared folder (inside the Views folder).

The Account folder contains pages for registering and logging in to user accounts.

The Home folder is used for storing application pages like the home page and the about page.

The Shared folder is used to store views shared between controllers (master pages and layout pages).

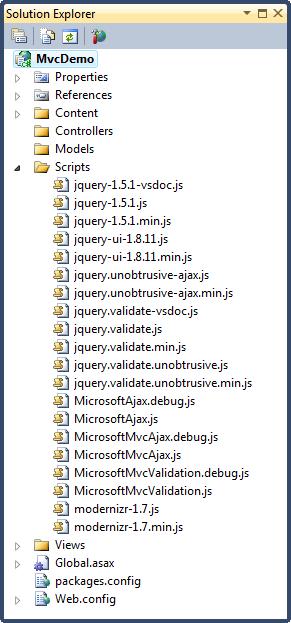


We will edit the layout files in the next chapter of this tutorial.

## The Scripts Folder

The Scripts folder stores the JavaScript files of the application.

By default Visual Web Developer fills this folder with standard MVC, Ajax, and jQuery files:



**Note:** The files named "modernizr" are JavaScript files used for supporting HTML5 and CSS3 features in the application.

[« Previous](http://www.w3schools.com/aspnet/mvc_app.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_layout.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET MVC - Styles and Layout

[« Previous](http://www.w3schools.com/aspnet/mvc_folders.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_controllers.asp)

To learn ASP.NET MVC, we are Building an Internet Application.

Part III: Adding Styles and a Consistent Look (Layout).

## Adding a Layout

The file \_Layout.cshtml represent the layout of each page in the application. It is located in the Shared folder inside the Views folder.

Open the file and swap the content with this:

<!DOCTYPE html>  
<html>  
<head>  
<meta charset="utf-8" />  
<title>@ViewBag.Title</title>  
<link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />  
<script src="@Url.Content("~/Scripts/jquery-1.5.1.min.js")"></script>  
<script src="@Url.Content("~/Scripts/modernizr-1.7.min.js")"></script>  
</head>  
<body>  
<ul id="menu">  
<li>@Html.ActionLink("Home", "Index", "Home")</li>  
<li>@Html.ActionLink("Movies", "Index", "Movies")</li>  
<li>@Html.ActionLink("About", "About", "Home")</li>  
</ul>   
<section id="main">  
@RenderBody()  
<p>Copyright W3schools 2012. All Rights Reserved.</p>  
</section>  
</body>  
</html>

## HTML Helpers

In the code above, HTML helpers are used to modify HTML output:

@Url.Content() - URL content will be inserted here.

@Html.ActionLink() - HTML link will be inserted here.

You will learn more about HTML helpers in a later chapter of this tutorial.

## Razor Syntax

In the code above, the code marked red are C# using Razor markup.

@ViewBag.Title - The page title will be inserted here.

@RenderBody() - The page content will be rendered here.

You can learn about Razor markup for both C# and VB (Visual Basic) in our [Razor tutorial](http://www.w3schools.com/aspnet/razor_intro.asp).

## Adding Styles

The style sheet for the application is called Site.css. It is located in the Content folder.

Open the file Site.css and swap the content with this:

body  
{  
font: "Trebuchet MS", Verdana, sans-serif;  
background-color: #5c87b2;  
color: #696969;  
}  
h1  
{  
border-bottom: 3px solid #cc9900;  
font: Georgia, serif;  
color: #996600;  
}  
#main  
{  
padding: 20px;  
background-color: #ffffff;  
border-radius: 0 4px 4px 4px;  
}  
a  
{  
color: #034af3;   
}  
/\* Menu Styles ------------------------------\*/   
ul#menu  
{  
padding: 0px;  
position: relative;  
margin: 0;  
}  
ul#menu li  
{  
display: inline;  
}  
ul#menu li a   
{  
background-color: #e8eef4;  
padding: 10px 20px;  
text-decoration: none;  
line-height: 2.8em;  
/\*CSS3 properties\*/  
border-radius: 4px 4px 0 0;  
}  
ul#menu li a:hover  
{  
background-color: #ffffff;  
}   
/\* Forms Styles ------------------------------\*/   
fieldset  
{  
padding-left: 12px;  
}   
fieldset label  
{  
display: block;  
padding: 4px;  
}  
input[type="text"], input[type="password"]  
{  
width: 300px;  
}  
input[type="submit"]  
{  
padding: 4px;  
}  
/\* Data Styles ------------------------------\*/   
table.data  
{  
background-color:#ffffff;  
border:1px solid #c3c3c3;  
border-collapse:collapse;  
width:100%;  
}  
table.data th  
{  
background-color:#e8eef4;  
border:1px solid #c3c3c3;  
padding:3px;  
}  
table.data td   
{  
border:1px solid #c3c3c3;  
padding:3px;  
}

## The \_ViewStart File

The \_ViewStart file in the Shared folder (inside the Views folder) contains the following content:

@{Layout = "~/Views/Shared/\_Layout.cshtml";}

This code is automatically added to all views displayed by the application.

If you remove this file, you must add this line to all views.

You will learn more about views in a later chapter of this tutorial.

[« Previous](http://www.w3schools.com/aspnet/mvc_folders.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_controllers.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET MVC - Controllers

[« Previous](http://www.w3schools.com/aspnet/mvc_layout.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_views.asp)

To learn ASP.NET MVC, we are Building an Internet Application.

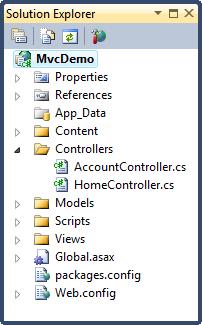
Part IV: Adding a Controller.

## The Controllers Folder

The **Controllers Folder** contains the controller classes responsible for handling user input and responses.

MVC requires the name of all controllers to end with "Controller".

In our example, Visual Web Developer has created the following files: **HomeController.cs** (for the Home and About pages) and **AccountController.cs** (For the Log On pages):



Web servers will normally map incoming URL requests directly to disk files on the server. For example: a URL request like "http://www.w3schools.com/default.asp" will map directly to the file "default.asp" at the root directory of the server.

The MVC framework maps differently. MVC maps URLs to methods. These methods are in classes called "Controllers".

Controllers are responsible for processing incoming requests, handling input, saving data, and sending a response to send back to the client.

## The Home controller

The controller file in our application **HomeController.cs**, defines the two controls **Index** and **About**.

Swap the content of the HomeController.cs file with this:

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.Mvc;  
  
namespace MvcDemo.Controllers  
{  
public class HomeController : Controller  
{  
public ActionResult Index()  
{return View();}  
  
public ActionResult About()  
{return View();}  
}  
}

## The Controller Views

The files **Index.cshtml** and **About.cshtml** in the Views folder defines the ActionResult views Index() and About() in the controller.

# ASP.NET MVC - Views

[« Previous](http://www.w3schools.com/aspnet/mvc_controllers.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_database.asp)

To learn ASP.NET MVC, we are Building an Internet Application.

Part V: Adding Views for Displaying the Application.

## The Views Folder

The **Views** folder stores the files (HTML files) related to the display of the application (the user interfaces). These files may have the extensions html, asp, aspx, cshtml, and vbhtml, depending on the language content.

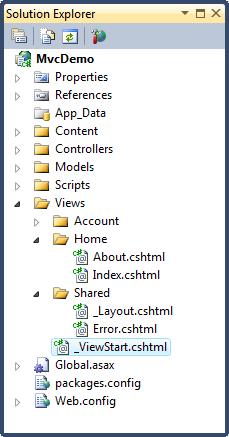
The Views folder contains one folder for each controller.

Visual Web Developer has created an Account folder, a Home folder, and a Shared folder (inside the Views folder).

The Account folder contains pages for registering and logging in to user accounts.

The Home folder is used for storing application pages like the home page and the about page.

The Shared folder is used to store views shared between controllers (master pages and layout pages).



## ASP.NET File Types

The following HTML file types can be found in the Views Folder:

|  |  |
| --- | --- |
| **File Type** | **Extention** |
| Plain HTML | .htm or .html |
| Classic ASP | .asp |
| Classic ASP.NET | .aspx |
| ASP.NET Razor C# | .cshtml |
| ASP.NET Razor VB | .vbhtml |

## The Index File

The file Index.cshtml represents the Home page of the application. It is the application's default file (index file).

Put the following content in the file:

@{ViewBag.Title = "Home Page";}  
  
<h1>Welcome to W3Schools</h1>  
  
<p>Put Home Page content here</p>

## The About File

The file About.cshtml represent the About page of the application.

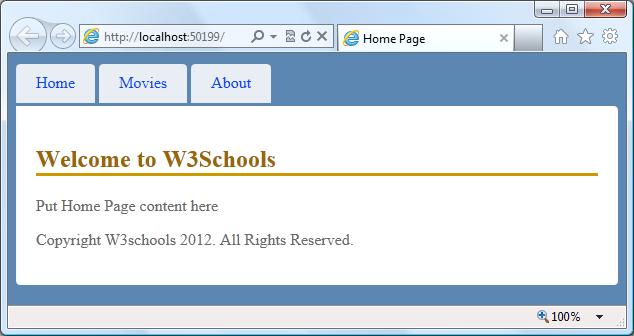
Put the following content in the file:

@{ViewBag.Title = "About Us";}  
  
<h1>About Us</h1>  
  
<p>Put About Us content here</p>

## Run the Application

Select Debug, Start Debugging (or F5) from the Visual Web Developer menu.

Your application will look like this:



Click on the "Home" tab and the "About" tab to see how it works.

## Congratulations

Congratulations. You have created your first MVC Application.

**Note:** You cannot click on the "Movies" tab yet. We will add code for the "Movies" tab in the next chapters of this tutorial.

[« Previous](http://www.w3schools.com/aspnet/mvc_controllers.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_database.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET MVC - SQL Database

[« Previous](http://www.w3schools.com/aspnet/mvc_views.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_models.asp)

To learn ASP.NET MVC, we are Building an Internet Application.

Part VI: Adding a Database.

## Creating the Database

Visual Web Developer comes with a free SQL database called SQL Server Compact.

The database needed for this tutorial can be created with these simple steps:

* Right-click the **App\_Data** folder in the **Solution Explorer** window
* Select **Add, New Item**
* Select **SQL Server Compact Local Database \***
* Name the database **Movies.sdf**.
* Click the **Add** button

**\*** If SQL Server Compact Local Database is not an option, you have not installed SQL Server Compact on your computer. Install it from this link: [SQL Server Compact](http://www.microsoft.com/web/gallery/install.aspx?appid=SQLCE;SQLCEVSTools_4_0)

Visual Web Developer automatically creates the database in the App\_Data folder.

**Note:** In this tutorial it is expected that you have some knowledge about SQL databases. If you want to study this topic first, please visit our [SQL Tutorial](http://www.w3schools.com/sql/default.asp).

## Adding a Database Table

Double-clicking the **Movies.sdf** file in the **App\_Data** folder will open a **Database Explorer** window.

To create a new table in the database, right-click the **Tables** folder, and select **Create Table**.

Create the following columns:

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Allow Nulls** |
| ID | int (primary key) | No |
| Title | nvarchar(100) | No |
| Director | nvarchar(100) | No |
| Date | datetime | No |

Columns explained:

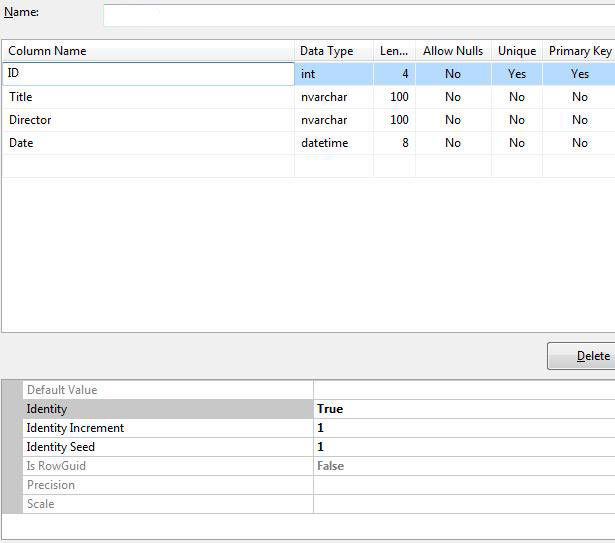
**ID** is an integer (whole number) used to identify each record in the table.

**Title** is a 100 character text column to store the name of the movie.

**Director** is a 100 character text column to store the director's name.

**Date** is a datetime column to store the release date of the movie.

After creating the columns described above, you must make the ID column the table's primary key (record identifier). To do this, click on the column name (ID) and select **Primary Key**. Also, in the **Column Properties** window, set the **Identity** property to **True**:



When you have finished creating the table columns, save the table and name it **MovieDBs**.

**Note:**

We have deliberately named the table "MovieDBs" (ending with s). In the next chapter, you will see the name "MovieDB" used for the data model. It looks strange, but this is the naming convention you have to use to make the controller connect to the database table.

## Adding Database Records

You can use Visual Web Developer to add some test records to the movie database.

Double-click the **Movies.sdf** file in the **App\_Data** folder.

Right-click the **MovieDBs** table in the Database Explorer window and select **Show Table Data**.

Add some records:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Title** | **Director** | **Date** |
| 1 | Psycho | Alfred Hitchcock | 01.01.1960 |
| 2 | La Dolce Vita | Federico Fellini | 01.01.1960 |

**Note:** The ID column is updated automatically. You should not edit it.

[« Previous](http://www.w3schools.com/aspnet/mvc_views.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_models.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET MVC - Models

[« Previous](http://www.w3schools.com/aspnet/mvc_database.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_security.asp)

To learn ASP.NET MVC, we are Building an Internet Application.

Part VII: Adding a Data Model.

## MVC Models

The MVC **Model** contains all application logic (business logic, validation logic, and data access logic), except pure view and controller logic.

With MVC, models both hold and manipulate application data.

## The Models Folder

The **Models Folder** contains the classes that represent the application model.

Visual Web Developer automatically creates an **AccountModels.cs** file that contains the models for application security.

**AccountModels** contains a **LogOnModel**, a **ChangePasswordModel**, and a **RegisterModel.**

## Adding a Database Model

The database model needed for this tutorial can be created with these simple steps:

* In the **Solution Explorer**, right-click the **Models** folder, and select **Add** and **Class**.
* Name the class **MovieDB.cs**, and click **Add**.
* Edit the class:

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Data.Entity;  
  
namespace MvcDemo.Models  
{  
public class MovieDB  
{  
public int ID { get; set; }  
public string Title { get; set; }  
public string Director { get; set; }  
public DateTime Date { get; set; }  
  
}  
public class MovieDBContext : DbContext  
{  
public DbSet<MovieDB> Movies { get; set; }   
}  
}

**Note:**

We have deliberately named the model class "MovieDB". In the previous chapter, you saw the name "MovieDBs" (ending with s) used for the database table. It looks strange, but this is the naming convention you have to use to make the model connect to the database table.

## Adding a Database Controller

The database controller needed for this tutorial can be created with these simple steps:

* Re-Build your project: Select **Debug**, and then **Build MvcDemo** from the menu.
* In the Solution Explorer, right-click the **Controllers** folder, and select **Add** and **Controller**
* Set controller name to **MoviesController**
* Select template: **Controller with read/write actions and views, using Entity Framework**
* Select model class: **MovieDB (MvcDemo.Models)**
* Select data context class: **MovieDBContext (MvcDemo.Models)**
* Select views **Razor (CSHTML)**
* Click **Add**

Visual Web Developer will create the following files:

* A **MoviesController.cs** file in the **Controllers** folder
* A **Movies** folder in the **Views** folder

## Adding Database Views

The following files are automatically created in the Movies folder:

* Create.cshtml
* Delete.cshtml
* Details.cshtml
* Edit.cshtml
* Index.cshtml

## Adding a Connection String

Add the following element to the **<connectionStrings>** element in your **Web.config** file:

<add name="MovieDBContext"  
connectionString="Data Source=|DataDirectory|\Movies.sdf"  
providerName="System.Data.SqlServerCe.4.0"/>

## Congratulations

Congratulations. You have added your first MVC data model to your application.

Now you can click on the "Movies" tab :-)

[« Previous](http://www.w3schools.com/aspnet/mvc_database.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_security.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET MVC - Security

[« Previous](http://www.w3schools.com/aspnet/mvc_models.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_htmlhelpers.asp)

To learn ASP.NET MVC, we are Building an Internet Application.

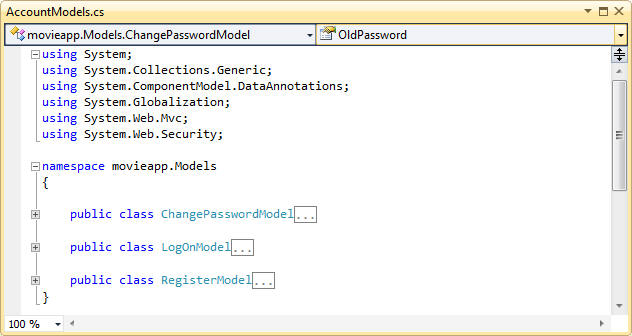
Part VIII: Adding Security.

## MVC Application Security

The **Models Folder** contains the classes that represent the application model.

Visual Web Developer automatically creates an **AccountModels.cs** file that contains the models for application authentication.

**AccountModels** contains a **LogOnModel**, a **ChangePasswordModel**, and a **RegisterModel**:



## The Change Password Model

public class ChangePasswordModel  
{  
  
[Required]  
[DataType(DataType.Password)]  
[Display(Name = "Current password")]  
public string OldPassword { get; set; }  
  
[Required]  
[StringLength(100, ErrorMessage = "The {0} must be at least {2}      characters long.", MinimumLength = 6)]  
[DataType(DataType.Password)]  
[Display(Name = "New password")]  
public string NewPassword { get; set; }  
  
[DataType(DataType.Password)]  
[Display(Name = "Confirm new password")]  
[Compare("NewPassword", ErrorMessage = "The new password and confirmation password do not match.")]  
public string ConfirmPassword { get; set; }  
  
}

## The Logon Model

public class LogOnModel  
{  
  
[Required]  
[Display(Name = "User name")]  
public string UserName { get; set; }  
  
[Required]  
[DataType(DataType.Password)]  
[Display(Name = "Password")]  
public string Password { get; set; }  
  
[Display(Name = "Remember me?")]  
public bool RememberMe { get; set; }  
  
}

## The Register Model

public class RegisterModel  
{  
  
[Required]  
[Display(Name = "User name")]  
public string UserName { get; set; }  
  
[Required]  
[DataType(DataType.EmailAddress)]  
[Display(Name = "Email address")]  
public string Email { get; set; }  
  
[Required]  
[StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]  
[DataType(DataType.Password)]  
[Display(Name = "Password")]  
public string Password { get; set; }  
  
[DataType(DataType.Password)]  
[Display(Name = "Confirm password")]  
[Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]  
public string ConfirmPassword { get; set; }  
  
}

[« Previous](http://www.w3schools.com/aspnet/mvc_models.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_htmlhelpers.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

**ASP.NET MVC - HTML Helpers**

[« Previous](http://www.w3schools.com/aspnet/mvc_security.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_publish.asp)

HTML Helpers are used to modify HTML output

**HTML Helpers**

With MVC, HTML helpers are much like traditional ASP.NET Web Form controls.

Just like web form controls in ASP.NET, HTML helpers are used to modify HTML. But HTML helpers are more lightweight. Unlike Web Form controls, an HTML helper does not have an event model and a view state.

In most cases, an HTML helper is just a method that returns a string.

With MVC, you can create your own helpers, or use the built in HTML helpers.

**Standard HTML Helpers**

MVC includes standard helpers for the most common types of HTML elements, like HTML links and HTML form elements.

**HTML Links**

The easiest way to render an HTML link in is to use the HTML.ActionLink() helper.

With MVC, the Html.ActionLink() does not link to a view. It creates a link to a controller action.

Razor Syntax:

@Html.ActionLink("About this Website", "About")

ASP Syntax:

<%=Html.ActionLink("About this Website", "About")%>

The first parameter is the link text, and the second parameter is the name of the controller action.

The Html.ActionLink() helper above, outputs the following HTML:

<a href="/Home/About">About this Website</a>

The Html.ActionLink() helper has several properties:

|  |  |
| --- | --- |
| **Property** | **Description** |
| .linkText | The link text (label) |
| .actionName | The target action |
| .routeValues | The values passed to the action |
| .controllerName | The target controller |
| .htmlAttributes | The set of attributes to the link |
| .protocol | The link protocol |
| .hostname | The host name for the link |
| .fragment | The anchor target for the link |

**Note:** You can pass values to a controller action. For example, you can pass the id of a database record to a database edit action:

Razor Syntax C#:

@Html.ActionLink("Edit Record", "Edit", new {Id=3})

Razor Syntax VB:

@Html.ActionLink("Edit Record", "Edit", New With{.Id=3})

The Html.ActionLink() helper above, outputs the following HTML:

<a href="/Home/Edit/3">Edit Record</a>

**HTML Form Elements**

There following HTML helpers can be used to render (modify and output) HTML form elements:

* BeginForm()
* EndForm()
* TextArea()
* TextBox()
* CheckBox()
* RadioButton()
* ListBox()
* DropDownList()
* Hidden()
* Password()

ASP.NET Syntax C#:

<%= Html.ValidationSummary("Create was unsuccessful. Please correct the errors and try again.") %>  
<% using (Html.BeginForm()){%>  
<p>  
<label for="FirstName">First Name:</label>  
<%= Html.TextBox("FirstName") %>  
<%= Html.ValidationMessage("FirstName", "\*") %>  
</p>  
<p>  
<label for="LastName">Last Name:</label>  
<%= Html.TextBox("LastName") %>  
<%= Html.ValidationMessage("LastName", "\*") %>  
</p>  
<p>  
<label for="Password">Password:</label>  
<%= Html.Password("Password") %>  
<%= Html.ValidationMessage("Password", "\*") %>  
</p>  
<p>  
<label for="Password">Confirm Password:</label>  
<%= Html.Password("ConfirmPassword") %>  
<%= Html.ValidationMessage("ConfirmPassword", "\*") %>  
</p>  
<p>  
<label for="Profile">Profile:</label>  
<%= Html.TextArea("Profile", new {cols=60, rows=10})%>  
</p>  
<p>  
<%= Html.CheckBox("ReceiveNewsletter") %>  
<label for="ReceiveNewsletter" style="display:inline">Receive Newsletter?</label>  
</p>  
<p>  
<input type="submit" value="Register" />  
</p>  
<%}%>

[« Previous](http://www.w3schools.com/aspnet/mvc_security.asp)

# ASP.NET Web Pages - WebMail Object

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_database.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_ref_helpers.asp)

With the WebMail object you can easily send emails from a web page.

## Description

The **WebMail Object** provides email for ASP.NET Web Pages using SMTP (Simple Mail Transfer Protocol).

## Example

See an example in the chapter: [Web Pages Email](http://www.w3schools.com/aspnet/webpages_email.asp).

## WebMail Object Reference - Properties

|  |  |
| --- | --- |
| **Properties** | **Description** |
| SmtpServer | The name the SMTP server that will send the emails |
| SmtpPort | The port the server will use to send SMTP emails |
| EnableSsl | True, if the server should use SSL encryption |
| UserName | The name of the SMTP account used to send the email |
| Password | The password of the SMTP account |
| From | The email to appear in the from address |

## WebMail Object Reference - Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| Send() | Sends an email message to an SMTP server for delivery |

The Send() method has the following parameters:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| to | String | The Email recipients (separated by semicolon) |
| subject | String | The subject line |
| body | String | The body of the message |

And the following optional parameters:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| from | String | The email of the sender |
| cc | String | The cc emails (separated by semicolon) |
| filesToAttach | Collection | Filenames |
| isBodyHtml | Boolean | True if the email body is in HTML |
| additionalHeaders | Collection | Additional headers |

## Technical Data

|  |  |
| --- | --- |
| **Name** | **Value** |
| Class | System.Web.Helpers.WebMail |
| Namespace | System.Web.Helpers |
| Assembly | System.Web.Helpers.dll |

## Initializing the WebMail Helper

To use the WebMail helper, you need access to an SMTP server. SMTP is the "output" part of email. If you use a web host, you probably already know the name of the SMTP server. If you work in a corporate network, your IT department can give you the name. If you are working at home, you might be able to use your ordinary email provider.

 In order to send an email you will need:

* The name of the SMTP server
* The port number (most often 25)
* An email user name
* An email password

In the root of your web, create a page (or edit the page ) named **\_AppStart.cshtml**.

Put the following code inside the file:

## \_AppStart.cshtml

@{  
WebMail.SmtpServer = "smtp.example.com";  
WebMail.SmtpPort = 25;  
WebMail.EnableSsl = false;  
WebMail.UserName = "support@example.com";  
WebMail.Password = "password";  
WebMail.From = "john@example.com"  
}

The code above will run each time the web site (application) starts. It feeds your **WebMail Object** with initial values.

Please substitute:

**smtp.example.com** with the name the SMTP server that will be used to send the emails.

**25** with the port number the server will use to send SMTP transactions (emails).

**false** with true, if the server should use SSL (Secure Socket Layer) encryption.

**support@example.com** with the name of the SMTP email account used to send emails.

**password** with the password of the SMTP email account.

**john@example** with the email to appear in the from address.

|  |  |
| --- | --- |
| **Note** | You don't **have to** initiate the WebMail object in your AppStart file, but you must set these properties before you call the **WebMail.Send()** method. |

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_database.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_ref_helpers.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET MVC - Publishing the Website

[« Previous](http://www.w3schools.com/aspnet/mvc_htmlhelpers.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_reference.asp)

Learn how to publish an MVC application without using Visual Web Developer.

## Publish Your Application Without Using Visual Web Developer

An ASP.NET MVC application can be published to a remote server by using the Publish commands in WebMatrix ,Visual Web Developer, or Visual Studio.

This function copies all your application files, controllers, models, images, and all the required DLL files for MVC, Web Pages, Razor, Helpers, and SQL Server Compact (if a database is used).

Sometimes you don't want to use this option. Maybe your hosting provider only supports FTP? Maybe you already have a web site based on classic ASP? Maybe you want to copy the files yourself? Maybe you want to use Front Page, Expression Web, or some other publishing software?

**Will you get a problem? Yes, you will. But you can solve it.**

To perform a web copy, you have to know how to include the right files, what DDL files to copy, and where store them.

Follow these steps:

## 1. Use the Latest Version of ASP.NET

Before you continue, make sure your hosting computer runs the latest version of ASP.NET (4.0).

## 2. Copy the Web Folders

Copy your website (all folders and content) from your development computer to an application folder on your remote hosting computer (server).

If your **App\_Data** folder contains test data, don't copy the App\_Data folder (see SQL Data below).

## 3. Copy the DLL Files

On the remote server create a bin folder in the root of your application. (If you have installed Helpers, you already have a bin folder)

Copy everything from your folders:

**C:\Program Files (x86)\Microsoft ASP.NET\ASP.NET Web Pages\v1.0\Assemblies**

**C:\Program Files (x86)\Microsoft ASP.NET\ASP.NET MVC 3\Assemblies**

to your application's bin folder on the remote server.

## 4. Copy the SQL Server Compact DLL Files

If your application has a SQL Server Compact database (an .sdf file in App\_Data folder), you must copy the SQL Server Compact DLL files:

Copy everything from your folder:

**C:\Program Files (x86)\Microsoft SQL Server Compact Edition\v4.0\Private**

to your application's bin folder on the remote server.

Create (or edit) the Web.config file for your application:

## Example C#

<?xml version="1.0" encoding="UTF-8"?>  
<configuration>  
<system.data>  
<DbProviderFactories>  
<remove invariant="System.Data.SqlServerCe.4.0" />  
  
<add invariant="System.Data.SqlServerCe.4.0"  
name="Microsoft SQL Server Compact 4.0"  
description=".NET Framework Data Provider for Microsoft SQL Server Compact" type="System.Data.SqlServerCe.SqlCeProviderFactory, System.Data.SqlServerCe, Version=4.0.0.1,Culture=neutral, PublicKeyToken=89845dcd8080cc91" />  
  
</DbProviderFactories>  
</system.data>  
</configuration>

## 5. Copy SQL Server Compact Data

Do you have .sdf files in your App\_Data folder that contains test data?

Do you want to publish the test data to the remote server?

Most likely not.

If you have to copy the SQL data files (.sdf files), you should delete everything in the database, and then copy the empty .sdf file from your development computer to the server.

**THAT'S IT. GOOD LUCK !**

[« Previous](http://www.w3schools.com/aspnet/mvc_htmlhelpers.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/mvc_reference.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET MVC - Reference

[« Previous](http://www.w3schools.com/aspnet/mvc_publish.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_intro.asp)

## Classes

|  |  |
| --- | --- |
| **Class** | **Description** |
| AcceptVerbsAttribute | Represents an attribute that specifies which HTTP verbs an action method will respond to. |
| ActionDescriptor | Provides information about an action method, such as its name, controller, parameters, attributes, and filters. |
| ActionExecutedContext | Provides the context for the ActionExecuted method of the ActionFilterAttribute class. |
| ActionExecutingContext | Provides the context for the ActionExecuting method of the ActionFilterAttribute class. |
| ActionFilterAttribute | Represents the base class for filter attributes. |
| ActionMethodSelectorAttribute | Represents an attribute that is used to influence the selection of an action method. |
| ActionNameAttribute | Represents an attribute that is used for the name of an action. |
| ActionNameSelectorAttribute | Represents an attribute that affects the selection of an action method. |
| ActionResult | Encapsulates the result of an action method and is used to perform a framework-level operation on behalf of the action method. |
| AdditionalMetadataAttribute | Provides a class that implements the IMetadataAware interface in order to support additional metadata. |
| AjaxHelper | Represents support for rendering HTML in AJAX scenarios within a view. |
| AjaxHelper(Of TModel) | Represents support for rendering HTML in AJAX scenarios within a strongly typed view. |
| AjaxRequestExtensions | Represents a class that extends the HttpRequestBase class by adding the ability to determine whether an HTTP request is an AJAX request. |
| AllowHtmlAttribute | Allows a request to include HTML markup during model binding by skipping request validation for the property. (It is strongly recommended that your application explicitly check all models where you disable request validation in order to prevent script exploits.) |
| AreaRegistration | Provides a way to register one or more areas in an ASP.NET MVC application. |
| AreaRegistrationContext | Encapsulates the information that is required in order to register an area within an ASP.NET MVC application. |
| AssociatedMetadataProvider | Provides an abstract class to implement a metadata provider. |
| AssociatedValidatorProvider | Provides an abstract class for classes that implement a validation provider. |
| AsyncController | Provides the base class for asynchronous controllers. |
| AsyncTimeoutAttribute | Represents an attribute that is used to set the timeout value, in milliseconds, for an asynchronous method. |
| AuthorizationContext | Encapsulates the information that is required for using an AuthorizeAttribute attribute. |
| AuthorizeAttribute | Represents an attribute that is used to restrict access by callers to an action method. |
| BindAttribute | Represents an attribute that is used to provide details about how model binding to a parameter should occur. |
| BuildManagerCompiledView | Represents the base class for views that are compiled by the BuildManager class before being rendered by a view engine. |
| BuildManagerViewEngine | Provides a base class for view engines. |
| ByteArrayModelBinder | Maps a browser request to a byte array. |
| ChildActionOnlyAttribute | Represents an attribute that is used to indicate that an action method should be called only as a child action. |
| ChildActionValueProvider | Represents a value provider for values from child actions. |
| ChildActionValueProviderFactory | Represents a factory for creating value provider objects for child actions. |
| ClientDataTypeModelValidatorProvider | Returns the client data-type model validators. |
| CompareAttribute | Provides an attribute that compares two properties of a model. |
| ContentResult | Represents a user-defined content type that is the result of an action method. |
| Controller | Provides methods that respond to HTTP requests that are made to an ASP.NET MVC Web site. |
| ControllerActionInvoker | Represents a class that is responsible for invoking the action methods of a controller. |
| ControllerBase | Represents the base class for all MVC controllers. |
| ControllerBuilder | Represents a class that is responsible for dynamically building a controller. |
| ControllerContext | Encapsulates information about an HTTP request that matches specified RouteBase and ControllerBase instances. |
| ControllerDescriptor | Encapsulates information that describes a controller, such as its name, type, and actions. |
| ControllerInstanceFilterProvider | Adds the controller to the FilterProviderCollection instance. |
| CustomModelBinderAttribute | Represents an attribute that invokes a custom model binder. |
| DataAnnotationsModelMetadata | Provides a container for common metadata, for the DataAnnotationsModelMetadataProvider class, and for the DataAnnotationsModelValidator class for a data model. |
| DataAnnotationsModelMetadataProvider | Implements the default model metadata provider for ASP.NET MVC. |
| DataAnnotationsModelValidator | Provides a model validator. |
| DataAnnotationsModelValidator(Of TAttribute) | Provides a model validator for a specified validation type. |
| [DataAnnotationsModelValidatorProvider](http://msdn.microsoft.com/en-us/library/system.web.mvc.dataannotationsmodelvalidatorprovider%28v=vs.98%29.aspx) | Implements the default validation provider for ASP.NET MVC. |
| [DataErrorInfoModelValidatorProvider](http://msdn.microsoft.com/en-us/library/system.web.mvc.dataerrorinfomodelvalidatorprovider%28v=vs.98%29.aspx) | Provides a container for the error-information model validator. |
| DefaultControllerFactory | Represents the controller factory that is registered by default. |
| DefaultModelBinder | Maps a browser request to a data object. This class provides a concrete implementation of a model binder. |
| DefaultViewLocationCache | Represents a memory cache for view locations. |
| DependencyResolver | Provides a registration point for dependency resolvers that implement IDependencyResolver or the Common Service Locator IServiceLocator interface. |
| DependencyResolverExtensions | Provides a type-safe implementation of GetService and GetServices. |
| DictionaryValueProvider(Of TValue) | Represents the base class for value providers whose values come from a collection that implements the IDictionary(Of TKey, TValue) interface. |
| EmptyModelMetadataProvider | Provides an empty metadata provider for data models that do not require metadata. |
| EmptyModelValidatorProvider | Provides an empty validation provider for models that do not require a validator. |
| EmptyResult | Represents a result that does nothing, such as a controller action method that returns nothing. |
| ExceptionContext | Provides the context for using the HandleErrorAttribute class. |
| ExpressionHelper | Provides a helper class to get the model name from an expression. |
| FieldValidationMetadata | Provides a container for client-side field validation metadata. |
| FileContentResult | Sends the contents of a binary file to the response. |
| FilePathResult | Sends the contents of a file to the response. |
| FileResult | Represents a base class that is used to send binary file content to the response. |
| FileStreamResult | Sends binary content to the response by using a Stream instance. |
| Filter | Represents a metadata class that contains a reference to the implementation of one or more of the filter interfaces, the filter's order, and the filter's scope. |
| FilterAttribute | Represents the base class for action and result filter attributes. |
| FilterAttributeFilterProvider | Defines a filter provider for filter attributes. |
| FilterInfo | Encapsulates information about the available action filters. |
| FilterProviderCollection | Represents the collection of filter providers for the application. |
| FilterProviders | Provides a registration point for filters. |
| FormCollection | Contains the form value providers for the application. |
| FormContext | Encapsulates information that is required in order to validate and process the input data from an HTML form. |
| FormValueProvider | Represents a value provider for form values that are contained in a NameValueCollection object. |
| FormValueProviderFactory | Represents a class that is responsible for creating a new instance of a form-value provider object. |
| GlobalFilterCollection | Represents a class that contains all the global filters. |
| GlobalFilters | Represents the global filter collection. |
| HandleErrorAttribute | Represents an attribute that is used to handle an exception that is thrown by an action method. |
| HandleErrorInfo | Encapsulates information for handling an error that was thrown by an action method. |
| HiddenInputAttribute | Represents an attribute that is used to indicate whether a property or field value should be rendered as a hidden input element. |
| HtmlHelper | Represents support for rendering HTML controls in a view. |
| HtmlHelper(Of TModel) | Represents support for rendering HTML controls in a strongly typed view. |
| HttpDeleteAttribute | Represents an attribute that is used to restrict an action method so that the method handles only HTTP DELETE requests. |
| HttpFileCollectionValueProvider | Represents a value provider to use with values that come from a collection of HTTP files. |
| HttpFileCollectionValueProviderFactory | Represents a class that is responsible for creating a new instance of an HTTP file collection value provider object. |
| HttpGetAttribute | Represents an attribute that is used to restrict an action method so that the method handles only HTTP GET requests. |
| HttpNotFoundResult | Defines an object that is used to indicate that the requested resource was not found. |
| HttpPostAttribute | Represents an attribute that is used to restrict an action method so that the method handles only HTTP POST requests. |
| HttpPostedFileBaseModelBinder | Binds a model to a posted file. |
| HttpPutAttribute | Represents an attribute that is used to restrict an action method so that the method handles only HTTP PUT requests. |
| HttpRequestExtensions | Extends the HttpRequestBase class that contains the HTTP values that were sent by a client during a Web request. |
| HttpStatusCodeResult | Provides a way to return an action result with a specific HTTP response status code and description. |
| HttpUnauthorizedResult | Represents the result of an unauthorized HTTP request. |
| JavaScriptResult | Sends JavaScript content to the response. |
| JsonResult | Represents a class that is used to send JSON-formatted content to the response. |
| JsonValueProviderFactory | Enables action methods to send and receive JSON-formatted text and to model-bind the JSON text to parameters of action methods. |
| LinqBinaryModelBinder | Maps a browser request to a LINQ Binary object. |
| ModelBinderAttribute | Represents an attribute that is used to associate a model type to a model-builder type. |
| ModelBinderDictionary | Represents a class that contains all model binders for the application, listed by binder type. |
| ModelBinderProviderCollection | Provides a container for model binder providers. |
| ModelBinderProviders | Provides a container for model binder providers. |
| ModelBinders | Provides global access to the model binders for the application. |
| ModelBindingContext | Provides the context in which a model binder functions. |
| ModelClientValidationEqualToRule | Provides a container for an equality validation rule that is sent to the browser. |
| ModelClientValidationRangeRule | Provides a container for a range-validation rule that is sent to the browser. |
| ModelClientValidationRegexRule | Provides a container for a regular-expression client validation rule that is sent to the browser. |
| ModelClientValidationRemoteRule | Provides a container for a remote validation rule that is sent to the browser. |
| ModelClientValidationRequiredRule | Provides a container for client validation for required field. |
| ModelClientValidationRule | Provides a base class container for a client validation rule that is sent to the browser. |
| ModelClientValidationStringLengthRule | Provides a container for a string-length validation rule that is sent to the browser. |
| ModelError | Represents an error that occurs during model binding. |
| ModelErrorCollection | A collection of ModelError instances. |
| ModelMetadata | Provides a container for common metadata, for the ModelMetadataProvider class, and for the ModelValidator class for a data model. |
| ModelMetadataProvider | Provides an abstract base class for a custom metadata provider. |
| ModelMetadataProviders | Provides a container for the current ModelMetadataProvider instance. |
| ModelState | Encapsulates the state of model binding to a property of an action-method argument, or to the argument itself. |
| ModelStateDictionary | Represents the state of an attempt to bind a posted form to an action method, which includes validation information. |
| ModelValidationResult | Provides a container for a validation result. |
| ModelValidator | Provides a base class for implementing validation logic. |
| ModelValidatorProvider | Provides a list of validators for a model. |
| ModelValidatorProviderCollection | Provides a container for a list of validation providers. |
| ModelValidatorProviders | Provides a container for the current validation provider. |
| MultiSelectList | Represents a list of items that users can select more than one item from. |
| MvcFilter | When implemented in a derived class, provides a metadata class that contains a reference to the implementation of one or more of the filter interfaces, the filter's order, and the filter's scope. |
| MvcHandler | Selects the controller that will handle an HTTP request. |
| MvcHtmlString | Represents an HTML-encoded string that should not be encoded again. |
| MvcHttpHandler | Verifies and processes an HTTP request. |
| MvcRouteHandler | Creates an object that implements the IHttpHandler interface and passes the request context to it. |
| MvcWebRazorHostFactory | Creates instances of MvcWebPageRazorHost files. |
| NameValueCollectionExtensions | Extends a NameValueCollection object so that the collection can be copied to a specified dictionary. |
| NameValueCollectionValueProvider | Represents the base class for value providers whose values come from a NameValueCollection object. |
| NoAsyncTimeoutAttribute | Provides a convenience wrapper for the AsyncTimeoutAttribute attribute. |
| NonActionAttribute | Represents an attribute that is used to indicate that a controller method is not an action method. |
| OutputCacheAttribute | Represents an attribute that is used to mark an action method whose output will be cached. |
| ParameterBindingInfo | Encapsulates information for binding action-method parameters to a data model. |
| ParameterDescriptor | Contains information that describes a parameter. |
| PartialViewResult | Represents a base class that is used to send a partial view to the response. |
| PreApplicationStartCode | Provides a registration point for ASP.NET Razor pre-application start code. |
| QueryStringValueProvider | Represents a value provider for query strings that are contained in a NameValueCollection object. |
| QueryStringValueProviderFactory | Represents a class that is responsible for creating a new instance of a query-string value-provider object. |
| RangeAttributeAdapter | Provides an adapter for the RangeAttribute attribute. |
| RazorView | Represents the class used to create views that have Razor syntax. |
| RazorViewEngine | Represents a view engine that is used to render a Web page that uses the ASP.NET Razor syntax. |
| RedirectResult | Controls the processing of application actions by redirecting to a specified URI. |
| RedirectToRouteResult | Represents a result that performs a redirection by using the specified route values dictionary. |
| ReflectedActionDescriptor | Contains information that describes a reflected action method. |
| ReflectedControllerDescriptor | Contains information that describes a reflected controller. |
| ReflectedParameterDescriptor | Contains information that describes a reflected action-method parameter. |
| RegularExpressionAttributeAdapter | Provides an adapter for the RegularExpressionAttribute attribute. |
| RemoteAttribute | Provides an attribute that uses the jQuery validation plug-in remote validator. |
| RequiredAttributeAdapter | Provides an adapter for the RequiredAttributeAttribute attribute. |
| RequireHttpsAttribute | Represents an attribute that forces an unsecured HTTP request to be re-sent over HTTPS. |
| ResultExecutedContext | Provides the context for the OnResultExecuted method of the ActionFilterAttribute class. |
| ResultExecutingContext | Provides the context for the OnResultExecuting method of the ActionFilterAttribute class. |
| RouteCollectionExtensions | Extends a RouteCollection object for MVC routing. |
| RouteDataValueProvider | Represents a value provider for route data that is contained in an object that implements the IDictionary(Of TKey, TValue) interface. |
| RouteDataValueProviderFactory | Represents a factory for creating route-data value provider objects. |
| SelectList | Represents a list that lets users select one item. |
| SelectListItem | Represents the selected item in an instance of the SelectList class. |
| SessionStateAttribute | Specifies the session state of the controller. |
| SessionStateTempDataProvider | Provides session-state data to the current TempDataDictionary object. |
| StringLengthAttributeAdapter | Provides an adapter for the StringLengthAttribute attribute. |
| TempDataDictionary | Represents a set of data that persists only from one request to the next. |
| TemplateInfo | Encapsulates information about the current template context. |
| UrlHelper | Contains methods to build URLs for ASP.NET MVC within an application. |
| UrlParameter | Represents an optional parameter that is used by the MvcHandler class during routing. |
| ValidatableObjectAdapter | Provides an object adapter that can be validated. |
| ValidateAntiForgeryTokenAttribute | Represents an attribute that is used to prevent forgery of a request. |
| ValidateInputAttribute | Represents an attribute that is used to mark action methods whose input must be validated. |
| ValueProviderCollection | Represents the collection of value-provider objects for the application. |
| ValueProviderDictionary | **Obsolete.** Represents a dictionary of value providers for the application. |
| ValueProviderFactories | Represents a container for value-provider factory objects. |
| ValueProviderFactory | Represents a factory for creating value-provider objects. |
| ValueProviderFactoryCollection | Represents the collection of value-provider factories for the application. |
| ValueProviderResult | Represents the result of binding a value (such as from a form post or query string) to an action-method argument property, or to the argument itself. |
| ViewContext | Encapsulates information that is related to rendering a view. |
| ViewDataDictionary | Represents a container that is used to pass data between a controller and a view. |
| ViewDataDictionary(Of TModel) | Represents a container that is used to pass strongly typed data between a controller and a view. |
| ViewDataInfo | Encapsulates information about the current template content that is used to develop templates and about HTML helpers that interact with templates. |
| ViewEngineCollection | Represents a collection of view engines that are available to the application. |
| ViewEngineResult | Represents the result of locating a view engine. |
| ViewEngines | Represents a collection of view engines that are available to the application. |
| ViewMasterPage | Represents the information that is needed to build a master view page. |
| ViewMasterPage(Of TModel) | Represents the information that is required in order to build a strongly typed master view page. |
| ViewPage | Represents the properties and methods that are needed to render a view as a Web Forms page. |
| ViewPage(Of TModel) | Represents the information that is required in order to render a strongly typed view as a Web Forms page. |
| ViewResult | Represents a class that is used to render a view by using an IView instance that is returned by an IViewEngine object. |
| ViewResultBase | Represents a base class that is used to provide the model to the view and then render the view to the response. |
| ViewStartPage | Provides an abstract class that can be used to implement a view start (master) page. |
| ViewTemplateUserControl | Provides a container for TemplateInfo objects. |
| ViewTemplateUserControl(Of TModel) | Provides a container for TemplateInfo objects. |
| ViewType | Represents the type of a view. |
| ViewUserControl | Represents the information that is needed to build a user control. |
| ViewUserControl(Of TModel) | Represents the information that is required in order to build a strongly typed user control. |
| VirtualPathProviderViewEngine | Represents an abstract base-class implementation of the IViewEngine interface. |
| WebFormView | Represents the information that is needed to build a Web Forms page in ASP.NET MVC. |
| WebFormViewEngine | Represents a view engine that is used to render a Web Forms page to the response. |
| WebViewPage | Represents the properties and methods that are needed in order to render a view that uses ASP.NET Razor syntax. |
| WebViewPage(Of TModel) | Represents the properties and methods that are needed in order to render a view that uses ASP.NET Razor syntax. |

## Interfaces

|  |  |
| --- | --- |
| **Interface** | **Description** |
| IActionFilter | Defines the methods that are used in an action filter. |
| IActionInvoker | Defines the contract for an action invoker, which is used to invoke an action in response to an HTTP request. |
| IAuthorizationFilter | Defines the methods that are required for an authorization filter. |
| IClientValidatable | Provides a way for the ASP.NET MVC validation framework to discover at run time whether a validator has support for client validation. |
| IController | Defines the methods that are required for a controller. |
| IControllerActivator | Provides fine-grained control over how controllers are instantiated using dependency injection. |
| IControllerFactory | Defines the methods that are required for a controller factory. |
| IDependencyResolver | Defines the methods that simplify service location and dependency resolution. |
| IExceptionFilter | Defines the methods that are required for an exception filter. |
| IFilterProvider | Provides an interface for finding filters. |
| IMetadataAware | Provides an interface for exposing attributes to the AssociatedMetadataProvider class. |
| IModelBinder | Defines the methods that are required for a model binder. |
| IModelBinderProvider | Defines methods that enable dynamic implementations of model binding for classes that implement the IModelBinder interface. |
| IMvcFilter | Defines members that specify the order of filters and whether multiple filters are allowed. |
| IResultFilter | Defines the methods that are required for a result filter. |
| IRouteWithArea | Associates a route with an area in an ASP.NET MVC application. |
| ITempDataProvider | Defines the contract for temporary-data providers that store data that is viewed on the next request. |
| IUnvalidatedValueProvider | Represents an IValueProvider interface that can skip request validation. |
| IValueProvider | Defines the methods that are required for a value provider in ASP.NET MVC. |
| IView | Defines the methods that are required for a view. |
| IViewDataContainer | Defines the methods that are required for a view data dictionary. |
| IViewEngine | Defines the methods that are required for a view engine. |
| IViewLocationCache | Defines the methods that are required in order to cache view locations in memory. |
| IViewPageActivator | Provides fine-grained control |

[« Previous](http://www.w3schools.com/aspnet/mvc_publish.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_intro.asp)

### [WEB HOSTING](http://www.w3schools.com/aspnet/aspnet_intro.asp)

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

## WF Tutorials

# ASP.NET Web Pages - Objects

[« Previous](http://www.w3schools.com/aspnet/webpages_forms.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_files.asp)

 Web Pages is much often about Objects.

## The Page Object

You have already seen some Page Object methods in use:

@RenderPage("header.cshtml")  
  
@RenderBody()

In the previous chapter you saw two Page Object properties being used (IsPost, and Request):

If (IsPost) {  
  
if (Request["Choice"] != null) {

## Some Page Object Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| href | Builds a URL using the specified parameters |
| RenderBody() | Renders the portion of a content page that is not within a named section (In layout pages) |
| RenderPage(page) | Renders the content of one page within another page |
| RenderSection(section) | Renders the content of a named section (In layout pages) |
| Write(object) | Writes the object as an HTML-encoded string |
| WriteLiteral | Writes an object without HTML-encoding it first. |

## Some Page Object Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| IsPost | Returns true if the HTTP data transfer method used by the client is a POST request |
| Layout | Gets or sets the path of a layout page |
| Page | Provides property-like access to data shared between pages and layout pages |
| Request | Gets the HttpRequest object for the current HTTP request |
| Server | Gets the HttpServerUtility object that provides web-page processing methods |

## The Page Property (of the Page Object)

The Page property of the Page Object, provides property-like access to data shared between pages and layout pages.

You can use (add) your own properties to the Page property:

* Page.Title
* Page.Version
* Page.anythingyoulike

The pages property is very helpful. For instance, it makes it possible to set the page title in content files, and use it in the layout file:

## Home.cshtml

@{  
Layout="~/Shared/Layout.cshtml";  
Page.Title="Home Page"  
}  
  
<h1>Welcome to W3Schools</h1>   
  
<h2>Web Site Main Ingredients</h2>  
  
<p>A Home Page (Default.cshtml)</p>  
<p>A Layout File (Layout.cshtml)</p>  
<p>A Style Sheet (Site.css)</p>

## Layout.cshtml

<!DOCTYPE html>  
<html>  
<head>  
<title>@Page.Title</title>  
</head>  
<body>  
@RenderBody()  
</body>  
</html>

[« Previous](http://www.w3schools.com/aspnet/webpages_forms.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_files.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Forms - HTML Pages

[« Previous](http://www.w3schools.com/aspnet/aspnet_intro.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_controls.asp)

A simple ASP.NET page looks just like an ordinary HTML page.

## Hello W3Schools

To start learning ASP.NET, we will construct a very simple HTML page that will display "Hello W3Schools" in an Internet browser like this:

|  |
| --- |
| Hello W3Schools! |

## Hello W3Schools in HTML

This code displays the example as an HTML page:

<html>  
<body bgcolor="yellow">  
<center>  
<h2>Hello W3Schools!</h2>  
</center>  
</body>  
</html>

If you want to try it yourself, save the code in a file called "**firstpage.htm**", and create a link to the file like this: [firstpage.htm](http://www.w3schools.com/aspnet/firstpage.htm)

## Hello W3Schools in ASP.NET

The simplest way to convert an HTML page into an ASP.NET page is to copy the HTML file to a new file with an **.aspx** extension.

This code displays our example as an ASP.NET page:

<html>  
<body bgcolor="yellow">  
<center>  
<h2>Hello W3Schools!</h2>  
</center>  
</body>  
</html>

If you want to try it yourself, save the code in a file called "**firstpage.aspx**", and create a link to the file like this: [firstpage.aspx](http://www.w3schools.com/aspnet/firstpage.aspx)

## How Does it Work?

Fundamentally an ASP.NET page is just the same as an HTML page.

An HTML page has the extension .htm. If a browser requests an HTML page from the server, the server sends the page to the browser without any modifications.

An ASP.NET page has the extension .aspx. If a browser requests an ASP.NET page, the server processes any executable code in the page, before the result is sent back to the browser.

The ASP.NET page above does not contain any executable code, so nothing is executed. In the next examples we will add some executable code to the page to demonstrate the difference between static HTML pages and dynamic ASP pages.

## Classic ASP

Active Server Pages (ASP) has been around for several years. With ASP, executable code can be placed inside HTML pages.

Previous versions of ASP (before ASP .NET) are often called Classic ASP.

ASP.NET is not fully compatible with Classic ASP, but most Classic ASP pages will work fine as ASP.NET pages, with only minor changes.

If you want to learn more about Classic ASP, please visit our [ASP Tutorial](http://www.w3schools.com/asp/default.asp).

## Dynamic Page in Classic ASP

To demonstrate how ASP can display pages with dynamic content, we have added some executable code (in red) to the previous example:

<html>  
<body bgcolor="yellow">  
<center>  
<h2>Hello W3Schools!</h2>  
<p><%Response.Write(Now())%></p>  
</center>  
</body>  
</html>

The code inside the <% --%> tags is executed on the server.

Response.Write is ASP code for writing something to the HTML output stream.

Now() is a function returning the servers current date and time.

If you want to try it yourself, save the code in a file called "**dynpage.asp**", and create a link to the file like this: [dynpage.asp](http://www.w3schools.com/aspnet/dynpage.asp)

## Dynamic Page in ASP .NET

This following code displays our example as an ASP.NET page:

<html>  
<body bgcolor="yellow">  
<center>  
<h2>Hello W3Schools!</h2>  
<p><%Response.Write(Now())%></p>  
</center>  
</body>  
</html>

If you want to try it yourself, save the code in a file called "**dynpage.aspx**", and create a link to the file like this: [dynpage.aspx](http://www.w3schools.com/aspnet/dynpage.aspx)

## ASP.NET vs Classic ASP

The previous examples didn't demonstrate any differences between ASP.NET and Classic ASP.

As you can see from the two latest examples there are no differences between the two ASP and ASP.NET pages.

In the next chapters you will see how server controls make ASP.NET more powerful than Classic ASP.

[« Previous](http://www.w3schools.com/aspnet/aspnet_intro.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_controls.asp)

# ASP.NET Web Forms - HTML Pages

[« Previous](http://www.w3schools.com/aspnet/aspnet_intro.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_controls.asp)

A simple ASP.NET page looks just like an ordinary HTML page.

## Hello W3Schools

To start learning ASP.NET, we will construct a very simple HTML page that will display "Hello W3Schools" in an Internet browser like this:

|  |
| --- |
| Hello W3Schools! |

## Hello W3Schools in HTML

This code displays the example as an HTML page:

<html>  
<body bgcolor="yellow">  
<center>  
<h2>Hello W3Schools!</h2>  
</center>  
</body>  
</html>

If you want to try it yourself, save the code in a file called "**firstpage.htm**", and create a link to the file like this: [firstpage.htm](http://www.w3schools.com/aspnet/firstpage.htm)

## Hello W3Schools in ASP.NET

The simplest way to convert an HTML page into an ASP.NET page is to copy the HTML file to a new file with an **.aspx** extension.

This code displays our example as an ASP.NET page:

<html>  
<body bgcolor="yellow">  
<center>  
<h2>Hello W3Schools!</h2>  
</center>  
</body>  
</html>

If you want to try it yourself, save the code in a file called "**firstpage.aspx**", and create a link to the file like this: [firstpage.aspx](http://www.w3schools.com/aspnet/firstpage.aspx)

## How Does it Work?

Fundamentally an ASP.NET page is just the same as an HTML page.

An HTML page has the extension .htm. If a browser requests an HTML page from the server, the server sends the page to the browser without any modifications.

An ASP.NET page has the extension .aspx. If a browser requests an ASP.NET page, the server processes any executable code in the page, before the result is sent back to the browser.

The ASP.NET page above does not contain any executable code, so nothing is executed. In the next examples we will add some executable code to the page to demonstrate the difference between static HTML pages and dynamic ASP pages.

## Classic ASP

Active Server Pages (ASP) has been around for several years. With ASP, executable code can be placed inside HTML pages.

Previous versions of ASP (before ASP .NET) are often called Classic ASP.

ASP.NET is not fully compatible with Classic ASP, but most Classic ASP pages will work fine as ASP.NET pages, with only minor changes.

If you want to learn more about Classic ASP, please visit our [ASP Tutorial](http://www.w3schools.com/asp/default.asp).

## Dynamic Page in Classic ASP

To demonstrate how ASP can display pages with dynamic content, we have added some executable code (in red) to the previous example:

<html>  
<body bgcolor="yellow">  
<center>  
<h2>Hello W3Schools!</h2>  
<p><%Response.Write(Now())%></p>  
</center>  
</body>  
</html>

The code inside the <% --%> tags is executed on the server.

Response.Write is ASP code for writing something to the HTML output stream.

Now() is a function returning the servers current date and time.

If you want to try it yourself, save the code in a file called "**dynpage.asp**", and create a link to the file like this: [dynpage.asp](http://www.w3schools.com/aspnet/dynpage.asp)

## Dynamic Page in ASP .NET

This following code displays our example as an ASP.NET page:

<html>  
<body bgcolor="yellow">  
<center>  
<h2>Hello W3Schools!</h2>  
<p><%Response.Write(Now())%></p>  
</center>  
</body>  
</html>

If you want to try it yourself, save the code in a file called "**dynpage.aspx**", and create a link to the file like this: [dynpage.aspx](http://www.w3schools.com/aspnet/dynpage.aspx)

## ASP.NET vs Classic ASP

The previous examples didn't demonstrate any differences between ASP.NET and Classic ASP.

As you can see from the two latest examples there are no differences between the two ASP and ASP.NET pages.

In the next chapters you will see how server controls make ASP.NET more powerful than Classic ASP.

[« Previous](http://www.w3schools.com/aspnet/aspnet_intro.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_controls.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Forms - Server Controls

[« Previous](http://www.w3schools.com/aspnet/aspnet_pages.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_events.asp)

Server controls are tags that are understood by the server.

## Limitations in Classic ASP

The listing below was copied from the previous chapter:

<html>  
<body bgcolor="yellow">  
<center>  
<h2>Hello W3Schools!</h2>  
<p><%Response.Write(now())%></p>  
</center>  
</body>  
</html>

The code above illustrates a limitation in Classic ASP: The code block has to be placed where you want the output to appear.

With Classic ASP it is impossible to separate executable code from the HTML itself. This makes the page difficult to read, and difficult to maintain.

## ASP.NET - Server Controls

ASP.NET has solved the "spaghetti-code" problem described above with server controls.

Server controls are tags that are understood by the server.

There are three kinds of server controls:

* HTML Server Controls - Traditional HTML tags
* Web Server Controls - New ASP.NET tags
* Validation Server Controls - For input validation

## ASP.NET - HTML Server Controls

HTML server controls are HTML tags understood by the server.

HTML elements in ASP.NET files are, by default, treated as text. To make these elements programmable, add a runat="server" attribute to the HTML element. This attribute indicates that the element should be treated as a server control. The id attribute is added to identify the server control. The id reference can be used to manipulate the server control at run time.

**Note:** All HTML server controls must be within a <form> tag with the runat="server" attribute. The runat="server" attribute indicates that the form should be processed on the server. It also indicates that the enclosed controls can be accessed by server scripts.

In the following example we declare an HtmlAnchor server control in an .aspx file. Then we manipulate the HRef attribute of the HtmlAnchor control in an event handler (an event handler is a subroutine that executes code for a given event). The Page\_Load event is one of many events that ASP.NET understands:

<script runat="server">  
Sub Page\_Load  
link1.HRef="http://www.w3schools.com"  
End Sub  
</script>  
  
<html>  
<body>  
  
<form runat="server">  
<a id="link1" runat="server">Visit W3Schools!</a>  
</form>  
  
</body>  
</html>

The executable code itself has been moved outside the HTML.

## ASP.NET - Web Server Controls

Web server controls are special ASP.NET tags understood by the server.

Like HTML server controls, Web server controls are also created on the server and they require a runat="server" attribute to work. However, Web server controls do not necessarily map to any existing HTML elements and they may represent more complex elements.

The syntax for creating a Web server control is:

<asp:control\_name id="some\_id" runat="server" />

In the following example we declare a Button server control in an .aspx file. Then we create an event handler for the Click event which changes the text on the button:

<script runat="server">  
Sub submit(Source As Object, e As EventArgs)  
button1.Text="You clicked me!"  
End Sub  
</script>  
  
<html>  
<body>  
  
<form runat="server">  
<asp:Button id="button1" Text="Click me!"  
runat="server" OnClick="submit"/>  
</form>  
  
</body>  
</html>

## ASP.NET - Validation Server Controls

Validation server controls are used to validate user-input. If the user-input does not pass validation, it will display an error message to the user.

Each validation control performs a specific type of validation (like validating against a specific value or a range of values).

By default, page validation is performed when a Button, ImageButton, or LinkButton control is clicked. You can prevent validation when a button control is clicked by setting the CausesValidation property to false.

The syntax for creating a Validation server control is:

<asp:control\_name id="some\_id" runat="server" />

In the following example we declare one TextBox control, one Button control, and one RangeValidator control in an .aspx file. If validation fails, the text "The value must be from 1 to 100!" will be displayed in the RangeValidator control:

## Example

<html>  
<body>  
  
<form runat="server">  
<p>Enter a number from 1 to 100:  
<asp:TextBox id="tbox1" runat="server" />  
<br /><br />  
<asp:Button Text="Submit" runat="server" />  
</p>  
  
<p>  
<asp:RangeValidator  
ControlToValidate="tbox1"  
MinimumValue="1"  
MaximumValue="100"  
Type="Integer"  
Text="The value must be from 1 to 100!"  
runat="server" />  
</p>  
</form>  
  
</body>  
</html>

[Show example »](http://www.w3schools.com/aspnet/valipage.aspx)

[« Previous](http://www.w3schools.com/aspnet/aspnet_pages.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_events.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Forms - Events

[« Previous](http://www.w3schools.com/aspnet/aspnet_controls.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_forms.asp)

An Event Handler is a subroutine that executes code for a given event.

## ASP.NET - Event Handlers

Look at the following code:

<%  
lbl1.Text="The date and time is " & now()  
%>  
  
<html>  
<body>  
<form runat="server">  
<h3><asp:label id="lbl1" runat="server" /></h3>  
</form>  
</body>  
</html>

When will the code above be executed? The answer is: "You don't know..."

## The Page\_Load Event

The Page\_Load event is one of many events that ASP.NET understands. The Page\_Load event is triggered when a page loads, and ASP.NET will automatically call the subroutine Page\_Load, and execute the code inside it:

## Example

<script runat="server">  
Sub Page\_Load  
lbl1.Text="The date and time is " & now()  
End Sub  
</script>  
  
<html>  
<body>  
<form runat="server">  
<h3><asp:label id="lbl1" runat="server" /></h3>  
</form>  
</body>  
</html>

[Show example »](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_pageload)

**Note:** The Page\_Load event contains no object references or event arguments!

## The Page.IsPostBack Property

The Page\_Load subroutine runs EVERY time the page is loaded. If you want to execute the code in the Page\_Load subroutine only the FIRST time the page is loaded, you can use the Page.IsPostBack property. If the Page.IsPostBack property is false, the page is loaded for the first time, if it is true, the page is posted back to the server (i.e. from a button click on a form):

## Example

<script runat="server">  
Sub Page\_Load  
if Not Page.IsPostBack then  
  lbl1.Text="The date and time is " & now()  
end if  
End Sub  
  
Sub submit(s As Object, e As EventArgs)  
lbl2.Text="Hello World!"  
End Sub  
</script>  
  
<html>  
<body>  
<form runat="server">  
<h3><asp:label id="lbl1" runat="server" /></h3>  
<h3><asp:label id="lbl2" runat="server" /></h3>  
<asp:button text="Submit" onclick="submit" runat="server" />  
</form>  
</body>  
</html>

[Show example »](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_pageispostback)

The example above will write the "The date and time is...." message only the first time the page is loaded. When a user clicks on the Submit button, the submit subroutine will write "Hello World!" to the second label, but the date and time in the first label will not change.

[« Previous](http://www.w3schools.com/aspnet/aspnet_controls.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_forms.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

[REPORT ERROR](http://www.w3schools.com/aspnet/aspnet_events.asp)

[PRINT PAGE](http://www.w3schools.com/aspnet/aspnet_events.asp)

[FORUM](http://www.w3schools.com/forum/default.asp)

# ASP.NET Web Forms - HTML Forms

[« Previous](http://www.w3schools.com/aspnet/aspnet_events.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_viewstate.asp)

All server controls must appear within a <form> tag, and the <form> tag must contain the runat="server" attribute.

## ASP.NET Web Forms

All server controls must appear within a <form> tag, and the <form> tag must contain the runat="server" attribute. The runat="server" attribute indicates that the form should be processed on the server. It also indicates that the enclosed controls can be accessed by server scripts:

<form runat="server">  
  
...HTML + server controls  
  
</form>

**Note:** The form is always submitted to the page itself. If you specify an action attribute, it is ignored. If you omit the method attribute, it will be set to method="post" by default. Also, if you do not specify the name and id attributes, they are automatically assigned by ASP.NET.

**Note:** An .aspx page can only contain ONE <form runat="server"> control!

If you select view source in an .aspx page containing a form with no name, method, action, or id attribute specified, you will see that ASP.NET has added these attributes to the form. It looks something like this:

<form name="\_ctl0" method="post" action="page.aspx" id="\_ctl0">  
  
...some code  
  
</form>

## Submitting a Form

A form is most often submitted by clicking on a button. The Button server control in ASP.NET has the following format:

<asp:Button id="id" text="label" OnClick="sub" runat="server" />

The id attribute defines a unique name for the button and the text attribute assigns a label to the button. The onClick event handler specifies a named subroutine to execute.

In the following example we declare a Button control in an .aspx file. A button click runs a subroutine which changes the text on the button:

[Example](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_button)

[« Previous](http://www.w3schools.com/aspnet/aspnet_events.asp)

# ASP.NET Web Forms - Maintaining the ViewState

[« Previous](http://www.w3schools.com/aspnet/aspnet_forms.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_textbox.asp)

You may save a lot of coding by maintaining the ViewState of the objects in your Web Form.

## Maintaining the ViewState

When a form is submitted in classic ASP, all form values are cleared. Suppose you have submitted a form with a lot of information and the server comes back with an error. You will have to go back to the form and correct the information. You click the back button, and what happens.......ALL form values are CLEARED, and you will have to start all over again! The site did not maintain your ViewState.

When a form is submitted in ASP .NET, the form reappears in the browser window together with all form values. How come? This is because ASP .NET maintains your ViewState. The ViewState indicates the status of the page when submitted to the server. The status is defined through a hidden field placed on each page with a <form runat="server"> control. The source could look something like this:

<form name="\_ctl0" method="post" action="page.aspx" id="\_ctl0">  
<input type="hidden" name="\_\_VIEWSTATE"  
value="dDwtNTI0ODU5MDE1Ozs+ZBCF2ryjMpeVgUrY2eTj79HNl4Q=" />  
  
.....some code  
  
</form>

Maintaining the ViewState is the default setting for ASP.NET Web Forms. If you want to NOT maintain the ViewState, include the directive <%@ Page EnableViewState="false" %> at the top of an .aspx page or add the attribute EnableViewState="false" to any control.

Look at the following .aspx file. It demonstrates the "old" way to do it. When you click on the submit button, the form value will disappear:

## Example

<html>  
<body>  
  
<form action="demo\_classicasp.aspx" method="post">  
Your name: <input type="text" name="fname" size="20">  
<input type="submit" value="Submit">  
</form>  
<%  
dim fname  
fname=Request.Form("fname")  
If fname<>"" Then  
Response.Write("Hello " & fname & "!")  
End If  
%>  
  
</body>  
</html>

[Show example »](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_classicasp)

Here is the new ASP .NET way. When you click on the submit button, the form value will NOT disappear:

## Example

Click view source in the right frame of the example to see that ASP .NET has added a hidden field in the form to maintain the ViewState

<script runat="server">  
Sub submit(sender As Object, e As EventArgs)  
lbl1.Text="Hello " & txt1.Text & "!"  
End Sub  
</script>  
  
<html>  
<body>  
  
<form runat="server">  
Your name: <asp:TextBox id="txt1" runat="server" />  
<asp:Button OnClick="submit" Text="Submit" runat="server" />  
<p><asp:Label id="lbl1" runat="server" /></p>  
</form>  
  
</body>  
</html>

[Show example »](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_aspnetviewstate)

[« Previous](http://www.w3schools.com/aspnet/aspnet_forms.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_textbox.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Forms - Maintaining the ViewState

[« Previous](http://www.w3schools.com/aspnet/aspnet_forms.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_textbox.asp)

You may save a lot of coding by maintaining the ViewState of the objects in your Web Form.

## Maintaining the ViewState

When a form is submitted in classic ASP, all form values are cleared. Suppose you have submitted a form with a lot of information and the server comes back with an error. You will have to go back to the form and correct the information. You click the back button, and what happens.......ALL form values are CLEARED, and you will have to start all over again! The site did not maintain your ViewState.

When a form is submitted in ASP .NET, the form reappears in the browser window together with all form values. How come? This is because ASP .NET maintains your ViewState. The ViewState indicates the status of the page when submitted to the server. The status is defined through a hidden field placed on each page with a <form runat="server"> control. The source could look something like this:

<form name="\_ctl0" method="post" action="page.aspx" id="\_ctl0">  
<input type="hidden" name="\_\_VIEWSTATE"  
value="dDwtNTI0ODU5MDE1Ozs+ZBCF2ryjMpeVgUrY2eTj79HNl4Q=" />  
  
.....some code  
  
</form>

Maintaining the ViewState is the default setting for ASP.NET Web Forms. If you want to NOT maintain the ViewState, include the directive <%@ Page EnableViewState="false" %> at the top of an .aspx page or add the attribute EnableViewState="false" to any control.

Look at the following .aspx file. It demonstrates the "old" way to do it. When you click on the submit button, the form value will disappear:

## Example

<html>  
<body>  
  
<form action="demo\_classicasp.aspx" method="post">  
Your name: <input type="text" name="fname" size="20">  
<input type="submit" value="Submit">  
</form>  
<%  
dim fname  
fname=Request.Form("fname")  
If fname<>"" Then  
Response.Write("Hello " & fname & "!")  
End If  
%>  
  
</body>  
</html>

[Show example »](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_classicasp)

Here is the new ASP .NET way. When you click on the submit button, the form value will NOT disappear:

## Example

Click view source in the right frame of the example to see that ASP .NET has added a hidden field in the form to maintain the ViewState

<script runat="server">  
Sub submit(sender As Object, e As EventArgs)  
lbl1.Text="Hello " & txt1.Text & "!"  
End Sub  
</script>  
  
<html>  
<body>  
  
<form runat="server">  
Your name: <asp:TextBox id="txt1" runat="server" />  
<asp:Button OnClick="submit" Text="Submit" runat="server" />  
<p><asp:Label id="lbl1" runat="server" /></p>  
</form>  
  
</body>  
</html>

[Show example »](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_aspnetviewstate)

[« Previous](http://www.w3schools.com/aspnet/aspnet_forms.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_textbox.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Forms - The Button Control

[« Previous](http://www.w3schools.com/aspnet/aspnet_textbox.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_databinding.asp)

The Button control is used to display a push button.

## The Button Control

The Button control is used to display a push button. The push button may be a submit button or a command button. By default, this control is a submit button.

A submit button does not have a command name and it posts the page back to the server when it is clicked. It is possible to write an event handler to control the actions performed when the submit button is clicked.

A command button has a command name and allows you to create multiple Button controls on a page. It is possible to write an event handler to control the actions performed when the command button is clicked.

The Button control's attributes and properties are listed in our [web controls reference page](http://www.w3schools.com/aspnet/aspnet_refwebcontrols.asp).

The example below demonstrates a simple Button control:

<html>  
<body>  
  
<form runat="server">  
<asp:Button id="b1" Text="Submit" runat="server" />  
</form>  
  
</body>  
</html>

## Add a Script

A form is most often submitted by clicking on a button.

In the following example we declare one TextBox control, one Button control, and one Label control in an .aspx file. When the submit button is triggered, the submit subroutine is executed. The submit subroutine writes a text to the Label control:

## Example

<script runat="server">  
Sub submit(sender As Object, e As EventArgs)  
lbl1.Text="Your name is " & txt1.Text  
End Sub  
</script>  
  
<html>  
<body>  
  
<form runat="server">  
Enter your name:  
<asp:TextBox id="txt1" runat="server" />  
<asp:Button OnClick="submit" Text="Submit" runat="server" />  
<p><asp:Label id="lbl1" runat="server" /></p>  
</form>  
  
</body>  
</html>

[Show example »](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_textbox)

# ASP.NET Web Forms - Data Binding

[« Previous](http://www.w3schools.com/aspnet/aspnet_button.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_arraylist.asp)

We may use data binding to fill lists with selectable items from an imported data source, like a database, an XML file, or a script.

## Data Binding

The following controls are list controls which support data binding:

* asp:RadioButtonList
* asp:CheckBoxList
* asp:DropDownList
* asp:Listbox

The selectable items in each of the above controls are usually defined by one or more asp:ListItem controls, like this:

<html>  
<body>  
  
<form runat="server">  
<asp:RadioButtonList id="countrylist" runat="server">  
<asp:ListItem value="N" text="Norway" />  
<asp:ListItem value="S" text="Sweden" />  
<asp:ListItem value="F" text="France" />  
<asp:ListItem value="I" text="Italy" />  
</asp:RadioButtonList>  
</form>  
  
</body>  
</html>

However, with data binding we may use a separate source, like a database, an XML file, or a script to fill the list with selectable items.

By using an imported source, the data is separated from the HTML, and any changes to the items are made in the separate data source.

In the next three chapters, we will describe how to bind data from a scripted data source.

[« Previous](http://www.w3schools.com/aspnet/aspnet_button.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_arraylist.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Forms - The ArrayList Object

[« Previous](http://www.w3schools.com/aspnet/aspnet_databinding.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_hashtable.asp)

The ArrayList object is a collection of items containing a single data value.

Examples

## Examples

[ArrayList DropDownList](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_arraylist_drop1)

[ArrayList RadioButtonList](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_arraylist_radio1)

## Create an ArrayList

The ArrayList object is a collection of items containing a single data value.

Items are added to the ArrayList with the Add() method.

The following code creates a new ArrayList object named mycountries and four items are added:

<script runat="server">  
Sub Page\_Load  
if Not Page.IsPostBack then  
  dim mycountries=New ArrayList  
  mycountries.Add("Norway")  
  mycountries.Add("Sweden")  
  mycountries.Add("France")  
  mycountries.Add("Italy")  
end if  
end sub  
</script>

By default, an ArrayList object contains 16 entries. An ArrayList can be sized to its final size with the TrimToSize() method:

<script runat="server">  
Sub Page\_Load  
if Not Page.IsPostBack then  
  dim mycountries=New ArrayList  
  mycountries.Add("Norway")  
  mycountries.Add("Sweden")  
  mycountries.Add("France")  
  mycountries.Add("Italy")  
  mycountries.TrimToSize()  
end if  
end sub  
</script>

An ArrayList can also be sorted alphabetically or numerically with the Sort() method:

<script runat="server">  
Sub Page\_Load  
if Not Page.IsPostBack then  
  dim mycountries=New ArrayList  
  mycountries.Add("Norway")  
  mycountries.Add("Sweden")  
  mycountries.Add("France")  
  mycountries.Add("Italy")  
  mycountries.TrimToSize()  
  mycountries.Sort()  
end if  
end sub  
</script>

To sort in reverse order, apply the Reverse() method after the Sort() method:

<script runat="server">  
Sub Page\_Load  
if Not Page.IsPostBack then  
  dim mycountries=New ArrayList  
  mycountries.Add("Norway")  
  mycountries.Add("Sweden")  
  mycountries.Add("France")  
  mycountries.Add("Italy")  
  mycountries.TrimToSize()  
  mycountries.Sort()  
  mycountries.Reverse()  
end if  
end sub  
</script>

## Data Binding to an ArrayList

An ArrayList object may automatically generate the text and values to the following controls:

* asp:RadioButtonList
* asp:CheckBoxList
* asp:DropDownList
* asp:Listbox

To bind data to a RadioButtonList control, first create a RadioButtonList control (without any asp:ListItem elements) in an .aspx page:

<html>  
<body>  
  
<form runat="server">  
<asp:RadioButtonList id="rb" runat="server" />  
</form>  
  
</body>  
</html>

Then add the script that builds the list and binds the values in the list to the RadioButtonList control:

## Example

<script runat="server">  
Sub Page\_Load  
if Not Page.IsPostBack then  
  dim mycountries=New ArrayList  
  mycountries.Add("Norway")  
  mycountries.Add("Sweden")  
  mycountries.Add("France")  
  mycountries.Add("Italy")  
  mycountries.TrimToSize()  
  mycountries.Sort()  
  rb.DataSource=mycountries  
  rb.DataBind()  
end if  
end sub  
</script>  
  
<html>  
<body>  
  
<form runat="server">  
<asp:RadioButtonList id="rb" runat="server" />  
</form>  
  
</body>  
</html>

[Show example »](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_arraylist_radio1)

The DataSource property of the RadioButtonList control is set to the ArrayList and it defines the data source of the RadioButtonList control. The DataBind() method of the RadioButtonList control binds the data source with the RadioButtonList control.

**Note:** The data values are used as both the Text and Value properties for the control. To add Values that are different from the Text, use either the Hashtable object or the SortedList object.

[« Previous](http://www.w3schools.com/aspnet/aspnet_databinding.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_hashtable.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

# [colorpicker](http://www.w3schools.com/tags/ref_colorpicker.asp)ASP.NET Web Forms - The Hashtable Object

[« Previous](http://www.w3schools.com/aspnet/aspnet_arraylist.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_sortedlist.asp)

The Hashtable object contains items in key/value pairs.

Examples

**Examples**

[Hashtable RadiobuttonList 1](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_hashtable_radio1)

[Hashtable RadiobuttonList 2](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_hashtable_radio2)

[Hashtable DropDownList](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_hashtable_drop1)

**Create a Hashtable**

The Hashtable object contains items in key/value pairs. The keys are used as indexes, and very quick searches can be made for values by searching through their keys.

Items are added to the Hashtable with the Add() method.

The following code creates a Hashtable named mycountries and four elements are added:

<script runat="server">  
Sub Page\_Load  
if Not Page.IsPostBack then  
  dim mycountries=New Hashtable  
  mycountries.Add("N","Norway")  
  mycountries.Add("S","Sweden")  
  mycountries.Add("F","France")  
  mycountries.Add("I","Italy")  
end if  
end sub  
</script>

**Data Binding**

A Hashtable object may automatically generate the text and values to the following controls:

* asp:RadioButtonList
* asp:CheckBoxList
* asp:DropDownList
* asp:Listbox

To bind data to a RadioButtonList control, first create a RadioButtonList control (without any asp:ListItem elements) in an .aspx page:

<html>  
<body>  
  
<form runat="server">  
<asp:RadioButtonList id="rb" runat="server" AutoPostBack="True" />  
</form>  
  
</body>  
</html>

Then add the script that builds the list:

<script runat="server">  
sub Page\_Load  
if Not Page.IsPostBack then  
  dim mycountries=New Hashtable  
  mycountries.Add("N","Norway")  
  mycountries.Add("S","Sweden")  
  mycountries.Add("F","France")  
  mycountries.Add("I","Italy")  
  rb.DataSource=mycountries  
  rb.DataValueField="Key"  
  rb.DataTextField="Value"  
  rb.DataBind()  
end if  
end sub  
</script>  
  
<html>  
<body>  
  
<form runat="server">  
<asp:RadioButtonList id="rb" runat="server" AutoPostBack="True" />  
</form>  
  
</body>  
</html>

Then we add a sub routine to be executed when the user clicks on an item in the RadioButtonList control. When a radio button is clicked, a text will appear in a label:

**Example**

<script runat="server">  
sub Page\_Load  
if Not Page.IsPostBack then  
  dim mycountries=New Hashtable  
  mycountries.Add("N","Norway")  
  mycountries.Add("S","Sweden")  
  mycountries.Add("F","France")  
  mycountries.Add("I","Italy")  
  rb.DataSource=mycountries  
  rb.DataValueField="Key"  
  rb.DataTextField="Value"  
  rb.DataBind()  
end if  
end sub  
  
sub displayMessage(s as Object,e As EventArgs)  
lbl1.text="Your favorite country is: " & rb.SelectedItem.Text  
end sub  
</script>  
  
<html>  
<body>  
  
<form runat="server">  
<asp:RadioButtonList id="rb" runat="server"  
AutoPostBack="True" onSelectedIndexChanged="displayMessage" />  
<p><asp:label id="lbl1" runat="server" /></p>  
</form>  
  
</body>  
</html>

[Show example »](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_hashtable_radio1)

**Note:** You cannot choose the sort order of the items added to the Hashtable. To sort items alphabetically or numerically, use the SortedList object.

[« Previous](http://www.w3schools.com/aspnet/aspnet_arraylist.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_sortedlist.asp)

# ASP.NET Web Pages - The WebMail Helper

[« Previous](http://www.w3schools.com/aspnet/webpages_chart.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_php.asp)

The WebMail Helper - One of many useful ASP.NET Web Helpers.

## The WebMail Helper

The WebMail Helper makes it easy to send an email from a web application using SMTP (Simple Mail transfer Protocol).

## Scenario: Email Support

To demonstrate the use of email, we will create an input page for support, let the user submit the page to another page, and send an email about the support problem.

## First: Edit Your AppStart Page

If you have built the Demo application in this tutorial, you already have a page called \_AppStart.cshtml with the following content:

## \_AppStart.cshtml

@{  
WebSecurity.InitializeDatabaseConnection("Users", "UserProfile", "UserId", "Email", true);  
}

To initiate the WebMail helper, add the the following WebMail properties to your AppStart page:

## \_AppStart.cshtml

@{  
WebSecurity.InitializeDatabaseConnection("Users", "UserProfile", "UserId", "Email", true);  
WebMail.SmtpServer = "smtp.example.com";  
WebMail.SmtpPort = 25;  
WebMail.EnableSsl = false;  
WebMail.UserName = "support@example.com";  
WebMail.Password = "password-goes-here";  
WebMail.From = "john@example.com";  
}

Properties explained:

**SmtpServer:** The name the SMTP server that will be used to send the emails.

**SmtpPort:** The port the server will use to send SMTP transactions (emails).

**EnableSsl:** True, if the server should use SSL (Secure Socket Layer) encryption.

**UserName:** The name of the SMTP email account used to send the email.

**Password:** The password of the SMTP email account.

**From:** The email to appear in the from address (often the same as UserName).

## Second: Create an Email Input Page

Then create an input page, and name it Email\_Input:

## Email\_Input.cshtml

<!DOCTYPE html>   
<html>   
<body>   
<h1>Request for Assistance</h1>   
  
<form method="post" action="EmailSend.cshtml">   
<label>Username:</label>  
<input type="text name="customerEmail" />  
<label>Details about the problem:</label>   
<textarea name="customerRequest" cols="45" rows="4"></textarea>   
<p><input type="submit" value="Submit" /></p>   
</form>   
  
</body>   
</html>

The purpose of the input page is to collect information, then submit the data to a new page that can send the information as an email.

## Third: Create An Email Send Page

Then create the page that will be used to send the email, and name it Email\_Send:

## Email\_Send.cshtml

@{ // Read input  
var customerEmail = Request["customerEmail"];  
var customerRequest = Request["customerRequest"];  
try  
{  
// Send email   
WebMail.Send(to:"someone@example.com", subject: "Help request from - " + customerEmail, body: customerRequest );   
}  
catch (Exception ex )  
{  
<text>@ex</text>   
}  
}

For more information about sending emails from a ASP.NET Web Pages application, please see the: [WebMail Object Reference](http://www.w3schools.com/aspnet/webpages_ref_webmail.asp).

[« Previous](http://www.w3schools.com/aspnet/webpages_chart.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_php.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Pages - PHP

[« Previous](http://www.w3schools.com/aspnet/webpages_email.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_publish.asp)

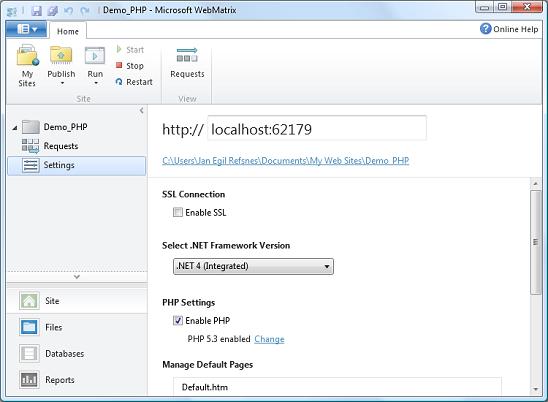
Attention PHP Developers. Web Pages can be written in PHP.

## WebMatrix Supports PHP

At first look, WebMatrix only supports Microsoft technologies. This is not true. In WebMatrix you can write full PHP applications with MySQL and all.

## Create a PHP Site

In WebMatrix, create an empty site named "Demo\_PHP", enable PHP (see picture below), create a new empty page of the type PHP, name it "index.php", and you have created your first PHP site.



## Create a PHP Page

Put the following code inside the "index.php" file:

## index.php

<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
phpinfo();  
?>  
  
</body>  
</html>

Run the file and see PHP at work.

[« Previous](http://www.w3schools.com/aspnet/webpages_email.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_publish.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Pages - Publishing the Website

[« Previous](http://www.w3schools.com/aspnet/webpages_php.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_examples.asp)

Learn how to publish a Web Pages application without using WebMatrix.

## Publish Your Application Without Using WebMatrix

An ASP.NET Web Pages application can be published to a remote server by using the Publish commands in WebMatrix (or Visual Studio).

This function copies all your application files, cshtml pages, images, and all the required DLL files for Web Pages, for Razor, for Helpers, and for SQL Server Compact (if a database is used).

Sometimes you don't want to use this option. Maybe your hosting provider only supports FTP? Maybe you already have a web site based on classic ASP? Maybe you want to copy the files yourself? Maybe you want to use Front Page, Expression Web, or some other publishing software?

**Will you get a problem? Yes, you will. But you can solve it.**

To perform a web copy, you have to know how to include the right files, what DDL files to copy, and where store them.

Follow these steps:

## 1. Use the Latest Version of ASP.NET

Before you continue, make sure your hosting computer runs the latest version of ASP.NET (4.0 or 4.5).

## 2. Copy the Web Folders

Copy your website (all folders and content) from your development computer to an application folder on your remote hosting computer (server).

|  |  |
| --- | --- |
| **Note** | If your application contains data, **don't copy the data** (see point 4 below). |

## 3. The DLL Files

Make sure the bin folder, on your remote hosting computer, contains the same dll files as on your development computer.

After copying the bin folder, it should contain files like this:

 Microsoft.Web.Infrastructure.dll  
NuGet.Core.dll  
System.Web.Helpers.dll  
System.Web.Razor.dll  
System.Web.WebPages.Administration.dll  
System.Web.WebPages.Deployment.dll  
System.Web.WebPages.dll  
System.Web.WebPages.Razor.dll  
WebMatrix.Data.dll  
WebMatrix.WebData

## 4. Copy Your Data

If your application contains data or a database. For instance an SQL Server Compact database (a .sdf file in App\_Data folder), consider the following:

Do you want to publish your test data to the remote server?

Most likely not.

If you have test data on your development computer, it may overwrite production data on your remote hosting computer.

If you have to copy an SQL database (.sdf file), perhaps you should delete everything in the database, and then copy the empty .sdf file from your development computer to the server.

**THAT'S IT. GOOD LUCK !**

[« Previous](http://www.w3schools.com/aspnet/webpages_php.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_examples.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Pages - Examples in C# and VB

[« Previous](http://www.w3schools.com/aspnet/webpages_publish.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_ref_classes.asp)

Learn ASP.NET Web Pages by C# and Visual Basic examples.

|  |  |
| --- | --- |
| **Examples in C#** | **Examples in VB** |
| **Basic Web Pages**  [Display Date and Time](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_001) [Reusable Header and Footer](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_002) [Basic HTML Form](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_009)  [Examples Explained](http://www.w3schools.com/aspnet/webpages_intro.asp) | **Basic Web Pages**  [Display Date and Time](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_001) [Reusable Header and Footer](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_002) [Basic HTML Form](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_009)  [Examples Explained](http://www.w3schools.com/aspnet/webpages_intro.asp) |
| **Basic C#**  [For Loop](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_005) [For Each Loop](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_006) [While Loop](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_007) [Array](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_008)  [If Condition](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_010) [If Else Condition](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_011) [Else If Condition](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_012) [Switch Condition](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_razor_cs_013)  [Examples Explained](http://www.w3schools.com/aspnet/razor_intro.asp) | **Basic VB**  [For Loop](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_005) [For Each Loop](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_006) [While Loop](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_007) [Array](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_008)  [If Condition](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_010) [If Else Condition](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_011) [Else If Condition](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_012) [Select Condition](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_razor_vb_013)  [Examples Explained](http://www.w3schools.com/aspnet/razor_intro.asp) |
| **Working with Databases**  [Display Database Data](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_003) [Display Data with WebGrid](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_004)  [Examples Explained](http://www.w3schools.com/aspnet/webpages_database.asp) | **Working with Databases**  [Display Database Data](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_003) [Display Data with WebGrid](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_004)  [Examples Explained](http://www.w3schools.com/aspnet/webpages_database.asp) |
| **Using the Chart Helper**  [Display a Bar Chart from an Array](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_005) [Display a Bar Chart from a Database](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_006) [Display a Pie Chart from a Database](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_007) [Display a Pie Chart from an XML File](http://www.w3schools.com/aspnet/showfile_c.asp?filename=try_webpages_cs_008)  [Examples Explained](http://www.w3schools.com/aspnet/webpages_chart.asp) | **Using the Chart Helper**  [Display a Bar Chart from an Array](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_005) [Display a Bar Chart from a Database](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_006) [Display a Pie Chart from a Database](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_007) [Display a Pie Chart from an XML File](http://www.w3schools.com/aspnet/showfile_vb.asp?filename=try_webpages_vb_008)  [Examples Explained](http://www.w3schools.com/aspnet/webpages_chart.asp) |

[« Previous](http://www.w3schools.com/aspnet/webpages_examples.asp)

## WP References

# ASP.NET Web Pages - Classes

[« Previous](http://www.w3schools.com/aspnet/webpages_examples.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_ref_websecurity.asp)

## ASP.NET Classes Reference

|  |  |
| --- | --- |
| **Method** | **Description** |
| AsBool(), AsBool(true|false) | Converts a string value to a Boolean value (true/false). Returns false or the specified value if the string does not represent true/false. |
| AsDateTime(), AsDateTime(value) | Converts a string value to date/time. Returns DateTime. MinValue or the specified value if the string does not represent a date/time. |
| AsDecimal(), AsDecimal(value) | Converts a string value to a decimal value. Returns 0.0 or the specified value if the string does not represent a decimal value. |
| AsFloat(), AsFloat(value) | Converts a string value to a float. Returns 0.0 or the specified value if the string does not represent a decimal value. |
| AsInt(), AsInt(value) | Converts a string value to an integer. Returns 0 or the specified value if the string does not represent an integer. |
| Href(path [, param1 [, param2]]) | Creates a browser-compatible URL from a local file path, with optional additional path parts. |
| Html.Raw(value) | Renders value as HTML markup instead of rendering it as HTML-encoded output. |
| IsBool(), IsDateTime(), IsDecimal(), IsFloat(), IsInt() | Returns true if the value can be converted from a string to the specified type. |
| IsEmpty() | Returns true if the object or variable has no value. |
| IsPost | Returns true if the request is a POST. (Initial requests are usually a GET.) |
| Layout | Specifies the path of a layout page to apply to this page. |
| PageData[key], PageData[index], Page | Contains data shared between the page, layout pages, and partial pages in the current request. You can use the dynamic Page property to access the same data, as in the following example: |
| RenderBody() | (Layout pages) Renders the content of a content page that is not in any named sections. |
| RenderPage(path, values)  RenderPage(path[, param1 [, param2]]) | Renders a content page using the specified path and optional extra data. You can get the values of the extra parameters from PageData by position (example 1) or key (example 2). |
| RenderSection(sectionName [, required = true|false]) | (Layout pages) Renders a content section that has a name. Set required to false to make a section optional. |
| Request.Cookies[key] | Gets or sets the value of an HTTP cookie. |
| Request.Files[key] | Gets the files that were uploaded in the current request. |
| Request.Form[key] | Gets data that was posted in a form (as strings). Request[key] checks both the Request.Form and the Request.QueryString collections. |
| Request.QueryString[key] | Gets data that was specified in the URL query string. Request[key] checks both the Request.Form and the Request.QueryString collections. |
| Request.Unvalidated(key)  Request.Unvalidated().QueryString|Form|Cookies|Headers[key] | Selectively disables request validation for a form element, query-string value, cookie, or header value. Request validation is enabled by default and prevents users from posting markup or other potentially dangerous content. |
| Response.AddHeader(name, value) | Adds an HTTP server header to the response. |
| Response.OutputCache(seconds [, sliding] [, varyByParams]) | Caches the page output for a specified time. Optionally set sliding to reset the timeout on each page access and varyByParams to cache different versions of the page for each different query string in the page request. |
| Response.Redirect(path) | Redirects the browser request to a new location. |
| Response.SetStatus(httpStatusCode) | Sets the HTTP status code sent to the browser. |
| Response.WriteBinary(data [, mimetype]) | Writes the contents of data to the response with an optional MIME type. |
| Response.WriteFile(file) | Writes the contents of a file to the response. |
| @section(sectionName) { content } | (Layout pages) Defines a content section that has a name. |
| Server.HtmlDecode(htmlText) | Decodes a string that is HTML encoded. |
| Server.HtmlEncode(text) | Encodes a string for rendering in HTML markup. |
| Server.MapPath(virtualPath) | Returns the server physical path for the specified virtual path. |
| Server.UrlDecode(urlText) | Decodes text from a URL. |
| Server.UrlEncode(text) | Encodes text to put in a URL. |
| Session[key] | Gets or sets a value that exists until the user closes the browser. |
| ToString() | Displays a string representation of the object's value. |
| UrlData[index] | Gets additional data from the URL (for example, /MyPage/ExtraData). |

[« Previous](http://www.w3schools.com/aspnet/webpages_examples.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_ref_websecurity.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Pages - WebSecurity Object

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_classes.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_ref_database.asp)

## Description

The **WebSecurity Object** provides security and authentication for ASP.NET Web Pages applications.

With the WebSecurity object you can create user accounts, login and logout users, reset or change passwords, and more.

## WebSecurity Object Reference - Properties

|  |  |
| --- | --- |
| **Properties** | **Description** |
| [CurrentUserId](http://www.w3schools.com/aspnet/prop_websecurity_currentuserid.asp) | Gets the ID for the current user |
| [CurrentUserName](http://www.w3schools.com/aspnet/prop_websecurity_currentusername.asp) | Gets the name of the current user |
| [HasUserId](http://www.w3schools.com/aspnet/prop_websecurity_hasuserid.asp) | Returns true if the current has a user ID |
| [IsAuthenticated](http://www.w3schools.com/aspnet/prop_websecurity_isauthenticated.asp) | Returns true if the current user is logged in |

## WebSecurity Object Reference - Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| [ChangePassword()](http://www.w3schools.com/aspnet/met_websecurity_changepassword.asp) | Changes the password for a user |
| [ConfirmAccount()](http://www.w3schools.com/aspnet/met_websecurity_confirmaccount.asp) | Confirms an account using a confirmation token |
| [CreateAccount()](http://www.w3schools.com/aspnet/met_websecurity_createaccount.asp) | Creates a new user account |
| [CreateUserAndAccount()](http://www.w3schools.com/aspnet/met_websecurity_createuserandaccount.asp) | Creates a new user account |
| [GeneratePasswordResetToken()](http://www.w3schools.com/aspnet/met_websecurity_generatepasswordresettoken.asp) | Generates a token that can be sent to as user by email |
| [GetCreateDate()](http://www.w3schools.com/aspnet/met_websecurity_getcreatedate.asp) | Gets the time the specified membership was created |
| [GetPasswordChangeDate()](http://www.w3schools.com/aspnet/met_websecurity_getpasswordchangedate.asp) | Gets the date and time when password was changed |
| [GetUserId()](http://www.w3schools.com/aspnet/met_websecurity_getuserid.asp) | Gets a user ID from a user name |
| [InitializeDatabaseConnection()](http://www.w3schools.com/aspnet/met_websecurity_initializedatabaseconnection.asp) | Initializes the WebSecurity system (database) |
| [IsConfirmed()](http://www.w3schools.com/aspnet/met_websecurity_isconfirmed.asp) | Checks if a user is confirmed |
| [IsCurrentUser()](http://www.w3schools.com/aspnet/met_websecurity_iscurrentuser.asp) | Checks if the current user matches a user name |
| [Login()](http://www.w3schools.com/aspnet/met_websecurity_login.asp) | Logs the user in by setting a token in the cookie |
| [Logout()](http://www.w3schools.com/aspnet/met_websecurity_logout.asp) | Logs the user out by removing the token cookie |
| [RequireAuthenticatedUser()](http://www.w3schools.com/aspnet/met_websecurity_requireauthenticateduser.asp) | Exits the page if the user is not an authenticated user |
| [RequireRoles()](http://www.w3schools.com/aspnet/met_websecurity_requireroles.asp) | Exits the page if the user is not a part of the specified roles |
| [RequireUser()](http://www.w3schools.com/aspnet/met_websecurity_requireuser.asp) | Exits the page if the user is not the specified user |
| [ResetPassword()](http://www.w3schools.com/aspnet/met_websecurity_resetpassword.asp) | Changes a user's password using a token |
| [UserExists()](http://www.w3schools.com/aspnet/met_websecurity_userexists.asp) | Checks if a given user exists |

## Technical Data

|  |  |
| --- | --- |
| **Name** | **Value** |
| Class | WebMatrix.WebData.WebSecurity |
| Namespace | WebMatrix.WebData |
| Assembly | WebMatrix.WebData.dll |

## Initializing the WebSecurity Database

You must create or initialize an WebSecurity database before you can use the WebSecurity object in your code.

In the root of your web, create a page (or edit the page ) named **\_AppStart.cshtml**.

Put the following code inside the file:

## \_AppStart.cshtml

@{  
WebSecurity.InitializeDatabaseConnection("Users", "UserProfile", "UserId", "Email", true);  
}

The code above will run each time the web site (application) starts. It initializes the WebSecurity database.

**"Users"** is the name of the WebSecurity database (Users.sdf).

**"UserProfile"** is the name of the database table that contains the user profile information.

**"UserId"** is the name of the column that contains the user IDs (primary key).

**"Email"** is the name of the column that contains user names.

The last parameter **true** is a boolean value indicating that the user profile and membership tables should be created automatically if they don't exist, otherwise **false**.

|  |  |
| --- | --- |
| **Note** | Although **true** indicates automatic creation of the database **tables,** the database itself will not be created automatically. It must exist. |

## The WebSecurity Database

The **UserProfile** table contains one record for each user, with a user ID (primary key) and the user's name (email):

|  |  |
| --- | --- |
| **UserId** | **Email** |
| 1 | john@johnson.net |
| 2 | peter@peterson.com |
| 3 | lars@larson.eut |

The **Membership table** will contain membership information about when the user was created and if (and when) the membership was confirmed.

Much like this (some columns are not shown):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Id** | **Create Date** | **Confirmation Token** | **Is Confirmed** | **Last Password Failure** | **Password** | **Password Change** |
| 1 | 12.04.2012 16:12:17 | NULL | True | NULL | AFNQhWfy.... | 12.04.2012 16:12:17 |

Note: If you want to see all columns and all content, open the database with WebMatrix and look inside each table.

## Simple Membership Configuration

You might get errors using the WebSecurity object, if your site is not configured to use the ASP.NET Web Pages membership system **SimpleMembership**.

This can occur if a hosting provider's server is configured differently than your local server. To fix this, add the following element to the site's Web.config file:

<appSettings>   
<add key="enableSimpleMembership" value="true" />   
</appSettings>

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_classes.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_ref_database.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Pages - Database Object

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_websecurity.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_ref_webmail.asp)

## ASP.NET Database Object Reference

|  |  |
| --- | --- |
| **Method** | **Description** |
| Database.Execute(SQLstatement [, parameters]) | Executes SQLstatement (with optional parameters) such as INSERT, DELETE, or UPDATE and returns a count of affected records. |
| Database.GetLastInsertId() | Returns the identity column from the most recently inserted row. |
| Database.Open(filename)  Database.Open(connectionStringName) | Opens either the specified database file or the database specified using a named connection string from the Web.config file. |
| Database.OpenConnectionString(connectionString) | Opens a database using the connection string. (This contrasts with Database.Open, which uses a connection string name.) |
| Database.Query(SQLstatement[, parameters]) | Queries the database using SQLstatement (optionally passing parameters) and returns the results as a collection. |
| Database.QuerySingle(SQLstatement [, parameters]) | Executes SQLstatement (with optional parameters) and returns a single record. |
| Database.QueryValue(SQLstatement [, parameters]) | Executes SQLstatement (with optional parameters) and returns a single value. |

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_websecurity.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_ref_webmail.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Pages - WebMail Object

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_database.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_ref_helpers.asp)

With the WebMail object you can easily send emails from a web page.

## Description

The **WebMail Object** provides email for ASP.NET Web Pages using SMTP (Simple Mail Transfer Protocol).

## Example

See an example in the chapter: [Web Pages Email](http://www.w3schools.com/aspnet/webpages_email.asp).

## WebMail Object Reference - Properties

|  |  |
| --- | --- |
| **Properties** | **Description** |
| SmtpServer | The name the SMTP server that will send the emails |
| SmtpPort | The port the server will use to send SMTP emails |
| EnableSsl | True, if the server should use SSL encryption |
| UserName | The name of the SMTP account used to send the email |
| Password | The password of the SMTP account |
| From | The email to appear in the from address |

## WebMail Object Reference - Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| Send() | Sends an email message to an SMTP server for delivery |

The Send() method has the following parameters:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| to | String | The Email recipients (separated by semicolon) |
| subject | String | The subject line |
| body | String | The body of the message |

And the following optional parameters:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| from | String | The email of the sender |
| cc | String | The cc emails (separated by semicolon) |
| filesToAttach | Collection | Filenames |
| isBodyHtml | Boolean | True if the email body is in HTML |
| additionalHeaders | Collection | Additional headers |

## Technical Data

|  |  |
| --- | --- |
| **Name** | **Value** |
| Class | System.Web.Helpers.WebMail |
| Namespace | System.Web.Helpers |
| Assembly | System.Web.Helpers.dll |

## Initializing the WebMail Helper

To use the WebMail helper, you need access to an SMTP server. SMTP is the "output" part of email. If you use a web host, you probably already know the name of the SMTP server. If you work in a corporate network, your IT department can give you the name. If you are working at home, you might be able to use your ordinary email provider.

 In order to send an email you will need:

* The name of the SMTP server
* The port number (most often 25)
* An email user name
* An email password

In the root of your web, create a page (or edit the page ) named **\_AppStart.cshtml**.

Put the following code inside the file:

## \_AppStart.cshtml

@{  
WebMail.SmtpServer = "smtp.example.com";  
WebMail.SmtpPort = 25;  
WebMail.EnableSsl = false;  
WebMail.UserName = "support@example.com";  
WebMail.Password = "password";  
WebMail.From = "john@example.com"  
}

The code above will run each time the web site (application) starts. It feeds your **WebMail Object** with initial values.

Please substitute:

**smtp.example.com** with the name the SMTP server that will be used to send the emails.

**25** with the port number the server will use to send SMTP transactions (emails).

**false** with true, if the server should use SSL (Secure Socket Layer) encryption.

**support@example.com** with the name of the SMTP email account used to send emails.

**password** with the password of the SMTP email account.

**john@example** with the email to appear in the from address.

|  |  |
| --- | --- |
| **Note** | You don't **have to** initiate the WebMail object in your AppStart file, but you must set these properties before you call the **WebMail.Send()** method. |

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_database.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_ref_helpers.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Pages - More Helpers

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_webmail.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/razor_intro.asp)

## ASP.NET Helpers - Objects References

## Analytics Object Reference (Google)

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Analytics.GetGoogleHtml(webPropertyId) | Renders the Google Analytics JavaScript code for the specified ID. |
| Analytics.GetStatCounterHtml(project, security) | Renders the StatCounter Analytics JavaScript code for the specified project. |
| Analytics.GetYahooHtml(account) | Renders the Yahoo Analytics JavaScript code for the specified account. |

## Bing Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Bing.SearchBox([boxWidth]) | Passes a search to Bing. To specify the site to search and a title for the search box, you can set the Bing.SiteUrl and Bing.SiteTitle properties. Normally you set these properties in the \_AppStart page. |
| Bing.AdvancedSearchBox([, boxWidth] [, resultWidth] [, resultHeight]   [, themeColor] [, locale]) | Displays Bing search results in the page with optional formatting. To specify the site to search and a title for the search box, you can set the Bing.SiteUrl and Bing.SiteTitle properties. Normally you set these properties in the \_AppStart page. |

## Chart Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
|  |  |
| Chart(width, height [, template] [, templatePath]) | Initializes a chart. |
| Chart.AddLegend([title] [, name]) | Adds a legend to a chart. |
| Chart.AddSeries([name] [, chartType] [, chartArea]   [, axisLabel] [, legend] [, markerStep] [, xValue]   [, xField] [, yValues] [, yFields] [, options]) | Adds a series of values to the chart. |

## Crypto Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Crypto.Hash(string [, algorithm])  Crypto.Hash(bytes [, algorithm]) | Returns a hash for the specified data. The default algorithm is sha256. |

## Facebook Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Facebook.LikeButton(href [, buttonLayout] [, showFaces] [, width] [, height] [, action] [, font] [, colorScheme] [, refLabel]) | Lets Facebook users make a connection to pages. |

## FileUpload Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| FileUpload.GetHtml([initialNumberOfFiles] [, allowMoreFilesToBeAdded]   [, includeFormTag] [, addText] [, uploadText]) | Renders UI for uploading files. |

## GamerCard Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| GamerCard.GetHtml(gamerTag) | Renders the specified Xbox gamer tag. |

## Gravatar Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Gravatar.GetHtml(email [, imageSize] [, defaultImage] [, rating]   [, imageExtension] [, attributes]) | Renders the Gravatar image for the specified email address. |

## Json Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Json.Encode(object) | Converts a data object to a string in the JavaScript Object Notation (JSON) format. |
| Json.Decode(string) | Converts a JSON-encoded input string to a data object that you can iterate over or insert into a database. |

## LinkShare Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| LinkShare.GetHtml(pageTitle [, pageLinkBack] [, twitterUserName]   [, additionalTweetText] [, linkSites]) | Renders social networking links using the specified title and optional URL. |

## ModelState Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| ModelStateDictionary.AddError(key, errorMessage) | Associates an error message with a form field. Use the ModelState helper to access this member. |
| ModelStateDictionary.AddFormError(errorMessage) | Associates an error message with a form. Use the ModelState helper to access this member. |
| ModelStateDictionary.IsValid | Returns true if there are no validation errors. Use the ModelState helper to access this member. |

## ObjectInfo Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| ObjectInfo.Print(value [, depth] [, enumerationLength]) | Renders the properties and values of an object and any child objects. |

## Recaptcha Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Recaptcha.GetHtml([, publicKey] [, theme] [, language] [, tabIndex]) | Renders the reCAPTCHA verification test. |
| ReCaptcha.PublicKey  ReCaptcha.PrivateKey | Sets public and private keys for the reCAPTCHA service. Normally you set these properties in the \_AppStart page. |
| ReCaptcha.Validate([, privateKey]) | Returns the result of the reCAPTCHA test. |
|  |  |

## ServerInfo Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| ServerInfo.GetHtml() | Renders status information about ASP.NET Web Pages. |

## Twitter Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Twitter.Profile(twitterUserName) | Renders a Twitter stream for the specified user. |
| Twitter.Search(searchQuery) | Renders a Twitter stream for the specified search text. |

## Video Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| Video.Flash(filename [, width, height]) | Renders a Flash video player for the specified file with optional width and height. |
| Video.MediaPlayer(filename [, width, height]) | Renders a Windows Media player for the specified file with optional width and height. |
| Video.Silverlight(filename, width, height) | Renders a Silverlight player for the specified .xap file with required width and height. |

## WebCache Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| WebCache.Get(key) | Returns the object specified by key, or null if the object is not found. |
| WebCache.Remove(key) | Removes the object specified by key from the cache. |
| WebCache.Set(key, value [, minutesToCache] [, slidingExpiration]) | Puts value into the cache under the name specified by key. |

## WebGrid Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| WebGrid(data) | Creates a new WebGrid object using data from a query. |
| WebGrid.GetHtml() | Renders markup to display data in an HTML table. |
| WebGrid.Pager() | Renders a pager for the WebGrid object. |

## WebImage Object Reference

|  |  |
| --- | --- |
| **Helper** | **Description** |
| WebImage(path) | Loads an image from the specified path. |
| WebImage.AddImagesWatermark(image) | Adds the specified image as a watermark. |
| WebImage.AddTextWatermark(text) | Adds the specified text to the image. |
| WebImage.FlipHorizontal()  WebImage.FlipVertical() | Flips the image horizontally or vertically. |
| WebImage.GetImageFromRequest() | Loads an image when an image is posted to a page during a file upload. |
| WebImage.Resize(width, height) | Resizes the image. |
| WebImage.RotateLeft()  WebImage.RotateRight() | Rotates the image to the left or the right. |
| WebImage.Save(path [, imageFormat]) | Saves the image to the specified path. |

[« Previous](http://www.w3schools.com/aspnet/webpages_ref_webmail.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/razor_intro.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# SP.NET Web Forms - HTML Controls

[« Previous](http://www.w3schools.com/aspnet/aspnet_examples.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_refwebcontrols.asp)

HTML server controls are HTML tags understood by the server.

## HTML Server Controls

HTML elements in ASP.NET files are, by default, treated as text. To make these elements programmable, add a runat="server" attribute to the HTML element. This attribute indicates that the element should be treated as a server control.

**Note:** All HTML server controls must be within a <form> tag with the runat="server" attribute!

**Note:** ASP.NET requires that all HTML elements must be properly closed and properly nested.

|  |  |
| --- | --- |
| **HTML Server Control** | **Description** |
| [HtmlAnchor](http://www.w3schools.com/aspnet/control_htmlanchor.asp) | Controls an <a> HTML element |
| [HtmlButton](http://www.w3schools.com/aspnet/control_htmlbutton.asp) | Controls a <button> HTML element |
| [HtmlForm](http://www.w3schools.com/aspnet/control_htmlform.asp) | Controls a <form> HTML element |
| [HtmlGeneric](http://www.w3schools.com/aspnet/control_htmlgeneric.asp) | Controls other HTML element not specified by a specific HTML server control, like <body>, <div>, <span>, etc. |
| [HtmlImage](http://www.w3schools.com/aspnet/control_htmlimage.asp) | Controls an <image> HTML element |
| [HtmlInputButton](http://www.w3schools.com/aspnet/control_htmlinputbutton.asp) | Controls <input type="button">, <input type="submit">, and <input type="reset"> HTML elements |
| [HtmlInputCheckBox](http://www.w3schools.com/aspnet/control_htmlinputcheckbox.asp) | Controls an <input type="checkbox"> HTML element |
| [HtmlInputFile](http://www.w3schools.com/aspnet/control_htmlinputfile.asp) | Controls an <input type="file"> HTML element |
| [HtmlInputHidden](http://www.w3schools.com/aspnet/control_htmlinputhidden.asp) | Controls an <input type="hidden"> HTML element |
| [HtmlInputImage](http://www.w3schools.com/aspnet/control_htmlinputimage.asp) | Controls an <input type="image"> HTML element |
| [HtmlInputRadioButton](http://www.w3schools.com/aspnet/control_htmlinputradiobutton.asp) | Controls an <input type="radio"> HTML element |
| [HtmlInputText](http://www.w3schools.com/aspnet/control_htmlinputtext.asp) | Controls <input type="text"> and <input type="password"> HTML elements |
| [HtmlSelect](http://www.w3schools.com/aspnet/control_htmlselect.asp) | Controls a <select> HTML element |
| [HtmlTable](http://www.w3schools.com/aspnet/control_htmltable.asp) | Controls a <table> HTML element |
| [HtmlTableCell](http://www.w3schools.com/aspnet/control_htmltablecell.asp) | Controls <td>and <th> HTML elements |
| [HtmlTableRow](http://www.w3schools.com/aspnet/control_htmltablerow.asp) | Controls a <tr> HTML element |
| [HtmlTextArea](http://www.w3schools.com/aspnet/control_htmltextarea.asp) | Controls a <textarea> HTML element |

[« Previous](http://www.w3schools.com/aspnet/aspnet_examples.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_refwebcontrols.asp)

### WEB HOSTING

[UK Reseller Hosting](https://www.heartinternet.uk/?utm_source=w3schools&utm_medium=cpc&utm_campaign=w3schools%20text%20link)

### WEB BUILDING

[FREE Website BUILDER](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink1_templatesnr-a-d-e-w3) [Free HTML5 Templates](http://www.wix.com/eteamhtml/templatesnr-a-d-e-w3?utm_campaign=ma_w3schools.com&experiment_id=ma_w3schools.comlink2_freehtml5templates_templatesnr-a-d-e-w3)

### W3SCHOOLS EXAMS

[HTML, CSS, JavaScript, PHP, jQuery, and XML Certifications](http://www.w3schools.com/cert/default.asp)

### SHARE THIS PAGE

### COLOR PICKER

[](http://www.w3schools.com/tags/ref_colorpicker.asp)

# ASP.NET Web Forms - Server Controls

[« Previous](http://www.w3schools.com/aspnet/aspnet_refhtmlcontrols.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_refvalidationcontrols.asp)

Web server controls are special ASP.NET tags understood by the server.

## Web Server Controls

Like HTML server controls, Web server controls are also created on the server and they require a runat="server" attribute to work. However, Web server controls do not necessarily map to any existing HTML elements and they may represent more complex elements.

The syntax for creating a Web server control is:

<asp:control\_name id="some\_id" runat="server" />

|  |  |
| --- | --- |
| **Web Server Control** | **Description** |
| [AdRotator](http://www.w3schools.com/aspnet/control_adrotator.asp) | Displays a sequence of images |
| [Button](http://www.w3schools.com/aspnet/control_button.asp) | Displays a push button |
| [Calendar](http://www.w3schools.com/aspnet/control_calendar.asp) | Displays a calendar |
| [CalendarDay](http://www.w3schools.com/aspnet/control_calendarday.asp) | A day in a calendar control |
| [CheckBox](http://www.w3schools.com/aspnet/control_checkbox.asp) | Displays a check box |
| [CheckBoxList](http://www.w3schools.com/aspnet/control_checkboxlist.asp) | Creates a multi-selection check box group |
| DataGrid | Displays fields of a data source in a grid |
| DataList | Displays items from a data source by using templates |
| [DropDownList](http://www.w3schools.com/aspnet/control_dropdownlist.asp) | Creates a drop-down list |
| [HyperLink](http://www.w3schools.com/aspnet/control_hyperlink.asp) | Creates a hyperlink |
| [Image](http://www.w3schools.com/aspnet/control_image.asp) | Displays an image |
| [ImageButton](http://www.w3schools.com/aspnet/control_imagebutton.asp) | Displays a clickable image |
| [Label](http://www.w3schools.com/aspnet/control_label.asp) | Displays static content which is programmable (lets you apply styles to its content) |
| [LinkButton](http://www.w3schools.com/aspnet/control_linkbutton.asp) | Creates a hyperlink button |
| [ListBox](http://www.w3schools.com/aspnet/control_listbox.asp) | Creates a single- or multi-selection drop-down list |
| [ListItem](http://www.w3schools.com/aspnet/control_listitem.asp) | Creates an item in a list |
| [Literal](http://www.w3schools.com/aspnet/control_literal.asp) | Displays static content which is programmable(does not let you apply styles to its content) |
| [Panel](http://www.w3schools.com/aspnet/control_panel.asp) | Provides a container for other controls |
| [PlaceHolder](http://www.w3schools.com/aspnet/control_placeholder.asp) | Reserves space for controls added by code |
| [RadioButton](http://www.w3schools.com/aspnet/control_radiobutton.asp) | Creates a radio button |
| [RadioButtonList](http://www.w3schools.com/aspnet/control_radiobuttonlist.asp) | Creates a group of radio buttons |
| [BulletedList](http://www.w3schools.com/aspnet/control_bulletedlist.asp) | Creates a list in bullet format |
| Repeater | Displays a repeated list of items bound to the control |
| [Style](http://www.w3schools.com/aspnet/control_style.asp) | Sets the style of controls |
| [Table](http://www.w3schools.com/aspnet/control_table.asp) | Creates a table |
| [TableCell](http://www.w3schools.com/aspnet/control_tablecell.asp) | Creates a table cell |
| [TableRow](http://www.w3schools.com/aspnet/control_tablerow.asp) | Creates a table row |
| [TextBox](http://www.w3schools.com/aspnet/control_textbox.asp) | Creates a text box |
| [Xml](http://www.w3schools.com/aspnet/control_xml.asp) | Displays an XML file or the results of an XSL transform |

[« Previous](http://www.w3schools.com/aspnet/aspnet_refhtmlcontrols.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/aspnet_refvalidationcontrols.asp)

# ASP.NET Web Forms - Validation Controls

[« Previous](http://www.w3schools.com/aspnet/aspnet_refwebcontrols.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/default.asp)

Validation server controls are used to validate user-input.

## Validation Server Controls

A Validation server control is used to validate the data of an input control. If the data does not pass validation, it will display an error message to the user.

The syntax for creating a Validation server control is:

<asp:control\_name id="some\_id" runat="server" />

|  |  |
| --- | --- |
| **Validation Server Control** | **Description** |
| [CompareValidator](http://www.w3schools.com/aspnet/control_comparevalidator.asp) | Compares the value of one input control to the value of another input control or to a fixed value |
| [CustomValidator](http://www.w3schools.com/aspnet/control_customvalidator.asp) | Allows you to write a method to handle the validation of the value entered |
| [RangeValidator](http://www.w3schools.com/aspnet/control_rangevalidator.asp) | Checks that the user enters a value that falls between two values |
| [RegularExpressionValidator](http://www.w3schools.com/aspnet/control_regularexpvalidator.asp) | Ensures that the value of an input control matches a specified pattern |
| [RequiredFieldValidator](http://www.w3schools.com/aspnet/control_reqfieldvalidator.asp) | Makes an input control a required field |
| [ValidationSummary](http://www.w3schools.com/aspnet/control_validationsummary.asp) | Displays a report of all validation errors occurred in a Web page |

[« Previous](http://www.w3schools.com/aspnet/aspnet_refwebcontrols.asp)

[Next Chapter »](http://www.w3schools.com/aspnet/default.asp)