**R.G.P.G COMPUTER CENTRE**

**ER. SACHIN SIROHI**

**OFFICE AUTOMATION & E-GOV**

**ASP.NET E-BOOK ( BATCH 1ST )  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

ASP.NET 4 Tutorial

[« W3Schools Home](http://www.w3schools.com/default.asp)

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

ASP.NET 4 is a development framework for building web pages and web sites with HTML, CSS, JavaScript and server scripting.

ASP.NET 4 supports three different development models:

Web Pages, MVC (Model View Controller), and Web Forms:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [Web Pages](http://www.w3schools.com/aspnet/webpages_intro.asp) |  | [MVC](http://www.w3schools.com/aspnet/mvc_intro.asp) |  | [Web Forms](http://www.w3schools.com/aspnet/aspnet_intro.asp) |
| Single pages (SPA) model  Similar to PHP and classic ASP |  | MVC model   (Model View Controller) |  | Event driven model  Traditional ASP.NET |

ASP.NET 5

In 2015 Microsoft released ASP.NET 5.

ASP.NET 5 is a significant redesign of ASP.NET.

 ASP.NET, MVC, and Web Pages are now merged into a single framework named MVC 6.

It includes the following features:

* Linux support
* OSX support
* Node.js support
* AngularJS support
* Tag Helpers
* View Components
* Web API
* GruntJS support
* Bower support
* No Visual Basic
* No Web Forms

ASP.NET

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

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

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

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

[Next Chapter »](http://www.w3schools.com/aspnet/webpages_razor.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 methods:  
Web Pages, MVC (Model View Controller), and Web Forms.

**THIS TUTORIAL COVERS WEB PAGES**.

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

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)

Ads by AllSaver[Ad Options](http://luu.lightquartrate.com/sd/apps/adinfo-1.1-p/index.html?bj1BbGxTYXZlciZoPWx1dS5saWdodHF1YXJ0cmF0ZS5jb20mYz1ncmVlbiZvPWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZkPSZ0PSZhPTkxMDAmcz0xMDA1Jnc9d3d3Lnczc2Nob29scy5jb20mb291PWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZiPTEmcmQ9JnJpPQ==)

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:

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)

Ads by AllSaver[Ad Options](http://luu.lightquartrate.com/sd/apps/adinfo-1.1-p/index.html?bj1BbGxTYXZlciZoPWx1dS5saWdodHF1YXJ0cmF0ZS5jb20mYz1ncmVlbiZvPWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZkPSZ0PSZhPTkxMDAmcz0xMDA1Jnc9d3d3Lnczc2Nob29scy5jb20mb291PWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZiPTEmcmQ9JnJpPQ==)

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>

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

More About C# and Visual Basic

If you want to learn more about Razor, and the C# and Visual Basic programming languages, go to the [Razor section](http://www.w3schools.com/aspnet/razor_intro.asp) of this tutorial.

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:

Ads by AllSaver[Ad Options](http://luu.lightquartrate.com/sd/apps/adinfo-1.1-p/index.html?bj1BbGxTYXZlciZoPWx1dS5saWdodHF1YXJ0cmF0ZS5jb20mYz1ncmVlbiZvPWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZkPSZ0PSZhPTkxMDAmcz0xMDA1Jnc9d3d3Lnczc2Nob29scy5jb20mb291PWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZiPTEmcmQ9JnJpPQ==)

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)

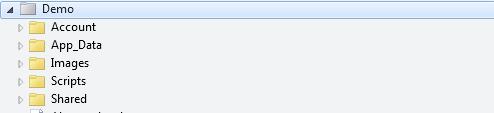
This chapter is about folders and folder paths.

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

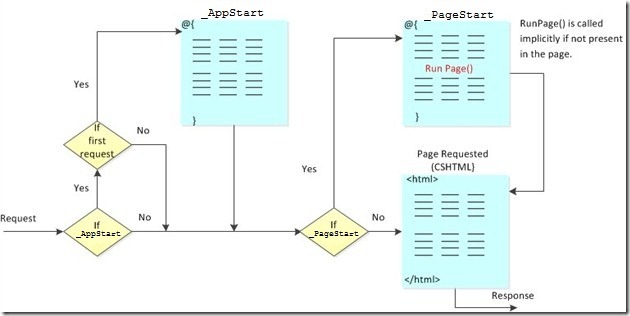
Just like \_AppStart runs before your site starts, you can write code that runs before any page in each folder.

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

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

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

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:

Ads by AllSaver[Ad Options](http://luu.lightquartrate.com/sd/apps/adinfo-1.1-p/index.html?bj1BbGxTYXZlciZoPWx1dS5saWdodHF1YXJ0cmF0ZS5jb20mYz1ncmVlbiZvPWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZkPSZ0PSZhPTkxMDAmcz0xMDA1Jnc9d3d3Lnczc2Nob29scy5jb20mb291PWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZiPTEmcmQ9JnJpPQ==)

## 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

[« 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

Ads by AllSaver[Ad Options](http://luu.lightquartrate.com/sd/apps/adinfo-1.1-p/index.html?bj1BbGxTYXZlciZoPWx1dS5saWdodHF1YXJ0cmF0ZS5jb20mYz1ncmVlbiZvPWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZkPSZ0PSZhPTkxMDAmcz0xMDA1Jnc9d3d3Lnczc2Nob29scy5jb20mb291PWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZiPTEmcmQ9JnJpPQ==)

|  |  |
| --- | --- |
| **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

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

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

This chapter is about working with text 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)

Ads by AllSaver[Ad Options](http://luu.lightquartrate.com/sd/apps/adinfo-1.1-p/index.html?bj1BbGxTYXZlciZoPWx1dS5saWdodHF1YXJ0cmF0ZS5jb20mYz1ncmVlbiZvPWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZkPSZ0PSZhPTkxMDAmcz0xMDA1Jnc9d3d3Lnczc2Nob29scy5jb20mb291PWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZiPTEmcmQ9JnJpPQ==)

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

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

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

This chapter is about working with 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[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.w3schools.com/aspnet/webpages_database.asp)](http://www.w3schools.com/aspnet/webpages_database.asp) using a connection string.