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.

ASP.NET Web Pages - Adding Razor 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

## 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

# ASP.NET Web Pages - Page Layout

## 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 a Layout Page

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 (**@{Layout="Layout.cshtml";}**)**.

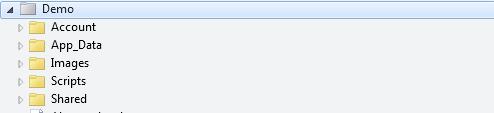
## Hiding Sensitive Information

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".

# ASP.NET Web Pages - Folders

## 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)

## URLs and Paths

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 |

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);

## 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

## 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.

## 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.

# 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)

# ASP.NET Razor - Markup

## What is Razor?

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

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

## 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

## 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:

# ASP.NET MVC Tutorial

## 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.

# ASP.NET MVC - Application Folders

## The App\_Data Folder

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

## The Content Folder

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

## 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".

## The Models Folder

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

## 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.

## The Scripts Folder

The Scripts folder stores the JavaScript files of the application.

## 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.

# ASP.NET MVC - Views

## 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.

## The Index File

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

# ASP.NET MVC - Models

## MVC Models

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

# ASP.NET MVC - HTML Helpers

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>

## 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 NAVIGATION**

## Web Site Navigation

Maintaining the menu of a large web site is difficult and time consuming.

In ASP.NET the menu can be stored in a file to make it easier to maintain. This file is normally called **web.sitemap**, and is stored in the root directory of the web.

In addition, ASP.NET has three new navigation controls:

* Dynamic menus
* TreeViews
* Site Map Path