Razor Syntax

Razor is one of the view engine supported in ASP.NET MVC. Razor allows you to write mix of HTML and server side code using C# or Visual Basic. Razor view with visual basic syntax has .vbhtml file extension and C# syntax has .cshtml file extension.

Razor syntax has following Characteristics:

- **Compact**: Razor syntax is compact which enables you to minimize number of characters and keystrokes required to write a code.
- **Easy to Learn**: Razor syntax is easy to learn where you can use your familiar language C# or Visual Basic.
- **Intellisense**: Razor syntax supports statement completion within Visual Studio.

Now, let's learn how to write razor code.

Inline expression

Start with @ symbol to write server side C# or VB code with Html code. For example, write @Variable_Name to display a value of a server side variable. For example, DateTime.Now returns a current date and time. So, write @DateTime.Now to display current datetime as shown below. A single line expression does not require a semicolon at the end of the expression.

C# Razor Syntax

```
<h1>Razor syntax demo</h1>

<h2>@DateTime.Now.ToShortDateString()</h2>
```

Output:

```
Razor syntax demo
08-09-2014
```

Multi-statement Code block

You can write multiple line of server side code enclosed in braces `@{ ... }`. Each line must ends with semicolon same as C#.

Example: Server side Code in Razor Syntax

```
@{
    var date = DateTime.Now.ToShortDateString();
    var message = "Hello World";
}
```

```
<h2>Today's date is: @date </h2>
<h3>@message</h3>
```

Output:

```
Today's date is: 08-09-2014
Hello World!
```

Display Text from Code Block

Use `@:` or `<text>/<text>` to display texts within code block.

Example: Display Text in Razor Syntax

```
@{
    var date = DateTime.Now.ToShortDateString();
    string message = "Hello World!";
    @:Today's date is: @date <br />
    @message
}
```

Output:

```
Today's date is: 08-09-2014
Hello World!
```

Display text using <text> within a code block as shown below.

Example: Text in Razor Syntax

```
@{
    var date = DateTime.Now.ToShortDateString();
    string message = "Hello World!";
    <text>Today's date is:</text> @date <br />
    @message
}
```

Output:

```
Today's date is: 08-09-2014
Hello World!
```

if-else condition

Write if-else condition starting with @ symbol. The if-else code block must be enclosed in braces { }, even for single statement.

Example: if else in Razor

```
@if(DateTime.IsLeapYear(DateTime.Now.Year) )
{
    @DateTime.Now.Year @:is a leap year.
}
else {
    @DateTime.Now.Year @:is not a leap year.
}
```

Output:

```
2014 is not a leap year.
```

for loop

Example: for loop in Razor

```
@for (int i = 0; i < 5; i++) {
    @i.ToString() <br />
}
```

Output:

```
0
1
2
3
4
```

Model

Use @model to use model object anywhere in the view.

Example: Use Model in Razor

```
@model Student

<h2>Student Detail:</h2>
<ul>
    <li>Student Id: @Model.StudentId</li>
    <li>Student Name: @Model.StudentName</li>
    <li>Age: @Model.Age</li>
</ul>
```

Output:

```
Student Detail:

- Student Id: 1
- Student Name: John
- Age: 18
```

Declare Variables

Declare a variable in a code block enclosed in brackets and then use those variables inside html with @ symbol.

Example: Variable in Razor

```
@{
    string str = "";

    if(1 > 0)
    {
        str = "Hello World!";
    }
}

<p>@str</p>
```

Output:

```
Hello World!
```

So this was some of the important razor syntaxes. Visit asp.net to learn razor syntax in detail.

Points to Remember :

1. Use @ to write server side code.
2. Server side code block starts with @{* code * }
3. Use @: or <text></<text> to display text from code block.
4. if condition starts with @if{ }
5. for loop starts with @for
6. @model allows you to use model object anywhere in the view.

HtmlHelper - TextBox

Learn how to generate textbox control using HtmlHelper in razor view in this section.

HtmlHelper class includes two extension methods which creates a textbox (<input type="text">) element in razor view: TextBox() and TextBoxFor(). The TextBox() method is loosely typed method whereas TextBoxFor() is a strongly typed method.

We will use following Student model with TextBox() and TextBoxFor() method.

Example: Student Model

```
public class Student
{
    public int StudentId { get; set; }
    [Display(Name="Name")]
    public string StudentName { get; set; }
    public int Age { get; set; }
    public bool isNewlyEnrolled { get; set; }
    public string Password { get; set; }
}
```

TextBox()

The Html.TextBox() method creates <input type="text" > element with specified name, value and html attributes.

TextBox() method signature

```
MvcHtmlString Html.TextBox(string name, string value, object htmlAttributes)
```

TextBox method has many overloads. Please visit MSDN to know all the overloads of TextBox() method.

The TextBox() method is a loosely typed method because name parameter is a string. The name parameter can be a property name of model object. It binds specified property with textbox. So it automatically displays a value of the model property in a textbox and visa-versa.

Example: Html.TextBox() in Razor View

```
@model Student

@Html.TextBox("StudentName", null, new { @class = "form-control" })
```

Html Result:

```
<input class="form-control"
        id="StudentName"
        name="StudentName"
        type="text"
        value="" />
```

In the above example, the first parameter is "StudentName" property of Student model class which will be set as a name & id of textbox. The second parameter is a value to display in a textbox, which is null in the above example because TextBox() method will automatically display a value of the StudentName property in the textbox. The third parameter will be set as class attribute. HtmlAttributes parameter is an object type, so it can be anonymous object and attributes name will be its properties starting with @ symbol.

You can also specify any name for the textbox. However, it will not be bind to a model.

Example: Html.TextBox() in Razor View

```
@Html.TextBox("myTextBox", "This is value", new { @class = "form-control" })
```

Html Result:

```
<input class="form-control"
        id="myTextBox"
        name="myTextBox"
        type="text"
        value="This is value" />
```

This is value

Output of TextBox() Html Helper method

TextBoxFor

TextBoxFor helper method is a strongly typed extension method. It generates a text input element for the model property specified using a lambda expression. TextBoxFor method binds a specified model object property to input text. So it automatically displays a value of the model property in a textbox and visa-versa.

TextBoxFor() method Signature

MvcHtmlString TextBoxFor(Expression<Func<TModel,TValue>> expression, object htmlAttributes)

Visit MSDN to know all the overloads of TextBoxFor() method.

Example: TextBoxFor() in Razor View

```
@model Student

@Html.TextBoxFor(m => m.StudentName, new { @class = "form-control" })
```

Html Result:

```
<input class="form-control"
        id="StudentName"
        name="StudentName"
        type="text"
        value="John" />
```

In the above example, the first parameter in TextBoxFor() method is a lambda expression which specifies StudentName property to bind with the textbox. It generates an input text element with id & name set to property name. The value attribute will be set to the value of a StudentName property e.g John. The following figure shows the input text element genered by above example.

John

Output of TextBoxFor() Html Helper method

Difference between TextBox and TextBoxFor

- @Html.TextBox() is loosely typed method whereas @Html.TextBoxFor() is a strongly typed (generic) extension method.
- TextBox() requires property name as string parameter where as TextBoxFor() requires lambda expression as a parameter.
- TextBox doesn't give you compile time error if you have specified wrong property name. It will throw run time exception.
- TextBoxFor is generic method so it will give you compile time error if you have specified wrong property name or property name changes. (Provided view is not compile at run time.)

HtmlHelper - TextArea

Learn how to generate TextArea control using HtmlHelper in razor view in this section.

HtmlHelper class includes two extension methods to generate a multi line <textarea> element in a razor view: TextArea() and TextAreaFor(). By default, it creates textarea with rows=2 and cols=20.

We will use the following Student model with the TextArea() and TextAreaFor() method.

Example: Student Model

```
public class Student
{
    public int StudentId { get; set; }
    [Display(Name="Name")]
    public string StudentName { get; set; }
    public string Description { get; set; }
}
```

TextArea()

The Html.TextArea() method creates <textarea rows="2" cols="20" > element with specified name, value and html attributes.

TextArea() method Signature

```
MvcHtmlString Html.TextArea(string name, string value, object htmlAttributes)
```

TextArea method has many overloads. Please visit MSDN to know all the overloads of TextArea method.

The TextArea() method is a loosely typed method because the name parameter is a string. The name parameter can be a property name of model object. It binds a specified property with the textarea. So it automatically displays a value of the model property in a textarea and visa-versa.

Example: Html.TextArea() in Razor View

```
@model Student

@Html.TextArea("Description", null, new { @class = "form-control" })
```

Html Result:

```
<textarea class="form-control"
            id="Description"
            name="Description"
            rows="2"
            cols="20">This is value</textarea>
```

In the above example, the first parameter is the "Description" property of Student model class which will be set as a name & id of textarea. The second parameter is a value to display in a textarea, which is null in the above example because TextArea() method will automatically display a value of the Description property in the textarea. The third parameter will be set as class attribute. HtmlAttributes parameter is an object type, so it can be anonymous object and attributes name will be its properties starting with @ symbol.

You can also specify any name for the textarea. However, it will not be bound to a model.

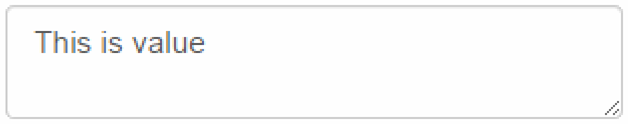
Example: Html.TextArea() in Razor View

```
@Html.TextArea("myTextArea", "This is value", new { @class = "form-control" })
```

Html Result:

```
<textarea class="form-control"
            cols="20"
            id="myTextArea"
            name="myTextArea"
            rows="2">This is value</textarea>
```

The above example would generate input elements as shown below.


This is value

Output of TextArea() Helper method

TextAreaFor

TextAreaFor helper method is a strongly typed extension method. It generates a multi line <textarea> element for the property in the model object specified using a lambda expression. TextAreaFor method binds a specified model object property to textarea element. So it automatically displays a value of the model property in a textarea and visa-versa.

TextBoxFor() method Signature

MvcHtmlString TextAreaFor(<Expression<Func<TModel,TValue>> expression, object htmlAttributes)

Visit MSDN to know all the overloads of TextAreaFor().

Example: TextAreaFor() in Razor View

```
@model Student

@Html.TextAreaFor(m => m.Description, new { @class = "form-control" })
```

Html Result:

```
<textarea class="form-control"
            cols="20"
            id="Description"
            name="Description"
            rows="2"></textarea>
```

In the above example, the first parameter in TextAreaFor() method is a lambda expression that specifies the model property to be bound with the textarea element. We have specified Description property in the above example. So, it generates <textarea> element with id & name set to property name - Description. The value of textarea will be set to the value of a Description property.

HtmlHelper - CheckBox

Learn how to generate checkbox control using HtmlHelper in razor view in this section.

HtmlHelper class includes two extension methods to generate a `<input type="checkbox">` element in razor view: CheckBox() and CheckBoxFor().

We will use following Student model with CheckBox() and CheckBoxFor() method.

Example: Student Model

```
public class Student
{
    public int StudentId { get; set; }
    [Display(Name="Name")]
    public string StudentName { get; set; }
    public int Age { get; set; }
    public bool isNewlyEnrolled { get; set; }
    public string Password { get; set; }
}
```

CheckBox()

The Html.CheckBox() is a loosely typed method which generates a `<input type="checkbox" >` with the specified name, isChecked boolean and html attributes.

CheckBox() method Signature

`MvcHtmlString CheckBox(string name, bool isChecked, object htmlAttributes)`

Please visit MSDN to know all the overloads of CheckBox() method.

Example: Html.CheckBox() in Razor View

```
@Html.CheckBox("isNewlyEnrolled", true)
```

Html Result:

```
<input checked="checked"
        id="isNewlyEnrolled"
        name="isNewlyEnrolled"
        type="checkbox"
        value="true" />
```

In the above example, first parameter is "isNewlyEnrolled" property of Student model class which will be set as a name & id of textbox. The second parameter is a boolean value, which checks or unchecks the checkbox.

CheckBoxFor

CheckBoxFor helper method is a strongly typed extension method. It generates <input type="checkbox"> element for the model property specified using a lambda expression. CheckBoxFor method binds a specified model object property to checkbox element. So it automatically checked or unchecked a checkbox based on the property value.

CheckBoxFor() method Signature

MvcHtmlString CheckBoxFor(<Expression<Func<TModel,TValue>> expression, object htmlAttributes)

Visit MSDN to know all the overloads of CheckBoxFor() method.

Example: Html.CheckBoxFor() in Razor View

```
@model Student

@Html.CheckBoxFor(m => m.isNewlyEnrolled)
```

Html Result:

```
<input data-val="true"
        data-val-required="The isNewlyEnrolled field is required."
        id="isNewlyEnrolled"
        name="isNewlyEnrolled"
        type="checkbox"
        value="true" />

<input name="isNewlyEnrolled" type="hidden" value="false" />
```

In the above example, the first parameter in CheckBoxFor() method is a lambda expression that specifies the model property to be bound with the checkbox element. We have specified isNewlyEnrolled property in the above example. So, it generates <input type="checkbox"> element with id & name set to property name - isNewlyEnrolled. The value attribute will be set to the value of a isNewlyEnrolled boolean property.

In the above Html result, notice that it has generated additional hidden field with the same name and value=false. This is because when you submit a form with a checkbox, the value is only posted if the checkbox is checked. So, if you leave the checkbox unchecked then nothing will be sent to the server when in many situations you would want false to be sent instead. As the hidden input has the same name as the checkbox, then if the checkbox is unchecked you'll still get a 'false' sent to the server.

HtmlHelper - RadioButton

Learn how to generate radio button control using HtmlHelper in razor view in this section.

HtmlHelper class include two extension methods to generate a <input type="radio"> element in a razor view: RadioButton() and RadioButtonFor().

We will use the following Student model with the RadioButton() and RadioButtonFor() method.

Example: Student Model

```
public class Student
{
    public int StudentId { get; set; }
    [Display(Name="Name")]
    public string StudentName { get; set; }
    public int Age { get; set; }
    public string Gender { get; set; }
}
```

RadioButton()

The Html.RadioButton() method creates an radio button element with a specified name, isChecked boolean and html attributes.

RadioButton() method Signature

MvcHtmlString RadioButton(string name, object value, bool isChecked, object htmlAttributes)

Please visit MSDN to know all the overloads of RadioButton() method.

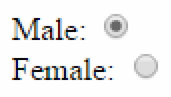
Example: Html.RadioButton() in Razor View

```
Male:   @Html.RadioButton("Gender","Male")
Female: @Html.RadioButton("Gender","Female")
```

Html Result:

```
Male: <input checked="checked"
        id="Gender"
        name="Gender"
        type="radio"
        value="Male" />

Female: <input id="Gender"
        name="Gender"
        type="radio"
        value="Female" />
```

In the above example, we have created two radio button for the "Gender" property. The second parameter is a value which will be sent to the server, if respective radio button is checked. If the Male radio button is selected, then the string value "Male" will be assigned to a model property Gender and submitted to the server. The above example creates two radio buttons as shown below.



Output of RadioButton() method

RadioButtonFor

RadioButtonFor helper method is a strongly typed extension method. It generates <input type="radio"> element for the property specified using a lambda expression. RadioButtonFor method binds a specified model object property to RadioButton control. So it automatically checked or unchecked a RadioButton based on the property value.

RadioButtonFor() method Signature

MvcHtmlString RadioButtonFor(<Expression<Func<TModel,TValue>> expression, object value, object htmlAttributes)

Visit MSDN to know all the overloads of RadioButtonFor().

Example: Html.RadioButtonFor() in Razor View

```
@model Student

@Html.RadioButtonFor(m => m.Gender,"Male")
@Html.RadioButtonFor(m => m.Gender,"Female")
```

Html Result:

```
<input checked="checked"
        id="Gender"
        name="Gender"
        type="radio"
        value="Male" />

<input id="Gender"
        name="Gender"
        type="radio"
        value="Female" />
```

In the above example, the first parameter in RadioButtonFor() method is a lambda expression that specifies the model property to be bind with the RadioButton element. We have created two radio button for the Gender property in the above example. So, it generates two <input type="RadioButton"> element with id & name set to property name - Gender. The second parameter is a value which will be sent to the server when form will be submitted.

HtmlHelper - DropDownList

Learn how to generate dropdownlist control using HtmlHelper in razor view in this section.

HtmlHelper class includes two extension methods to generate a <select> element in a razor view: DropDownList() and DropDownListFor().

We will use the following Student model with DropDownList() and DropDownListFor() method.

Example: Student Model

```
public class Student
{
    public int StudentId { get; set; }
    [Display(Name="Name")]
    public string StudentName { get; set; }
    public Gender StudentGender { get; set; }
}

public enum Gender
{
    Male,
    Female
}
```

DropDownList()

The Html.DropDownList() method generates a select element with specified name, list items and html attributes.

DropDownList() method signature

MvcHtmlString Html.DropDownList(string name, IEnumerable<SelectLestItem> selectList, string optionLabel, object htmlAttributes)

Please visit MSDN to know all the overloads of DropDownList() method.

Example: Html.DropDownList() in Razor View

```
@using MyMVCApp.Models

@model Student

@Html.DropDownList("StudentGender",
                    new SelectList(Enum.GetValues(typeof(Gender))),
                    "Select Gender",
                    new { @class = "form-control" })
```

Html Result:

```
<select class="form-control" id="StudentGender" name="StudentGender">
    <option>Select Gender</option>
    <option>Male</option>
    <option>Female</option>
</select>
```

In the above example, the first parameter is a property name for which we want to display list items. The second parameter is list of values to be included in the dropdownlist. We have used Enum methods to get the Gender enum values. The third parameter is a label which will be the first list item and the fourth parameter is for html attributes like css to be applied on the dropdownlist.

Please note that you can add `MyMVCApp.Models` namespace into `<namespaces>` section in web.config in the Views folder instead of using @using to include namespces in all the views.

DropDownListFor

DropDownListFor helper method is a strongly typed extension method. It generates <select> element for the property specified using a lambda expression. DropDownListFor method binds a specified model object property to dropdownlist control. So it automatically list items in DropDownList based on the property value.

DropDownListFor() method signature

MvcHtmlString Html.DropDownListFor(Expression<Func<dynamic,TProperty>> expression, IEnumerable<SelectLestItem> selectList, string optionLabel, object htmlAttributes)

Visit MSDN to know all the overloads of DropDownListFor().

The following example creates dropdown list for the above Gender enum.

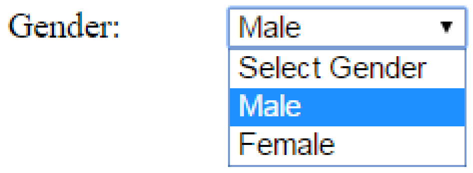
Example: Html.DropDownListFor() in Razor View

```
@using MyMVCApp.Models

@model Student

@Html.DropDownListFor(m => m.StudentGender,
                        new SelectList(Enum.GetValues(typeof(Gender))),
                        "Select Gender")
```

Html Result:

```
<select class="form-control" id="StudentGender" name="StudentGender">
    <option>Select Gender</option>
    <option>Male</option>
    <option>Female</option>
</select>
```

In the above example, the first parameter in DropDownListFor() method is a lambda expression that specifies the model property to be bind with the select element. We have specified StudentGender property of enum type. The second parameter specifies the items to show into dropdown list using SelectList. The third parameter is optionLabel which will be the first item of dropdownlist. So now, it generates <select> element with id & name set to property name - StudentGener and two list items - Male & Female as shown below.



Output of DropDownList() or DropDownListFor() method

HtmlHelper - Hidden field

Learn how to generate hidden field using HtmlHelper in razor view in this section.

HtmlHelper class includes two extension methods to generate a hidden field (<input type="hidden">) element in a razor view: Hidden() and HiddenFor().

We will use the following Student model with Hidden() and HiddenFor() method.

Example: Student Model

```
public class Student
{
    public int StudentId { get; set; }
    [Display(Name="Name")]
    public string StudentName { get; set; }
    public int Age { get; set; }
    public bool isNewlyEnrolled { get; set; }
    public string Password { get; set; }
}
```

Hidden()

The Html.Hidden() method generates a input hidden field element with specified name, value and html attributes.

Hidden() method signature

MvcHtmlString Html.Hidden(string name, object value, object htmlAttributes)

Hidden() method has many overloads. Please visit MSDN to know all the overloads of Hidden() method.

The following example creates a hidden field for StudentId property of Student model. It binds StudentId with the hidden field, so that it can assign value of StudentId to the hidden field and visa-versa.

Example: Html.Hidden() in Razor View

```
@model Student

@Html.Hidden("StudentId")
```

Html Result:

```
<input id="StudentId"
            name="StudentId"
            type="hidden"
            value="1" />
```

HiddenFor

HiddenFor helper method is a strongly typed extension method. It generates a hidden input element for the model property specified using a lambda expression. HiddenFor method binds a specified model object property to <input type="hidden">. So it automatically sets a value of the model property to hidden field and visa-versa.

HiddenFor() method signature

```
MvcHtmlString        Html.HiddenFor(Expression<Func<dynamic,TProperty>>
expression)
```

Visit MSDN to know all the overloads of HiddenFor() method.

Example: HiddenFor() in Razor View

```
@model Student

@Html.HiddenFor(m => m.StudentId)
```

Html Result:

```
<input data-val="true"
            data-val-number="The field StudentId must be a number."
            data-val-required="The StudentId field is required."
            id="StudentId"
            name="StudentId"
            type="hidden"
            value="1" />
```

In the above example, the first parameter in HiddenFor() method is a lambda expression that specifies the model property to be bind with the hidden field.

We have specified StudentId property in the above example. So, it generates input text element with id & name set to property name. The value attribute will be set to the value of a StudentId property which is 1 in the above example.

Please notice that it has created data- attribute of html5 that is used for the validation in ASP.Net MVC.

HtmlHelper - Password

Learn how to generate Password field using HtmlHelper in razor view in this section.

HtmlHelper class includes two extension methods to generate a password field (<input type="password">) element in a razor view: Password() and PasswordFor().

We will use following Student model with Password() and PasswordFor() method.

Example: Student Model

```
public class Student
{
    public int StudentId { get; set; }
    [Display(Name="Name")]
    public string StudentName { get; set; }
    public int Age { get; set; }
    public bool isNewlyEnrolled { get; set; }
    public string OnlinePassword { get; set; }
}
```

Password()

The Html.Password() method generates a input password element with specified name, value and html attributes.

Password() method signature

```
MvcHtmlString Html.Password(string name, object value, object htmlAttributes)
```

Password() method has many overloads. Please visit MSDN to know all the overloads of Password() method.

Example: Html.Password() in Razor View

```
@model Student

@Html.Password("OnlinePassword")
```

Html Result:

```
<input
        id="OnlinePassword"
        name="OnlinePassword"
        type="password"
        value="" />
```

The above example will create password field for "OnlinePassword" property as shown below.

Output of Password() or PasswordFor() method

PasswordFor()

PasswordFor helper method is a strongly typed extension method. It generates a <input type="password"> element for the model object property specified using a lambda expression. PasswordFor method binds a specified model object property to <input type="password">. So it automatically sets a value of the model property to password field and visa-versa.

PasswordFor() method signature

MvcHtmlString Html.PasswordFor(Expression<Func<dynamic,TProperty>> expression, object htmlAttributes)

Visit MSDN to know all the overloads of PasswordFor() method.

Example: PasswordFor() in Razor View

```
@model Student

@Html.PasswordFor(m => m.Password)
```

Html Result:

```
<input id="Password" name="Password" type="password" value="mypassword" />
```

In the above example, the first parameter in PasswordFor() method is a lambda expression that specifies the model property to be bind with the password textbox. We have specified Password property in the above example. So, it generates input password element with id & name set to property name. The value attribute will be set to the value of a Password property which is "mypassword" in the above example.

HtmlHelper - Display HTML String

Learn how to create html string literal using HtmlHelper in razor view in this section.

HtmlHelper class includes two extension methods to generate html string : Display() and DisplayFor().

We will use the following Student model with the Display() and DisplayFor() method.

Example: Student Model

```
public class Student
{
    public int StudentId { get; set; }
    public string StudentName { get; set; }
    public int Age { get; set; }
}
```

Display()

The Html.Display() is a loosely typed method which generates a string in razor view for the specified property of model.

Display() method Signature: `MvcHtmlString Display(string expression)`

Display() method has many overloads. Please visit MSDN to know all the overloads of Display() method

Example: Html.Display() in Razor View

```
@Html.Display("StudentName")
```

Html Result:

```
"Steve"
```

DisplayFor

DisplayFor helper method is a strongly typed extension method. It generates a html string for the model object property specified using a lambda expression.

DisplayFor() method Signature: `MvcHtmlString DisplayFor(<Expression<Func<TModel,TValue>> expression)`

Visit MSDN to know all the [overloads of DisplayFor() method](#).

Example: DisplayFor() in Razor View

```
@model Student

@Html.DisplayFor(m => m.StudentName)
```

Html Result:

```
" Steve  "
```

In the above example, we have specified StudentName property of Student model using lambda expression in the DisplayFor() method. So, it generates a html string with the value of StudentName property, which is "Steve" in the above example.

HtmlHelper - Label

Learn how to create <label> element using HtmlHelper in razor view in this section.

HtmlHelper class includes two extension methods to generate html label : Label() and LabelFor().

We will use following Student model with to demo Label() and LabelFor() method.

Example: Student Model

```
public class Student
{
    public int StudentId { get; set; }
    [Display(Name="Name")]
    public string StudentName { get; set; }
    public int Age { get; set; }
}
```

Label()

The Html.Label() method generates a `<label>` element for a specified property of model object.

Label() method Signature: `MvcHtmlString Label(string expression, string labelText, object htmlAttributes)`

Label() method has many overloads. Please visit MSDN to know all the overloads of Label() method

Example: Html.Label() in Razor View

```
@Html.Label("StudentName")
```

Html Result:

```
<label for="StudentName">Name</label>
```

In the above example, we have specified a StudentName property as a string. So, it will create <label> element that display *Name*.

You can specify another label text instead of property name as shown below.

Example: Html.Label() in Razor View

```
@Html.Label("StudentName","Student-Name")
```

Html Result:

```
<label for="StudentName">Student-Name</label>
```

LabelFor

LabelFor helper method is a strongly typed extension method. It generates a html label element for the model object property specified using a lambda expression.

LabelFor() method Signature: `MvcHtmlString LabelFor(<Expression<Func<TModel,TValue>> expression)`

Visit MSDN to know all the overloads of LabelFor() method.

Example: LabelFor() in Razor View

```
@model Student

@Html.LabelFor(m => m.StudentName)
```

Html Result:

```
<label for="StudentName">Name</label>
```

In the above example, we have specified the StudentName property of Student model using lambda expression in the LabelFor() method. So, it generates <label> and set label text to the same as StudentName property name.

HtmlHelper - Editor

We have seen different HtmlHelper methods used to generated different html elements in the previous sections. ASP.NET MVC also includes a method that generates html input elements based on the datatype. Editor() or EditorFor() extension method generates html elements based on the data type of the model object's property.

The following table list the html element created for each data type by Editor() or EditorFor() method.

Property DataType	Html Element
string	<input type="text" >
int	<input type="number" >
decimal, float	<input type="text" >
boolean	<input type="checkbox" >
Enum	<input type="text" >
DateTime	<input type="datetime" >

We will use the following model class with Editor and EditorFor method.

Example: Student Model

```
public class Student
{
    public int StudentId { get; set; }
    [Display(Name="Name")]
    public string StudentName { get; set; }
    public int Age { get; set; }
    public bool isNewlyEnrolled { get; set; }
    public string Password { get; set; }
    public DateTime DoB { get; set; }
}
```

Editor()

Editor() method requires a string expression parameter to specify the property name. It creats a html element based on the datatype of the specified property.

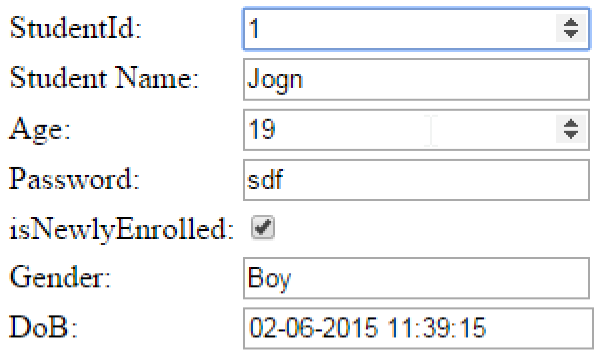
Editor() signature: `MvcHtmlString Editor(string propertyname)`

Visit MSDN to know all the [overloads of Editor() method]()

Consider the following example to understand the Editor() method.

Example: Editor() in Razor view

```
StudentId:       @Html.Editor("StudentId")
Student Name:    @Html.Editor("StudentName")
Age:             @Html.Editor("Age")
Password:        @Html.Editor("Password")
isNewlyEnrolled: @Html.Editor("isNewlyEnrolled")
Gender:          @Html.Editor("Gender")
DoB:             @Html.Editor("DoB")
```



StudentId:	1
Student Name:	Jogn
Age:	19
Password:	sdf
isNewlyEnrolled:	✓
Gender:	Boy
DoB:	02-06-2015 11:39:15

Output of Editor() and EditorFor() method

In the above example, we have specified property names of Student model as a string. So, Editor() method created the appropriate input elements based on the datatype as shown in the above figure.

EditorFor

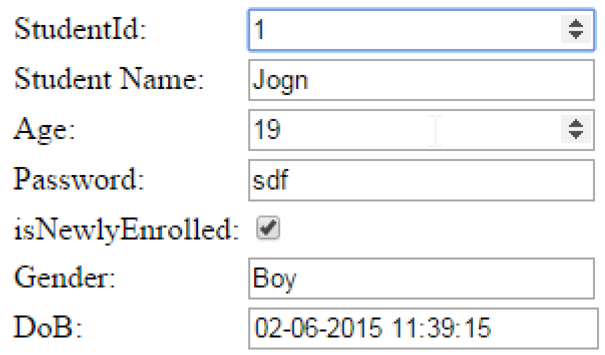
EditorFor() method is a strongly typed method. It requires the lambda expression to specify a property of the model object.

EditorFor() signature: `MvcHtmlString EditorFor(<Expression<Func<TModel,TValue>> expression)`

Visit MSDN to know all the overloads of EditorFor() method

Example: EditorFor() in Razor view

```
StudentId:       @Html.EditorFor(m => m.StudentId)
Student Name:    @Html.EditorFor(m => m.StudentName)
Age:             @Html.EditorFor(m => m.Age)
Password:        @Html.EditorFor(m => m.Password)
isNewlyEnrolled: @Html.EditorFor(m => m.isNewlyEnrolled)
Gender:          @Html.EditorFor(m => m.Gender)
DoB:             @Html.EditorFor(m => m.DoB)
```



StudentId:	1
Student Name:	Jogn
Age:	19
Password:	sdf
isNewlyEnrolled:	✓
Gender:	Boy
DoB:	02-06-2015 11:39:15

Output of Editor() and EditorFor() method

In the above example of EditorFor() method, we have specified the property name using the lambda expression. The result would be the same as the Editor() method as shown in the above figure.