



CHỦ ĐỀ



VIEWS

GVGD: PHẠM THỊ KIM NGOAN

Email: ngoanptk@ntu.edu.vn

Nội dung



1. Giới thiệu Views

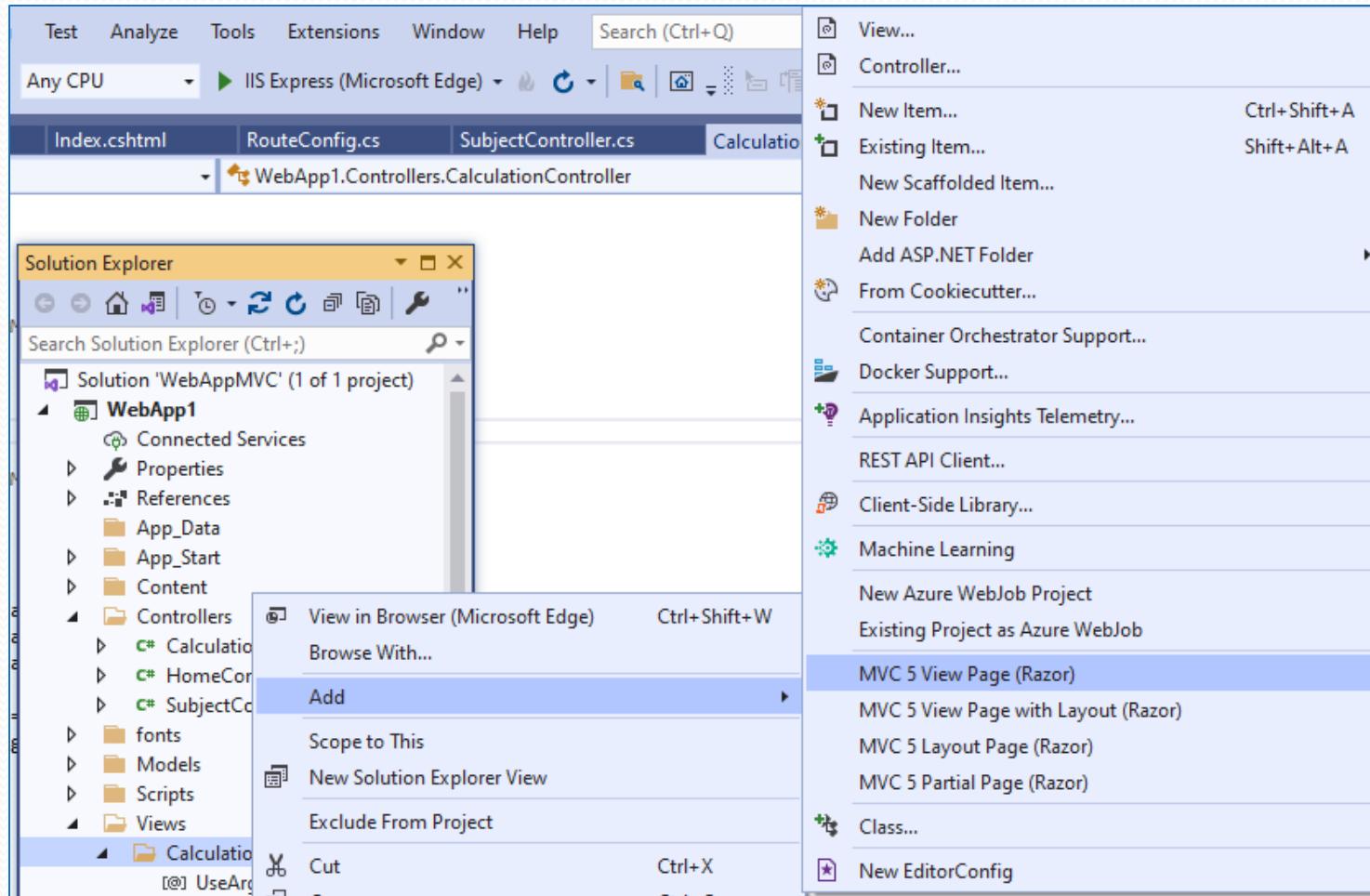
- **Views** - khuôn mẫu dành cho **hiển thị dữ liệu**.
- **Views** là các thành phần chịu trách nhiệm hiển thị các thông tin lên cho người dùng thông qua giao diện.
- View kết hợp với dữ liệu nó nhận được, **phát sinh** nội dung **HTML** trả về cho client.
- **View:** Là các file **.cshtml** tích hợp sẵn cú pháp **Razor** (razor engine), trong đó chứa mã HTML, có thể nhúng mã C# (những đoạn nhúng mã server side thì bắt đầu bằng ký hiệu @)
- Các View được tổ chức thành các thư mục cho từng **Controller** và được lưu trong **thư mục Views**.

1. Giới thiệu Views

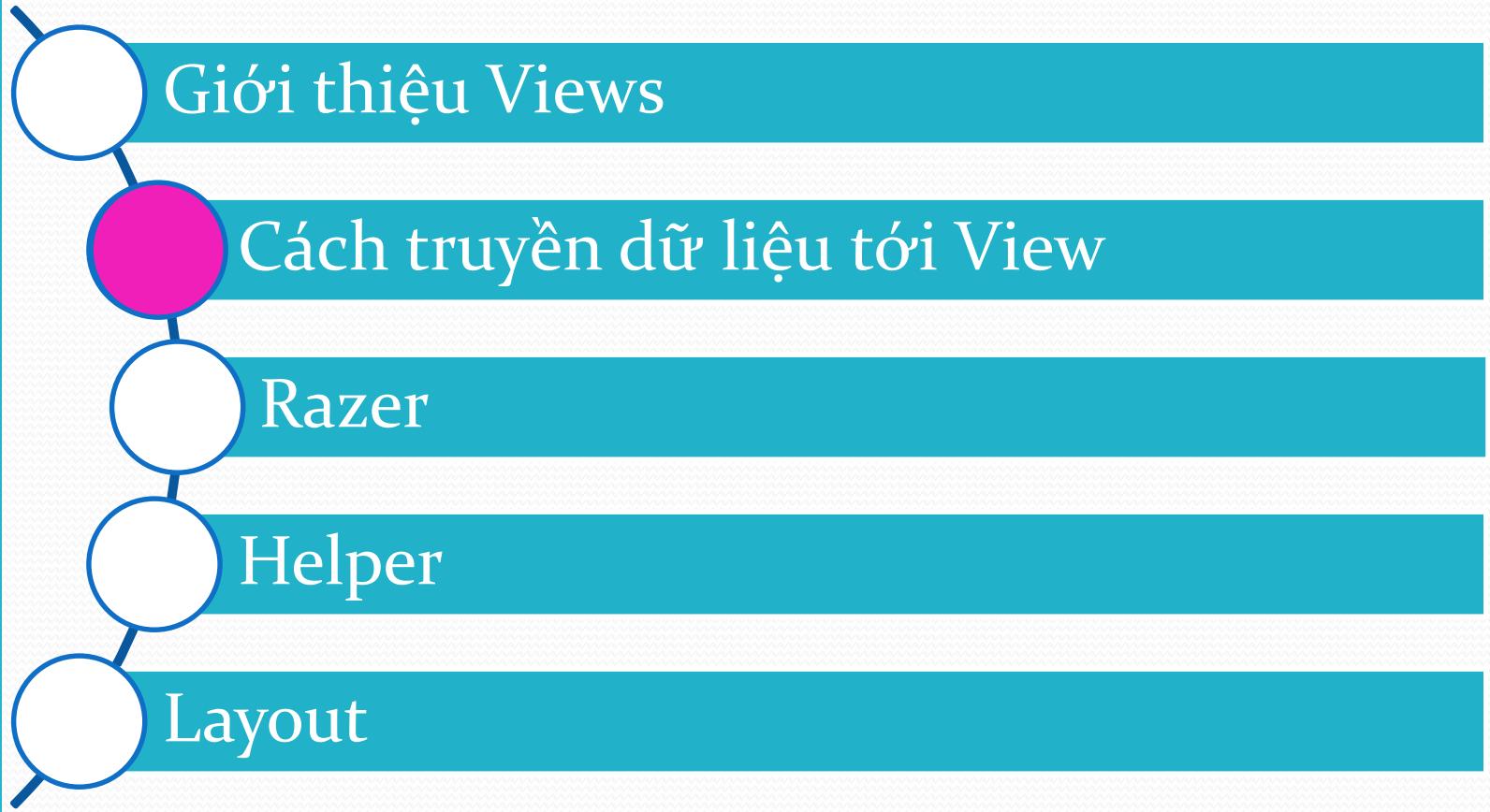
- Các View cho các Action mặc định lưu theo cấu trúc thư mục
`/View/ControllerName/ActionName.cshtml`
- Khi một Action trả về đối tượng **ViewResult** thì nó sẽ sử dụng **View.cshtml** và Razor làm việc để tạo nội dung HTML. Nó sẽ tìm đến thư mục **Views**, tìm đến thư mục cùng tên Controller.
- Ví dụ **Home**, sau đó là file **.cshtml** cùng tên với Action - file **.cshtml** tìm được sẽ sinh HTML.

1. Giới thiệu Views

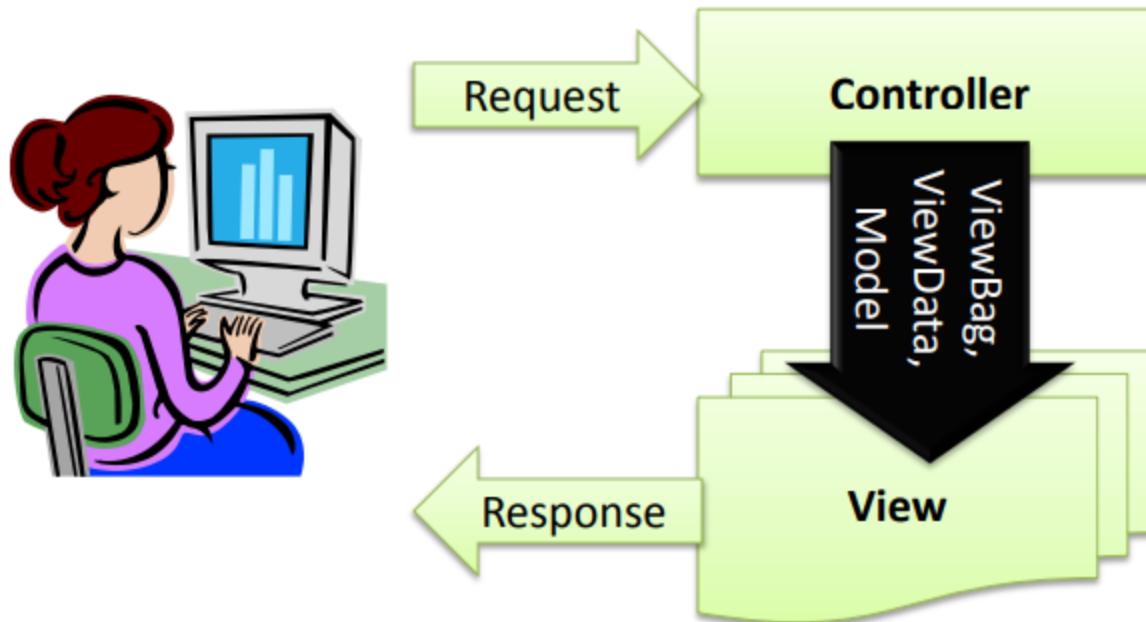
➤ Cách tạo View: R_Click vào thư mục trong Views → Add ...



Nội dung



2. Cách truyền dữ liệu Controller tới View

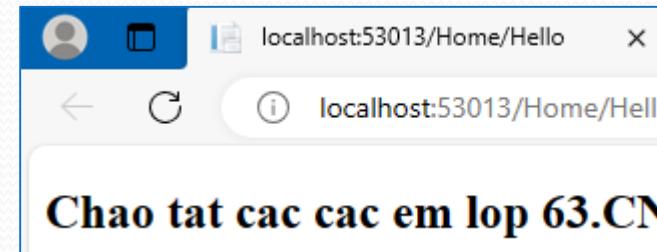


- **ViewBag/ViewData** và **Model** được sử dụng để chia sẻ dữ liệu giữa Controller và View.

2. Cách truyền dữ liệu Controller tới View

- ViewData, ViewBag là kiểu Dynamic, có thể thiết lập dữ liệu vào nó với cặp key/value (key: chuỗi)
- Khi Controller thiết lập dữ liệu qua ViewBag, ViewData thì ở View dùng ký hiệu @ để truy cập.

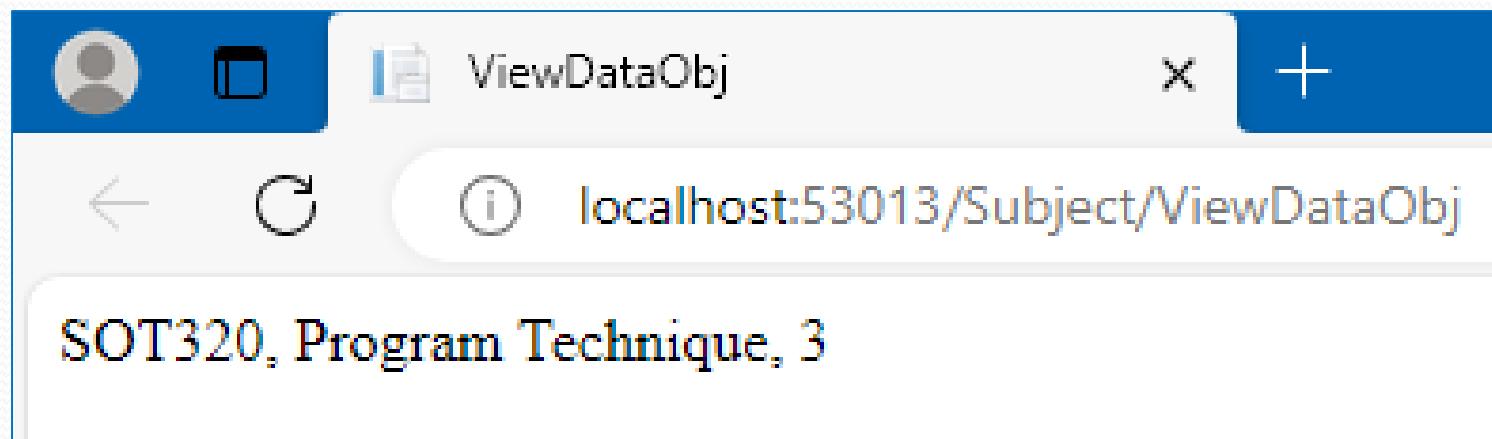
```
//use ViewBag, ViewData
public ActionResult Hello()
{
    ViewBag.Greeting = "Chao tat cac cac em lop";
    ViewData["class"] = "63.CNTT-4,5";
    return View();
}
```



```
<body>
    <div>
        <h2>@ViewBag.Greeting @ViewData["class"]</h2>
    </div>
</body>
```

Ví dụ truyền 1 đối tượng qua ViewBag

- Kết quả chạy trên trình duyệt: Thông tin của 1 đối tượng thuộc lớp học phần gồm 3 thuộc tính: mã số, tên học phần, số tín chỉ.



Ví dụ truyền 1 đối tượng qua ViewBag

- Trong thư mục Models

```
public class SubjectModel
{
    public string Id { get; set; }
    public string Name { get; set; }
    public byte Num { get; set; }
    public SubjectModel (string i,string s,byte n)
    {
        Id = i;Name = s;Num = n;
    }
}
```

Ví dụ truyền 1 đối tượng qua ViewBag

- Trong thư mục Controllers

```
public class SubjectController : Controller
{
    //dùng ViewBag
    public ActionResult ViewBagObj()
    {
        SubjectModel sub = new SubjectModel("SOT320", "Program
Technique", 3);
        ViewBag.Subject = sub;
        return View();
    }
}
```

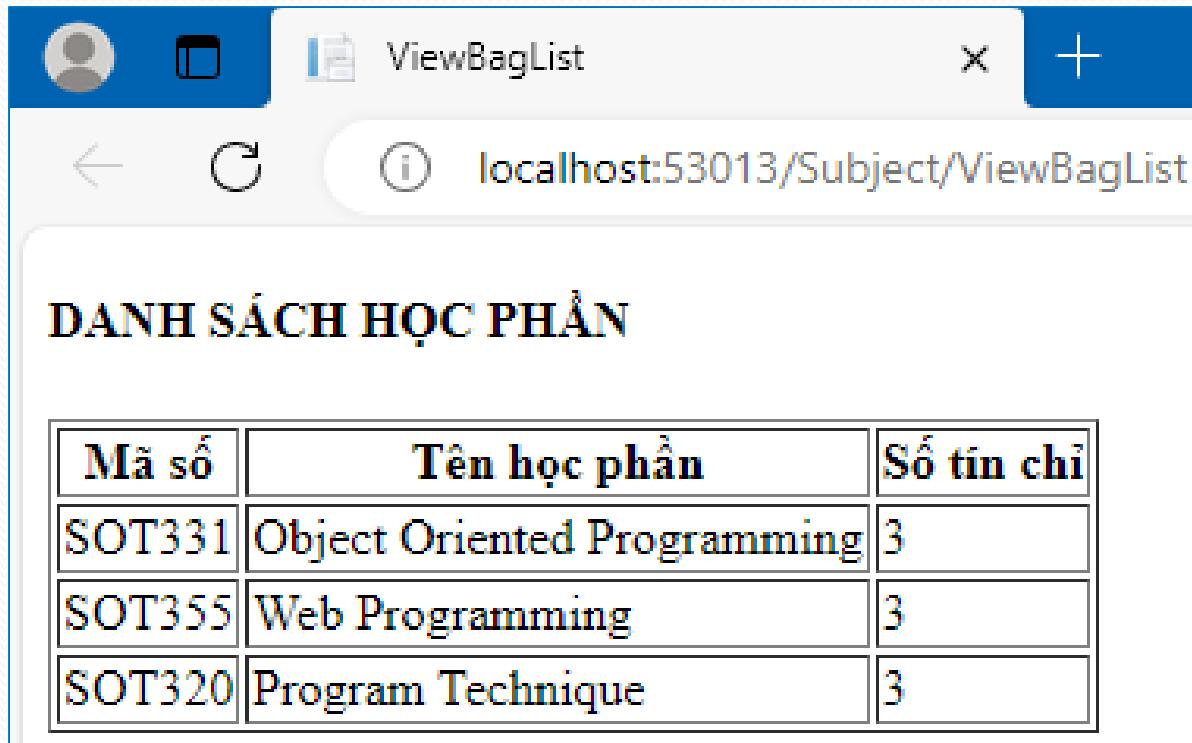
Ví dụ truyền 1 đối tượng qua ViewBag

- Trong thư mục Views\Subject

```
@{Layout = null;
    WebApp1.Models.SubjectModel subject =
(WebApp1.Models.SubjectModel)ViewBag.Subject;
}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>ViewBagObj</title>
</head>
<body>
    <div>
        @subject.Id, @subject.Name, @subject.Num
    </div>
</body>
</html>
```



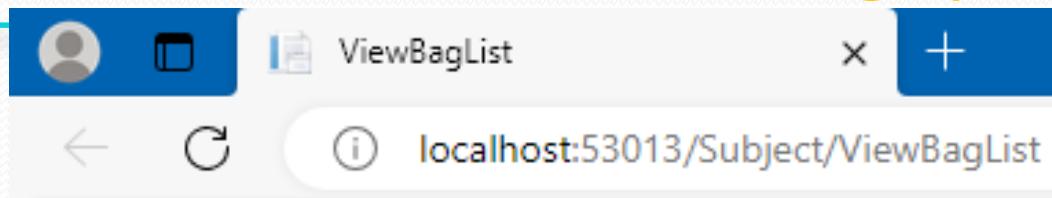
VD truyền 1 danh sách đối tượng qua ViewBag



The screenshot shows a web browser window with the title "ViewBagList". The address bar displays "localhost:53013/Subject/ViewBagList". The main content area has a header "DANH SÁCH HỌC PHẦN". Below it is a table with four rows, each containing a subject's code, name, and credit value.

Mã số	Tên học phần	Số tín chỉ
SOT331	Object Oriented Programming	3
SOT355	Web Programming	3
SOT320	Program Technique	3

VD truyền 2 danh sách đối tượng qua ViewBag



DANH SÁCH HỌC PHẦN

Mã số	Tên học phần	Số tín chỉ
SOT331	Object Oriented Programming	3
SOT355	Web Programming	3
SOT320	Program Technique	3
SOT382	General Informatic	2

DANH SÁCH HỌC PHẦN CÓ SỐ TÍN CHỈ LÀ 3

Mã số	Tên học phần	Số tín chỉ
SOT331	Object Oriented Programming	3
SOT355	Web Programming	3
SOT320	Program Technique	3

2. Cách truyền dữ liệu Controller tới View

➤ Sử dụng Model:

- Tạo class trong Models
- Thiết lập Model cho View trong phương thức View của Controller.
- Trong View .cshtml sử dụng chỉ thị **@model** để thiết lập kiểu truyền tới cho **Model** của View.

2. Cách truyền dữ liệu tới View

➤ Models

```
public class SubjectModel
{
    public string Id { get; set; }
    public string Name { get; set; }
    public byte Num { get; set; }
    public SubjectModel (string i,string s,byte n)
    {   Id = i;Name = s;Num = n;           }
}
```

➤ Controllers

```
public class SubjectController : Controller
{
    public ActionResult List()
    {   var model = new List<SubjectModel>()
        {   new SubjectModel ("SOT331", "Object Oriented Programming", 3),
            new SubjectModel("SOT355","Web Programming",3)
        };
        return View(model);
    }
}
```

2. Cách truyền dữ liệu tới View

➤ Views

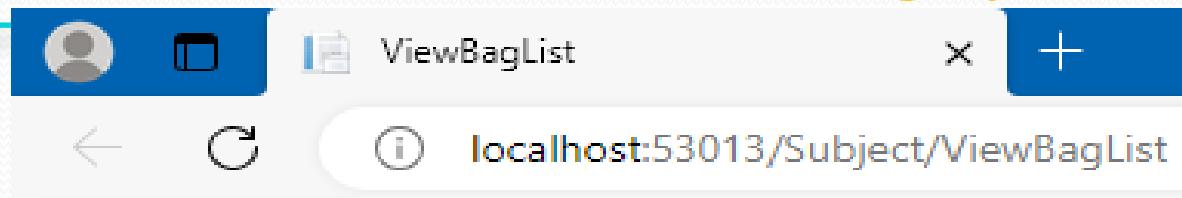
```
@model List<WebApp1.Models.SubjectModel>
<!DOCTYPE html>
<html>
    ...
    <body>
        <div>
            <ul>
                @foreach (var i in Model)
                {
                    <li>@i.Id <i>@i.Name </i> @i.Num</li>
                }
            </ul>
        </div>
    </body>
</html>
```

➤ Lưu ý: phân biệt @model và @Model

- @model: dùng để khai báo kiểu của @Model
- @Model: đối tượng chứa dữ liệu truyền từ Controller



VD truyền 2 danh sách đối tượng qua Model



A screenshot of a web browser window. The title bar says "ViewBagList". The address bar shows "localhost:53013/Subject/ViewBagList". The main content area displays two tables of course information.

DANH SÁCH HỌC PHẦN

Mã số	Tên học phần	Số tín chỉ
SOT331	Object Oriented Programming	3
SOT355	Web Programming	3
SOT320	Program Technique	3
SOT382	General Informatic	2

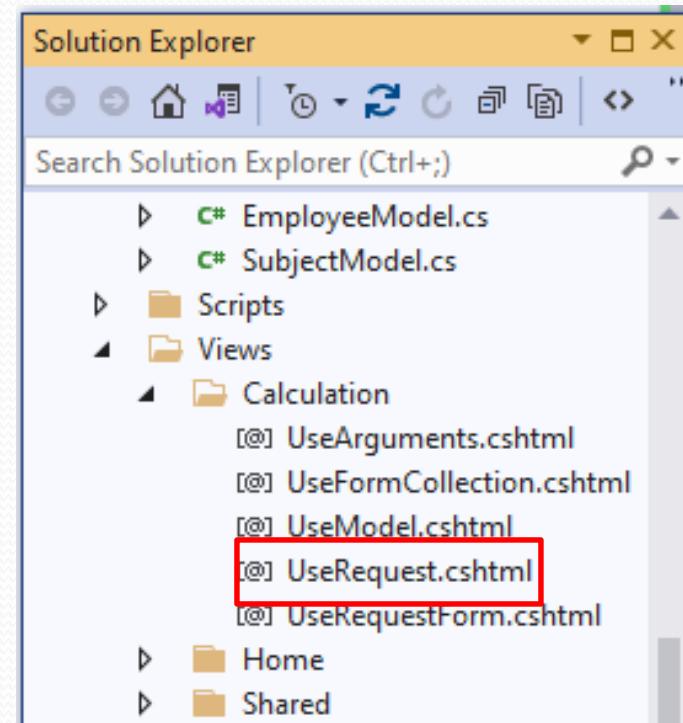
DANH SÁCH HỌC PHẦN CÓ SỐ TÍN CHỈ LÀ 3

Mã số	Tên học phần	Số tín chỉ
SOT331	Object Oriented Programming	3
SOT355	Web Programming	3
SOT320	Program Technique	3

View

- Tên của Action trong Controller và tên của View không giống nhau.

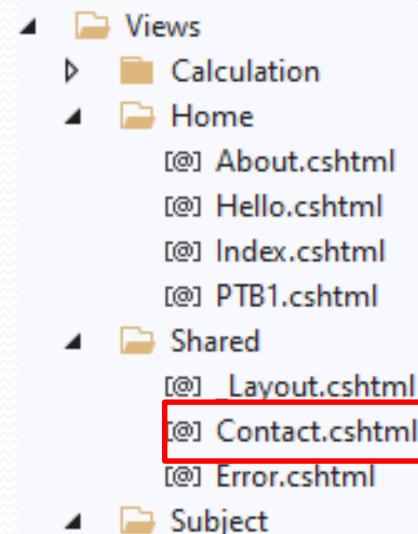
```
//tên action và View không giống nhau
0 references
public ActionResult Cal()
{
    return View("UseRequest");
}
//Sử dụng Request
[HttpPost]
0 references
public ActionResult Cal(string pt)
{
    double a = double.Parse(Request["a"]);
}
```



Shared View

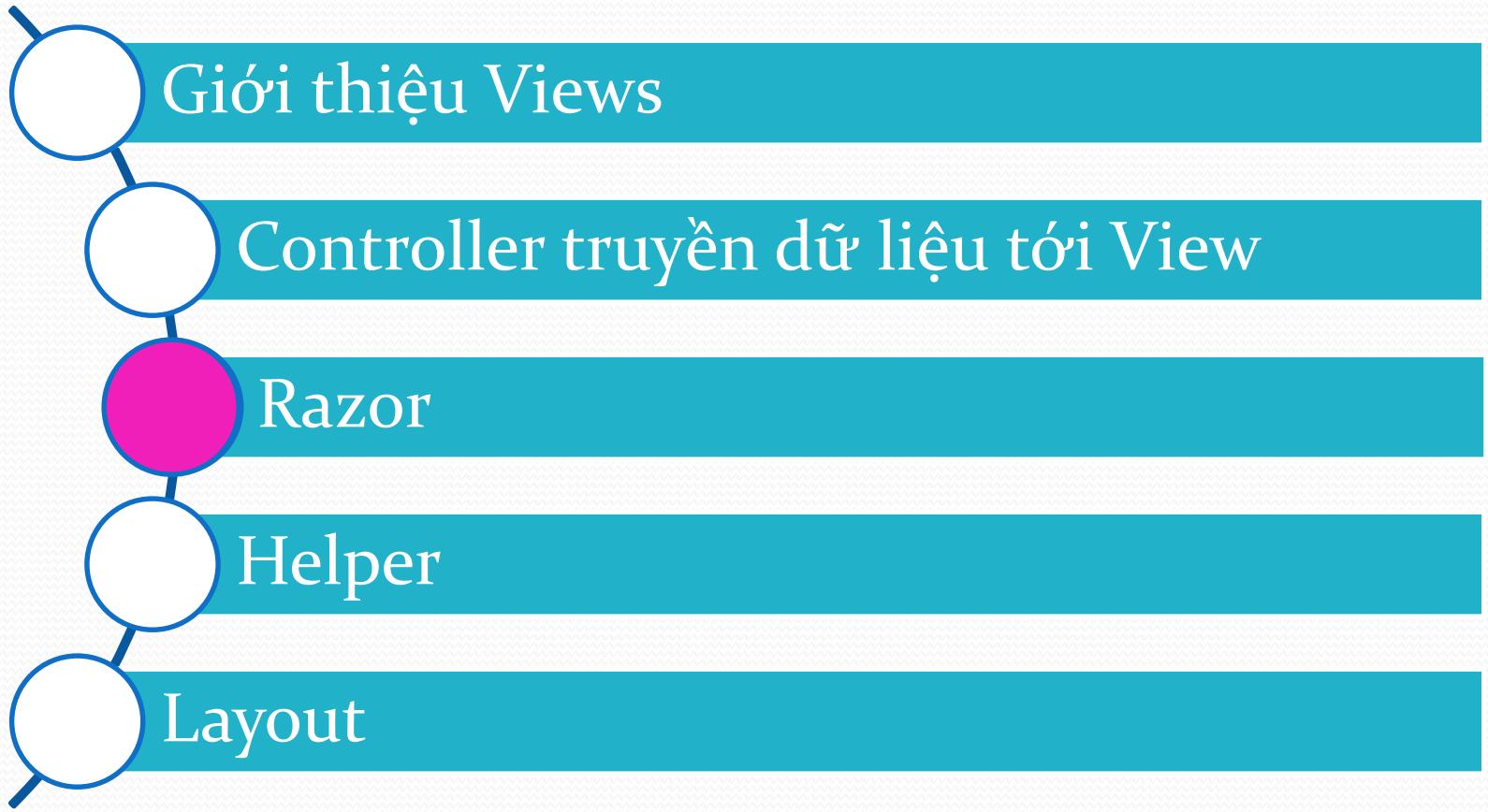
- View dùng chung cho nhiều Controller để trong thư mục Shared.

```
public class HomeController : Controller
{
    //shared view
    0 references
    public ActionResult Contact()
    {
        ViewBag.Message = "Your contact page.";
        ViewBag.phone = "0912345678";
        return View();
    }
}
0 references
```



```
public class SubjectController : Controller
{
    //shared view
    0 references
    public ActionResult Contact()
    {
        ViewBag.Message = "Your contact page.";
        ViewBag.phone = "1111111";
        return View();
    }
}
```

Nội dung



Razor

- **Razor** là ngôn ngữ ngắn gọn, rõ ràng và hữu ích cho phép tạo ra các giao diện ứng dụng ASP.NET MVC.
- Trong khối lệnh `@{...}` là mã C# hoặc VB.NET trộn HTML.
- Cú pháp Razor có các đặc điểm sau:
 - **Compact**: Razor là nhỏ gọn cho phép giảm thiểu số lượng ký tự và tổ hợp phím cần thiết để viết mã.
 - **Easy to Learn**: Razor dễ học có thể sử dụng các ngôn ngữ lập trình: C#, Visual Basic.
 - **Intellisense**: Razor hỗ trợ các câu lệnh trong Visual Studio.
 - **Unit Testable**: Razor hỗ trợ khả năng unit test cho các view mà không cần các controller hoặc web-server.

Razor

- **Cú pháp Razor:** Razor sử dụng ký tự @ để chuyển HTML sang C#.
- Có 2 cách để khai báo:
 - Sử dụng Razor expression
 - Sử dụng khối lệnh Razor
- Các biểu thức được xử lý bởi Razor View Engine và được gán vào response.

Razor

- **Khối lệnh:** có thể viết nhiều dòng lệnh phía máy chủ được đặt trong dấu ngoặc nhọn @ {...}. Mỗi dòng phải kết thúc bằng dấu chấm phẩy.

```
<!-- Khối lệnh đơn -->
@{ var message = "Hello World"; }
```

```
<!-- Biểu thức nội tuyến -->
<p>Giá trị của message là: @message</p>
```

```
<!-- Khối nhiều dòng mã lệnh -->
#{@
    var greeting = "Welcome to our site!";
    var weekDay = DateTime.Now.DayOfWeek;
    var greetingMessage = greeting + " Today is: " + weekDay;
}
<p>Lời chào là: @greetingMessage</p>
```

Razor

Khối mã	<pre>@{ int x = 123; string y = "because.>"; }</pre>
Biểu thức (đã mã hóa HTML)	<pre>@model.Message</pre>
Biểu thức (chưa mã hóa HTML)	<pre>@Html.Raw(model.Message)</pre>
Kết hợp text và HTML	<pre>@foreach(var item in items) { @item.Prop }</pre>
Trộn code và text	<pre>@if (foo) { <text>Plain Text</text> }</pre>
Trộn code và text	<pre>@if (foo) { @:Plain Text is @bar }</pre>

Razor

Khối using	<pre>@ using (Html.BeginForm()) { <input type="text" value="input here"> }</pre>
Địa chỉ email	Hi philha@example.com
Biểu thức (tường minh)	<code>ISBN@(isbnNumber)</code>
Mã hóa ký hiệu @	<code>In Razor, you use the @@foo to display the value of foo</code>
Chú thích phía server	<code>@* This is a server side multiline comment *@</code>
Trộn biểu thức và text	Hello @title. @name.

Razor

- Ví dụ hiển thị văn bản trong khối lệnh: Sử dụng @: hoặc <text> / <text>

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

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

Razor

- Ví dụ câu lệnh if-else: bắt đầu bằng ký hiệu @. Khối mã if-else phải được đặt trong dấu ngoặc {}.

```
@if(DateTime.IsLeapYear(DateTime.Now.Year) )  
{  
    @DateTime.Now.Year @:is a leap year;  
}  
else  
{  
    @DateTime.Now.Year <text> is not a leap year. </text>  
}
```

- Tương tự cho các câu lệnh lặp.

Razor

- Ví dụ khai báo và sử dụng biến:

```
@{  
    string str = "";  
    if(1 > 0)  
    {  
        str = "Hello World!";  
    }  
}  
<p>@str</p>
```

Razor

- Ví dụ: Thực hiện các phép toán số học trên 2 số a, b sử dụng Razor
- Phương thức Action trong TestRazorController

```
public class TestRazorController : Controller
{
    //thuc hien phap toan
    public ActionResult Cal()
    {
        return View();
    }
    [HttpPost]
    public ActionResult Cal(string i)
    {
        return View();
    }
}
```

Razor

- Ví dụ: Thực hiện các phép toán số học trên 2 số a, b
- Cal.cshtml trong Views/TesRazorController

```
@{ Layout = null;
    var Result = "";
    if (IsPost)
    {
        float a = float.Parse(Request["a"]);
        float b = Request["b"].AsFloat();
        char pt = char.Parse(Request["pt"]);
        switch (pt)
        {
            case '+': Result = (a + b).ToString(); break;
            case '-': Result = (a - b).ToString(); break;
            case '*': Result = (a * b).ToString(); break;
            case '/': Result = (a / b).ToString(); break;
        }
    }
}
```

Razor

- Ví dụ: Thực hiện các phép toán số học trên 2 số a, b
- Cal.cshtml trong Views/TesRazorController

```
<!DOCTYPE html>
<html>
<head>  <meta name="viewport" content="width=device-width" />
        <title>TestRazor</title>
</head>
<body>
    <form name="f1" action="/TestRazor/Cal" method="post">
        <div class="container">
            <div class="form">
                <h2>THỰC HIỆN PHÉP TOÁN CƠ BẢN</h2>
                Nhập a: <input type="text" name="a" /> <br />
                Nhập b: <input type="text" name="b" /> <br />
                <input type="radio" checked name="pt" value="+" />Cộng &nbsp;
                <input type="radio" name="pt" value="-" />Trừ &nbsp;
                <input type="radio" name="pt" value="*" />Nhân&nbsp;
                <input type="radio" name="pt" value="/" />Chia&nbsp;<br />
                <input type="submit" value="Tính toán" /><br />
                Kết quả: <input type="text" name="kq" value=@Result />
            </div>
        </div>
    </form>
</body>
</html>
```

Razor

- Sử dụng Razor viết mã lệnh trong file.cshtml để thực hiện giải phương trình bậc nhất.

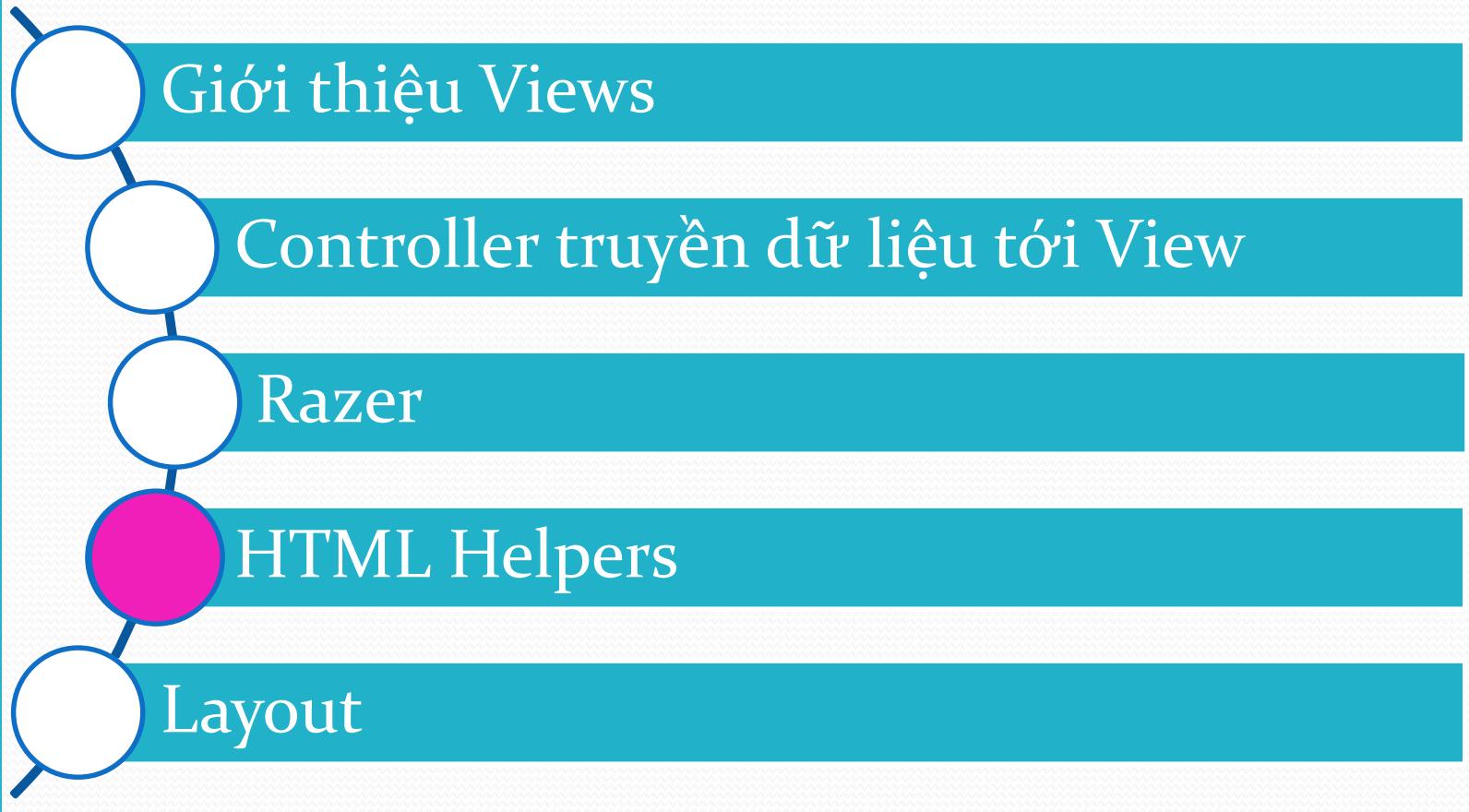
GIẢI PHƯƠNG TRÌNH BẬC NHẤT

Nhập a:

Nhập b:

Kết quả:

Nội dung



4. HTML Helpers

- HTML Helpers là các **phương thức** trả về HTML được sử dụng trong View.
- HTML Helpers: các thành phần **sinh giao diện web** phù hợp buộc dữ liệu với model để duy trì thông tin.
- **HTML Helper** giúp **viết phần tử HTML trong Razor** sử dụng cú pháp thân thiện với HTML, code được xử lý bởi Razor Engine trên server.
- Được dùng để **tạo ra (render)** các thẻ **HTML** thay vì viết tay => Đơn giản việc viết mã sinh giao diện.
- Trong MVC có built-in HTML helpers, ngoài ra có thể tạo các HTML helpers tùy chỉnh.

4. HTML Helpers

➤ Các loại HTML Helpers:

- **Inline** HTML Helper
- **Built-in** HTML Helper
 - Standard HTML helper
 - Strongly Typed HTML helper
 - Templated HTML Helper
- **Custom** HTML Helper

Inline HTML Helpers

- Đây là loại helper được sử dụng trên **một View đơn** và được sử dụng trên cùng một trang.
- HTML Helper nội tuyến có thể được tạo bằng thẻ **@helper**.

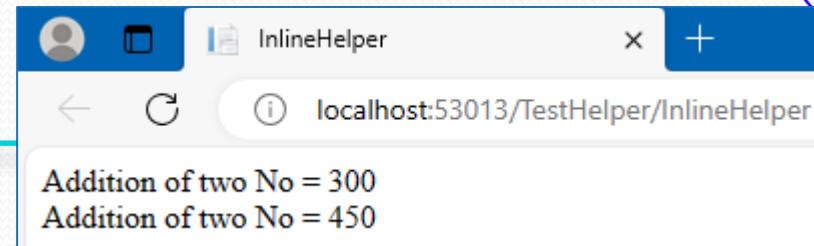
```
@helper HelperName(parameters)
{
    // code
}
```

```
//sử dụng helper
@HelperName(parameters)
```

Inline HTML Helpers

➤ Ví dụ

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>InlineHelper</title>
</head>
<body>
    <div>
        @helper AddHelper(int a, int b)
        {
            <label> Addition of two No = @(a + b)</label>
        }
    </div>
    <div><label> @AddHelper(100, 200) </label> </div>
    <div><label> @AddHelper(150, 300) </label> </div>
</body>
</html>
```

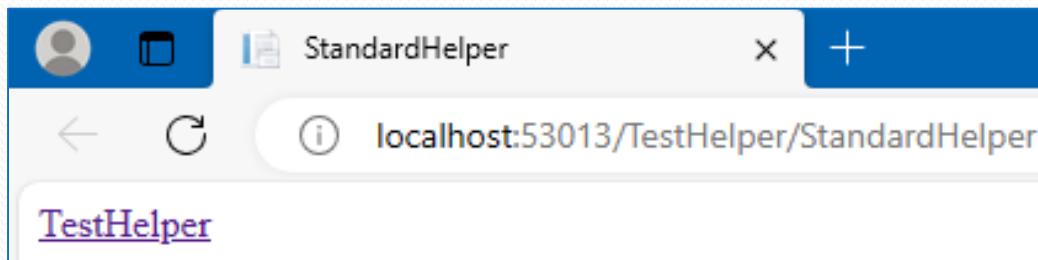


Standard HTML helpers

- Được sử dụng để hiển thị các loại phần tử HTML phổ biến như HTML Label, TextBox, Password, TextArea, CheckBox, RadioButton, DropDownList, Listbox, Display, Editor, ActionLink.
- **ActionLink**: chuyển hướng đến trang sẽ hiển thị.

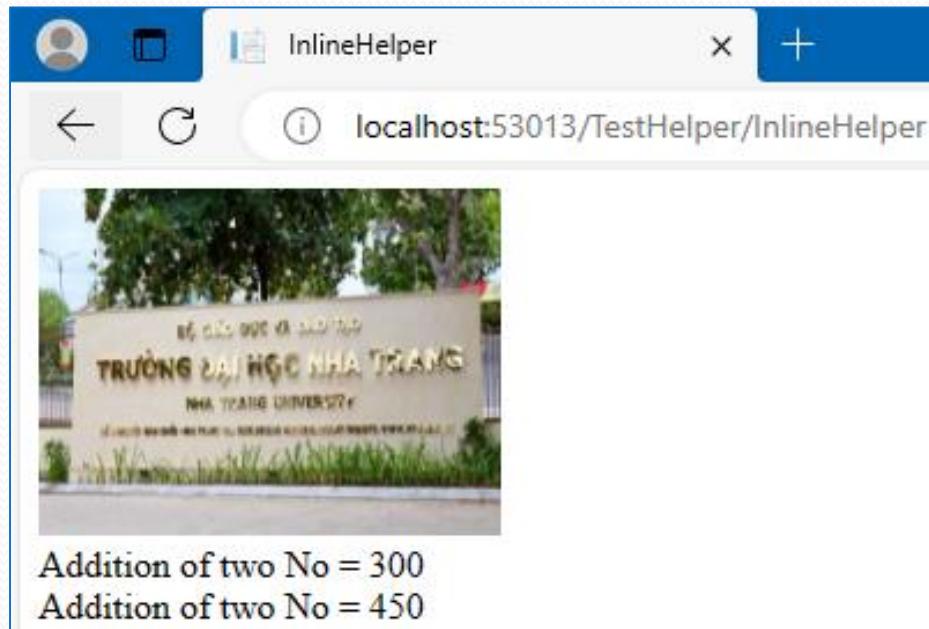
```
@Html.ActionLink("link text", "action", "controller")
```

- Ví dụ hiển thị chuỗi “TestHelper” là link tới trang InlineHelper



Standard HTML helpers

```
@Html.ActionLink("TestHelper", "InlineHelper", "TestHelper")
```



Standard HTML helpers

- `@Html.ActionLink()` được sử dụng để **sinh liên kết**

```
@Html.ActionLink("Giới thiệu", "About" )  
<a href="/Home/About">Giới thiệu</a>
```

- `@Html.ActionLink()` nhận một số tham số:

- **linkText** – nhãn của liên kết
- **actionName** – tên action
- **routeValues** – tập các giá trị truyền đến action.
- **controllerName** – tên controller
- **htmlAttributes** – tập thuộc tính HTML của thẻ `<a>`

- Ví dụ:

```
@Html.ActionLink("Edit Record", "Edit", new {Id=3})  
<a href="/Store/Edit/3">Edit Record</a>
```

- Liên kết chứa ảnh

```
<a href="@Url.Action("Delete")">  
</a>
```

Standard HTML helpers

- **Form**: để tạo một HTML form có thể sử dụng phương thức **BeginForm()** và **EndForm()** của lớp HTML helper.

```
<div>
  @{
    Html.BeginForm("action", "controller", "POST");

    Html.EndForm();
  }
</div>
```

- **Textbox**: để tạo một textbox có thể sử dụng phương thức **TextBox()** của lớp HTML helper.

```
@{
  Html.TextBox("name");
}
```



Standard HTML helpers

➤ Checkbox

```
@{  
    Html.CheckBox("name", true);  
}
```

➤ Radio Button

```
@{  
    Html.RadioButton("name", "This is name", true);  
}
```

➤ DropDownList

```
@{  
    Html.DropDownList("Subject", new SelectList(new []  
        {"OOP", "Web", "TP"}));  
}
```

Standard HTML helpers

Helper	HTML
@Html.BeginForm()	Sinh thẻ <form> bắt đầu
@Html.EndForm()	Sinh thẻ </form> kết thúc
@Html.CheckBox()	Sinh thẻ <input type="checkbox">
@Html.Hidden()	Sinh thẻ <input type="hidden">
@Html.Password()	Sinh thẻ <input type="password">
@Html.RadioButton()	Sinh thẻ <input type="radio">
@Html.TextArea()	Sinh thẻ <textarea></textarea>
@Html.TextBox()	Sinh thẻ <input type="text">
@Html.DropDownList()	Sinh thẻ <select><option></select>
@Html.ListBox()	Sinh thẻ <select multiple><option></select>

Standard HTML helpers

Full Name	<pre>@{Html.BeginForm("Action", "Controller");} <div>Full Name</div> @Html.TextBox("FullName")</pre>
Password	<pre><div>Password</div> @Html.Password("Password")</pre>
Photo	<pre><div>Photo</div> <input ><="" name="Photo" pre="" type="file"/></pre>
Chọn tệp Không có tệp	
Married Status	<pre><div>Married Status</div> <label>@Html.CheckBox("Status") Single</label></pre>
<input checked="" type="checkbox"/> Single	
Gender	<pre><div>Gender</div> <label>@Html.RadioButton("Gender", true) Male</label> <label>@Html.RadioButton("Gender", false) Female</label></pre>
Male Female	
Description	<pre><div>Description</div> @Html.TextArea("Description")</pre>
	<pre>@Html.Hidden("Active")</pre>
	<pre><hr /></pre>
Submit	<pre><input type="submit" value="Submit" /> @{Html.EndForm();}</pre>

Strongly HTML Helpers

- **Strongly HTML Helpers** hoạt động trên các biểu thức **lambda**.
- Đối tượng mô hình được truyền dưới dạng **giá trị** cho biểu thức **lambda**, có thể chọn **trường/ thuộc tính** từ **đối tượng** mô hình sẽ được sử dụng để đặt các thuộc tính id, tên và giá trị của HTML Helper.
- **TextBox:**
 - Dùng **TextBoxFor(lambda expression)**
 - Ví dụ
 - `@Html.TextBoxFor(model => model.Id)`

Strongly HTML Helpers

➤ Danh sách Strongly HTML Helpers:

- @Html.HiddenFor()
- @Html.LabelFor()
- @Html.TextBoxFor()
- @Html.RadioButtonFor()
- @Html.DropDownListFor()
- @Html.CheckBoxFor()
- @Html.TextAreaFor()
- @Html.PasswordFor()
- @Html.ListBoxFor()

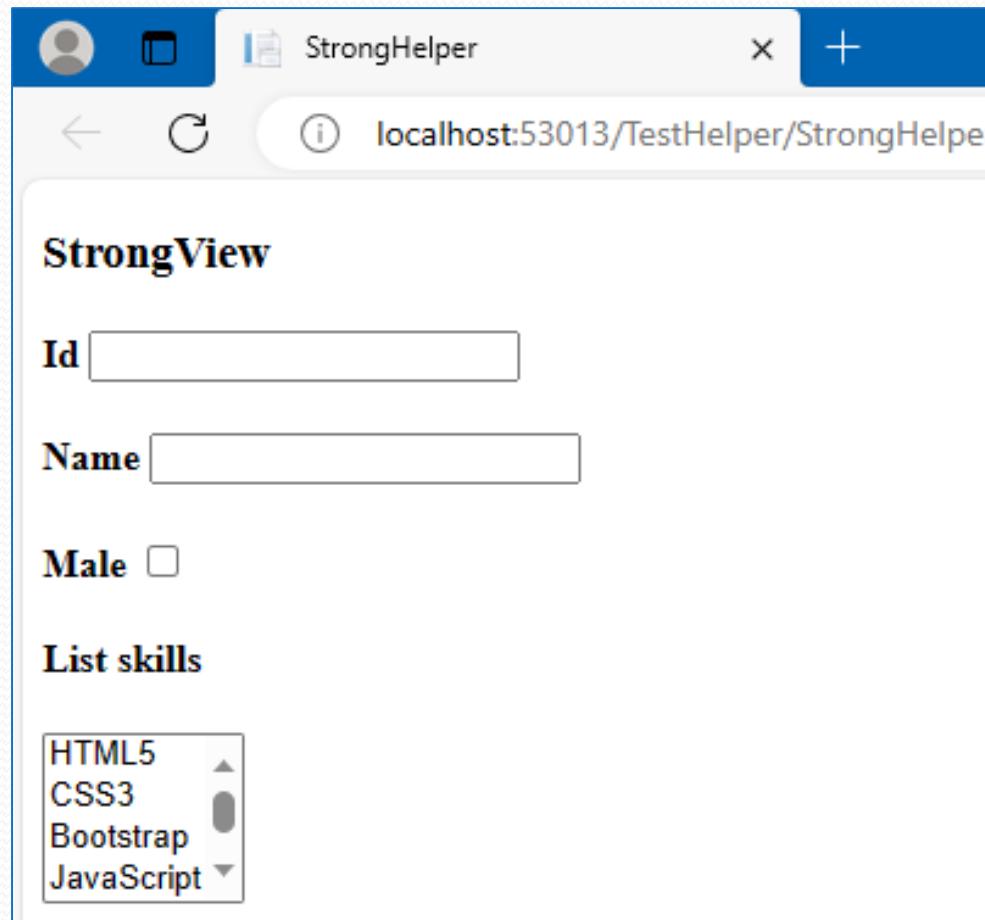
Strongly HTML Helpers

- ❑ Chỉ định loại control đối với các thuộc tính

Helper	Mô tả
@Html.TextBoxFor (m => m.Id)	<input type="text" name="Id" id="Id">
@Html.PasswordFor (m => m.Pwd)	<input type="password" name="Pwd" id="Pwd">
@Html.TextAreaFor (m => m.Notes)	<textarea name="Notes" id="Notes"></textarea>
@Html.CheckBoxFor (m => m.Status)	<input type="checkbox" name="Status" id="Status">
@Html.RadioButtonFor (m => m.Gender)	<input type="radio" name="Gender" id="Gender">
@Html.HiddenFor (m=> m.Name)	<input type="hidden" name="Name" id="Name">
@Html.DropDownListFor (m=> m.Blood)	<select id="Blood" name="Blood">...</select>
@Html.ListBoxFor (m=> m.Jobs)	<select id="Jobs" name="Jobs" multiple>...</select>
@Html.LabelFor (m=> m.Name)	<label for="Name"> Name </label>

Strongly HTML Helpers

➤ Ví dụ



The screenshot shows a web browser window titled "StrongHelper". The URL bar displays "localhost:53013/TestHelper/StrongHelper". The page content is titled "StrongView". It contains the following fields:

- Id**: An input text field.
- Name**: An input text field.
- Male**: A checkbox labeled "Male" with an unchecked state.
- List skills**: A dropdown menu containing the following items:
 - HTML5
 - CSS3
 - Bootstrap
 - JavaScript

Strongly HTML Helpers

- Tạo class StrongView trong thư mục Models

```
namespace WebApp1.Models
{
    //sử dụng strongly Html Helper
    public class StrongView
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Gender { get; set; }
        public skills skills { get; set; }
    }
    public enum skills
    {
        HTML5,
        CSS3,
        Bootstrap,
        MVC,
        WebAPI
    }
}
```

Strongly HTML Helpers

- Tạo phương thức **StrongHelper** trong class **TestHelperController** trong thư mục Controllers

```
namespace WebApp1.Controllers
{
    public class TestHelperController : Controller
    {
        //test strongly HtmlHelper
        public ActionResult StrongHelper()
        {
            return View();
        }
        ...
    }
}
```

Strongly HTML Helpers

- Tạo **StrongHelper.cshtml** trong thư mục Views

```
@model WebApp1.Models.EmployeeModel
@{ Layout = null;}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>StrongHelper</title>
</head>
<body>
    <div>
        <h3>StrongView</h3>
        <h4>Id @Html.TextBoxFor(model => model.Id)</h4>
        <h4>Name @Html.TextBoxFor(model => model.Name)</h4>
        <h4>Male @Html.CheckBoxFor(model => model.Gender)</h4>
        <h4> List skills </h4>
        @Html.ListBoxFor(model => model.skills, new
            SelectList(Enum.GetValues(typeof(WebApp1.Models.skills))))
    </div>
</body>
</html>
```

Templated HTML Helper

- **Templated HTML Helper** được sử dụng để **nhập** và **hiển thị** dữ liệu.
- Nó **tự động** tạo HTML theo **thuộc tính** mô hình.
- Hai loại mẫu HTML Helper:
 - **Display Template**
 - **Editor Template**

Templated HTML Helper

➤ Display Template:

- @Html.Display()
- @Html.DisplayFor()
- @Html.DisplayName()
- @Html.DisplayNameFor()
- @Html.DisplayText()
- @Html.DisplayTextFor()
- @Html.DisplayModelFor()

➤ Editor Template:

- @Html.Editor()
- @Html.EditorFor()
- @Html.EditorForModel()

Templated HTML Helper

- ❑ `@Html.DisplayNameFor (m=> m.Property)`
 - ↳ Hiển thị tên của thuộc tính Property
- ❑ `@Html.DisplayFor (m=>m.Property)`
 - ↳ Hiển thị giá trị cho thuộc tính Property
- ❑ `@Html.DisplayForModel ()`
 - ↳ Hiển thị giá trị của tất cả các thuộc tính
- ❑ `@Html.Display ("Mail")`
 - ↳ Hiển thị giá trị của tất cả các thuộc tính của đối tượng trong ViewData hay ViewBag

Helper	Mô tả
<code>Html.EditorFor(m=>m.Property)</code>	Sinh 1 control cho 1 thuộc tính.
<code>Html.EditorForModel()</code>	Sinh toàn form theo các thuộc tính của Model
<code>Html.Editor(object)</code>	Sinh toàn form theo các thuộc tính của Object đặt trong ViewBag

Ví dụ Templatized HTML Helper

- Tạo phương thức **TemplateHelper** trong class **TestHelperController** trong thư mục Controllers

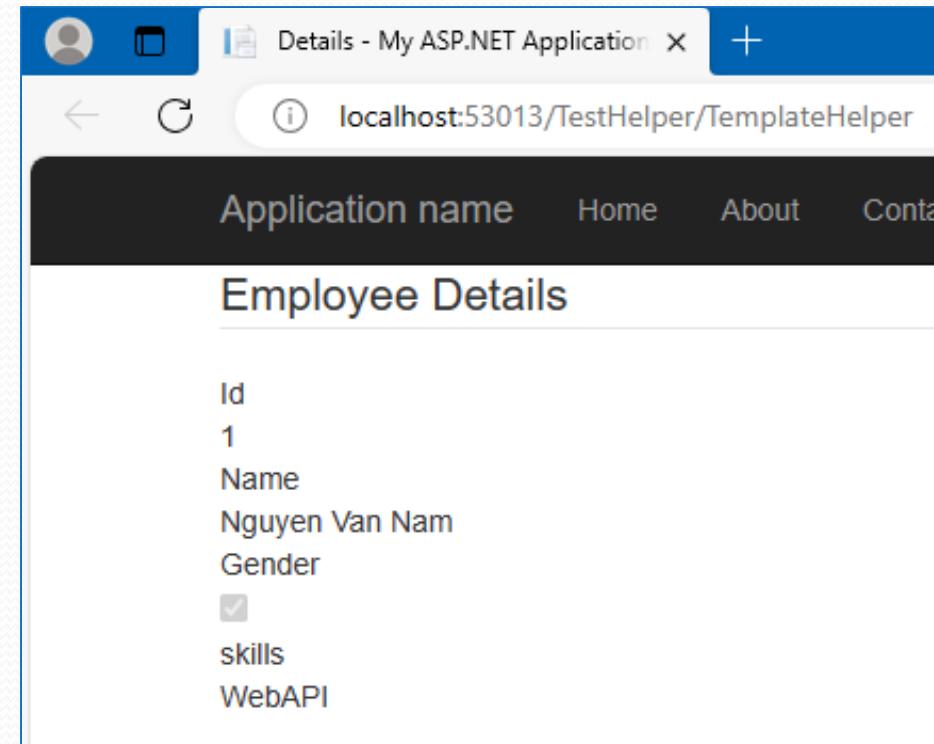
```
namespace WebApp1.Controllers
{
    public class TestHelperController : Controller
    {
        //test templated HtmlHelper
        public ActionResult TemplateHelper()
        {
            EmployeeModel employee = new EmployeeModel()
            {
                Id = 1,
                Name = "Nguyen Van Nam",
                Gender = true,
                skills = skills.WebAPI
            };
            ViewData["EmployeeData"] = employee;
            return View();
        }
    }
}
```

Templated HTML Helper

- Tạo TemplateHelper.cshtml trong thư mục Views

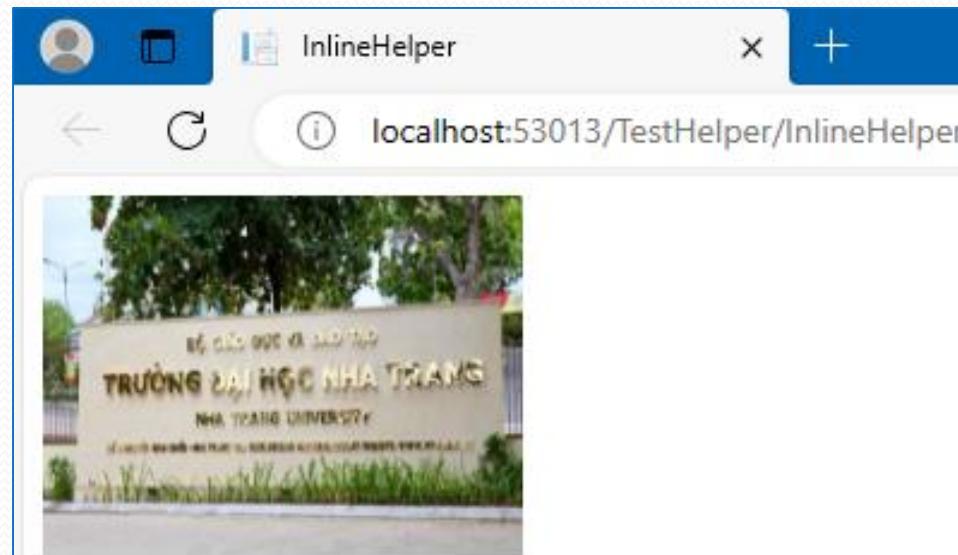
```
@{  
    ViewBag.Title = "Details";  
}  
<fieldset>  
    <legend> Employee Details</legend>  
    @Html.Display("EmployeeData")  
</fieldset>
```

- Kết quả



Custom HTML helpers

- Cách tạo Custom HTML Helpers:
 - Sử dụng **static methods**
 - Sử dụng **extension methods**
- Ví dụ dùng **static methods** để hiển thị file hình ảnh



Custom HTML helpers

➤ Các bước thực hiện:

- Tạo thư mục **CustomHelper**
- Tạo **class CustomHelper** trong thư mục CustomHelper, Tạo một phương thức tĩnh (static methods) trả về chuỗi được mã hóa HTML sử dụng MvcHtmlString.
- Gọi CustomHelper từ **View**.

Custom HTML helpers

```
namespace WebApp1.CustomHelper
{
    public class CustomHelper
    {
        public static MvcHtmlString Image(string source, string altTxt, string
width, string height)
        {
            //TagBuilder creates a new tag with the tag name specified
            var ImageTag = new TagBuilder("img");
            //MergeAttribute Adds attribute to the tag
            ImageTag.MergeAttribute("src", source);
            ImageTag.MergeAttribute("alt", altTxt);
            ImageTag.MergeAttribute("width", width);
            ImageTag.MergeAttribute("height", height);
            //Return an HTML encoded string with SelfClosing TagRenderMode
            return
                MvcHtmlString.Create(ImageTag.ToString(TagRenderMode.SelfClosing));
        }
    }
}
```

Custom HTML helpers

```
@using WebApp1.CustomHelper
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>InlineHelper</title>
</head>
<body>
    <div>
        <div>
            @CustomHelper.Image("/Images/NTU.jpg", "NTUPic", "200", "150")
        </div>
        @helper AddHelper(int a, int b)
        {
            <label> Addition of two No = @(a + b)</label>
        }
    </div>
    <div><label> @AddHelper(100, 200) </label> </div>
    <div><label> @AddHelper(150, 300) </label> </div>
</body>
</html>
```

Helper định dạng

Helper	Mô tả
@Html.FormatValue (value, format)	Định dạng một giá trị số, chuỗi hoặc thời gian
@String.Format(format, value1, value2...)	Định dạng nhiều giá trị hỗ hợp
@Html.Raw (html)	Giải mã chuỗi đã mã hóa HTML

- Số bình thường: 12345.8765
- Phân nhóm: 12,345.877
- Tiền tệ: \$12,345.88
- Phần trăm: 72.00 %

- Ngày bình thường: 5/27/2014 9:26:09 PM
- Định dạng D: Tuesday, May 27, 2014
- Định dạng ISO: 2014-05-27
- Định dạng English: 05/27/2014
- Định dạng 24 giờ: 21:26:09
- Định dạng 12 giờ: 09:26:09 PM

- Có mã hóa HTML: Hello
- Không mã hóa HTML : Hello

Helper định dạng

Ký hiệu	Mô tả
{0:C}	Currency – tiền tệ theo ngôn ngữ
{0:P}	Percent – số phần trăm
{0:#,###.##0}	Number – số phân nhóm và 3 số lẻ

```
@{
    var number1 = 12345.8765;
    var number2 = 0.72;
}
<ul>
    <li>Số bình thường: @number1</li>
    <li>Phân nhóm: @Html.FormatValue(number1, "{0:#,###.##0}")</li>
    <li>Tiền tệ: @Html.FormatValue(number1, "{0:c}")</li>
    <li>Phần trăm: @Html.FormatValue(number2, "{0:p}")</li>
</ul>
```



- Số bình thường: 12345.8765
- Phân nhóm: 12,345.877
- Tiền tệ: \$12,345.88
- Phần trăm: 72.00 %

Helper định dạng

Ký hiệu	Mô tả
{0:D}	Date – theo ngôn ngữ được chọn
{0:MMMM-dd-yyyy hh:mm:ss tt}	<ul style="list-style-type: none"> ✓ M,MM,MMM,MMMM: tháng 1, 2 ký tự số, 3 ký tự viết tắt, tên tháng đầy đủ ✓ d,dd: ngày 1, 2 ký tự ✓ yy,yyyy: năm 2, 4 ký tự số ✓ H, HH, h, hh: 1,2 ký tự giờ 24 hoặc 12 giờ mỗi ngày ✓ m,mm: 1,2 ký tự số phút ✓ s,ss: 1,2 ký tự số giây ✓ tt: 2 ký tự sáng/chiều

- Ngày bình thường: 5/27/2014 9:26:09 PM
- Định dạng D: Tuesday, May 27, 2014
- Định dạng ISO: 2014-05-27
- Định dạng English: 05/27/2014
- Định dạng 24 giờ: 21:26:09
- Định dạng 12 giờ: 09:26:09 PM

```

@{
    var now = DateTime.Now;
}
<ul>
    <li>Ngày bình thường: @now</li>
    <li>Định dạng D: @Html.FormatValue(now, "{0:D}")</li>
    <li>Định dạng ISO: @Html.FormatValue(now, "{0:yyyy-MM-dd}")</li>
    <li>Định dạng English: @Html.FormatValue(now, "{0:MM/dd/yyyy}")</li>
    <li>Định dạng 24 giờ: @Html.FormatValue(now, "{0:HH:mm:ss}")</li>
    <li>Định dạng 12 giờ: @Html.FormatValue(now, "{0:hh:mm:ss tt}")</li>
</ul>

```

Ví dụ

❑ Dựa vào các đặc điểm của thuộc tính trong lớp model để sinh ra giao diện người dùng.

- Sinh các control tương minh
- Sinh các control ngầm định

```
public class Student
{
    [DisplayName("Mã sinh viên")]
    public String Id { get; set; }
    [DisplayName("Mật khẩu")]
    public String Password { get; set; }
    [DisplayName("Họ và tên")]
    public String FullName { get; set; }
    [DisplayName("Giới tính")]
    public bool Gender { get; set; }
    [DisplayName("Ngày sinh")]
    public DateTime Birthday { get; set; }
    [DisplayName("Ghi chú")]
    public String Notes { get; set; }
}
```



Đăng ký thành viên

Mã sinh viên	<input type="text"/>
Mật khẩu	<input type="password"/>
Họ và tên	Nguyễn Nghiêm
Giới tính	<input checked="" type="radio"/> Nam <input type="radio"/> Nữ
Ngày sinh	9/1/1971 12:00:00 AM
Ghi chú	<input type="text"/>
<input type="button" value="Register"/>	

Ví dụ

```

@model Mvc5CodeDemo.Models.Student
<h2>Đăng ký thành viên</h2>
@using (Html.BeginForm())
{
    <table><tr>
        <td>@Html.LabelFor(m => m.Id)</td>
        <td>@Html.TextBoxFor(m => m.Id)</td>
    </tr><tr>
        <td>@Html.LabelFor(m => m.Password)</td>
        <td>@Html.PasswordFor(m => m.Password)</td>
    </tr><tr>
        <td>@Html.LabelFor(m => m.FullName)</td>
        <td>@Html.TextBoxFor(m => m.FullName)</td>
    </tr><tr>
        <td>@Html.LabelFor(m => m.Gender)</td>
        <td>
            <label>@Html.RadioButtonFor(m => m.Gender, true) Nam</label>
            <label>@Html.RadioButtonFor(m => m.Gender, false) Nữ</label>
        </td>
    </tr><tr>
        <td>@Html.LabelFor(m => m.Birthday)</td>
        <td>@Html.TextBoxFor(m => m.Birthday)</td>
    </tr><tr>
        <td>@Html.LabelFor(m => m.Notes)</td>
        <td>@Html.TextAreaFor(m => m.Notes)</td>
    </tr><tr>
        <td>&nbsp;</td>
        <td><input type="submit" value="Register" /></td>
    </tr></table>
}

```

Kiểu của Model

Sinh <label for="Id">Mã sinh viên</label>

Sinh <input type="text" name="Id" id="Id"> từ thuộc tính Id của Model

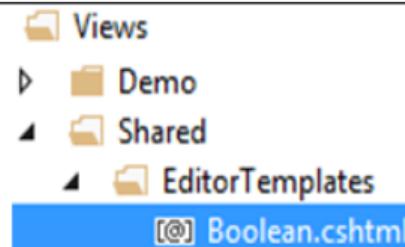
Ví dụ

- ❑ Tự sinh loại control phù hợp với đặc điểm của thuộc tính của lớp model.

Helper	Mô tả
Html.EditorFor(m=>m.Property)	Sinh 1 control cho 1 thuộc tính.
Html.EditorForModel()	Sinh toàn form theo các thuộc tính của Model
Html.Editor(object)	Sinh toàn form theo các thuộc tính của Object đặt trong ViewBag

Bổ sung thêm template để hiển thị giới tính dạng RadioButtonList bằng cách thêm Boolean.cshtml vào thư mục Views/Shared/EditorTemplates

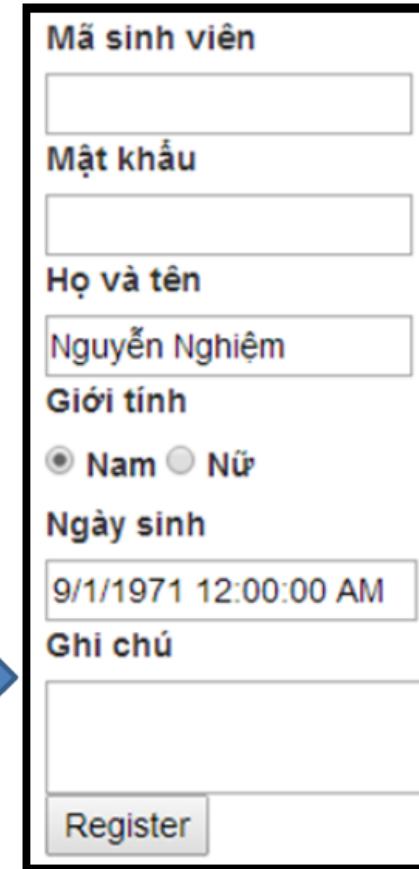
```
@model Boolean
<label>@Html.RadioButton("Gender", true, @Model == true)
Nam</label>
<label>@Html.RadioButton("Gender", false, @Model == false)
Nữ</label>
```



Ví dụ

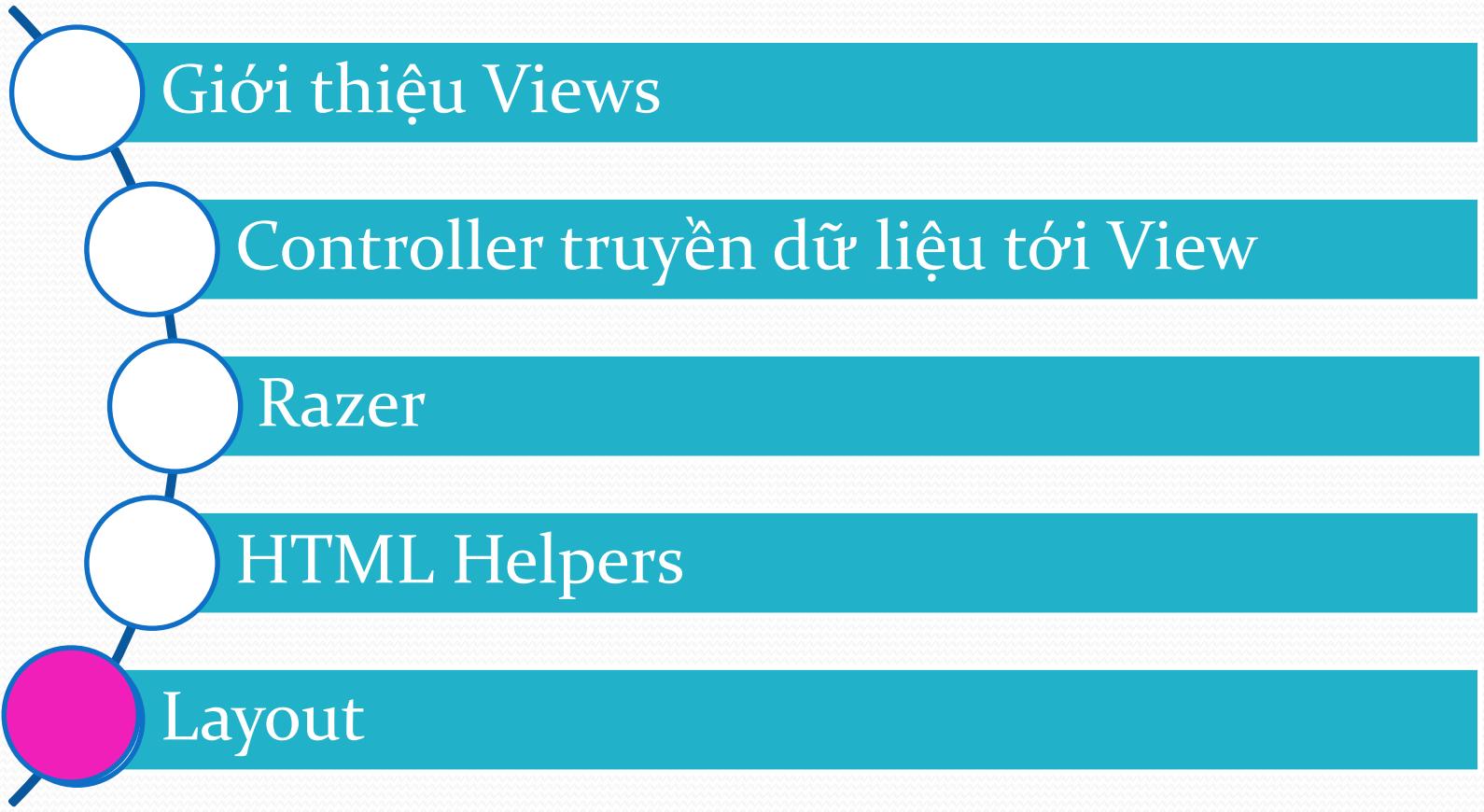
```
public class Student
{
    [DisplayName("Mã sinh viên")]
    public String Id { get; set; }
    [DisplayName("Mật khẩu"), DataType(DataType.Password)]
    public String Password { get; set; }
    [DisplayName("Họ và tên")]
    public String FullName { get; set; }
    [DisplayName("Giới tính")]
    public bool Gender { get; set; }
    [DisplayName("Ngày sinh")]
    public DateTime Birthday { get; set; }
    [DisplayName("Ghi chú"), DataType(DataType.MultilineText)]
    public String Notes { get; set; }
}
```

```
@model Mvc5CodeDemo.Models.Student
@using (Html.BeginForm())
{
    @Html.EditorForModel()
    <input type="submit" value="Register" />
}
```



The diagram illustrates the mapping between the `Student` model and the corresponding view code. On the left, the `Student` class is shown with its properties and their `[DisplayName]` attributes. In the middle, the `View` code uses `EditorForModel()` to generate input fields for each property. An arrow points from this code to the right, where a screenshot of a registration form is displayed. The form contains fields for `Mã sinh viên`, `Mật khẩu`, `Họ và tên` (with the value `Nguyễn Nghiêm` entered), `Giới tính` (with radio buttons for `Nam` and `Nữ`), `Ngày sinh` (with the value `9/1/1971 12:00:00 AM` entered), and `Ghi chú`. A `Register` button is at the bottom.

Nội dung



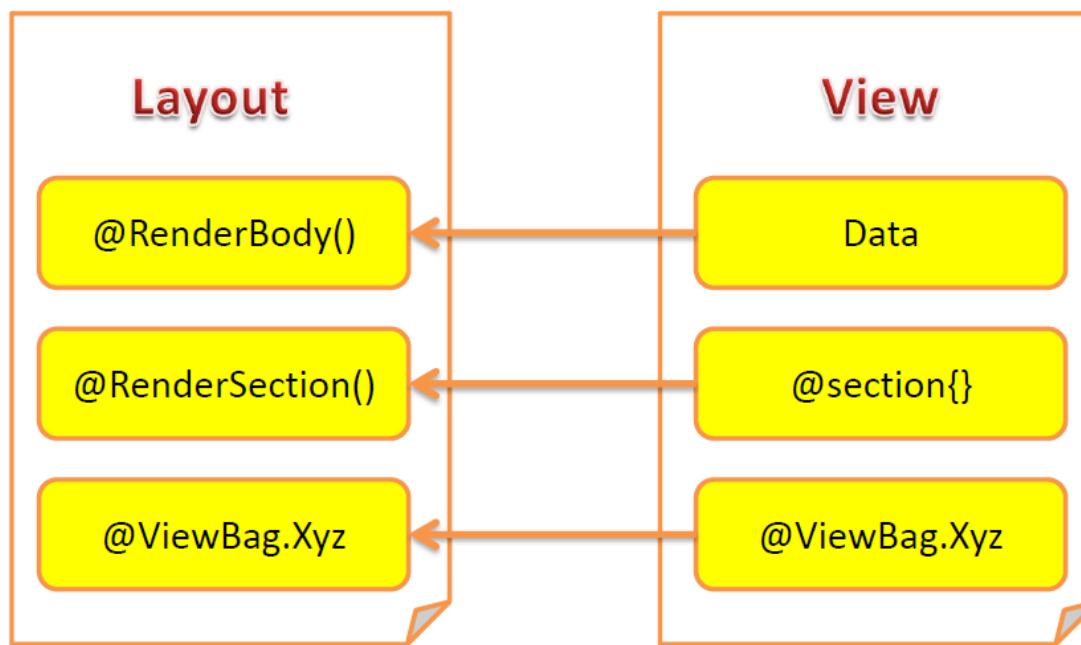
Layout

- ❑ Layout trong MVC là Masterpage trong Webform
- ❑ Layout được thiết kế và sử dụng các style CSS đã học
- ❑ Layout trong MVC chứa trong thư lục **Views/Shared** với phần mở rộng là **.cshtml**

Layout

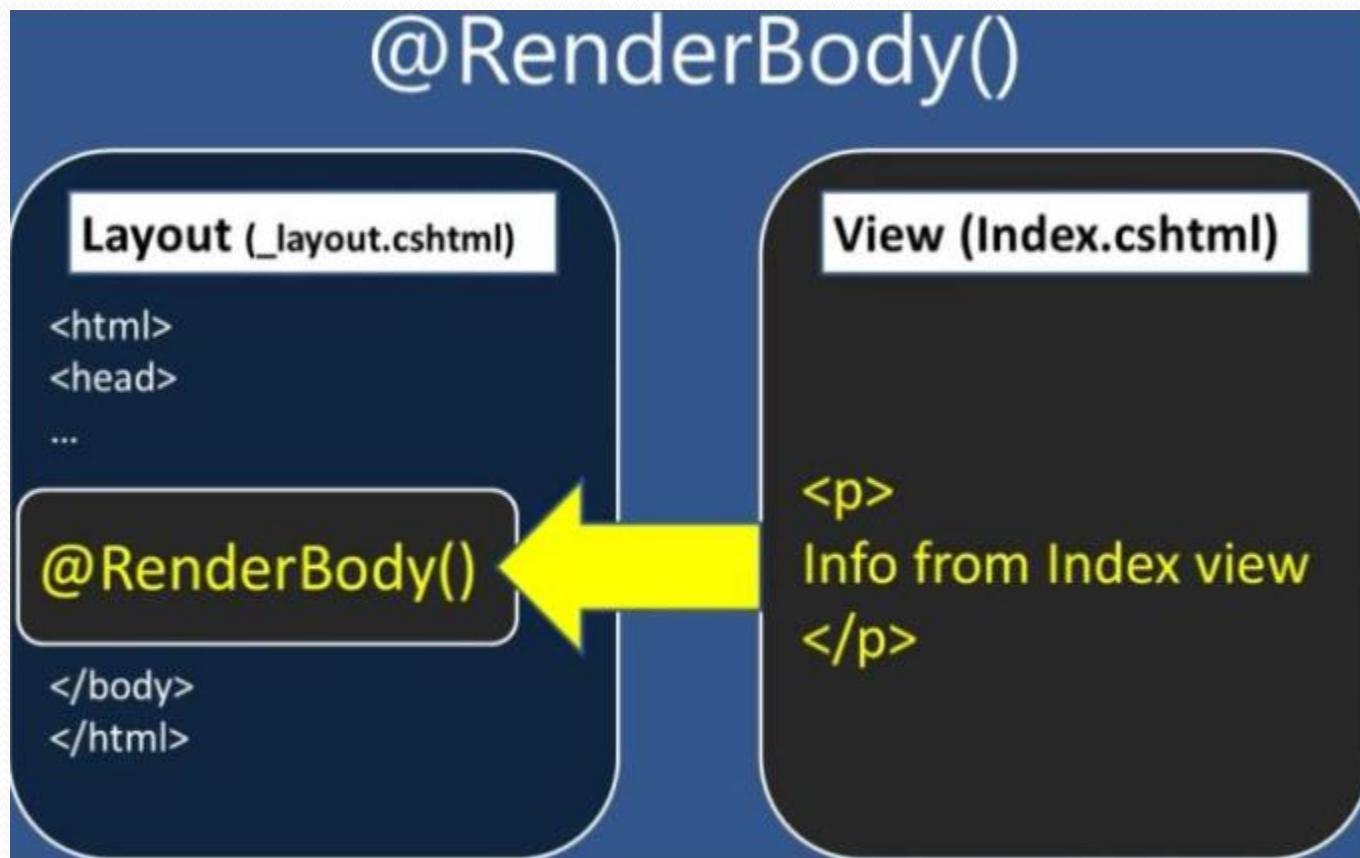
❑ Layout chứa

- ☞ **Một và chỉ một @RenderBody()** để giữ chỗ cho nội dung trong view
- ☞ **Không hoặc nhiều @RenderSection()** để giữ chỗ cho các phần được đánh dấu @section trong view



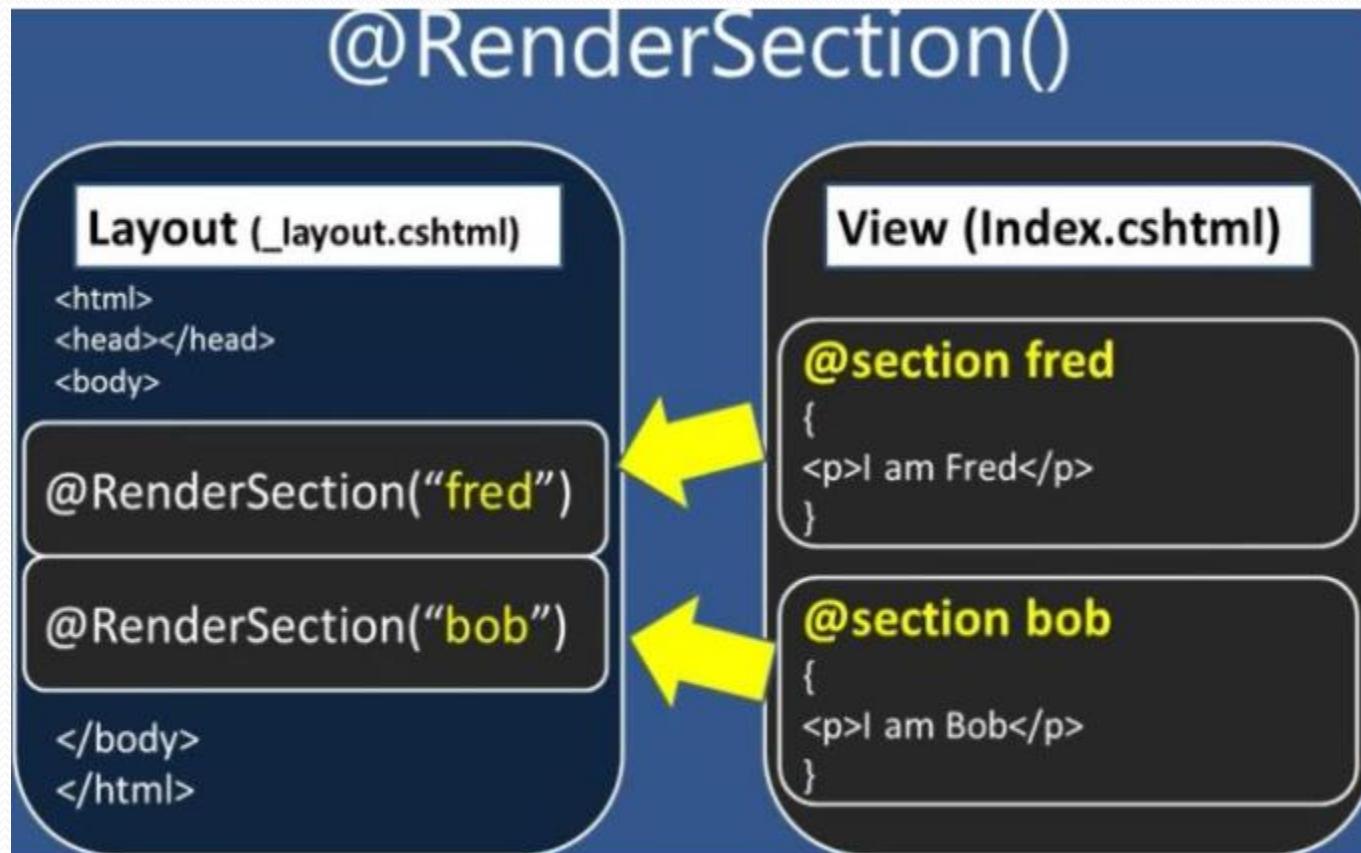
Layout

- Tất cả nội dung của những view mà kế thừa 1 layout nào đó sẽ chứa trong RenderBody().



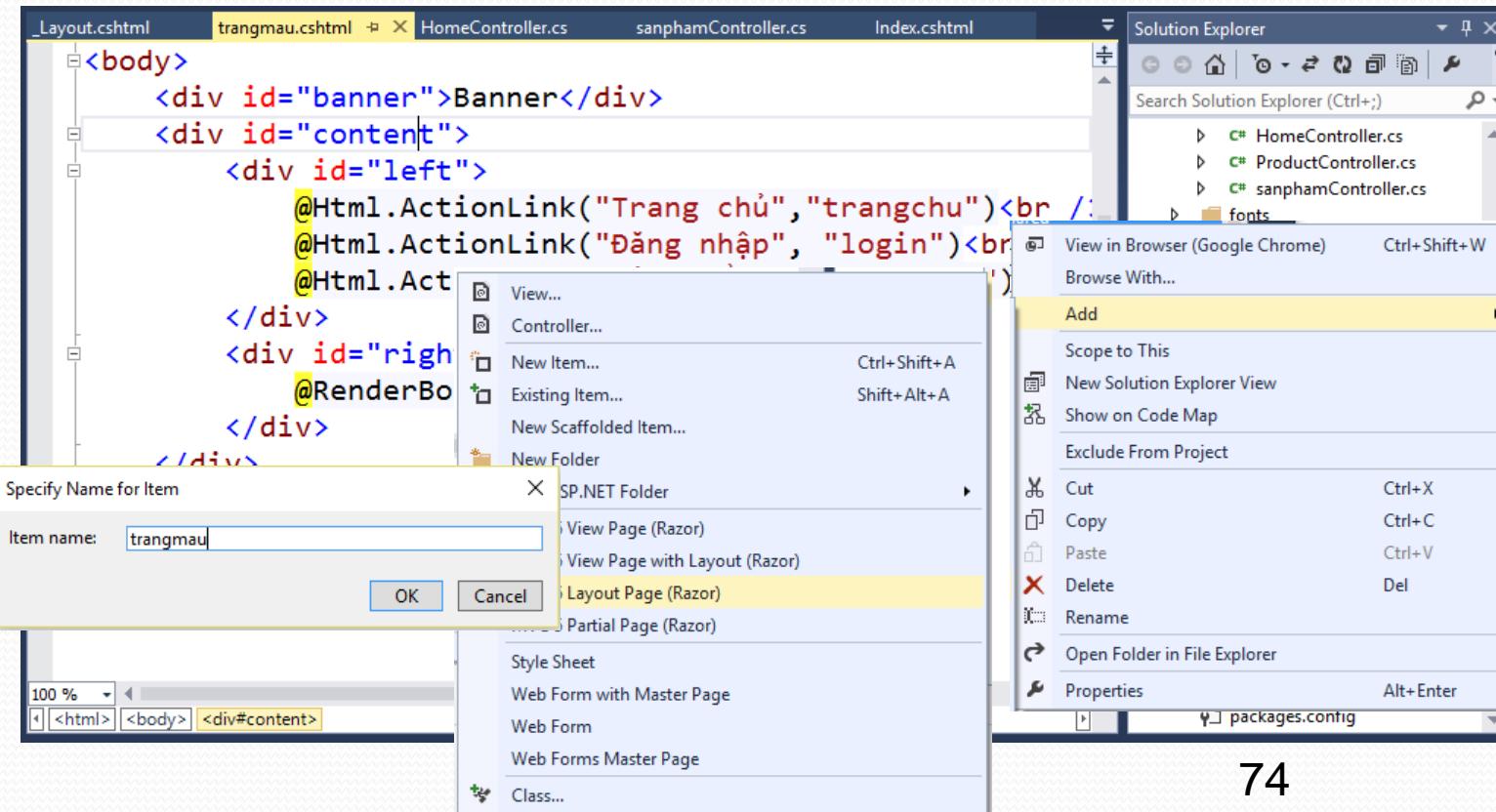
Layout

- Hiển thị nội dung của 1 thành phần nhỏ trong layout.



Cách tạo Layout

- Right Click View\Shared → Chọn Add → Chọn MVC Layout Page(Razor)



74

Cách tạo Layout

➤ Layout được tạo như sau:

@RenderBody là phần nội dung được thay đổi

```
<!DOCTYPE html>

<html>
  <head>
    <meta name="viewport" content="width=device-width" />
    <title>@ViewBag.Title</title>
  </head>
  <body>
    <div>
      @RenderBody()
    </div>
  </body>
</html>
```

Thiết lập layout cơ bản

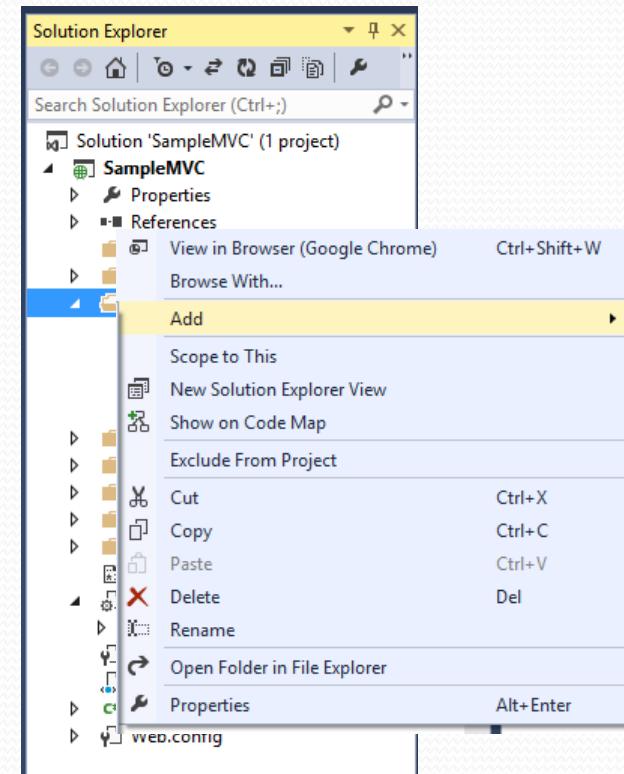
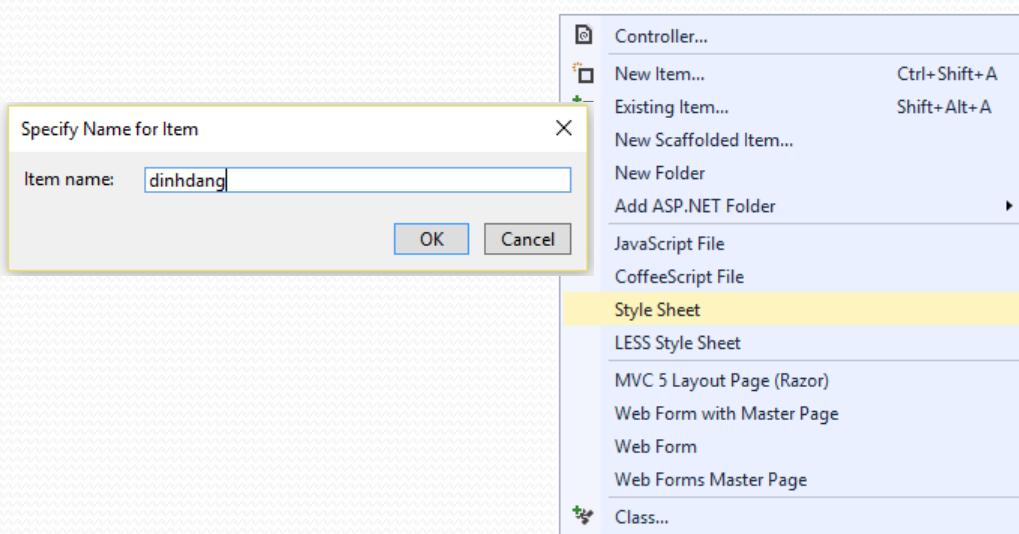


Layout

```
plate.cshtml      _Layout.cshtml      trangmau.cshtml*  ✖  HomeController.cs      sanphamController.cs      Danhsa
  <!DOCTYPE html>
  |<html>
  |<head>
  |<meta name="viewport" content="width=device-width" />
  |<title>@ViewBag.Title</title>
  |
  |<link href="~/Content/dinhdang.css" rel="stylesheet" />
  |</head>
  |<body>
  |<div id="banner">Banner</div>
  |<div id="content">
  |<div id="left">
  |<@Html.ActionLink("Trang chủ", "trangchu")<br />
  |<@Html.ActionLink("Đăng nhập", "login")<br />
  |<@Html.ActionLink("Sản phẩm", "Chitietsp")>
  |</div>
  |<div id="right">
  |<@RenderBody()
  |</div>
  |</div>
  |<div id="footer">Copy right ...</div>
  |</body>
  |</html>
```

Áp dụng CSS cho Layout

- Trong thư mục Content → Right Click → Add Style Sheet
- Đặt tên cho file css

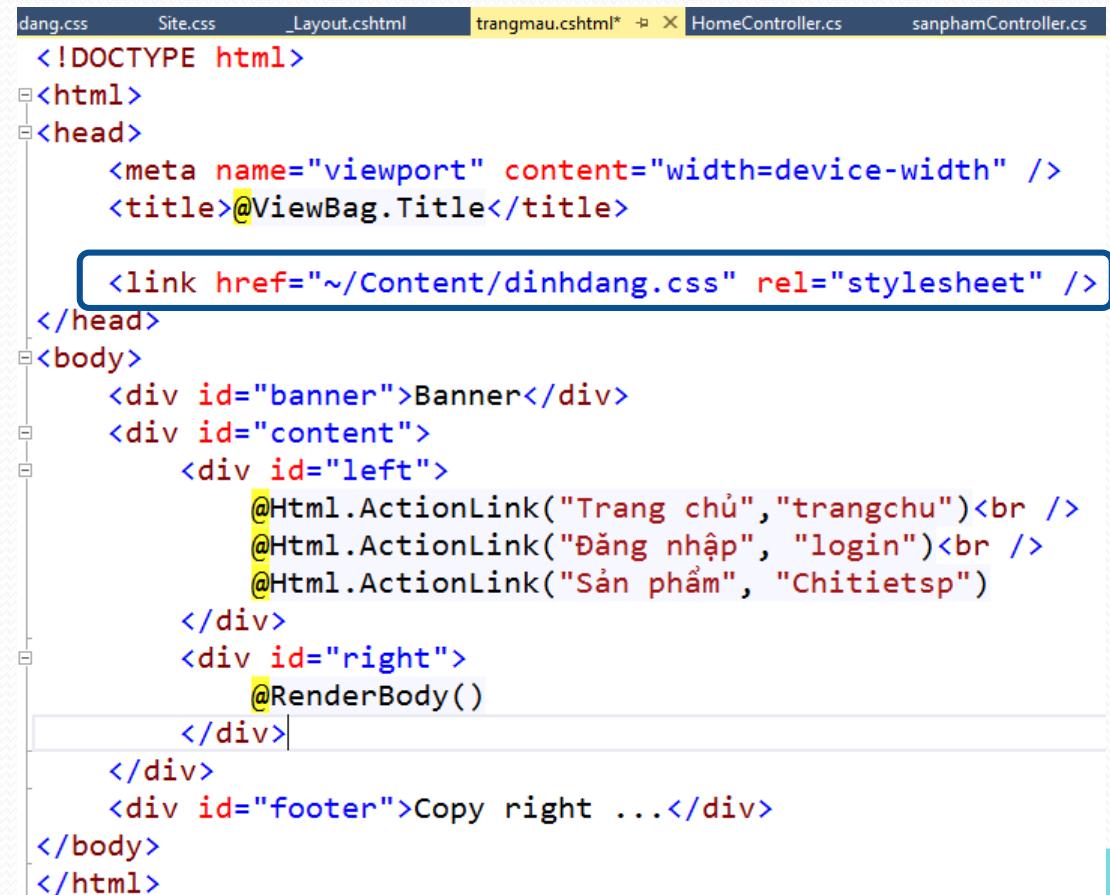


Áp dụng CSS cho layout

```
body {  
margin-top:0px;  
margin-left:100px;  
}  
  
#banner {  
width:800px;  
background-color: aqua;  
height:80px;  
border:1px solid black;  
float:left;  
}  
  
#footer {  
width:800px;  
background-color: violet;  
height:50px;  
border:1px solid black;  
float:left;  
}  
  
#content {  
width:800px;  
background-color: pink;  
height:400px;  
float:left;  
}  
  
#left {  
width:150px;  
background-color: aquamarine;  
border:1px solid black;  
height:398px;  
float:left;  
}  
  
#right {  
width:645px;  
background-color:beige;  
border:1px solid black;  
height:398px;  
float:left;  
}  
  
#container {  
width:800px;  
height:auto;  
float:left;  
margin:auto;  
border:5px solid red;  
}
```

Áp dụng CSS cho Layout

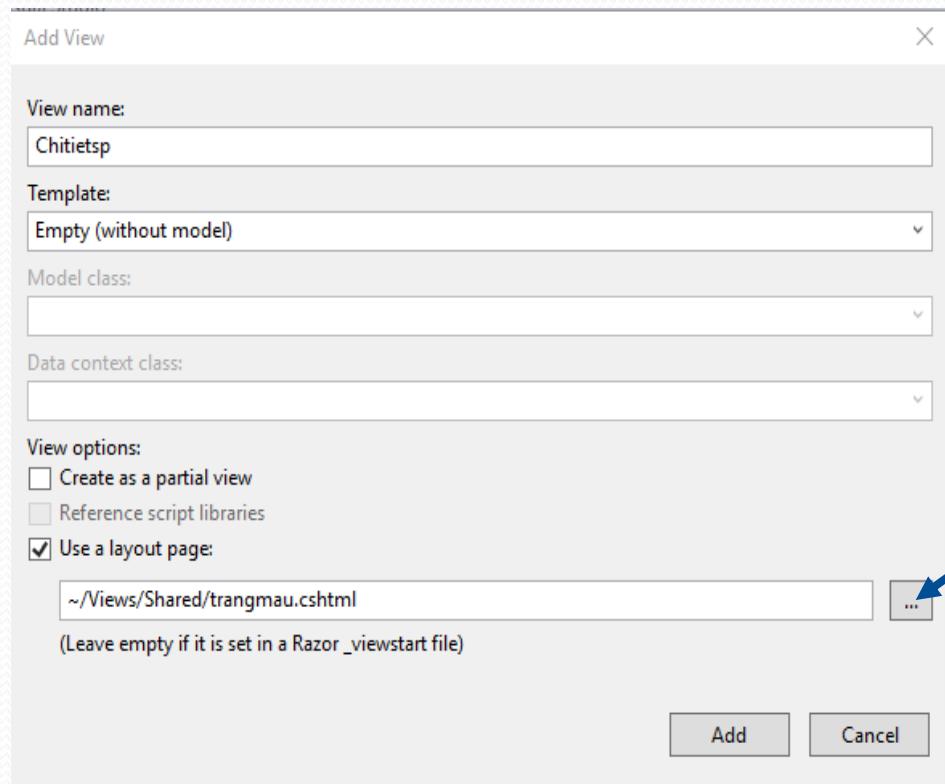
- Nhúng file CSS vào trong layout bằng cách kéo file vào trang layout
- Mã lệnh :



```
dang.css      Site.css      _Layout.cshtml      trangmau.cshtml*  X  HomeController.cs      sanphamController.cs
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>@ViewBag.Title</title>
    <link href="~/Content/dinhdang.css" rel="stylesheet" />
</head>
<body>
    <div id="banner">Banner</div>
    <div id="content">
        <div id="left">
            @Html.ActionLink("Trang chủ", "trangchu")<br />
            @Html.ActionLink("Đăng nhập", "login")<br />
            @Html.ActionLink("Sản phẩm", "Chitietsp")
        </div>
        <div id="right">
            @RenderBody()
        </div>
    </div>
    <div id="footer">Copy right ...</div>
</body>
</html>
```

Tạo view với Layout có sẵn

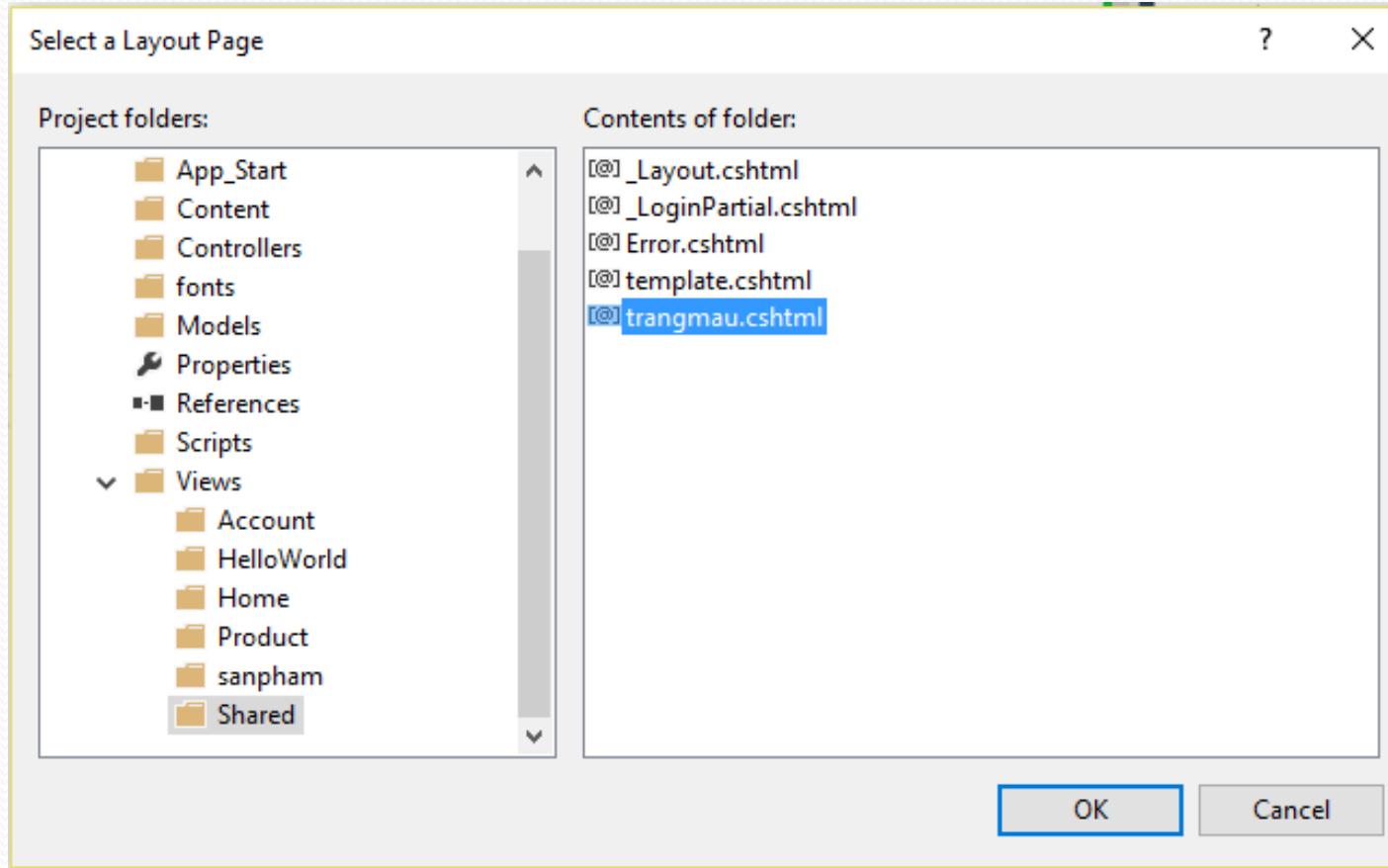
- Trong thư mục View → Add View



Click chọn danh sách các Layout đã tạo

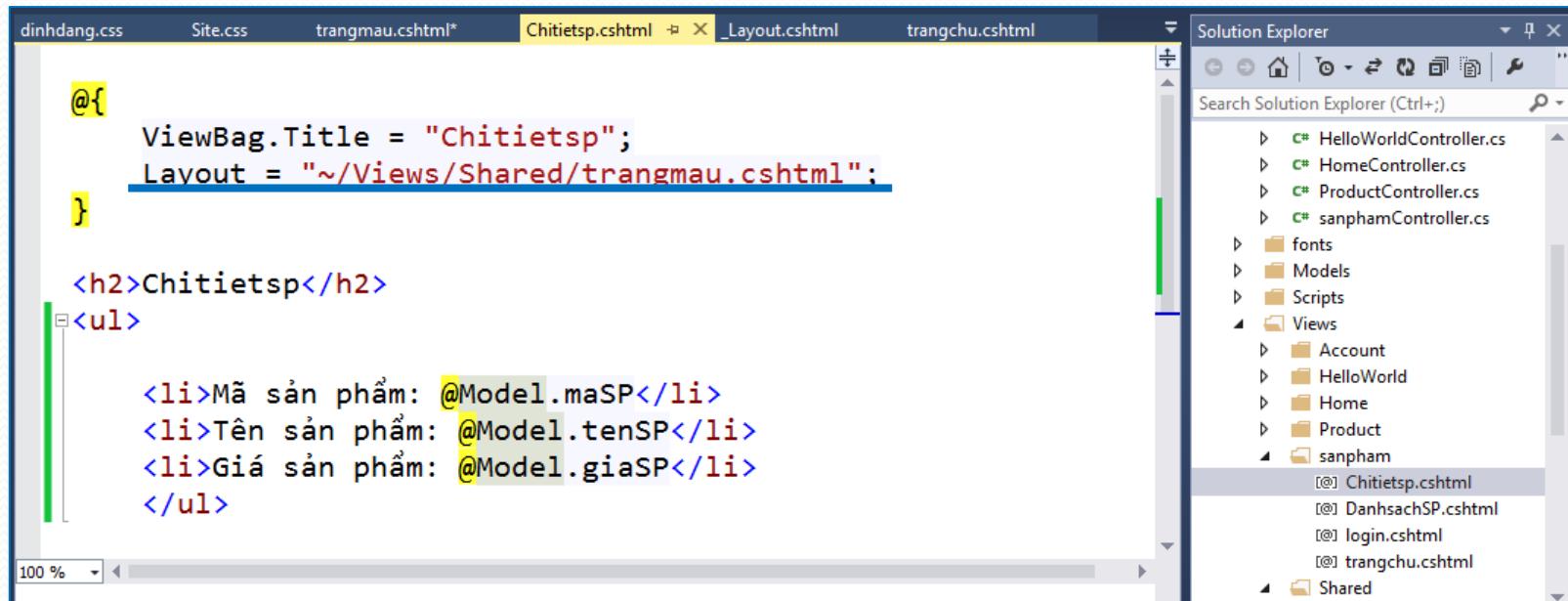
Tạo view với Layout có sẵn

- Chọn trang layout tương ứng



Tạo view với Layout có sẵn

- Tại View có sử dụng layout vừa chọn sẽ có mã lệnh tương ứng



The screenshot shows a Visual Studio interface. On the left is the code editor with the file `Chitietsp.cshtml` open. The code defines a title and a layout:

```
@{  
    ViewBag.Title = "Chitietsp";  
    Layout = "~/Views/Shared/trangmau.cshtml";  
}  
  


## Chitietsp



- Mã sản phẩm: @Model.maSP
- Tên sản phẩm: @Model.tenSP
- Giá sản phẩm: @Model.giaSP

```

On the right is the Solution Explorer window, which lists the project structure:

- Controllers: HelloWorldController.cs, HomeController.cs, ProductController.cs, sanphamController.cs
- Fonts
- Models
- Scripts
- Views
 - Account
 - HelloWorld
 - Home
 - Product
 - sanpham
 - Chitietsp.cshtml
 - Danh sach SP.cshtml
 - login.cshtml
 - trangchu.cshtml
- Shared

PartialView Helper

❑ @Html.Action()

➤ Nhúng một Action

➤ Action này phải trả về PartialView để *loại bỏ Layout*

➤ Action này có thể đánh dấu [ChildActionOnly] để
không cho truy xuất trực tiếp

❑ Sử dụng @Html.Partial()

➤ Nhúng một View *không bao gồm layout*

Sử dụng @Html.Partial()

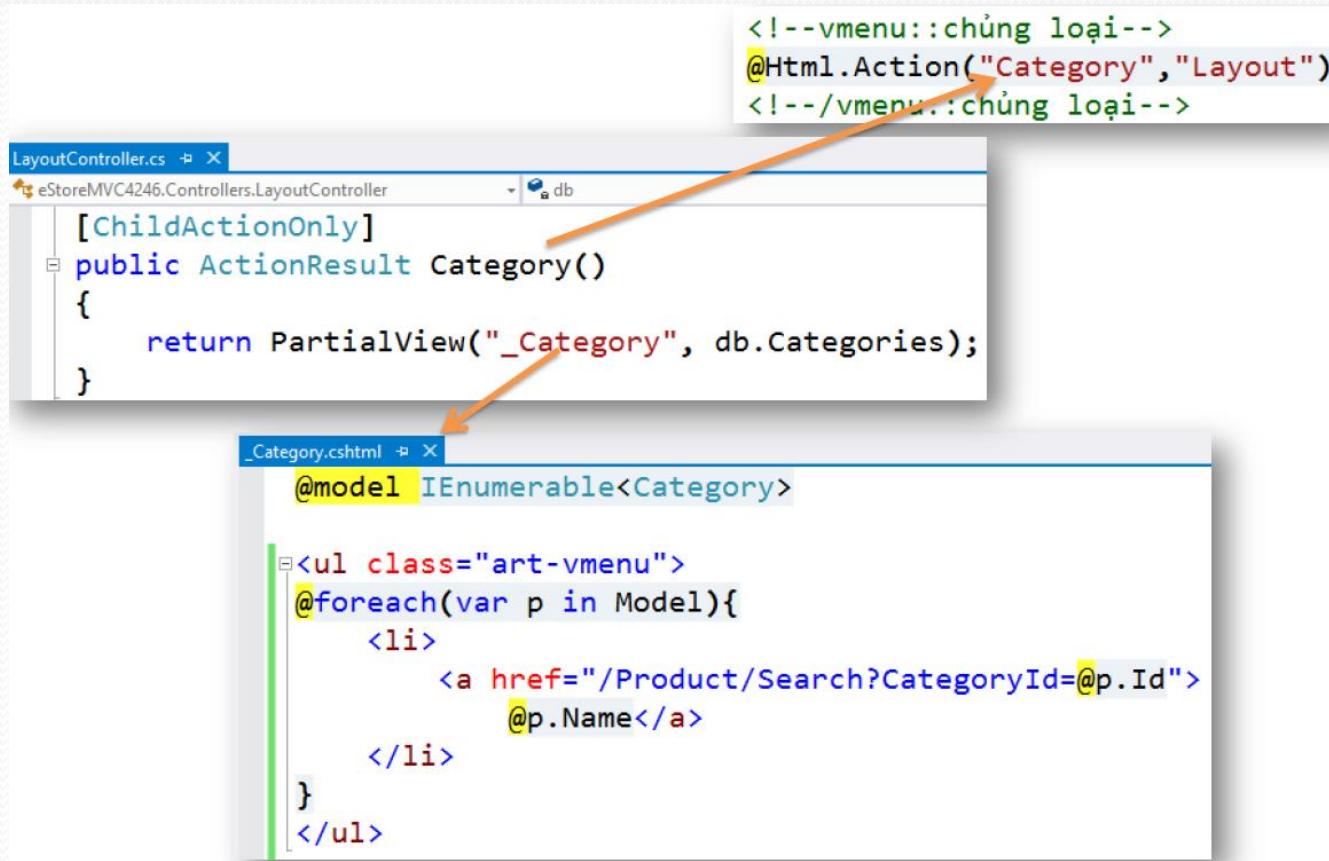
_Layout.cshtml

❑ `@Html.Partial("_LoginPartial")`



```
_LoginPartial.cshtml + X
@if (Request.IsAuthenticated) {
    <text>
        Hello, @Html.ActionLink(User.Identity.Name,
            @using (Html.BeginForm("LogOff", "Account",
                @Html.AntiForgeryToken()
                <a href="javascript:document.getElementById('logOff').submit()">Log off</a>
            )
        </text>
    } else {
        <ul>
            <li>@Html.ActionLink("Register", "Register", "Account")
            <li>@Html.ActionLink("Log in", "Login", "Account")
        </ul>
    }
}
```

Sử dụng @Html.Action()



The image shows two code files in Visual Studio:

- LayoutController.cs**:

```
[ChildActionOnly]
public ActionResult Category()
{
    return PartialView("_Category", db.Categories);
```
- _Category.cshtml**:

```
@model IEnumerable<Category>



@foreach(var p in Model){
    <li>
        <a href="/Product/Search?CategoryId=@p.Id">
            @p.Name</a>
    </li>
}
```

An orange arrow points from the highlighted line in `LayoutController.cs` (`@Html.Action("Category", "Layout")`) to the corresponding line in `_Category.cshtml` (`@model IEnumerable<Category>`).

BỔ SUNG

- WebGrid
- Button: Text, color, Image
- ActionLink

WebGrid

- Thêm Microsoft.AspNet.WebPages.WebData từ NuGet.

```
@using WebMatrix.Data;
@{
    var db = Database.Open("ModelUD4"); //Chuỗi kết
nối trong web.config
    var selectStr = "SELECT * FROM NhanVien";
    var data = db.Query(selectStr);
    var grid = new WebGrid(data);
}
<h1>DANH SÁCH NHÂN VIÊN</h1>
<div id="grid">
    @grid.GetHtml(tableStyle: "table")
</div>
```

Các thuộc tính thường dùng của WebGrid

- source //*Nguồn dữ liệu*
- rowsPerPage: n //*Phân trang n dòng trên 1 trang*
- defaultSort: “Tên thuộc tính”//*Sắp xếp mặc định*
- canPage: true/false //*Cho phép phân trang?*
- canSort: true/false //*Cho phép sắp xếp?*

WebGrid GetHtml

```
➤ tableStyle: "table",
➤ mode: WebGridPagerModes.Numeric,
➤ columns: new[] // columns in grid
{
    grid.Column("Tên thuộc tính", "Tên mới hiển thị", định dạng),
    ...
}
```

WebGrid GetHtml

```
@grid.GetHtml(
    tableStyle: "table",
    mode: WebGridPagerAdapter.Numeric,
    columns: new[] // columns in grid
    {
        grid.Column("MaNV", "Mã nhân viên"), //the model fields to
display
        grid.Column("HoNV", "Họ nhân viên"),
        grid.Column("TenNV", "Tên nhân viên"),
        grid.Column("GioiTinh", "Giới tính", format: (@item) =>
@item.GioiTinh==true?"Nam":"Nữ"),
        grid.Column("NgaySinh", "Ngày sinh", format:@<text>
@item.NgaySinh.ToString("dd/MM/yyyy")</text>),
        grid.Column("DiaChi", "Địa chỉ"),
        grid.Column("MaPB", "Phòng ban"),
        grid.Column("ChucNang", format: (item) =>
Html.ActionLink("Cập nhật", "Edit", new { id = item.MaNV })),
```

WebGrid GetHtml

```
grid.Column("Chức năng", format: @<text>
    <a href="@Url.Action("Edit",
        new { id = item.MaNV})" class="edit-
btn"><abbr title="Cập nhật dữ liệu"></abbr></a>
    <a href="@Url.Action("Details",
        new { id = item.MaNV})" class="edit-
btn"><abbr title="Xem chi tiết"></abbr></a>
    <a href="@Url.Action("Delete",
        new { id = item.MaNV})" class="edit-btn">
        <abbr title="Xóa dữ liệu">
            </abbr> </a>
    </text>),
}
)
```

Kết quả thể hiện trên trình duyệt

DANH SÁCH NHÂN VIÊN

Mã nhân viên	Họ nhân viên	Tên nhân viên	Giới tính	Ngày sinh	Địa chỉ	Phòng ban	Chức năng	Chức năng
XT	2	3	Nam	01/01/1999	1234	1	Cập nhật	  
2	3	4	Nam	01/01/1999	1234	KD	Cập nhật	  
ZT	4	5	Nữ	01/01/1999	345	KT	Cập nhật	  
6	6	7	Nam	01/01/1999	789	KT	Cập nhật	  
12								

Button



```
<input type="submit" value="Thêm" style="padding-left:  
28px; margin-left: 2px; background-image:  
url('/images/edit.png'); background-repeat:no-repeat;  
background-position: left; background-size:20px; "/>
```

```
<input type="submit" value="Thêm" style="margin-left: 2px;  
width:74px; background-color: blue; color:white" />
```

Kết thúc chủ đề 3

