**OW TO DEFINE VARIABLES AND DATATYPES IN RAZOR MARKUP**

**There are two types of variable initialization in Razor Markup.**

1. **Implicitly Typed Variable :**Using **var** keyword you can initialize any type of variable in razor syntax. It is known as **Implicitly-typed** variable. You must initialize the variable with value if it is var variable.  
     
   **Example:**  
   1. var s; @\* **Wrong** way. **Rasie** **Error** \*@
   2. var s = "Hello"; @\* **Correct** **Way** \*@

1. **Explicitly Typed Variable :**This variable is defined with data typed as **int**, **string**, **float**etc.

**PROGRAMMING EXAMPLE**

1. <html>
2. <head>
3. <title></title>
4. </head>
5. <body>
6. @{
7. @\* //Using var Keyword – Implicitly Typed \*@
8. var message = "Add Two Numbers";
9. var num1 = 5;
10. var sum = num1 + num1;
11. <h5>@message</h5>
12. <h5>@num1 + @num1 = @sum</h5>
14. @\* //Using Explicit Data Type \*@
15. int i = 10;
16. int result = i \* i;
17. <h5>Using Explicit Data Types</h5>
18. <h5>@i x @i = @result</h5>
19. }
20. </body>
21. </html>

**Output**

Add Two Numbers  
5 + 5 = 10  
Using Explicit Data Types  
10 x 10 = 100

**DATATYPES**

You can use all the C# or VB data types in Razor Syntax as the way you use in programming. But the process of converting data type is different and you must know the concept of conversion in Razor markup.

**CONVERTING DATA TYPES**

**1.** **Converting Integer** – Convert string that has value like **integer**, example "239" or "45". There are two methods for converting int. **IsInt()** Method and **AsInt()** Method.  
 **IsInt()** – It checks whether provided variable has integer value or not.  
 **AsInt()** – It converts variable into integer datatypes.

**Example Converting Integer**

1. <html>
2. <head>
3. <title>Converting Integer</title>
4. </head>
5. <body>
6. @{
7. var number = "525";
8. int num;
9. @\* //This line will raise error because you cannot store string into int variable \*@
10. @\* //num = number; \*@
11. if(number.IsInt()==true)
12. {
13. num=number.AsInt();
14. <h5>@num is Integer Value</h5>
15. }
16. else
17. {
18. <h5>@number is Not Integer</h5>
19. }
20. }
21. </body>
22. </html>

**Output**

525 is Integer Value

**2. Converting Bool** – Converts string **"true"** and **"false"** into **Boolean** type. There are two methods for converting Bool. **IsBool()** Method and **AsBool()** Method.  
 **IsBool()**– It checks whether provided variable has **Boolean** value or not.  
 **AsBool()** – It converts variable into **Boolean** datatypes.

**Example Converting Bool**

1. <html>
2. <head>
3. <title>Converting Bool</title>
4. </head>
5. <body>
6. @{
7. var b = "false";
8. Boolean bln;
9. if(b.IsBool()==true)
10. {
11. bln=b.AsBool();
12. <h5>@bln</h5>
13. }
14. }
15. </body>
16. </html>

**Output**

False

**3. Converting Float** – Converts a string that has a decimal value like "2.5" or "6.123". There are two methods for converting **Float**. **IsFloat()** Method and **AsFloat()** Method.  
 **IsFloat()** – It checks whether provided variable has Float value or not.  
 **AsFloat()** – It converts variable into Float datatypes.

**Example Converting Float**

1. <html>
2. <head>
3. <title>Converting Float</title>
4. </head>
5. <body>
6. @{
7. var v = "2.63";
8. float f;
9. if(v.IsFloat()==true)
10. {
11. f=v.AsFloat();
12. <h5>@f</h5>
13. }
14. }
15. </body>
16. </html>

**Output**

2.63

**4.** **Converting Decimal**- Converts string that has a decimal value like "2.6" or "6.1". There are two methods for converting **Decimal**. **IsDecimal()** Method and **AsDecimal()** Method.  
 **IsDecimal()** – It checks whether provided variable has Decimal value or not.  
 **AsDecimal()** – It converts variable into Decimal datatypes.

**Example**

1. <html>
2. <head>
3. <title>Converting Decimal</title>
4. </head>
5. <body>
6. @{
7. var v = "2.63";
8. decimal d;
9. if(v.IsDecimal()==true)
10. {
11. d=v.AsDecimal();
12. <h5>@d</h5>
13. }
14. }
15. </body>
16. </html>

**Output**

2.63

**5.** **Converting DateTime** – Converts string that has value in **date time** format. There are two methods for converting **DateTime**. **IsDateTime()** Method and **AsDateTime()** Method.  
 **IsDateTime()** – It checks whether provided variable has **DateTime** value or not.  
 **AsDateTime()** – It converts variable into **DateTime**datatypes.

**Example**

1. <html>
2. <head>
3. <title>Converting DateType</title>
4. </head>
5. <body>
6. @{
7. var v = "24/08/2016";
8. DateTime d;
9. if(v.IsDateTime()==true)
10. {
11. d=v.AsDateTime();
12. <h5>Date is : @d</h5>
13. }
14. }
15. </body>
16. </html>

**Output**

Date is : 24-08-2016 AM 12:00:00

**6.** **Converting String** – **ToString()** method converts any data type into string.

**Example**

1. <html>
2. <head>
3. <title>Converting String</title>
4. </head>
5. <body>
6. @{
7. var num = 2342;
8. DateTime d=DateTime.Now;
9. bool b = true;
11. <h5>@num.ToString() is String Now.</h5>
12. <h5>@d.ToString() is String Now.</h5>
13. <h5>@b.ToString() is String Now.</h5>
14. }
15. </body>
16. </html>

**Output**

2342 is String Now.  
24-08-2016 PM 08:21:00 is String Now.  
True is String Now.

**SUMMARY**

In this chapter you learned to **define variable** and **converting data types** in Razor Syntax. In the next chapter you will learn to use Operators in Razor Syntax

### HOW TO USE OPERATOR IN RAZOR MARKUP

Razor Markup supports many **operators** of C# and VB.Net. There is no special rule for using operators in razor markup. You can simply use operators same as you use in c# programming or vb programming. Most common operators that is used in Razor markup is follows.

**1.** **Math Operators** : It is **+ (Plus)**, **- (Minus)** , **\* (Multiplication)**, **/ (Divide)**. It is used for mathematical calculation.

**Example**

1. <html>
2. <head>
3. <title>Math Operators</title>
4. </head>
5. <body>
6. @{
7. int num1, num2, plus, minus, multi, divide;
8. num1 = 45;
9. num2 = 9;
10. plus = num1 + num2;
11. minus = num1 - num2;
12. multi = num1 \* num2;
13. divide = num1 / num2;
14. }
15. <h5>Addition : @num1 + @num2 = @plus</h5>
16. <h5>Subtraction : @num1 - @num2 = @minus</h5>
17. <h5>Multiplication : @num1 \* @num2 = @multi</h5>
18. <h5>Divide : @num1 / @num2 = @divide</h5>
19. </body>
20. </html>

**Output**

Addition : 45 + 9 = 54  
Subtraction : 45 - 9 = 36  
Multiplication : 45 \* 9 = 405  
Divide : 45 / 9 = 5

**2. Assignment Operator** : It is **=(equal to)**. It is used for assigning right side value to left side variable.

**Example**  
**int num = 55;**

**3. Equality Operator** : It is **== (Equality)**. It returns true if left hand side and right hand side value are equal otherwise return false.

**Example**

1. <html>
2. <head>
3. <title>Equality Operators</title>
4. </head>
5. <body>
6. @{
7. int[] num1={12,52,66,92,28};
8. int[] num2={73,12,98,23,52};
10. for(int i=0; i<5; i++)
11. {
12. for(int loop=0; loop<5; loop++)
13. {
14. if(num1[i]==num2[loop])
15. {
16. <h5>Match Found : @num1[i]</h5>
17. }
18. }
19. }
20. }
21. </body>
22. </html>

**Output**

Match Found : 12  
Match Found : 52

**4. Inequality Operator** : It is **!= (Inequality)**. It returns true if both value are not equal.

**Example**

1. <html>
2. <head>
3. <title>Inequality Operator </title>
4. </head>
5. <body>
6. @{
7. int num1 = 5, num2 = 10;
8. if(num1 != num2)
9. {
10. <h5>Returns True</h5>
11. }
12. }
13. </body>
14. </html>

**Output**

Returns True

**5. Comparison Operator** : It is **<, >, <=, =>**. It is used for comparing two integer or decimal values.

**Example**

1. <html>
2. <head>
3. <title>Comparison Operator </title>
4. </head>
5. <body>
6. @{
7. int num1 = 5, num2 = 10;
8. if(num1 < num2)
9. {
10. <h5>@num1 is less than @num2</h5>
11. }
12. else
13. {
14. <h5>@num1 is greater than @num2</h5>
15. }
16. }
17. </body>
18. </html>

**Output**

5 is less than 10

**6. Concatenation Operator** : It is **+ (Plus)**. It is used for joining two strings.

**Example**

1. <html>
2. <head>
3. <title>Concatenation Operator</title>
4. </head>
5. <body>
6. @{
7. string msg1 = "This is Razor Tutorial.";
8. string msg2 = "And you are learning at CompleteCsharpTutorial.com";
9. string finalmsg= msg1 + " " + msg2;
10. <h3>@finalmsg</h3>
11. }
12. </body>
13. </html>

**Output**

This is Razor Tutorial. And you are learning at CompleteCsharpTutorial.com

**7. Increment and Decrement Operators** : It is **++ or += and -- or -=**. It is used for incrementing or decrementing integer value.

**Example**

1. <html>
2. <head>
3. <title>Increment and Decrement Operators</title>
4. </head>
5. <body>
6. @{
7. int inc = 5;
8. <h3>Increment of @inc is @{inc++;} : @inc</h3>
9. <h3>Decrement of @inc is @{inc--;} : @inc</h3>
11. <h3>Increment of @inc is @{inc+=1;} : @inc</h3>
12. <h3>Decrement of @inc is @{inc-=1;} : @inc</h3>
13. }
14. </body>
15. </html>

**Output**

Increment of 5 is : 6  
Decrement of 6 is : 5  
Increment of 5 is : 6  
Decrement of 6 is : 5

**8. Dot Operator** : It is **. (Dot)**. It is used to distinguish objects and their properties and methods.

**Example**

1. <html>
2. <head>
3. <title>Dot Operator</title>
4. </head>
5. <body>
6. @{
7. var num = "55";
8. int i;
9. i=num.AsInt();
10. <h3>@i</h3>
11. <h3>Browser Name : @Request.Browser.Id</h3>
12. }
13. </body>
14. </html>

**Output**

55 Browser Name : chrome

**9. Parentheses** : It is **()**. It is used to group expression and to pass parameters to methods.

**Example**

1. <html>
2. <head>
3. <title>Parentheses</title>
4. </head>
5. <body>
6. @{
7. int num1 = 5, num2 = 10;
8. if((num1 + num2) == 15)
9. {
10. <h2>Addition is 15</h2>
11. }
12. }
13. </body>
14. </html>

**Output**

Addition is 15

**10. Bracket** : It is **[]**. It is used for accessing value in arrays of collections.

**Example**

1. <html>
2. <head>
3. <title>Bracket</title>
4. </head>
5. <body>
6. @{
7. int[] arr={12,562,62,98,388};
8. <h3>@arr[3]</h3>
9. }
10. </body>
11. </html>

**Output**

98

**11. Not Operator** : It is **! (Not)**. It **reverse**true into false and false into true.

**Example**

1. <html>
2. <head>
3. <title>Not Operator</title>
4. </head>
5. <body>
6. @{
7. int num1 = 5, num2 = 10;
8. if(!(num1 == num2))
9. {
10. <h3>However conditions are false still this message appears because you have used not operator and it reverse the condition.</h3>
11. }
12. }
13. </body>
14. </html>

**Output**

However conditions are false still this message appears because you have used not operator and it reverse the condition.

**12. Logical Operator** : It is **&& (AND)** and **|| (OR)**. It is used to link conditions together and helps to make logic much better.**&&** - Returns true if all the specified conditions are true otherwise returns false.  
**||** - Returns true if one or all the conditions are true.

**Example**

1. <html>
2. <head>
3. <title>Logical Operator </title>
4. </head>
5. <body>
6. @{
7. string username = "Steven";
8. string password = "Svn78\*5";
10. if(username == "Steven" && password == "Svn78\*")
11. {
12. <h3>Login Successful</h3>
13. }
14. else if(username == "Steven" || password == "12345")
15. {
16. <h3>Hello Steven, you have entered wrong password.</h3>
17. }
18. else
19. {
20. <h3>Unauthorized Login</h3>
21. }
22. }
23. </body>
24. </html>

**Output**

Hello Steven, you have entered wrong password.

### SUMMARY

In this chapter you learned to use different types of operator with programming example that is used in razor markup. In the next chapter you will learn how to use Conditional Statements in Razor Markup.

# Razor - Loop Statements With Programming Example

**In this chapter you will learn**

* How to use looping statements inside razor syntax.
* Programming example

You can use looping statements as **for loop**, **while loop**, **do while loop** or **foreach loop** inside razor syntax as follow:

### FOR LOOP

1. <html lang="en">
2. <head>
3. <meta charset="utf-8" />
4. <title></title>
5. </head>
6. <body>
7. @{
8. for(int i = 0; i < 5; i++)
9. {
10. <h3>Prints 5 times</h3>
11. }
12. }
13. </body>
14. </html>

**Output**

**Prints 5 times  
Prints 5 times  
Prints 5 times  
Prints 5 times  
Prints 5 times**

### WHILE LOOP

1. <html lang="en">
2. <head>
3. <meta charset="utf-8" />
4. <title></title>
5. </head>
6. <body>
7. @{
8. int i=0;
9. while(i<5)
10. {
11. <h3>This while loop print 5 times</h3>
12. i++;
13. }
14. }
15. </body>
16. </html>

**Output**

**This while loop print 5 times  
This while loop print 5 times  
This while loop print 5 times  
This while loop print 5 times  
This while loop print 5 times**

### FOREACH LOOP

1. <html lang="en">
2. <head>
3. <meta charset="utf-8" />
4. <title></title>
5. </head>
6. <body>
7. @{
8. int[] arr={1,3,5,7,11,13,17,19};
9. foreach(int x in arr)
10. {
11. <span>@x, </span>
12. }
13. }
14. </body>
15. </html>

**Output**

1, 3, 5, 7, 11, 13, 17, 19,

### SUMMARY

In this chapter you learned to use looping statements like for loop, while loop, foreach loop in razor syntax. In the next chapter you will learn array with razor.

# How To Use Collection Inside Razor Syntax

**In this chapter you will learn**

* How to use Collection inside Razor Syntax
* Programming Example

### HOW TO USE COLLECTION INSIDE RAZOR SYNTAX

A **collection**is a group of objects of the same type. There are mainly two common collections are used in razor, Array and Dictionary. You have learned Array in previous chapter so we will focus on Dictionary in this chapter.

### DICTIONARY

A dictionary is a collection of key/value pairs and you can use it in programming as follow:

1. <html lang="en">
2. <head>
3. <meta charset="utf-8" />
4. <title></title>
5. </head>
6. <body>
7. @{
8. var result = new Dictionary<string, int>();
9. result.Add("Jack", 75);
10. result.Add("Steven", 80);
11. result.Add("Clark", 95);
12. }
13. <h3>Results of the students are:</h3>
14. <h3>Jack : @result["Jack"]</h3>
15. <h3>Steven : @result["Steven"]</h3>
16. <h3>Clark : @result["Clark"]</h3>
17. </body>
18. </html>

**Output**

Results of the students are:  
Jack : 75  
Steven : 80  
Clark : 95

### SUMMARY

In this chapter you learned to use **Dictionary**in razor syntax. You are free to use any types of collection in razor syntax. In the next chapter you will learn creating Methods with parameter in Razor Syntax.

# Handling Errors In Razor Syntax

**In this chapter you will learn**

* How to handle errors in Razor Syntax?
* Try Catch Finally
* Programming Example

### HOW TO HANDLE ERRORS IN RAZOR SYNTAX?

As you know razor allows you to full implementation of c# or vb so you can use exception handling inside razor syntax with try, catch, finally block.  
  
**Programming Example**

1. <html lang="en">
2. <head>
3. <meta charset="utf-8" />
4. <title></title>
5. </head>
6. <body>
7. @functions{
8. public int divide(int x, int y)
9. {
10. string errormessage;
11. try
12. {
13. return x/y;
14. }
15. catch(DivideByZeroException d)
16. {
17. errormessage = d.ToString();
18. WriteLiteral("<br />Cannot Divide by Zero Exception found <br /><br />");
19. WriteLiteral("Additional Info <br /><br />" + errormessage);
20. return 0;
21. }
22. }
23. }
24. <h3>Division of 10 / 5 is @divide(10,5)</h3>
25. <h3>Add of 12 + 0 is @divide(12,0)</h3>
26. </body>
27. </html>

**Output**

Division of 10 / 5 is 2  
Add of 12 + 0 is  
Cannot Divide by Zero Exception foundAdditional Info  
  
System.DivideByZeroException: Attempted to divide by zero. at ASP.\_Page\_index\_cshtml.divide(Int32 x, Int32 y) in c:\Users\Prashant\Documents\My Web Sites\EmptySite5\index.cshtml:line 190

### SUMMARY

In this chapter you learned how to use try catch exception handling in razor syntax. In the next chapter you will get some practice question to solve. You must solve them using programming logic. It will help you to gain better understanding of Razor Syntax.