# ASP.NET Razor - C# Code Syntax

## Index

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

## Razor Comments

```
@*  A one-line code comment. *@

@*
    This is a multiline code comment.
    It can continue for any number of lines.
*@


@{
    @* This is a comment. *@
    var theVar = 17;
}
```

back to top

## Single statement block

```
@{ var myMessage = "Hello World"; }
```

## Inline expression

```
@{ var myMessage = "Hello World"; }
```

## 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> }
```

back to top

## Store String

```
/* A string is a sequence of characters that are treated as text. To specify a string, you enclose it in double quota

@{ var welcomeMessage = "Welcome, new members!"; }
<p>@welcomeMessage</p>
```

## Store Date

```
@{ var year = DateTime.Now.Year; }
```

back to top

## Read User Input

```
@{
var totalMessage = "";
if(IsPost)
    {
    var num1 = Request["text1"];
    var num2 = Request["text2"];
    var total = num1.AsInt() + num2.AsInt();
    totalMessage = "Total = " + total;
}
}
```

back to top

## Variables

```
@{
    // Assigning a string to a variable.
    var greeting = "Welcome!";

    // Assigning a number to a variable.
    var theCount = 3;

    // Assigning an expression to a variable.
    var monthlyTotal = theCount + 5;

    // Assigning a date value to a variable.
    var today = DateTime.Today;

    // Assigning the current page's URL to a variable.
    var myPath = this.Request.Url;
    // Declaring variables using explicit data types.
    string name = "Joe";
    int count = 5;
    DateTime tomorrow = DateTime.Now.AddDays(1);
}
```

### Display Variables

```
@{
    // Embedding the value of a variable into HTML markup.
    <p>@greeting, friends!</p>

    // Using variables as part of an inline expression.
    <p>The predicted annual total is: @( monthlyTotal * 12)</p>

    // Displaying the page URL with a variable.
    <p>The URL to this page is: @myPath</p>
}
```

back to top

## Convert Data Types

| Method | Description | Examples |
|---|---|---|
| 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();` |

### Coverting Data Types example

```
@{
    var total = 0;

    if(IsPost) {
        // Retrieve the numbers that the user entered.
        var num1 = Request["text1"];
        var num2 = Request["text2"];
        // Convert the entered strings into integers numbers and add.
        total = num1.AsInt() + num2.AsInt();
    }
}
```

back to top

## Loops

### Standard Loop

```
@for (var i = 0; i < people.Length; i++)
{
    var person = people[i];
    <text>Name: @person.Name</text>
}
```

### ForEach Loops

```
<ul>
@foreach (var myItem in Request.ServerVariables)
{
    <li>@myItem</li>
}
</ul>
```

### While Loops

```
@{
    var countNum = 0;
    while (countNum < 50)
    {
        countNum += 1;
        <p>Line #@countNum: </p>
    }
}
```

back to top

### Arrays

```
@{
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>
```

back to top

## Conditionals

### If

```
@{
  var showToday = true;
  if(showToday)
  {
    @DateTime.Today;
  }
}
```

### If Else

```
@{
  var showToday = false;
  if(showToday)
  {
    @DateTime.Today;
  }
  else
  {
    <text>Sorry!</text>
  }
}
```

### Else If

```
@{
    var theBalance = 4.99;
    if(theBalance == 0)
    {
        <p>You have a zero balance.</p>
    }
    else if (theBalance  > 0 && theBalance <= 5)
    {
        <p>Your balance of $@theBalance is very low.</p>
    }
    else
    {
        <p>Your balance is: $@theBalance</p>
    }
}
```

## Switch Statement

```
@{
    var weekday = "Wednesday";
    var greeting = "";

    switch(weekday)
    {
        case "Monday":
            greeting = "Ok, it's a marvelous Monday";
            break;
        case "Tuesday":
            greeting = "It's a tremendous Tuesday";
            break;
        case "Wednesday":
            greeting = "Wild Wednesday is here!";
            break;
        default:
            greeting = "It's some other day, oh well.";
            break;
    }

    <p>Since it is @weekday, the message for today is: @greeting</p>
}
```

## Try Catch Finally

```
@try
{
    throw new InvalidOperationException("You did something invalid.");
}
catch (Exception ex)
{
    <p>The exception message: @ex.Message</p>
}
finally
{
    <p>The finally statement.</p>
}
```

back to top

## Using

```
/* The @using directive adds the C# using directive to the generated view:*/

@using System.IO
@{
    var dir = Directory.GetCurrentDirectory();
}
<p>@dir</p>
```

back to top

## Models View

```
// The @model directive specifies the type of the model passed to a view:

@model TypeNameOfModel
```

## Access Model

```
<div>The Login Email: @Model.Email</div>
```

## Dependency Injection

```
@inject +ServiceName
```

back to top

## Add Functions

```
@functions {
    public string GetHello()
    {
        return "Hello";
    }
}

<div>From method: @GetHello()</div>
```

back to top

## Create Templates

Create a class

```
public class Pet
{
    public string Name { get; set; }
}
```

create a `.cshtml` page

```
@{
    Func<dynamic, object> petTemplate = @<p>You have a pet named @item.Name.</p>;

    var pets = new List<Pet>
    {
        new Pet { Name = "Rin Tin Tin" },
        new Pet { Name = "Mr. Bigglesworth" },
        new Pet { Name = "K-9" }
    };

     <!-- The template is rendered with pets supplied by a foreach statement: -->

     @foreach (var pet in pets)
{
    @petTemplate2(pet)
}
}
```

Rendered output

```
<p>You have a pet named <strong>Rin Tin Tin</strong>.</p>
<p>You have a pet named <strong>Mr. Bigglesworth</strong>.</p>
<p>You have a pet named <strong>K-9</strong>.</p>
```

back to top

## Conditional Attributes

```
@{
    string divStyle = null;
    if(Request.QueryString["style"] != null)
    {
        divStyle = "background-color: yellow;";
    }
}
<div style="@divStyle">Hello, world!</div>
```

back to top

## Forms

```
<form method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Movie.Title" class="control-label"></label>
                <input asp-for="Movie.Title" class="form-control" />
                <span asp-validation-for="Movie.Title" class="text-danger"></span>
            </div>
            <div class="form-group">
```

```
                <label asp-for="Movie.ReleaseDate" class="control-label"></label>
                <input asp-for="Movie.ReleaseDate" class="form-control" />
                <span asp-validation-for="Movie.ReleaseDate" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
</form>
```

back to top

## Add Partials

```
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

back to top

## Add link to a page

```
<div>
    <a asp-page="./Index">Back to List</a>
</div>
```

back to top

## Loop through a list and output

```
<tbody>
        @foreach (var item in Model.Movie)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Title)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.ReleaseDate)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Genre)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Price)
                </td>
                <td>
                    <a asp-page="./Edit" asp-route-id="@item.ID">Edit</a> |
                    <a asp-page="./Details" asp-route-id="@item.ID">Details</a> |
                    <a asp-page="./Delete" asp-route-id="@item.ID">Delete</a>
                </td>
            </tr>
        }
    </tbody>
```

back to top