

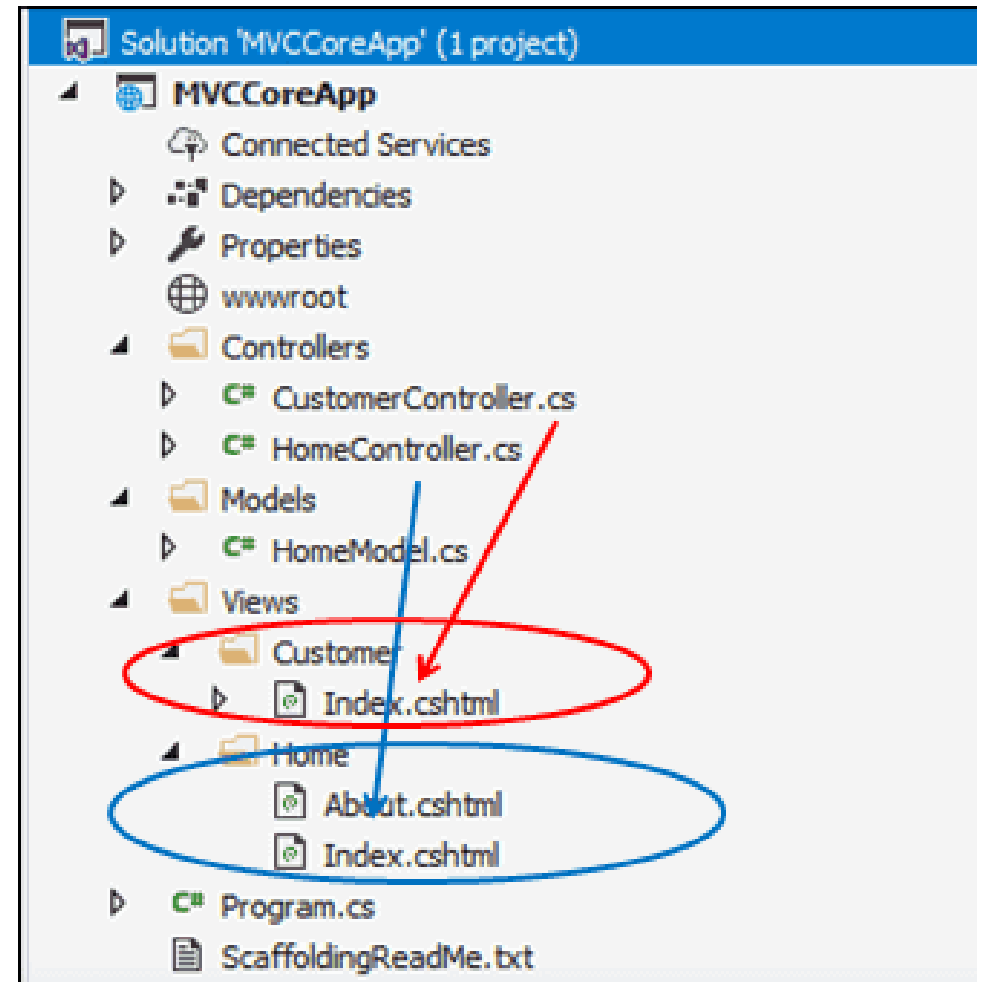
View

# View?

- View មានតួនាទីតែម្យ៉ាងគត់គឺវាធ្វើការបង្ហាញទិន្នន័យ ( Model ) ទៅអោយអ្នកប្រើប្រាស់មើលឃើញ។
- View អាចមានទំរង់ផ្សេងៗគ្នាគឺដូចជា៖ html, json, xml ឬទំរង់ផ្សេងៗទៀត។

# ការបង្កើត view ?

- គ្រប់ view ទាំងអស់ត្រូវស្ថិតនៅក្នុងថត views
- គ្រប់ views ដែលបោះពុម្ពជា markup ត្រូវមានភ្ជាប់ជា \*.cshtml
- ថតនីមួយៗ (sub directory) របស់ view ត្រូវមានឈ្មោះដូចគ្នានឹងឈ្មោះរបស់ controller



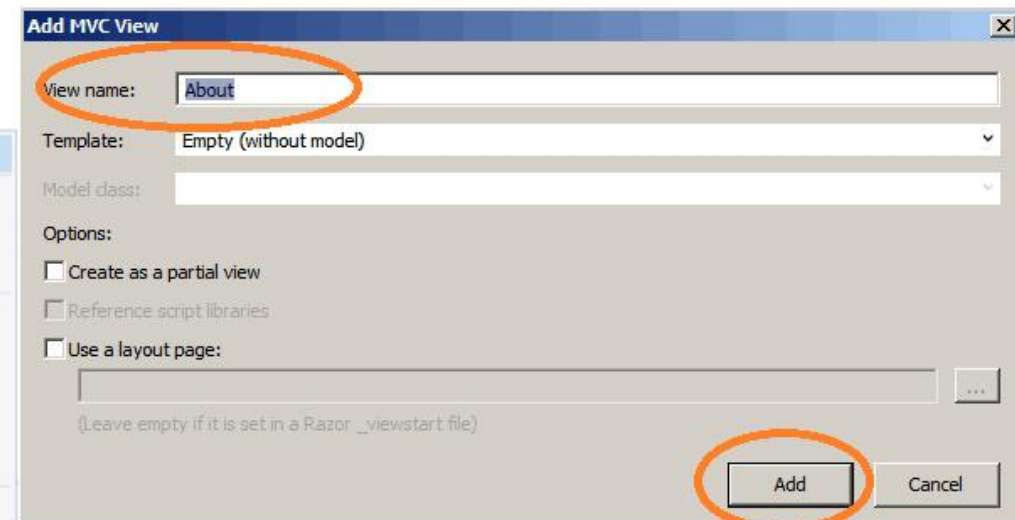
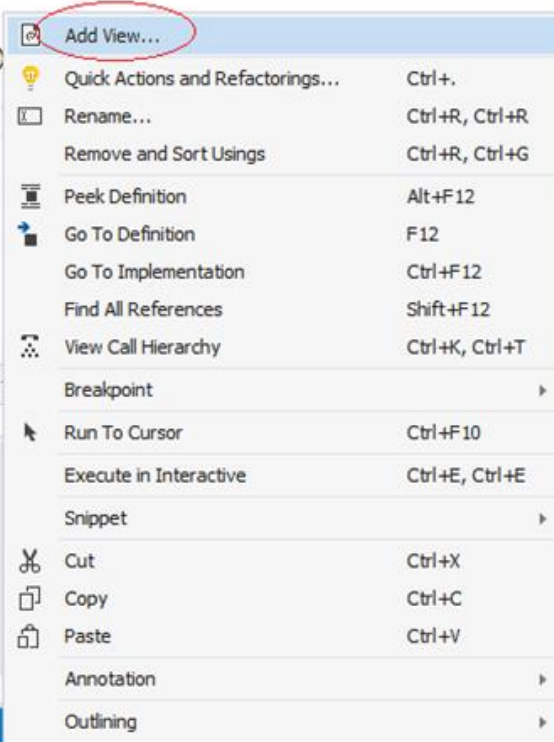
# ការបង្កើត view ?


- គេអាចបង្កើត view បានតាមពីរបៀបគឺ៖
  - ការបង្កើតចេញពី action method របស់ controller ណាមួយ
  - នឹងទី២បង្កើតview ចេញពីថត (sub directory)ដោយផ្ទាល់តែម្តង

# ดูจาบรรณ

```
public IActionResult Index()  
{  
    HomeModel message = new HomeModel();  
    return View(message);  
}
```

```
public IActionResult About()  
{  
    return View();  
}
```



View... 

Controller...

New Item... Ctrl+Shift+A

Existing Item... Shift+Alt+A

New Empty File... Shift+F2

New Scaffolded Item...

New Folder

Container Orchestrator Support...

Docker Support...

Client-Side Library...



New Azure WebJob Project

Existing Project as Azure WebJob

Class...

Solution Explorer (Ctrl+;) Solution 'Middleware' (1 of 1 project)

**Middleware**

- Connected Services
- Dependencies
- Properties
- wwwroot
- Controllers
- C# ProductController.cs
- Entities
- Middleware
- ViewModels
- Views**
- Product 
- View in Browser (Microsoft Edge) Ctrl+Shift+W
- Browse With...
- Add** 
- File Nesting
- Scope to This
- New Solution Explorer View
- Exclude From Project

## Add MVC View



View name:

Detailed



Template:

Empty (without model)



Model class:



Options:

☐ Create as a partial view

☐ Reference script libraries

☒ Use a layout page:



...

(Leave empty if it is set in a Razor \_viewstart file)



Add

Cancel

Razor syntax



# Razor syntax

- Razor ជា view engine ដែលអនុញ្ញាតិអោយយើងសរសេរ C# ក្នុងច្របល់ជាមួយ html បាន។
- គេមានវិធីសាស្ត្រចំនួនពីរដើម្បីសរសេរ razor syntax
  - តាមរយៈ Razor code expressions
  - និង Razor code blocks.

# Razor code blocks

- ដើម្បីអោយ view engine ស្គាល់ថាមួយណាជា server side code ដូច្នេះ រាល់ c# កូដទាំងអស់ត្រូវចាប់ផ្តើមដោយសញ្ញា @ និងបន្តដោយ { ហើយ បញ្ចប់ទៅវិញដោយសញ្ញា }

### <h3>Code Block</h3>

```
@{  
    var greeting = "Welcome to our site!";  
    var weekDay = DateTime.Now.DayOfWeek;  
}  
  
@{  
    var cust = new MVCCoreApp.Models.Customer()  
    {  
        name = "Rahul Dravid",  
        address = "Bangalore"  
    };  
}
```

# Razor Code Expressions

- Razor code expression ត្រូវចាប់ផ្តើមដោយសញ្ញា @ បន្តឡើងដោយ c# code ។

<h3>Code Expression</h3>

<p>@greeting</p>

<p>@DateTime.Now</p>

<p>Today is : @WeekDay thank you </p>

# Using directive

- @using directive ប្រើដើម្បី import namespace ចូលទៅក្នុង view ។
- ឧទាហរណ៍ ៖
  - @using WebApp.Model

# ការប្រើប្រាស់ model directive

- @model directive ប្រើសម្រាប់ import custom class ដើម្បីប្រើប្រាស់គ្រប់ទីតាំងអស់ក្នុង view ។
- រូបមន្ត
  - @model ClassName

```
@model Product
```

```
@{
```

```
    ViewData["Title"] = "Create";
```

```
}
```

```
<h1>Create</h1>
```

```
<form asp-action="create" asp-controller="product" method="post">
```

```
    <div>
```

```
        <label asp-for="ProductName"></label>
```

```
        <input asp-for="ProductName" />
```

```
        <span asp-validation-for="ProductName"></span>
```

```
    </div>
```

```
    <input type="submit" value="Create" />
```

```
</form>
```



# ការប្រកាសអថេរ

- ដើម្បីប្រកាសអថេរគេអាចប្រើនូវ 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>
```

# ការបង្ហាញទិន្នន័យ

- បង្ហាញតែមួយបន្ទាត់

```
@{
```

```
@:Hello from the Code block
```

```
}
```

- បង្ហាញច្រើនបន្ទាត់

```
@{
```

```
<text>Hello from the multiline text </text>
```

```
}
```

# សក្ខីណា

```
@{int value = 200;}
```

```
@if (value > 100)
```

```
{
```

```
<p>Value is greater than 100.</p>
```

```
}
```

```
else
```

```
{
```

```
<p>Value is less than 100.</p>
```

```
}
```

# ស៊ីណ

```
@{  
    var value = 200;  
    if (value > 100)  
    {  
        <p>The value is greater than 100 </p>  
    }  
    else  
    {  
        <p>This value is less than 100.</p>  
    }  
}
```

# បញ្ជាប្តូរ ( switch )

```
@switch (value)
{
    case 0:
        @: value is Zero
        break;
    case 100:
        <p>Value is 100 </p>
        break;
    case 200:
        <p>Value is @value </p>
        break;
    case 300:
        <text>Value is 300</text>
        break;
    default:
        <p>Invalid Value </p>
        break;
}
```

# រង្វិលជុំ ( foreach )

```
<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>
```

# រង្វិលជុំ ( for )

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

# រង្វិលជុំ ( while )

<h3>While loop</h3>

@{

var r = 0;

while (r < 5)

{

r += 1;

<span> @r</span>

}

}



Layout

# Page Layout

- ក្នុងការកសាង website ជាទូទៅ website ត្រូវផ្សំឡើងដោយ menu, header, footer និង sidebar ជាដើម។
- Layout page ជួយអោយយើងកំណត់នូវធាតុនៃ common user's interface ដែលមានដូចជា៖ navigation menu, header, footer និងផ្សេងៗទៀតក្នុងកន្លែងតែមួយនិងអាចប្រើប្រាស់គ្រប់ទីកន្លែងទាំងអស់។
- Layout page ត្រូវស្ថិតនៅក្នុង views/shared directory និង ឈ្មោះរបស់វាត្រូវចាប់ផ្តើមដោយ underscore( \_ ) និងបន្តដោយឈ្មោះ។

# បង្កើត layout

- ដំបូងបង្កើត directory មួយដែលមានឈ្មោះថា **shared** (បើមិនមាន) ក្នុង views directory
- បន្ទាប់មកចុចម៉ៅស្កាំលើ shared directory → Add → New Item  
→ Razor Layout → Add

## Installed

PHP (PeachPie)

## Visual C#

## ASP.NET Core

Code

Data

General

## Web

ASP.NET

Scripts

Content

## Online

Sort by: Default



Search (Ctrl+E)



Controller Class

Visual C#



API Controller Class

Visual C#



Razor Component

Visual C#



Razor Page

Visual C#



Razor View

Visual C#



Razor Layout

Visual C#



Razor View Start

Visual C#



Razor View Imports

Visual C#



Tag Helper Class

Visual C#



Middleware Class

Visual C#



Startup Class

Visual C#



App Settings File

Visual C#

**Type:** Visual C#

Razor View Layout Page

Name:

\_Layout.cshtml

Add

Cancel

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>Layout Example</title>
</head>
<body>
  <div id="header">
    <h1>Layout example in HTML</h1>
  </div>
  <div id="content">
    @RenderBody()
  </div>
  <div id="Footer">
    <p>This is Footer</p>
  </div>
</body>
</html>
```

# @renderBody()

- វាជាmethod ពិសេសមួយដែលសម្គាល់ទីតាំងដែល child view ប្រើប្រាស់layout នេះដើម្បីបង្ហាញនូវ child contentដែលគេហៅម្យ៉ាងទៀតជា placeholder។

# ការប្រើប្រាស់ layout

```
@{
```

```
    Layout = "~/views/shared/_layout.cshtml";
```

```
    ViewBag.Title = "Edit";
```

```
}
```

```
<h2>this child page content</h2>
```

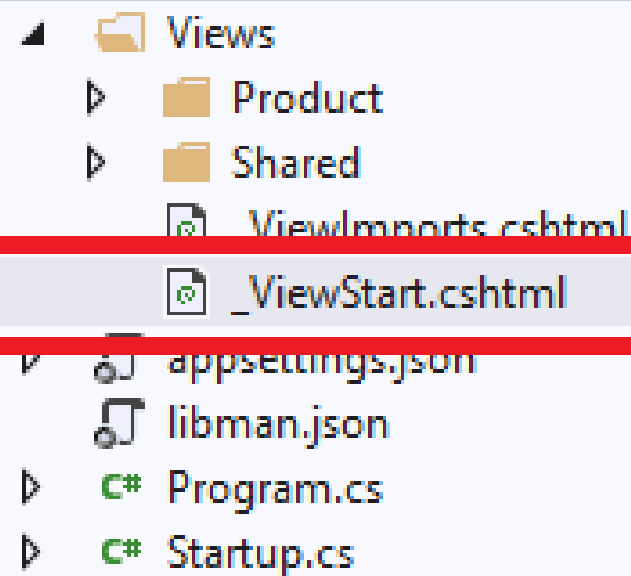
# ការប្រើប្រាស់ \_ViewStart

- គោលបំណងនៃការប្រើប្រាស់ \_viewstart.cshtml គឺដើម្បីកំណត់តំលៃដើមទោយ views ទាំងអស់ក្នុង views directory និង subdirectory របស់វា។
- \_viewstart ត្រូវស្ថិតនៅក្នុង views directory



```
1 @{  
2     Layout = "_Layout";  
3 }  
4
```

\_ViewStart.cshhtml



- Views
  - Product
  - Shared
  - \_ViewImports.cshhtml**
  - \_ViewStart.cshhtml**
- appsettings.json
- libman.json
- Program.cs
- Startup.cs

# \_ViewImport

- \_ViewImport ប្រើសម្រាប់ import namespace ដើម្បីប្រើប្រាស់គ្រប់ views ទាំងអស់។
- \_ViewImport.cshtml ត្រូវស្ថិតនៅក្នុង views directory

\_ViewImports.cshtml

```
1 @using Middleware.Entities
2 @using Middleware.ViewModels
3 @using Middleware.Middlewarees
4 @using Middleware.Controllers
5 @addTagHelper *,Microsoft.AspNetCore.Mvc.TagHelpers
```

# @RenderSection

- វាជាmethod មួយសម្រាប់បង្ហាញនូវ child section ណាដែល  
layout page ។
- រូបមន្ត
  - @RenderSection(string name,required:false|true)

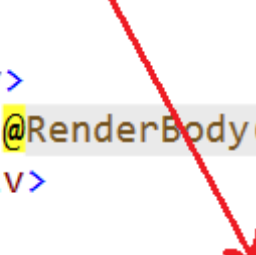
# ការបង្កើត section

edit.cshtml ( child page )

```
@{
    Layout = "~/views/shared/_layout.cshtml";
    ViewBag.Title = "Edit";
}
<h2>this child page content</h2>
@section mysection{
    <h1>this child content sections</h1>
}
```

\_layout.cshtml ( parent page )

```
14 <div>
15     <div>
16         @RenderBody()
17     </div>
18
19     @RenderSection("mysection", false)
20 </div>
21 <footer style="background-color:brown">
22     <h2>this footer section</h2>
23 </footer>
```



# Partial view

- Partial view ជាបំណែកតូចនៃ web page ដែលគេប្រើម្តងហើយម្តងទៀតបាន។
- រូបមន្ត
  - `@html.RenderPartial(string name)` ឬ `@html.RenderPartialAsync(string name)`
  - `<partial name="view name">`

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta name="viewport" content="width=device-width" />
5      <title>@ViewBag.Title</title>
6  </head>
7  <body>
8      <header style="background-color:blue">
9          <h3>this header section</h3>
10     </header>
11     <partial name="_navbar" />
12     <div>
13         <div>
14             @RenderBody()
15         </div>
16         @RenderSection("mysection", false)
17     </div>
18
19     @{ await Html.RenderPartialAsync("_footer"); }
20 </body>
21 </html>
```

# ViewBag និង ViewData

- ViewBag ជា dynamic property ដែលអាចផ្ទុកតំលៃអ្វីក៏បាន
- ViewData វាជា property មួយរបស់ controller base class ដែលបោះតំលៃជា ViewDataDictionary(ផ្ទុកតំលៃជា key-value paired)។  
ViewData ជាចំនាត់ត្រូវបំបែកជាមុនសិនទើបអាចប្រើប្រាស់បាន។
- ទាំង ViewBag និង ViewData គេប្រើដើម្បីបញ្ជូនទិន្នន័យពី controller ទៅ view។



# ViewBag

- រូបមន្ត
  - ViewBag.PropertyName=value;
- ការទាញទិន្នន័យ
  - ViewBag.PropertyName

```
@model Middleware.Entities.Product
@{
    ViewData["Title"] = "Create";
}
<h1>Create</h1>
<form asp-action="create" asp-controller="product" method="post">
    <div>
        <label asp-for="ProductName"></label>
        <input asp-for="ProductName" />
        <span asp-validation-for="ProductName"></span>
    </div>
    <input type="submit" value="Create" />
</form>

<h3>Description</h3>
<li>Id=@ViewBag.Product?.ProductId</li>
<li>Id=@ViewBag.Product?.CategoryId</li>
<li>Id=@ViewBag.Product?.ProductName</li>

@ViewBag.Test
```

```
public class ProductController : Controller
{
    [HttpGet]
    0 references
    public IActionResult Create()
    {
        return View();
    }
    [HttpPost]
    0 references
    public IActionResult Create([Bind("ProductName")] Product product)
    {
        if (ModelState.IsValid)
        {
            ViewBag.Test = "Hello world.";
            ViewBag.Product = new Product
            {
                ProductId=1,
                CategoryId=1,
                ProductName=product.ProductName
            };
            return View();
        }
        return View(product);
    }
}
```

# ViewData

- រូបមន្ត
  - ViewData[string key]=value;
- ទាញយកទិន្នន័យ
  - (type)ViewData[string key];

```
@model Product
@{
    ViewData["Title"] = "Create";
}
<h1>Create</h1>
<form asp-action="create" asp-controller="product" method="post">
    <div>
        <label asp-for="ProductName"></label>
        <input asp-for="ProductName" />
        <span asp-validation-for="ProductName"></span>
    </div>
    <input type="submit" value="Create" />
</form>

<h3>Description</h3>
@{
    var product = (Product)ViewData["Product"];
    <li>Id=@product?.ProductId</li>
    <li>CategoryId=@product?.CategoryId</li>
    <li>Product Name=@product?.ProductName</li>
}
@ViewData["Test"]
```

```
[HttpGet]
0 references
public IActionResult Create()
{
    return View();
}
[HttpPost]
0 references
public IActionResult Create([Bind("ProductName")] Product product)
{
    if (ModelState.IsValid)
    {
        ViewData["Test"] = "Hello world.";
        ViewData["Product"] = new Product
        {
            ProductId=1,
            CategoryId=1,
            ProductName=product.ProductName
        };
        return View();
    }
    return View(product);
}
```