

***Entity Framework***

**Lab Guides**

|  |  |
| --- | --- |
| Document Code | 25e-BM/HR/HDCV/FSOFT |
| Version | 1.1 |
| Effective Date | 20/11/2012 |

**Hanoi, 07/2019**

RECORD OF CHANGES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Effective Date | Change Description | Reason | Reviewer | Approver |
|  | 01/Oct/2018 | TuTB create new | Draft |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Contents

[Lab: Programming Using the Razor Syntax 4](#_Toc13570999)

[Objectives: 4](#_Toc13571000)

[Prerequisites: 4](#_Toc13571001)

[Problem Description: 4](#_Toc13571002)

[Guidelines: 4](#_Toc13571003)

[Rule 1: Add code to a page using the @ character 4](#_Toc13571004)

[Rule 2: enclose code blocks in braces 5](#_Toc13571005)

[Rule 3: inside a block, end each code statement with a semicolon 5](#_Toc13571006)

[Rule 4: use variables to store values 6](#_Toc13571007)

[Rule 5: enclose literal string values in double quotation marks 6](#_Toc13571008)

[Rule 6: Code is case sensitive 7](#_Toc13571009)

[Rule 7: Much of your coding involves objects 8](#_Toc13571010)

[Rule 8: can write code that makes decisions 8](#_Toc13571011)

|  |  |
| --- | --- |
|  | **CODE: NWEB.M.L006**  **TYPE: MEDIUM**  **LOC: 300**  **DURATION: 60 MINUTES** |

# Lab: Programming Using the Razor Syntax

Objectives:

* Overview of programming with ASP.NET Web Pages using the Razor syntax. ASP.NET is Microsoft's technology for running dynamic web pages on web servers. This articles focuses on using the C# programming language.

Prerequisites:

* Complete lab NWEB.S.L001.

Problem Description:

* The top 8 programming tips for getting started with programming ASP.NET Web Pages using Razor syntax.
* Basic programming concepts you'll need.
* What ASP.NET server code and the Razor syntax is all about.

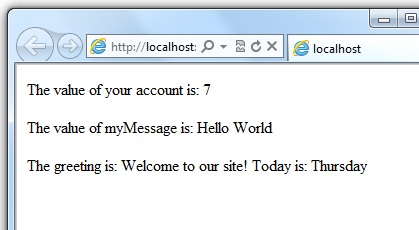
Guidelines:

### Rule 1: Add code to a page using the @ character

The @ character starts inline expressions, single statement blocks, and multi-statement blocks:

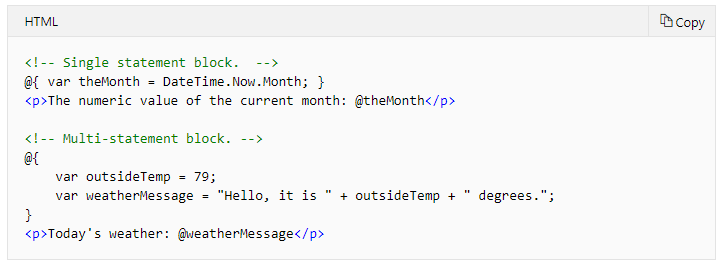


This is what these statements look like when the page runs in a browser:

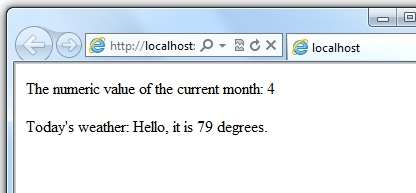


### Rule 2: enclose code blocks in braces

A *code block* includes one or more code statements and is enclosed in braces.



The result displayed in a browser:



### Rule 3: inside a block, end each code statement with a semicolon

Inside a code block, each complete code statement must end with a semicolon. Inline expressions don't end with a semicolon.

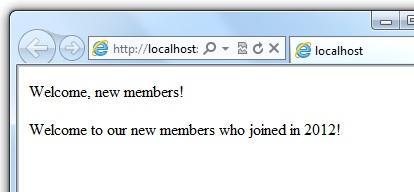


### Rule 4: use variables to store values

You can store values in a variable, including strings, numbers, and dates, etc. You create a new variable using the var keyword. You can insert variable values directly in a page using @.

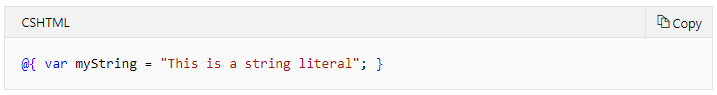


The result displayed in a browser:

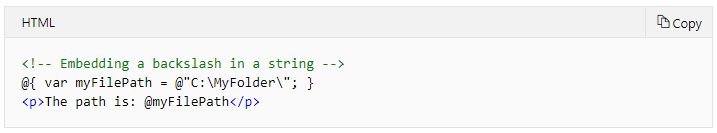


### Rule 5: enclose literal string values in double quotation marks

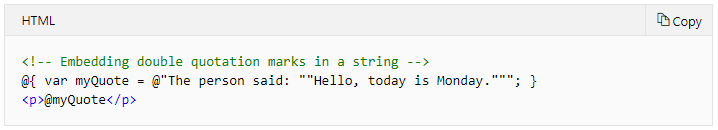
A string is a sequence of characters that are treated as text. To specify a string, you enclose it in double quotation marks:



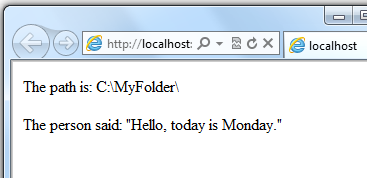
If the string that you want to display contains a backslash character ( \ ) or double quotation marks ( " ), use a verbatim string literal that's prefixed with the @ operator. (In C#, the \ character has special meaning unless you use a verbatim string literal.)



To embed double quotation marks, use a verbatim string literal and repeat the quotation marks:

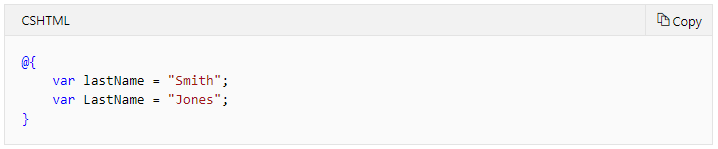


Here's the result of using both of these examples in a page:



### Rule 6: Code is case sensitive

In C#, keywords (like var, true, and if) and variable names are case sensitive. The following lines of code create two different variables, lastName and LastName.



If you declare a variable as var lastName = "Smith"; and if you try to reference that variable in your page as @LastName, an error results because LastName won't be recognized.

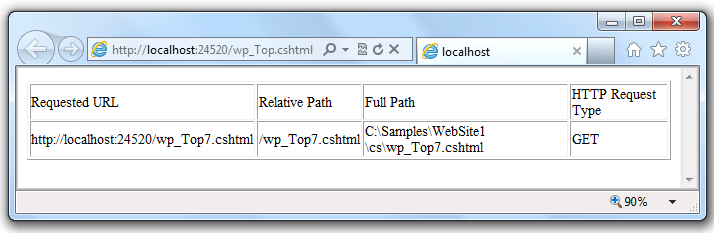
### Rule 7: Much of your coding involves objects

An object represents a thing that you can program with — a page, a text box, a file, an image, a web request, an email message, a customer record (database row), etc. Objects have properties that describe their characteristics and that you can read or change — a text box object has a Text property (among others), a request object has a Url property, an email message has a From property, and a customer object has a FirstName property. Objects also have methods that are the "verbs" they can perform. Examples include a file object's Save method, an image object's Rotate method, and an email object's Send method.

You'll often work with the Request object, which gives you information like the values of text boxes (form fields) on the page, what type of browser made the request, the URL of the page, the user identity, etc. The following example shows how to access properties of the Request object and how to call the MapPath method of the Request object, which gives you the absolute path of the page on the server:



The result displayed in a browser:



### Rule 8: can write code that makes decisions

A key feature of dynamic web pages is that you can determine what to do based on conditions. The most common way to do this is with the if statement (and optional else statement).



The statement if(IsPost) is a shorthand way of writing if(IsPost == true). Along with if statements, there are a variety of ways to test conditions, repeat blocks of code, and so on, which are described later in this article.

The result displayed in a browser (after clicking Submit):

