អំពី View

I. អ្វីទៅជា View?

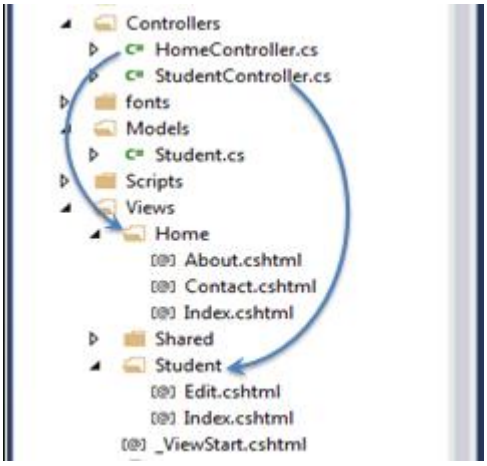
View ជា user interface ដែលវាទទួលខុសត្រូវក្នុងការបង្ហាញទិន្នន័យពីmodel ដែលបង្កើតឡើងដោយ action method របស់ controllerទៅអោយ user មើលឃើញនិងអោយពួកគេធ្វើការផ្លាស់ប្តូរព័ត៌មានបាន(modify the data)។

<u>លក្ខណៈរបស់វា view</u>

- មានតួនាទីបង្ហាញទិន្នន័យ(model)
- មិនគួមាន business logic និងមិនត្រូវធ្វើតំណើរផ្សេងៗ
- View មានទំរង់ច្រើនផ្សេងៗពីគ្នា(HTML JSON XML ...)

1. ការបង្កើត View

គ្រប់Viewsទាំងអស់ត្រូវរក្សាទុកក្នុងថត(directory) views/ControllerName/និងមានកន្ទុយជា *.cshtml។ឈ្មោះរបស់វាគួរមានឈ្មោះដូចនិងaction methodរបស់ controller។ឧទាហរណ៍ថា យើងមាន StudentController.cs និង example() action methodដូចនេះ view គួរដាក់ឈ្មោះថា example.cshtml។



<u>គេអាចបង្កើត view បានតាមពីរបៀបគឺ៖</u>

- ការបង្កើតចេញពី action method របស់ controller ណាមួយ
- និងទី២បង្កើតview ចេញពីថត (sub directory)ដោយផ្ទាល់តែម្តង

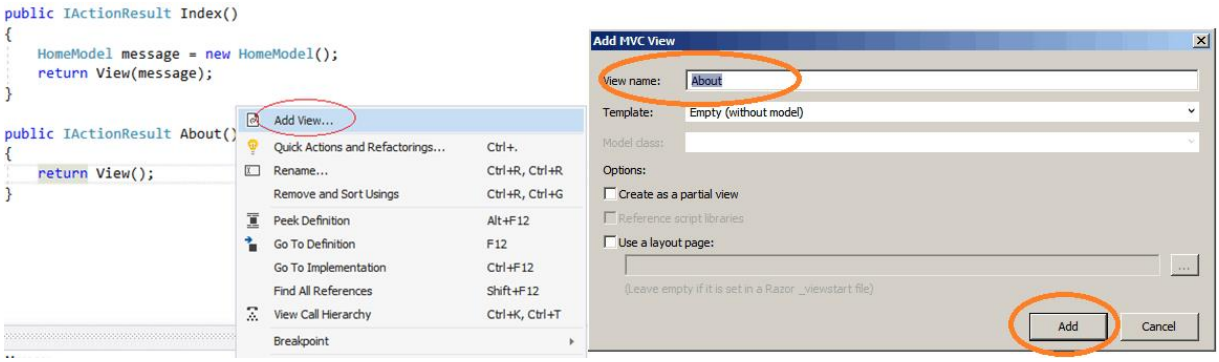
1.1. ការបង្កើតចេញពី action method

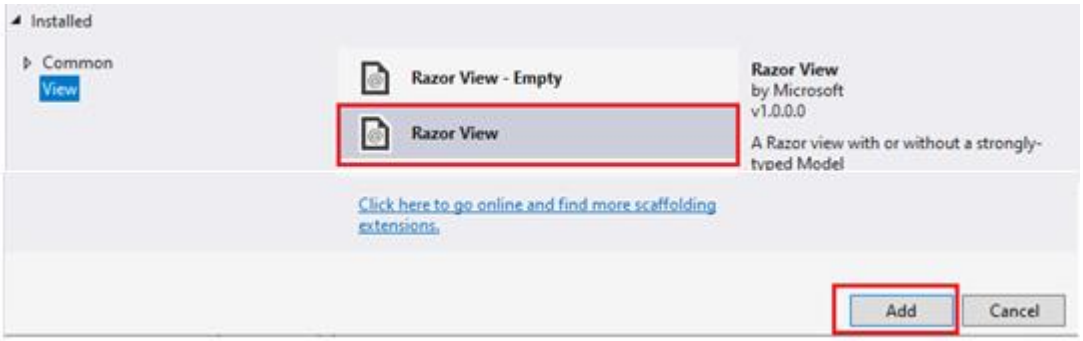
ដើម្បីបង្កើត view តាមរយៈ action method ត្រូវអនុវត្តតាមវិធីខាងក្រោម៖

a. ត្រូវជ្រើសរើសcontroller

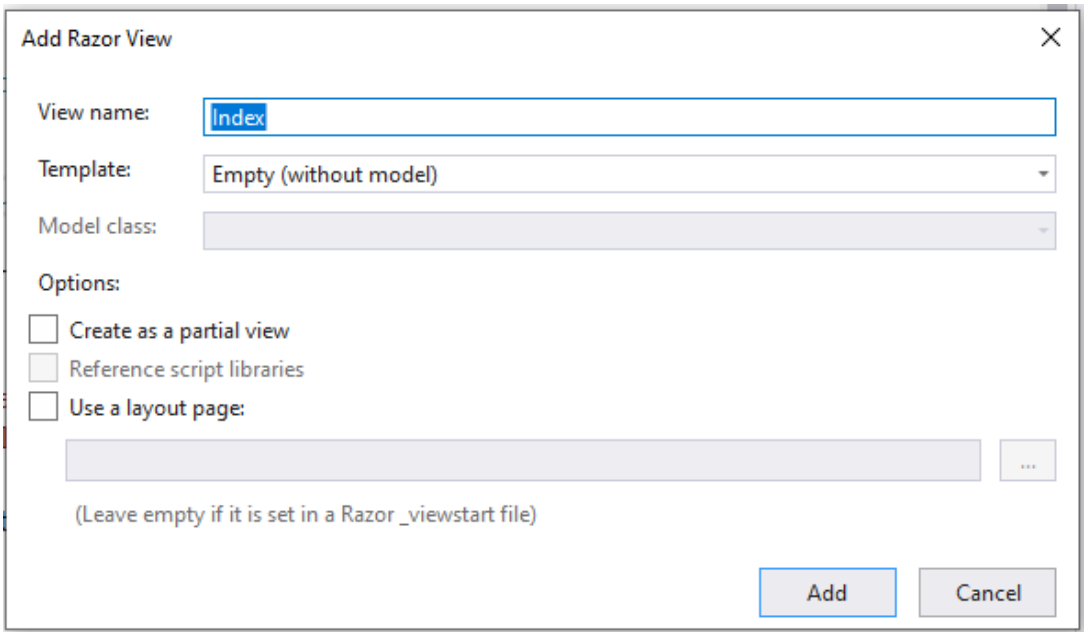
b. ត្រូវជ្រើសរើស action method មួយណាដែលត្រូវបង្កើត view

c. ចុចម៉ៅស្តាំលើ action method → add view → ចុចលើ Razor View

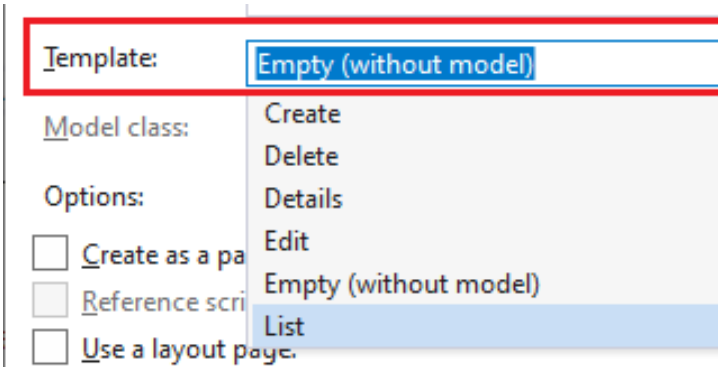




d. ចុចលើពាក្យ Add → ដាក់ឈ្មោះអោយ view → ជ្រើរើស template(empty without model) → ជ្រើសរើសយក model class (ទុកចំហរ) → Add



Templates



- Empty(without model)បង្កើត view ដោយមិនត្រូវការmodel class
- Createបង្កើតview សម្រាប់បញ្ចូលទិន្នន័យតាម model object
- Delete បង្កើតview សម្រាប់លុបទិន្នន័យ
- Details បង្កើតview សម្រាប់បង្ហាញព័ត៌មានលំអិតតាម model class
- Edit បង្កើតview សម្រាប់កែទិន្នន័យតាមmodel class
- List បង្កើតview សម្រាប់បង្ហាញព័ត៌មានទាំងអស់ពី ដាតាបេសតាម model class

Options

- Create as partial viewយើងធីកយកវាបើយើងចង់បង្កើត partial view តែបើមិនចង់ទេកុំធីកយកវាអី។
- Use a layout page បញ្ជាក់ថាតើ view របស់ចង់ប្រើប្រាស់ layout ដែលមានស្រាប់ឬទេតែបើមិនចង់ទេកុំធីកយកវាអី។

e. និងចុចលើ ប៊ូតុង Add

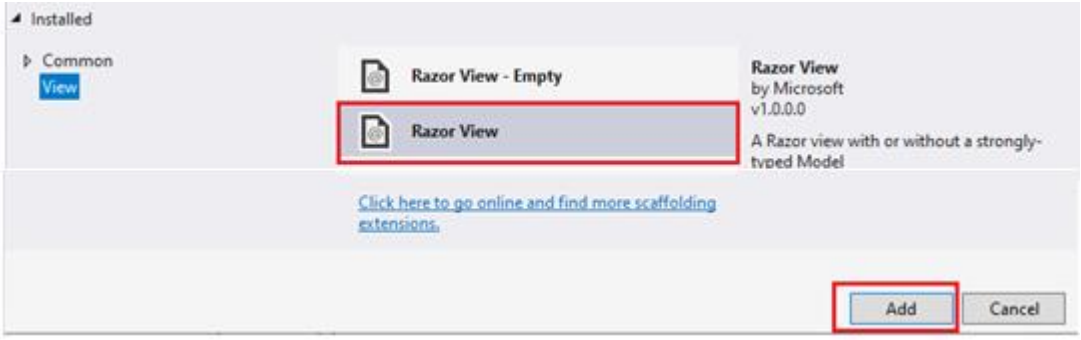
1.2. បង្កើតview ចេញពីថត (sub directory)ដោយផ្ទាល់តែម្តង

ដើម្បីបង្កើត view តាមរយៈ action method ត្រូវអនុវត្តតាមវិធីខាងក្រោម៖

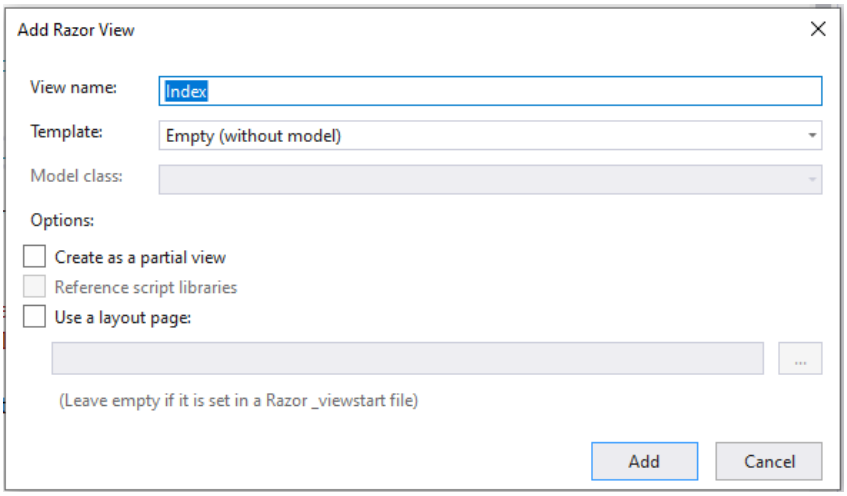
a. ចូលទៅកាន់ view directory

b. ត្រូវជ្រើសរើស sub directory ណាមួយដែលចង់បង្កើត

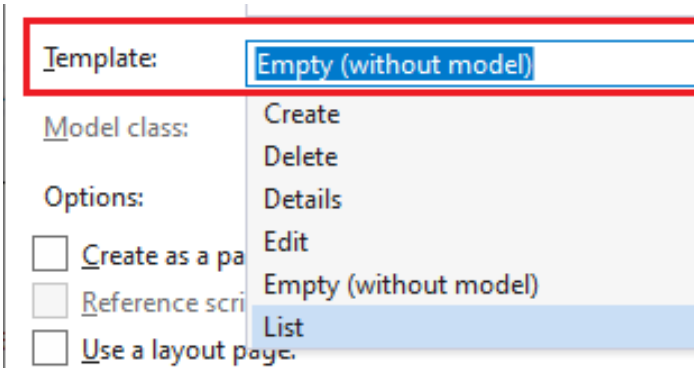
c. ចុចម៉ៅស្តាំលើ sub directoty(example) → add → view → ចុចលើ Razor View



d. ចុចលើពាក្យ Add → ដាក់ឈ្មោះអោយ view → ជ្រើរើស template(empty without model) → ជ្រើសរើសយក model class (ទុកចំហរ) → Add



Templates



- Empty(without model)បង្កើត view ដោយមិនត្រូវការmodel class
- Createបង្កើតview សម្រាប់បញ្ចូលទិន្នន័យតាម model object
- Delete បង្កើតview សម្រាប់លុបទិន្នន័យ
- Details បង្កើតview សម្រាប់បង្ហាញព័ត៌មានលំអិតតាម model class
- Edit បង្កើតview សម្រាប់កែទិន្នន័យតាមmodel class
- List បង្កើតview សម្រាប់បង្ហាញព័ត៌មានទាំងអស់ពី ដាតាបេសតាម model class

Options

- Create as partial viewយើងធីកយកវាបើយើងចង់បង្កើត partial view តែបើមិនចង់ទេកុំធីកយកវាអី។
- Use a layout page បញ្ជាក់ថាតើ view របស់ចង់ប្រើប្រាស់ layout ដែលមានស្រាប់ឬទេតែបើមិនចង់ទេកុំធីកយកវាអី។

e. និងចុចលើ ប៊ូតុង Add

2. ការដំណើរ View

View វាមិនបង្ហាញយើងភ្លាមៗនោះទេគឺវាតំណើរការពេលដែលគេធ្វើការ requestទៅ controller action ណាមួយ។យើងដឹងហើយថា action method និងបោះតំលៃគ្រប់យ៉ាងនូវអ្វីដែលគេចង់បានដូចជា អក្សរ លេខជាដើម។

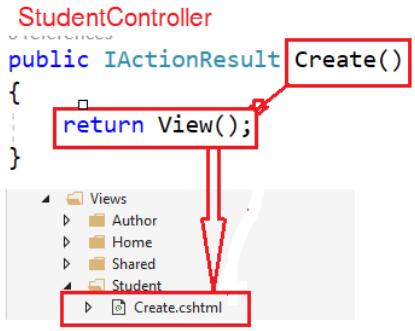
2.1. ការបង្ហាញ View

ដូច្នេះដើម្បីបោះតំលៃជា view (html markup)គេត្រូវប្រើនូវ View() helper methodរបស់ controller។

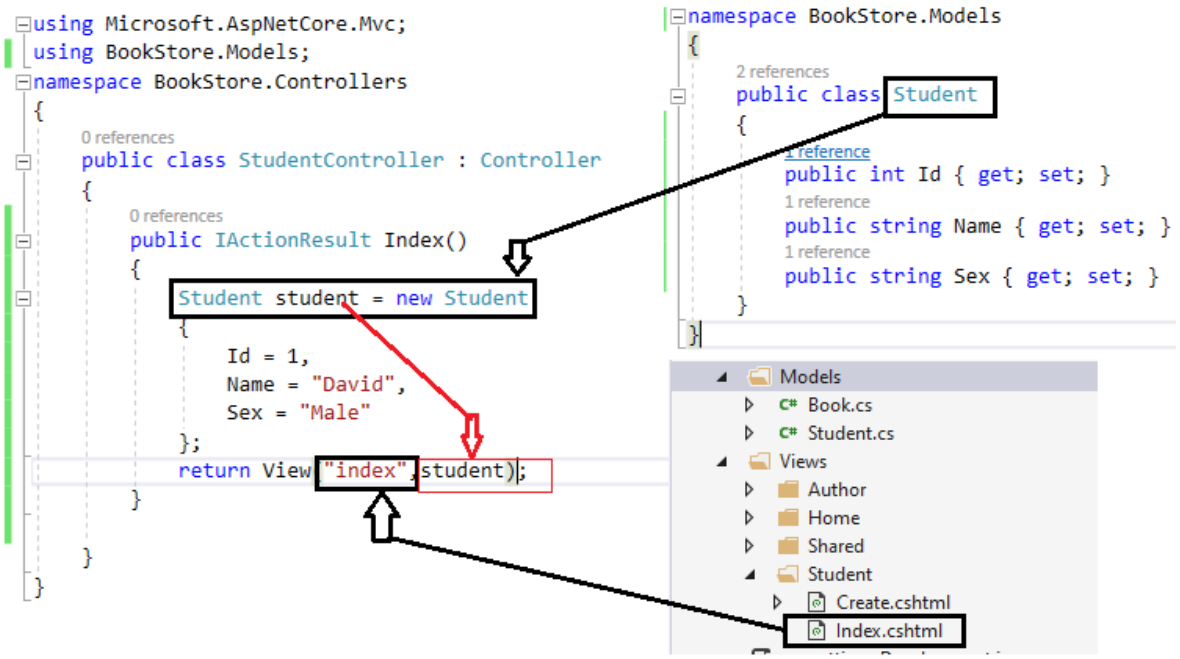
<u>រូបមន្ត</u>

- View()
- View(object model)
- View(string viewName)
- View(string viewName,object model)

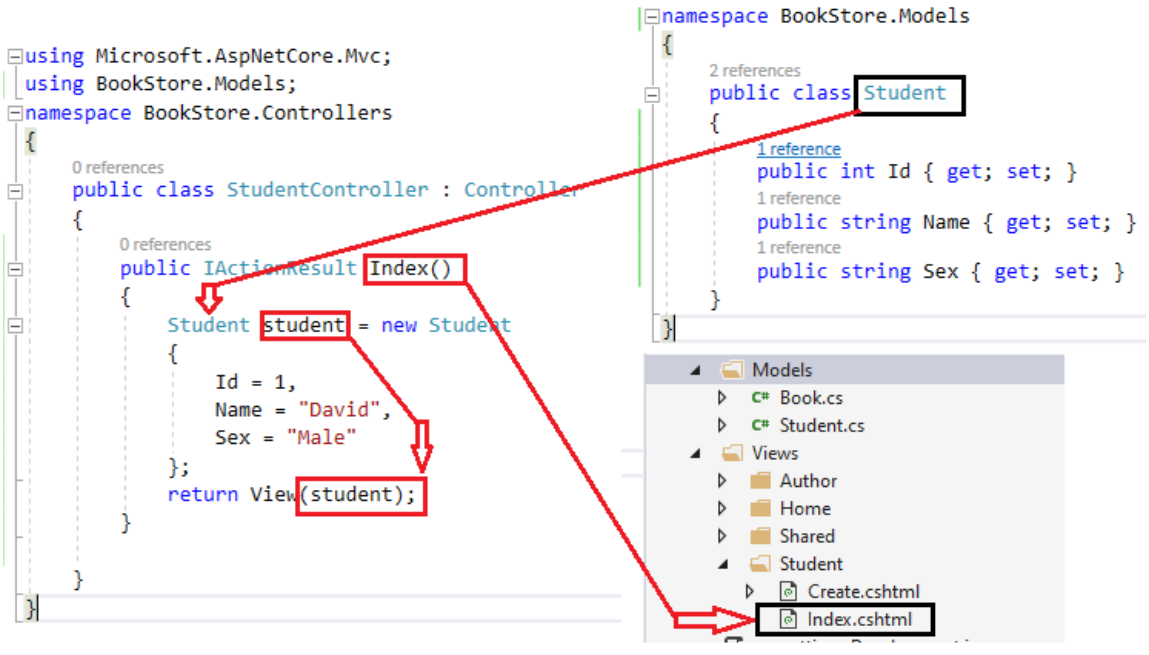
<u>ឧទាហរណ៍២.១</u>



<u>ឧទាហរណ៍២.២</u>



ឧទាហរណ៍២.៣



2.2. ការបង្ហាញ Partial View

ដូច្នេះដើម្បីបោះតំលៃជា view (html markup)គេត្រូវប្រើនូវ PartialView() helper methodរបស់ controller។

ប្រមន្ត

- PartialView()
- PartialView(object model)
- PartialView(string viewName)
- PartialView(string viewName,object model)

II. Razor View Engine

Razor វាមិនមែនជាprogramming language ទេតែវាជាview engine ដែលអនុញ្ញាតិអោយយើងសរសេរ server side (C#) កូដច្របល់ជាមួយ html បាន។ ដើម្បីអោយ view engine ស្គាល់ថាមួយណាជា server side code ដូច្នេះរាល់ C# កូដទាំងអស់ត្រូវចាប់ផ្តើមដោយសញ្ញា @។

គេមានវិធីសាស្ត្រចំនួនពីរដើម្បីសរសេរ razor syntax

- តាមរយៈ Razor code expressions
- និង Razor code blocks.

1. Inline expression

ដើម្បីអោយ view engine ស្គាល់ថាមួយណាជា server side code ដូច្នេះរាល់ c# កូដទាំងអស់ត្រូវចាប់ផ្តើមដោយសញ្ញា @និងបន្តដោយ C# code។

ឧទាហរណ៍៖

C# Razor Syntax

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

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

Output:

Razor syntax demo
08-09-2014

2. Razor code blocks

ដើម្បីអោយ view engine ស្គាល់ថាមួយណាជា server side code ដូច្នេះរាល់ C# កូដទាំងអស់ត្រូវ ចាប់ផ្តើមដោយសញ្ញា @ និងបន្តដោយ { …… }។

ឧទាហរណ៍៖

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!
```

3. ការបង្ហាញទិន្នន័យ

គេប្រើនូវ @: ឬ <text>your text here </text>ដើម្បីបង្ហាទិន្នន័យទៅកាន់ browser។

ឧទាហរណ៍៖

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!
```

ឧទាហរណ៍៖

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!
```

4. អំពីលក្ខ័ណ្ឌនិងរង្វិលជុំ

a. លក្ខ័ណ្ឌ

- If-else 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.
```

- Switch statement

```
@{
   ViewBag.Title = "RazorControlStructure";
   var value = 20;
}
<hr />
@switch (value)
{
   case 1:
      <p>You Entered 1</p>
      break;
   case 25:
      <p>You Entered 25</p>
      break;
   default:
      <p>You entered something than 1 and 25.</p>
      break;
}
```

Output:

You entered something than 1 and 25.

b. រង្វិលជុំ

- For

Example: for loop in Razor

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

Output:

0
1
2
3
4

- While

```
<h3>While loop</h3>
@{
  var r = 0;
  while (r < 5)
  {
    r += 1;
    <span> @r</span>
  }
}
```

```
@{ var s = 0; }
@while (s < 5)
   {
      s += 1;
      <span> @s</span>
   }
}
```

- Foreach

```
@{
   var custList = new List<Customer>()
   {
      new Customer() { name = "Rahul", address = "Bangalore" },
      new Customer() { name = "Sachin", address = "Mumbai" }
   };
}

<table>
   <thead>
      <tr><td>Name</td><td>Address</td></tr>
   </thead>
   @foreach (Customer custvar in custList)
   {
      <tr>
         <td>@custvar.name</td>
         <td>@custvar.address</td>
      </tr>
   }
</table>
```

5. ការប្រកាសអថេរ

ដើម្បីប្រកាអថេរគេអាចប្រើនូវ var keyword ឬប្រើប្រាស់នូវ C# data type។

ឧទាហរណ៍៖

```
<!-- Storing a string -->
@{ var message = "Welcome to our website"; }

<!-- Storing a date -->
@{ DateTime date = DateTime.Now; }

<p>@message</p>
<p> The current date is @date</p>
```

6. ការប្រើប្រាស់ @using directive

@using directive ប្រើដើម្បី import namespace ចូលទៅក្នុង viewហើយវាត្រូវស្ថិតនៅលើគេបង្អស់និងបន្តបន្ទាប់គ្នា។

រូបមន្ត

@using your_namespace

ឧទាហរណ៍៖

```
1
2   @using WebApplication1.Models
3
```

@using WebApp.Model

7. ការប្រើប្រាស់ @model directive

@model directive ប្រើសម្រាប់ import class object ដើម្បីយកមកប្រើប្រាស់គ្រប់ទីតាំងអស់ក្នុង view។ក្រោយពីប្រើប្រាស់ @model រួចមកគ្រប់properties ដែលមាននៅក្នុង class objectវានិងផ្ទុកនៅក្នុង Model object ដែលជាbuild-in object របស់ asp.net mvcដែលវាអនុញ្ញាតិអោយហើយយើងaccess នូវ properties បាន។ @model directive ត្រូវតែនៅលើគេបន្ទាប់ពី @using directive។

រូបមន្ត

@model YourClassName

ឧទាហរណ៍៖

```
public class Product
{
    0 references
    public int Id { get; set; }
    1 reference
    public string ProductName { get; set; }
    1 reference
    public decimal Price { get; set; }
}
```

```
@using WebApplication1.Models
@addTagHelper *,Microsoft.AspNetCore.Mvc.TagHelpers

@model Product
@{
    ViewData["Title"] = "Create";
}
<!doctype html>
<html>
<head>
    <title></title>
</head>
<body>
    <form action="/book/create" method="post">
        <p>
        <label asp-for="ProductName"></label>
        <input type="text" asp-for="ProductName" />
        </p>
        <p>
            <label asp-for="Price"></label>
            <input type="text" asp-for="Price"/>
        </p>
        <p><input type="submit" value="Create" /></p>
    </form>
</body>
</html>
```

III. អំពីLayout