Câu1 .Session là gì?

Một session được hiểu như một phiên làm việc giữa client và server, bắt đầu khi có request từ client, kết thúc khi người dùng đóng ứng dụng hoặc khi session hết thời gian timeout. Cũng giống như Cookie, Session được dùng để lưu trữ thông tin liên quan đến người dùng, nhưng thay vì được lưu trên máy client, các biến session được lưu trên máy server. Mỗi session có một định danh duy nhất (SessionID).

Cách sử dụng Session trong ASP.NET

Cách tạo và sử dụng session trong ASP.NET rất đơn giản. Ví dụ:

Session ["UserID"] = "101";

if(Session["UserID"] ==null)

{

    Response.Write("Page has expired.”);

}

Cách xóa Session trong ASP.NET

Thông thường, đối tượng session sẽ tự hủy sau khi hết thời gian timeout. Thời gian này mặc định là 20 phút nếu không được chỉ ra. Bạn cũng có thể hủy session bằng cách gọi đến phương thức Session.Abandon().

Ưu điểm của Session so với Cookie

- Session có thể lưu được những đối tượng phức tạp hơn.

- Session có độ bảo mật cao hơn (do được lưu trên server).

Cơ chế hủy Session của server

Chúng ta vẫn thường thấy rằng, khi bạn đóng trình duyệt rồi vào lại một trang web, bạn sẽ phải đăng nhập lại nếu bạn không dùng chức năng ghi nhớ trước đó. Như vậy bạn đang làm việc trong một session mới. Phải chăng session cũ đã bị hủy ngay sau khi bạn đóng trình duyệt?

Không phải vậy. Bạn nên nhớ một khái niệm cơ bản của giao  thức HTTP rằng nó làm việc theo nguyên tắc request/response, nghĩa là client chỉ kết nối đến server khi có request từ phía client mà thôi. Nếu không có request, server hoàn toàn mù tịt về phía client. Điều đó cho thấy, server hoàn toàn không biết người dụng đóng trình duyệt lúc nào để mà hủy session đi.

Như bạn biết, mỗi session được định danh duy nhất bởi SessionID. Và SessionID này thực tế được lưu trên máy client trong Session Cookie. Khi bạn đóng trình duyệt, Session Cookie này được hủy ngay lập tức. Lúc này, các biến session vẫn còn tồn tại trên server. Khi mở lại trình duyệt vào lại trang web cũ, Session Cookie mới được tạo ra, một SessionID mới được ghi vào Session Cookie mới (do trong Session Cookie mới không có SessionID của session cũ), một phiên làm việc mới (session) bắt đầu. Các biến session cũ của phiên làm việc trước đó sẽ được giải phóng (hủy) khi hết thời gian timeout.

Câu 3 link : <https://freetuts.net/hieu-ve-request-va-response-trong-lap-trinh-web-676.html>

Request có thể hiểu nhanh là thông tin gửi từ client lên server. Khi bạn lên trình duyệt [browser](https://freetuts.net/glossary/browser-la-gi-12.html) gõ một địa chỉ nào đó, ví dụ bạn gõ là freetuts.net thì ngay lập tức trình duyệt sẽ dựa vào tên domain để gửi yêu cầu truy cập đến địa chỉ IP mà domain này đang trỏ tới (bạn đừng quan tâm đến domain này do ai quản lý), lúc này phía server sẽ phân tích yêu cầu và sẽ gửi luồng xử lý tới vị trí vị trí lưu trữ của mã nguồn PHP (hoặc mã nguồn bất kì) và nhiệm vụ của các mã nguồn là tiếp nhận yêu cầu, phân tích request đó và trả kết quả lại cho client.

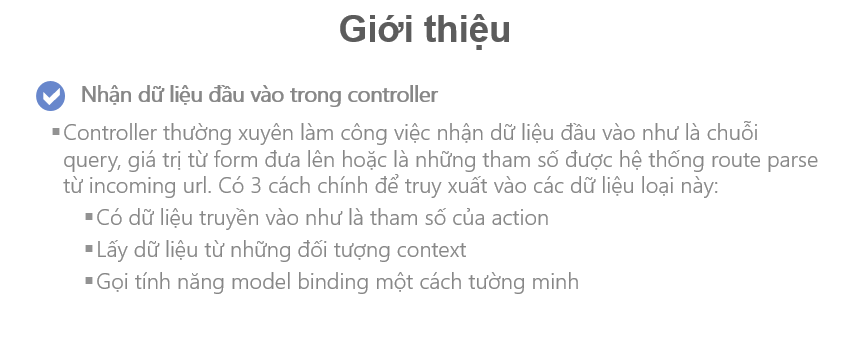
Chúng ta có hai phương thức (2 cách) để gửi request từ client lên server đó là sử dụng [phương thức GET và phương thức POST](https://freetuts.net/phuong-thuc-get-va-post-trong-php-19.html)

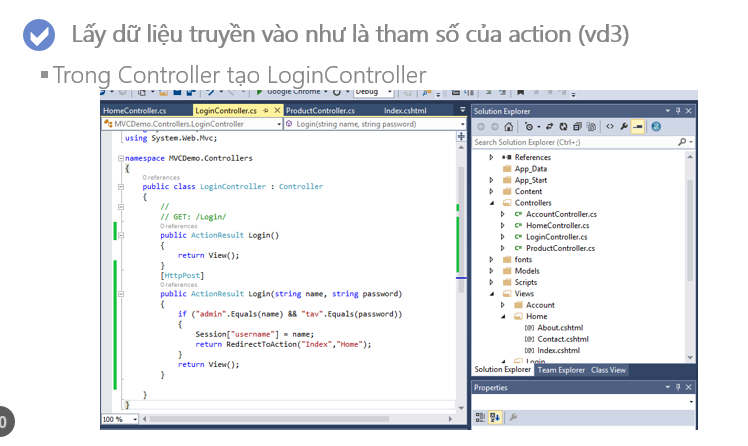
Khi bạn gửi request thì sẽ có rất nhiều thông tin đính kèm theo bao gồm các thông tin của trình duyệt đang sử dụng, địa chỉ IP của mạng mà máy tính bạn đang kết nối và nhiều thông tin khác.

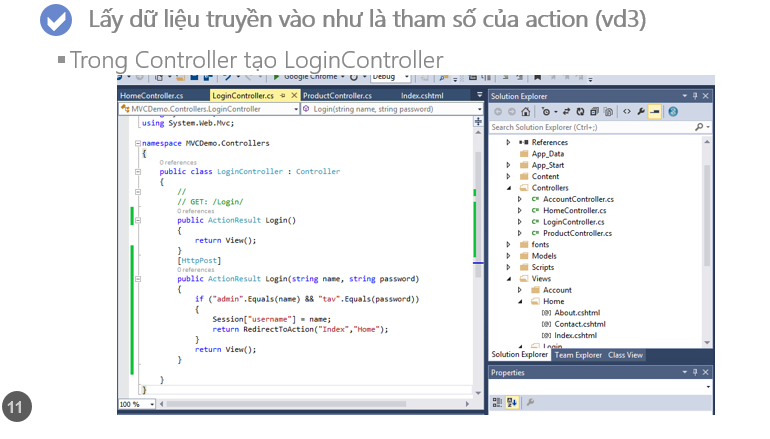
**Câu 4**

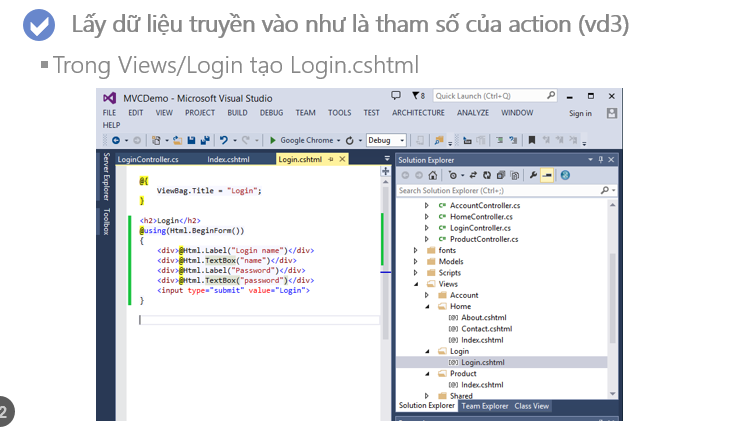
Reponse là dữ liệu mà server trả về cho client. Ví dụ khi bạn nhập vào địa chỉ freetuts.net thì kết quả trả về (response) chính là giao diện của website và các thông tin của header. Như vậy dữ liệu mà server trả về là những đoạn mã HTML kèm theo các thông tin của header

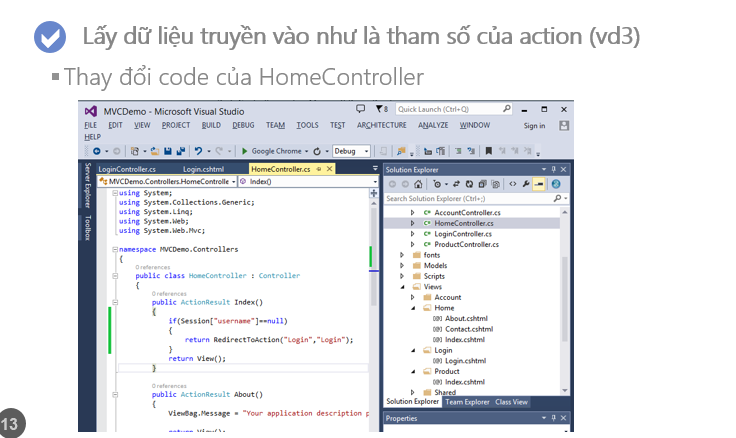
**Câu 5> slide 9 bài 6**

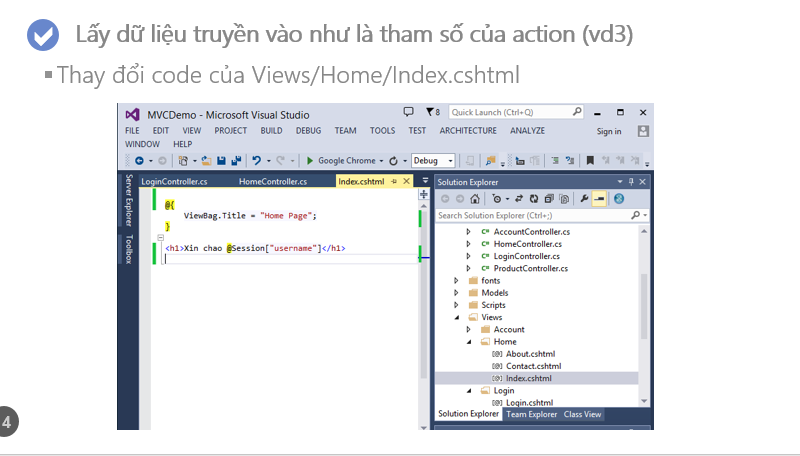






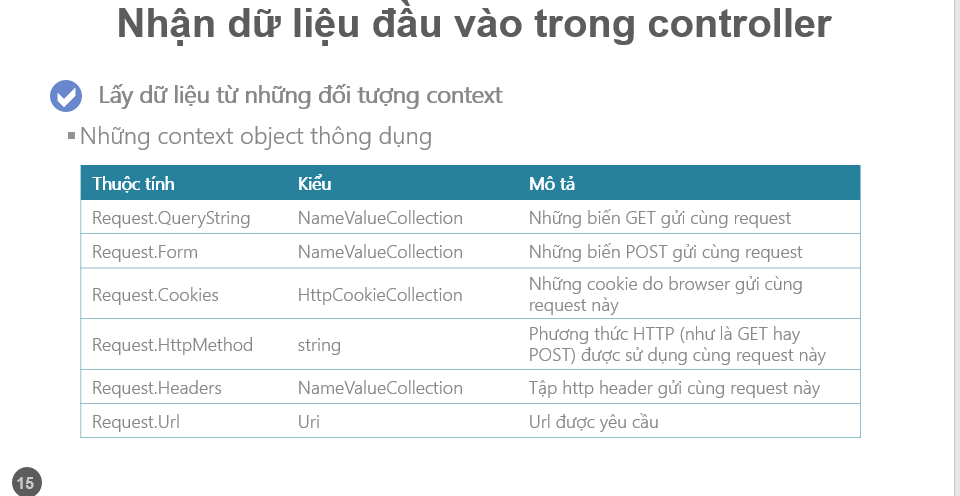


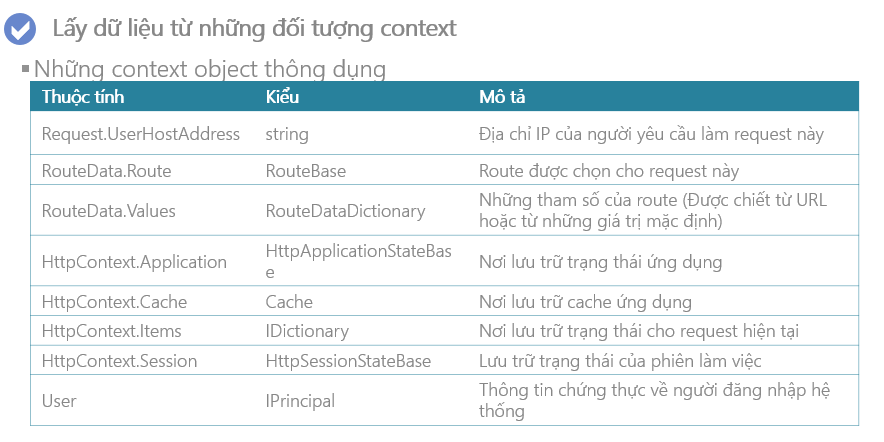




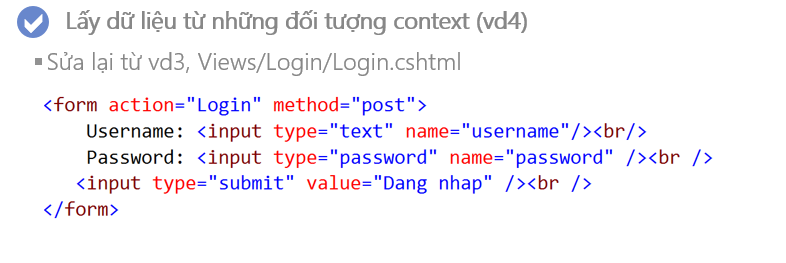
**Câu 6**

**Slide 15 bài 6**

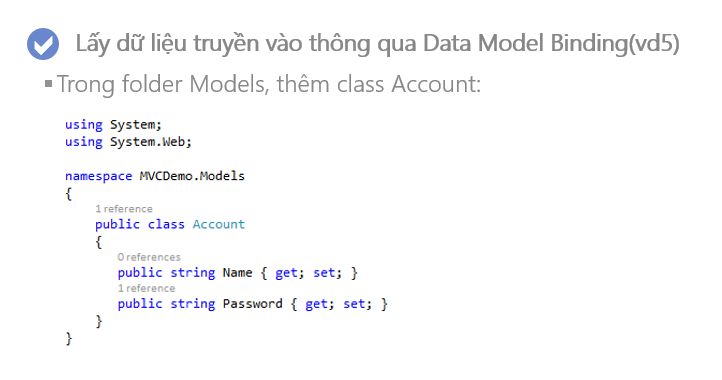




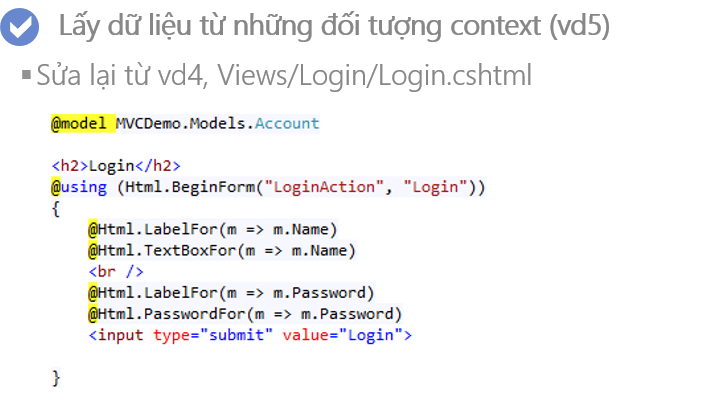




**Câu 7 slide 19 bài 6**

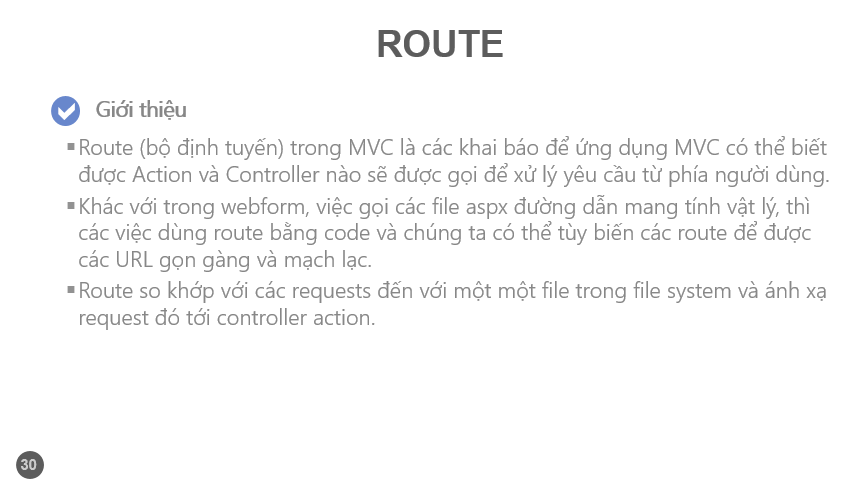




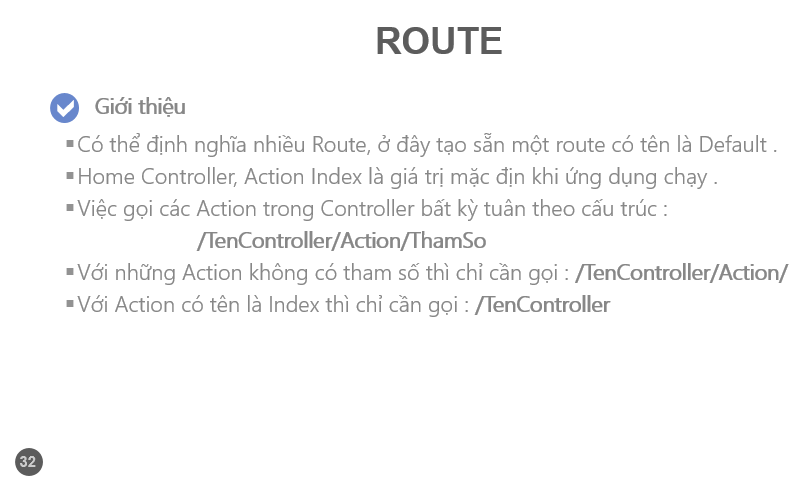


**Câu 8-10 slide 25 bài 6 tự xem nha.**

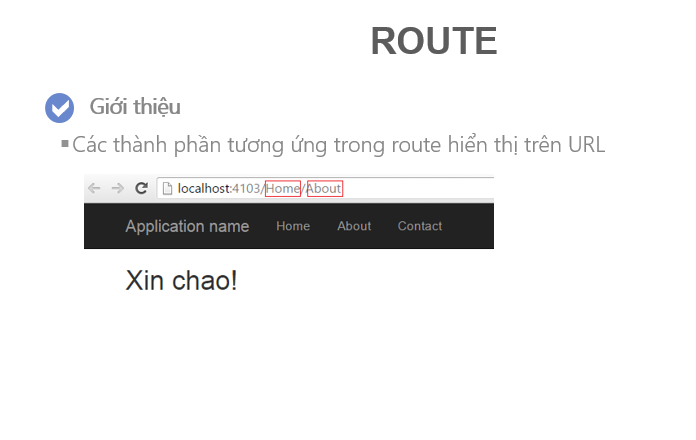
Câu 10:





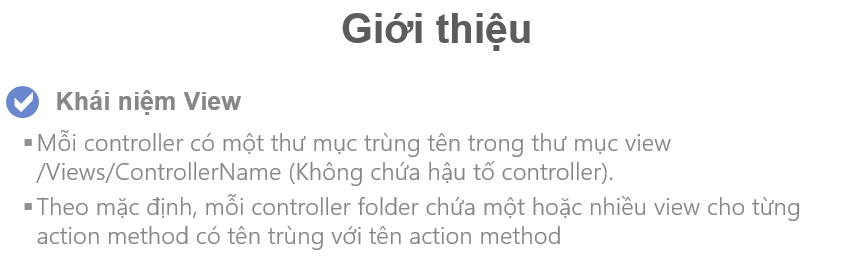




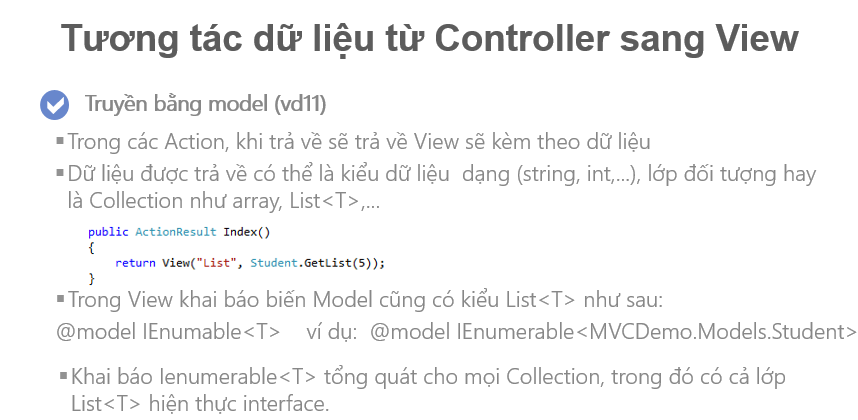


Câu 11 (slide bài 7)



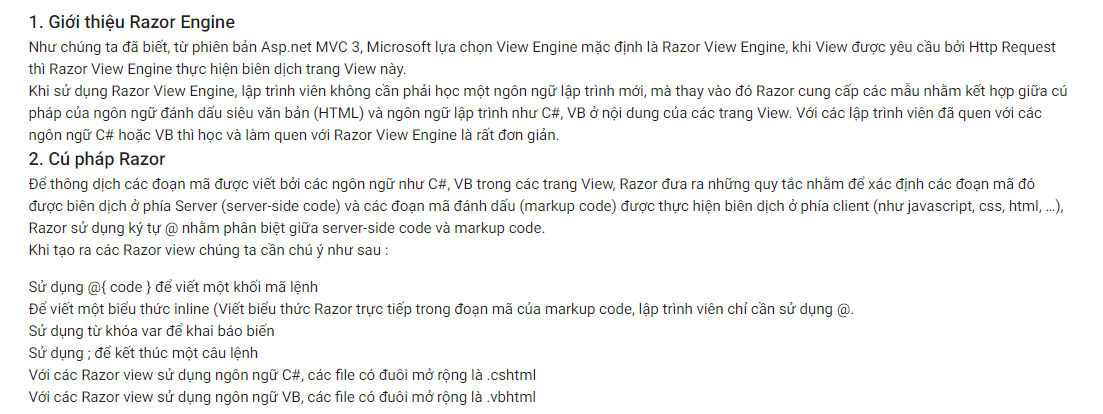


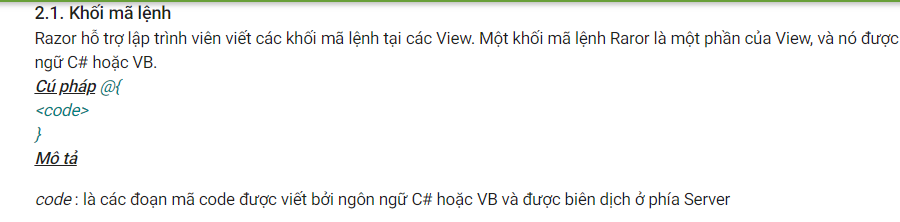




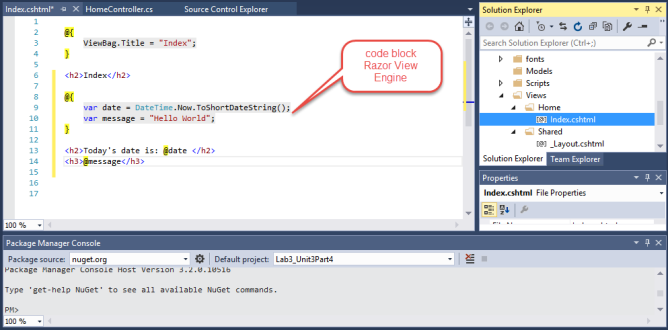
Câu12 học theo slide 17 bài 7 cũng đc . hoặc học theo dưới nay nha

link tham khảo: http://itprotraining.vn/vi/asp-net-mvc/hoc-lap-trinh-asp-net-mvc-5-0-view-trong-asp-net-mvc-visual-studio-2015-phan-4-bai-so-3



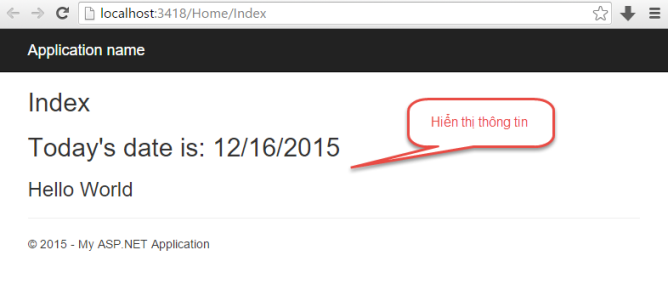


*Ví dụ*



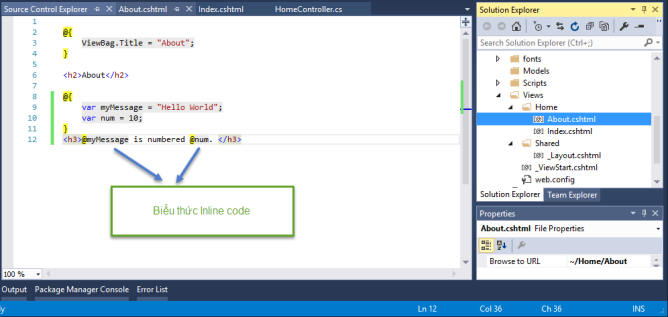
*Hình số 1 : Khối mã lệnh Razor View*

Kết quả khi chạy chương trình :

**

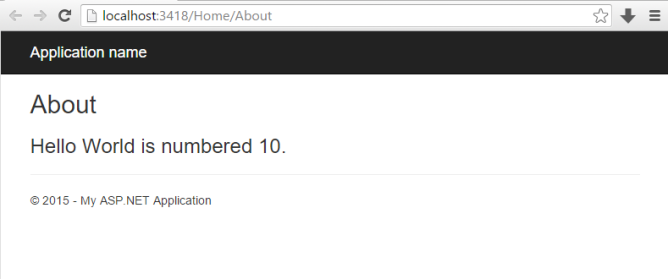
*Hình số 2 : Kết quả chạy chương trình với khối lệnh Razor*

2.2. Biểu thức Inline - Inline Expression  
Tương tự như các khối lệnh, Razor sử dụng @ để đánh dấu biểu thức Inline.  
*Ví dụ*



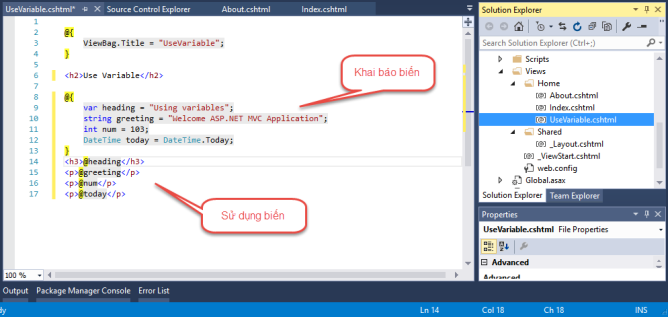
*Hình số 3 : Biểu thức Inline*

 Kết quả chạy chương trình :



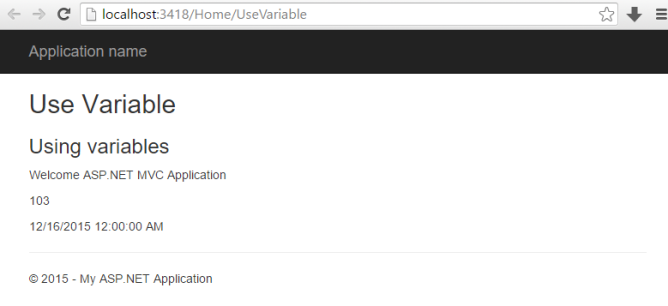
*Hình số 4 : Kết quả chạy chương trình với Biểu thức Inline*

2.3. Biến  
Giống như các ngôn ngữ lập trình C#, VB, Razor sử dụng từ khóa *var* cung cấp cách thức để khai báo biến.  
*Ví dụ* : Mô tả cách khai báo biến với từ khóa *var* và cách sử dụng biến trong view



*Hình số 5 : Khai báo và sử dụng biến trong Razor*

Kết quả chạy chương trình

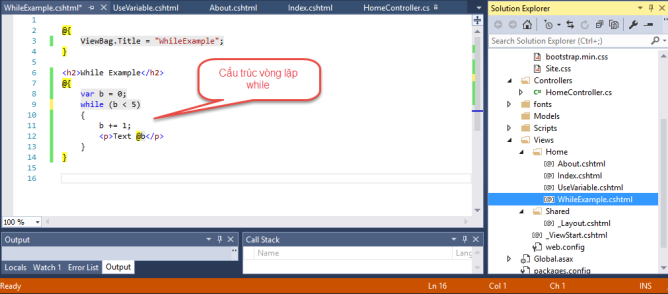


*Hình số 6 : Kết quả chạy chương trình với biến trong Razor*

2.4. Vòng lặp  
Giống như ngôn ngữ lập trình C#, khi chọn C# thực hiện tại Razor view engine, tại view hỗ trợ các cấu trúc lặp như sau :

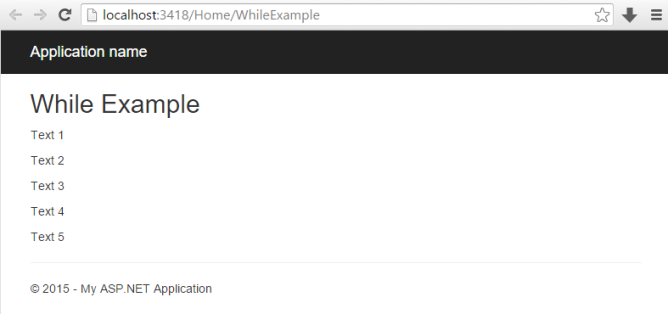
* Cấu trúc while
* Cấu trúc for
* Cấu trúc do .. while
* Cấu trúc foreach

Do cấu trúc vòng lặp đã quá quen thuộc đối với các lập trình viên, do vậy chúng ta sẽ lấy ví dụ về các cấu trúc vòng lặp này. Ở đây chúng ta chỉ lấy 1 ví dụ về cấu trúc vòng lặp while  
*Ví dụ*:



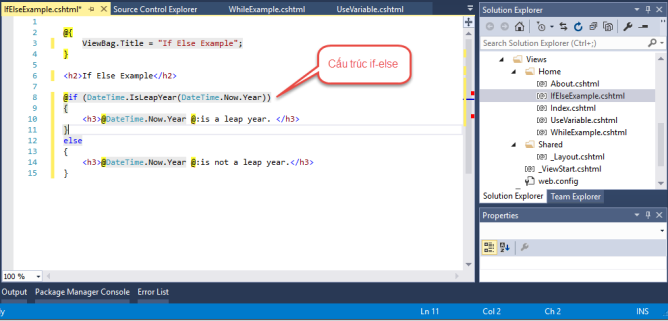
*Hình số 7 : Cấu trúc vòng lặp while*

Kết quả chạy chương trình



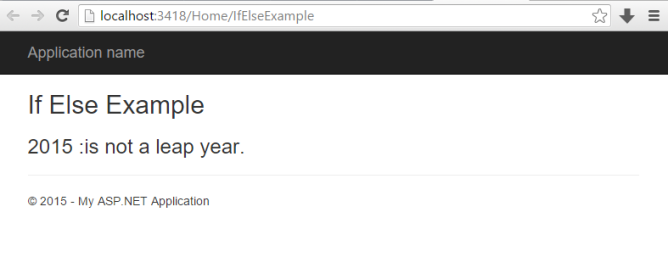
*Hình số 8 : Kết quả chạy chương trình với cấu trúc while*

2.5. Điều kiện if-else, switch  
Trong quá trình xây dựng ứng dụng asp.net mvc, quá trình hiển thị và xử lý dữ liệu động tại View là không thể tránh khỏi, nội dung dữ liệu động là phụ thuộc vào điều kiện hiển thị. Trong trường hợp này lập trình viên có thể sử dụng cấu trúc if-else hoặc switch để giải quyết vấn đề này.  
*Ví dụ* : Cấu trúc if-else



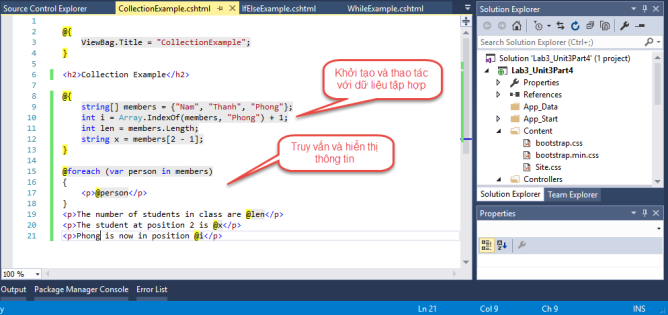
*Hình số 9 : Cấu trúc điều kiện if-else*

Kết quả chạy chương trình :



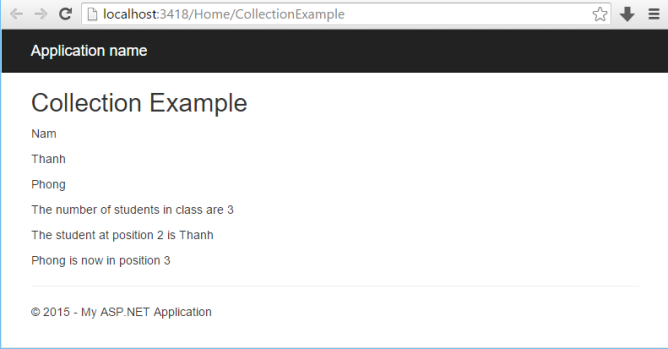
*Hình số 10 : Kết qủa chạy chương trình với cấu trúc điều kiện if-else*

2.6. Thao tác với dữ liệu tập hợp  
Razor view engine cho phép lập trình viên có thể thao tác với dữ liệu tập hợp như Array, List<T>,… Các thao tác với dữ liệu tập hợp tại View bao gồm : Khởi tạo dữ liệu tập hợp, truy vấn và hiển thị dữ liệu tập hợp …  
*Ví dụ* : Thao tác với dữ liệu tập hợp



*Hình số 11 : Thao tác với dữ liệu tập hợp*

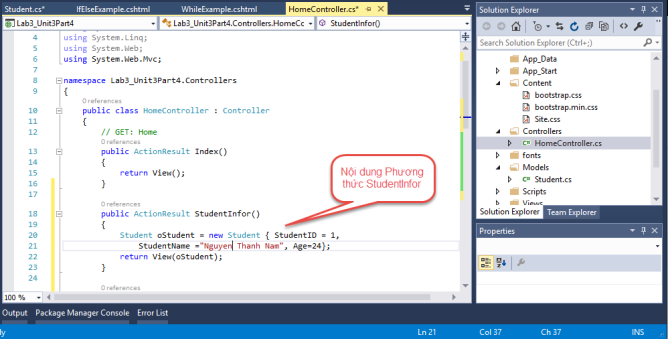
Kết quả chạy chương trình :



*Hình số 12 : Kết quả hiển thị với dữ liệu tập hợp*

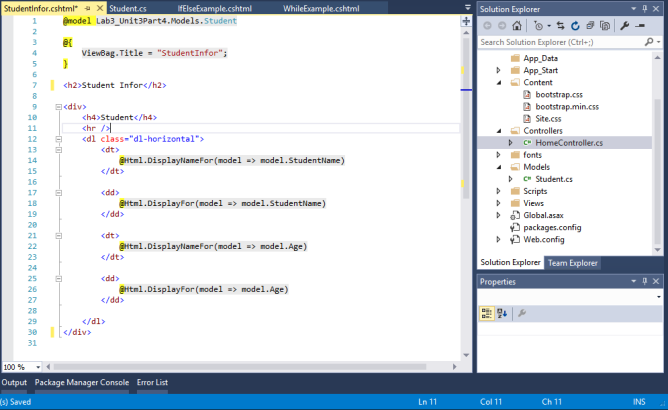
2.7. Model  
Sử dụng @model để thao tác với đối tượng model ở bất kỳ đâu trong View, đây là thao tác phổ biến nhất trong lập trình asp.net mvc.  
*Ví dụ* :  Tại Model tạo class Student (StudentID, StudentName, Age). Sau đó tại HomeController tạo phương thức StudentInfor với mục tiêu tạo View tương ứng để hiển thị thông tin chi tiết về Student. Quá trình thực hiện qua các bước sau :

* *Bước 1* : Tạo controller và phương thức Action StudentInfor



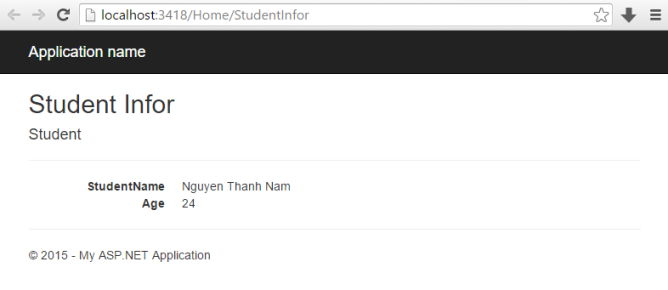
*Hình số 13 : Phương thức Action StudentInfor*

* *Bước 2 : Tạo View tương ứng với phương thức StudentInfor*



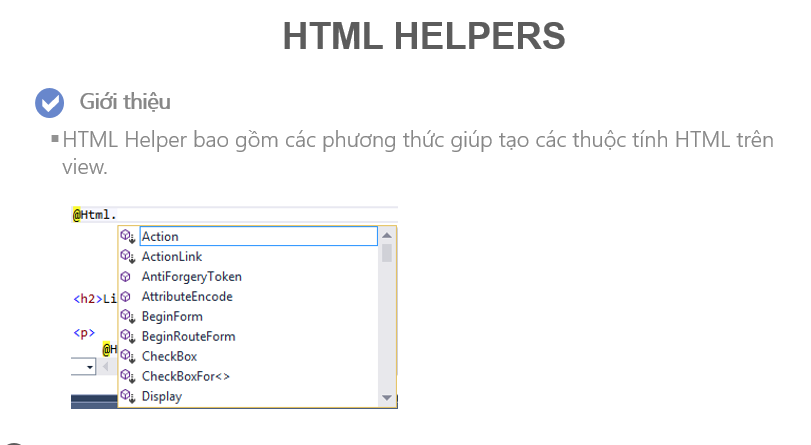
*Hình số 14 : Tạo View và sử dụng Model để truy vấn dữ liệu*

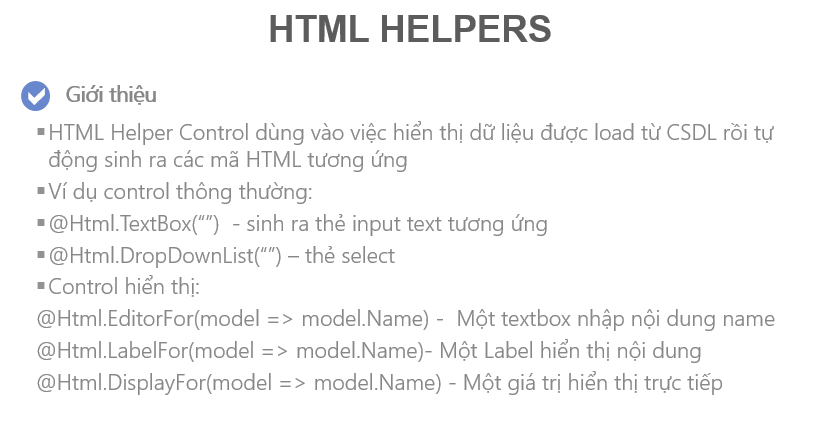
Kết quả chạy chương trình :



*Hình số 15 : Kết quả hiển thị thông tin khi sử dụng View*

Câu 13 slide 22 bài 7

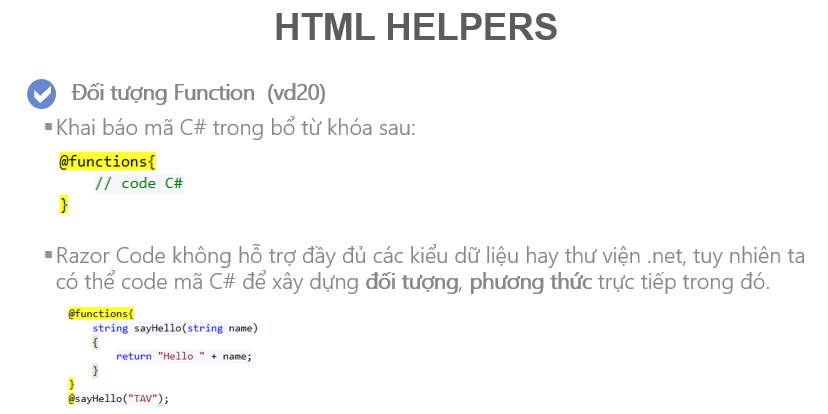




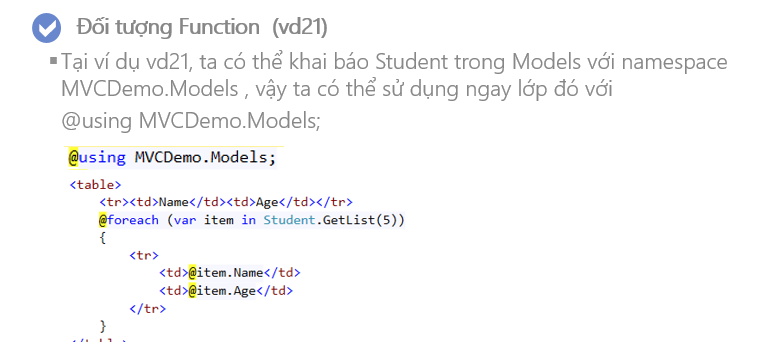


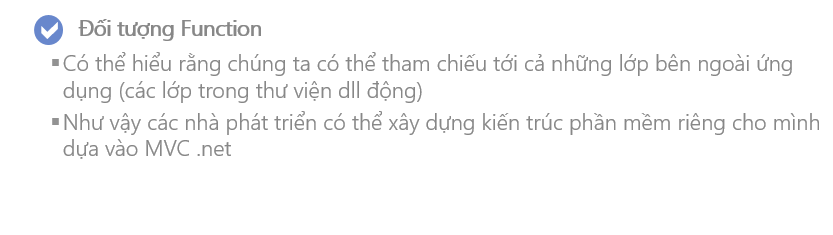


Cau 14 slide 27 bài 7









Câu 15

🞦 Là view cho phép sử dụng lại trong web  
application giống như user control.  
🞦 Partial view giống như view có đuôi mở rộng  
trong Razor Engine là *.cshtml*.  
🞦 Class Helper cung cấp một số phương thức cho  
view gọi partial view:  
▪ Html.Partial()  
▪ Html.RenderPartial()

🞦 Trong Controller ta có thể thay thế return  
View() thành return PartialView()  
public ActionResult PartialViewExaple(){  
return PartialView(\_sanpham.\_listSanPham);  
}  
▪ return View(); //Trả về trang HTML đầy đủ, render ra  
normal view cũng như Partial view.  
▪ return PartialView(); //Trả về một phần của HTML có  
thể được gọi thông qua AJAX requests.

-bổ sung trong slide 31 bài 7

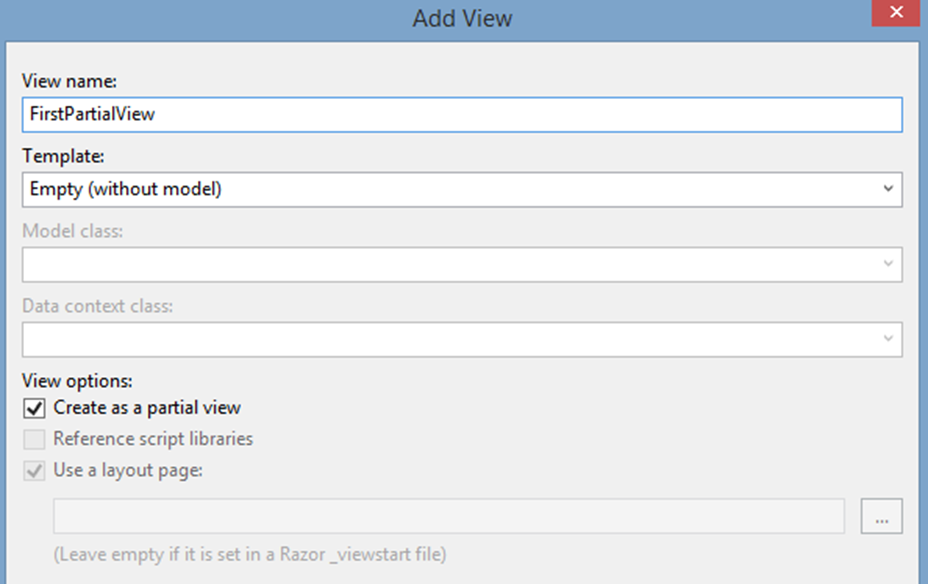
* Html.Partial, Html.RenderPartial dùng để gọi một PartialView.
* PartialView là một loại thẻ đặc biệt, chứa các thẻ html thuần hoặc các thẻ html chứa model hiển thị dữ liệu.
* Cấu trúc của HTML.Partial



Ví dụ :

**Tạo Partial**

* Trong Views/Login chọn
* Add/View



* Nhập nội dung cho FirstPartialView



* Trong file Login.cshtml

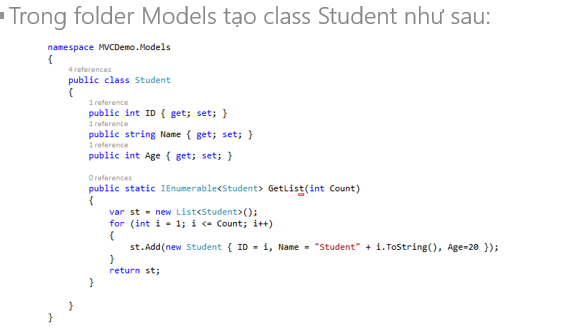


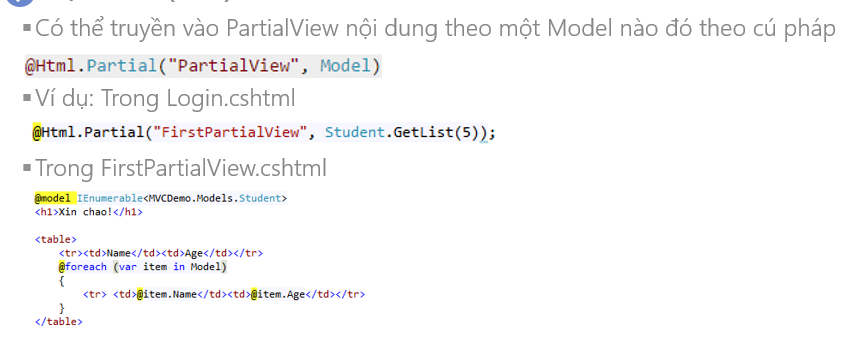


Ví dụ 2: Có thể truyền vào PảtialView nội dung theo một Model nào đó theo cú pháp:

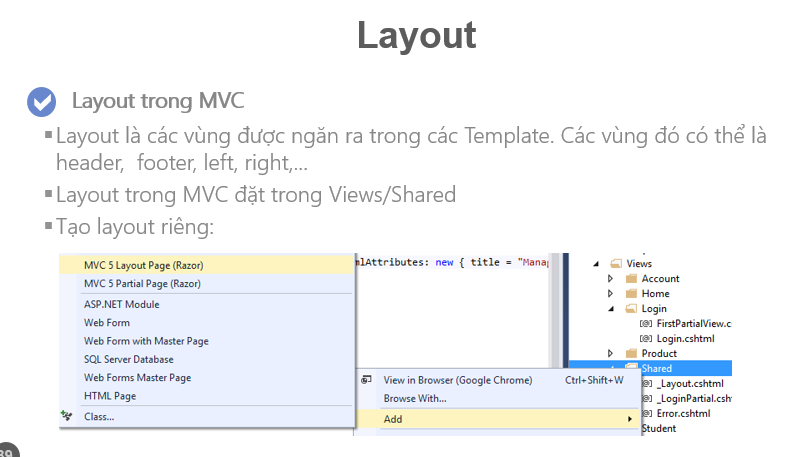
@Html.Partial( “PartialView “ , Model )

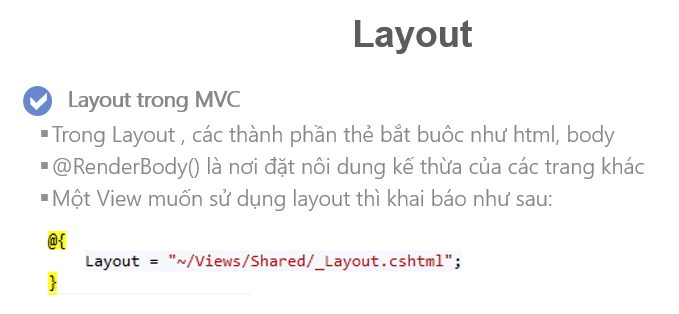
Ví dụ :

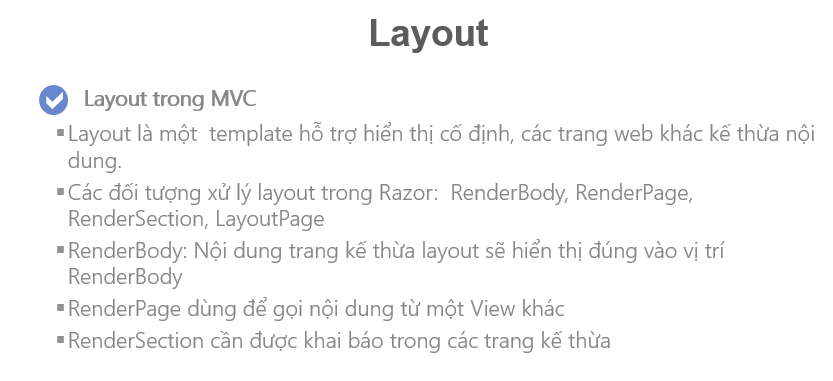




Câu 16 slide 39 bài 7





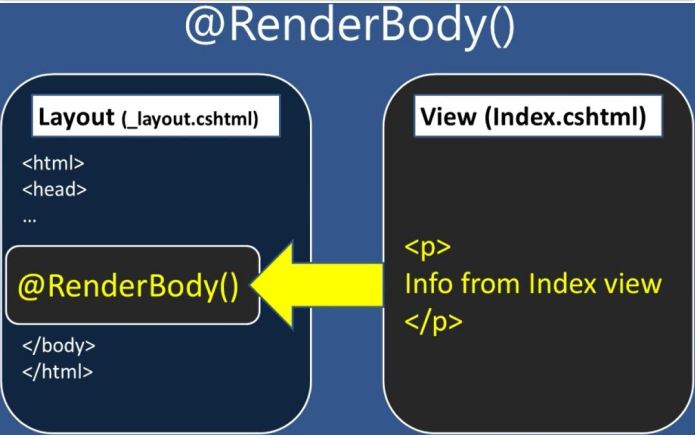


Câu17 -19

http://blog.chuyenlaptrinh.net/2017/05/tim-hieu-ve-renderbody-rendersection.html

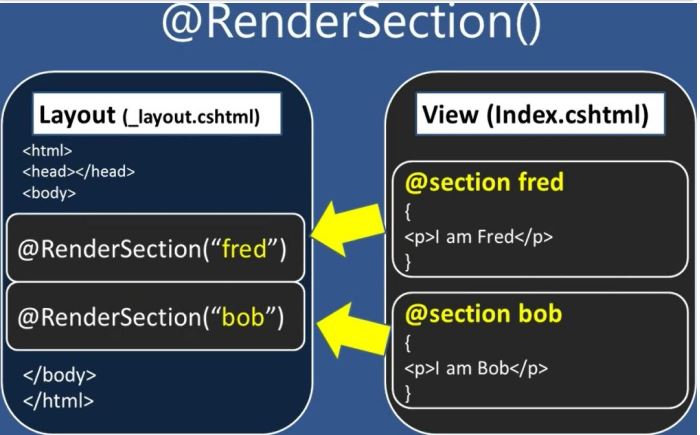
RenderBody  
- RenderBody là phương thức dùng gọi các trang kế thừa .Được khai báo ở layout   
RenderPage   
// RenderPage dùng để gọi nội dung html từ 1 View khác  
RenderSection   
// Được gọi trong Layout   
// Được khai báo trong các trang kế thừa   
Sample :   
Tại 1 View kế thừa Layout khai báo 1 section :  
@section Section1 {  
<h1>Section Content</h1>   
}  
// Trong Layout gọi Section với cú pháp :   
<div>  
@RenderSection("Section1",false)  
</div>  
Hoặc :   
<div>  
@RenderSection("Section1",required:false)  
</div>  
Các tham số required chỉ định thông báo lỗi hay không khi tìm section   
Hoặc có thể kiểm tra như sau :   
@if (IsSectionDefined("Section1"))  
{  
@RenderSection("Section1",required:false)  
}  
else  
{  
<h1>Default Content</h1>  
}

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

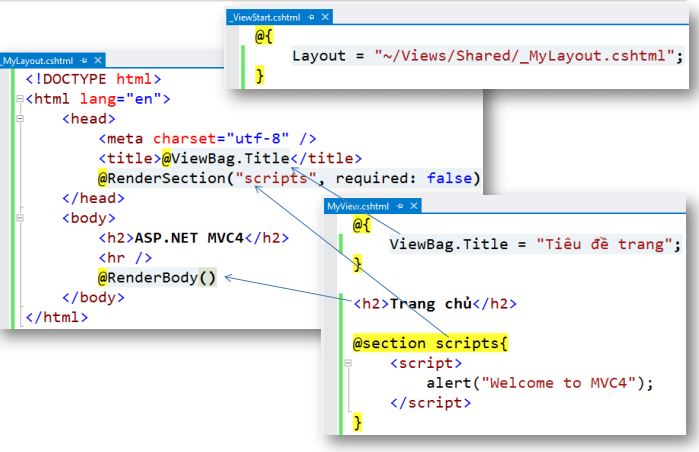
**[](https://1.bp.blogspot.com/-uJR44BPMXWE/WSBQ-_P5w3I/AAAAAAAAClE/vyQpKL05w6UShIqPDJBtuTOxJ5UjzcBpgCLcB/s1600/Capture.JPG)**

3. RenderSection()

Hiển thị nội dung của 1 thành phần nhỏ trong layout

**[](https://3.bp.blogspot.com/-FATG9EsY2v0/WSBTdxWRxqI/AAAAAAAAClQ/kSsgmYPl72IsVXwEZGWyFiBijsNhfCYSACLcB/s1600/Capture.JPG)**

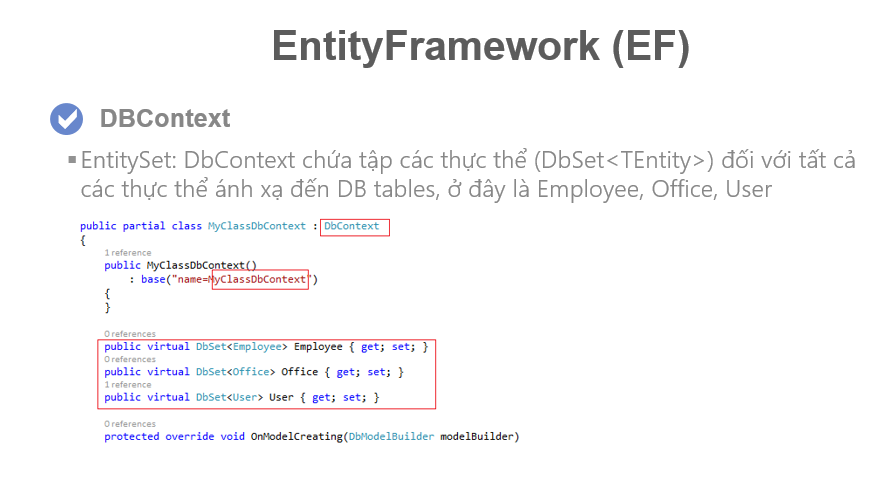
Ví dụ về RenderSection

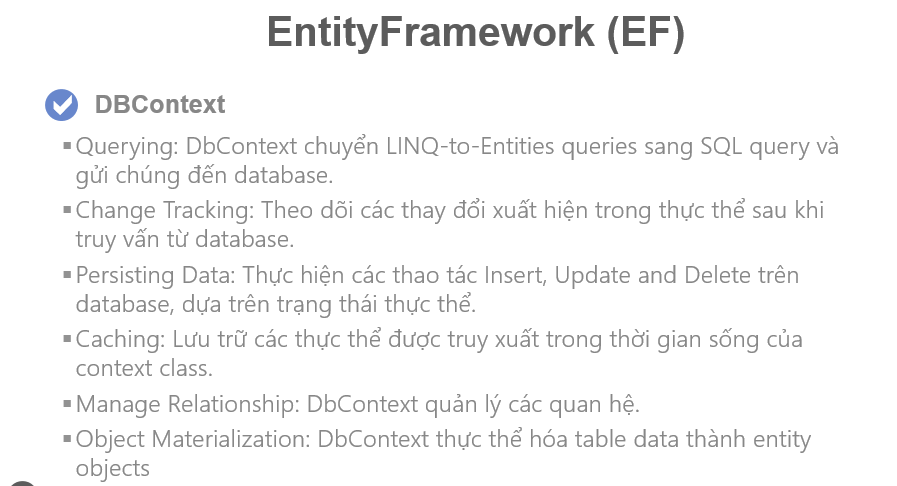
**[](https://4.bp.blogspot.com/-6NHG8UHNm6c/WSBV6ld-ktI/AAAAAAAAClc/Jj1uKcT3_hozFWD1PRNRTfap5fIQtDjhACLcB/s1600/Capture.JPG)**

Method RenderSection() có 2 tham số tham số thứ nhất là tên section tham số thứ 2 là*required* kiểu *bool*. Nếu các bạn không muốn sử dụng RenderSection trong tất cả các trang web thì thì truyến giá trị false vào tham số *required.*

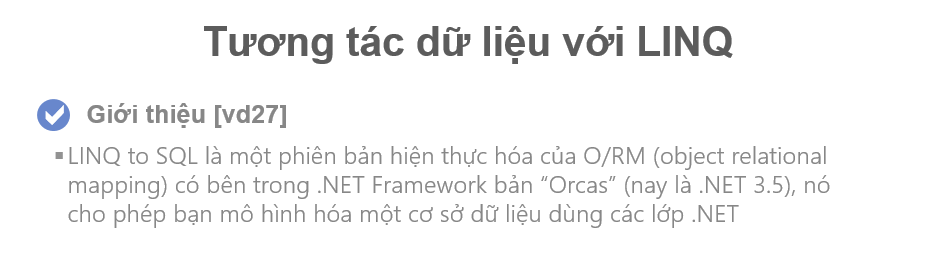


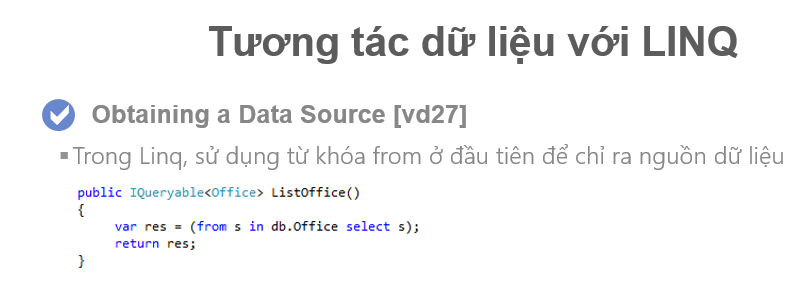
Câu 21 slide 17 bài 8

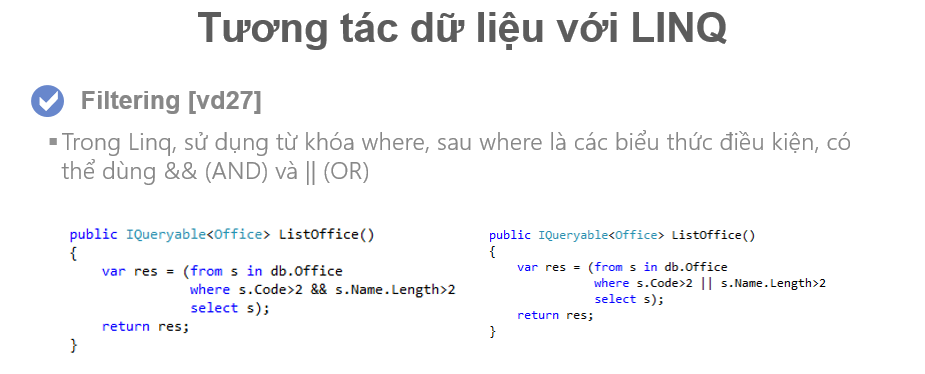


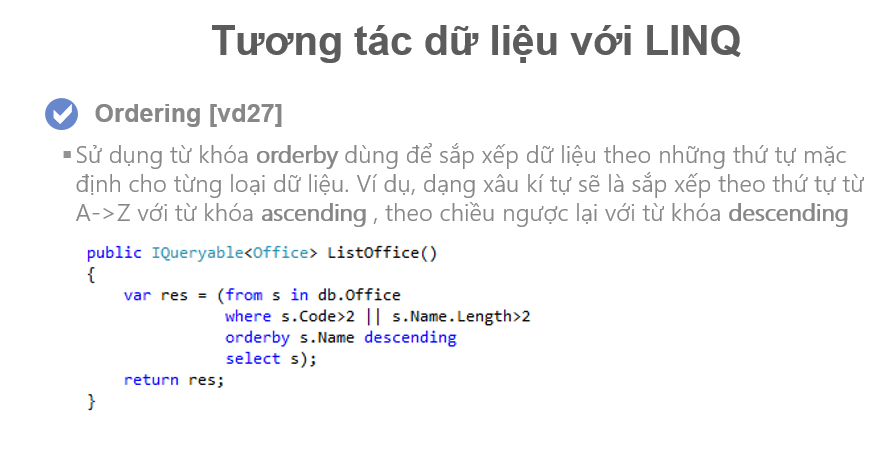


Câu 22 slide 20 bài 8









-Đọc thêm router

1. Giới Thiệu

[ASP.NET](http://asp.net/) Routing cho phép bạn sử dụng các URL không dẫn đến một file cụ thể in Website. Bởi vì URL không dẫn đến một file cụ thể trong ứng dụng web, bạn có thể làm cho URL trở nên thân thiện hơn với người dùng. Ở [ASP.NET](http://asp.net/) application không sử dụng routing ([ASP.NET](http://asp.net/) webform - đã bị khai tử), một request từ một URL thường được dẫn đến một tập tin xử lý request đó. Chẳng hạn:

http://server/application/Products.aspx?id=4

sẽ dẫn đến file có tên Product.aspx có chứa code xử lý và render để trả lời trình duyệt. Web page này sử dụng query string id = 4 để xác định nội dung được hiển thị. (Quen thuộc quá phải không các bạn, ngày xưa bạn nào làm webform chắc không lạ gì.)

Với MVC, ở ví dụ sau:

http://server/application/Products/show/beverages

URL sẽ không truy vấn tới file Products/show/beverages mà sẽ được định nghĩa bởi routing, chẳng hạn nếu Routing định nghĩa URL là server/application/{area}/{action}/{category} thì ở request URL trên, are = Product, action = show và category = beverages.

2. Router

Browser (trình duyệt) yêu cầu một địa chỉ từ Controller Acction trong [ASP.NET](http://asp.net/) MVC Framework được gọi là định tuyến URL (URL routing). URL routing sẽ chỉ định yêu cầu (request) tới Controller Action. URL routing sử dụng một bảng định tuyến để điều khiển các yêu cầu (request). Bảng định tuyến được tạo khi ứng dụng được chạy lần đầu tiên. Bảng định tuyến được thiết lập trong file Global.asax.

3. URL Patterns

Một URL pattern có thể chứa các parameters. các parameters ngăn cách nhau bởi dấu /(slash character).

Khi một request URL được yêu cầu, URL được phân tách thành cách đoạn như trong Routing đã định nghĩa, các giá trị được truyền vào từ URL sẽ được xử lý dưới dạng key - value.

Trong URL pattern ta xác định chúng bằng cách để vào dấu {}, và định nghĩa dấu phân cách cho chúng, VD /, -. Ví dụ {language}-{country}/{action} là một URL pattern đúng còn {language}{country}/{action} là một URL pattern sai vì không có dấu phân cách.

Bảng dưới đây cho ta một vài ví dụ về URL pattern đúng và request URL tương ứng.

| Định nghĩa Router | Ví dụ về URL đúng |
| --- | --- |
| {controller}/{action}/{id} | /Products/show/beverages |
| {table}/Details.aspx | /Products/Details.aspx |
| blog/{action}/{entry} | /blog/show/123 |
| {reporttype}/{year}/{month}/{day} | /sales/2008/1/5 |
| {locale}/{action} | /US/show |
| {language}-{country}/{action} | /en-US/show |

Với [ASP.NET](http://asp.net/) MVC, URL điển hình sẽ chứa {controller} và {action}.

4. Thêm Routing vào MVC application

Thông thường, nếu bạn không cần các URL tương đối đặc biệt, bạn có thể sử dụng Routing được định nghĩa sẵn khi tạo project MVC, nếu bạn muốn chỉnh sửa routes trong ứng dụng MVC, bạn có thể sử dụng MapRoute. Ví dụ dưới đây là đoạn code định nghĩa routes trong ứng dụng MVC.

public class MvcApplication : System.Web.HttpApplication

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

"Default", // Route name

"{controller}/{action}/{id}", // URL with parameters

new { controller = "Home", action = "Index", id = "" } // Parameter defaults

);

}

protected void Application\_Start()

{

RegisterRoutes(RouteTable.Routes);

}

}

5. Setting Default Values cho URL parameters

Khi bạn định nghĩa một route, bạn có thể xác định default value cho paramete của route. Default value sẽ được sử dụng khi trong URL không chứa parameter. Ví dụ dưới đây cho chúng ta thấy cách thêm default values, sử dụng MapPageRoute

void Application\_Start(object sender, EventArgs e)

{

RegisterRoutes(RouteTable.Routes);

}

public static void RegisterRoutes(RouteCollection routes)

{

routes.MapPageRoute("",

"Category/{action}/{categoryName}",

"~/categoriespage.aspx",

true,

new RouteValueDictionary

{{"categoryName", "food"}, {"action", "show"}});

}

Khi [ASP.NET](http://asp.net/) xử lý URL request, route sẽ được định nghĩa như bảng dưới đây:

| URL | Parameter values |
| --- | --- |
| /Category | action = "show" (default value) categoryName = "food" (default value) |
| Category/add | action = "add" categoryName = "food" (default value |
| /Category/add/beverages | action = "add" categoryName= "beverages" |

6. Làm thế nào URL khớp với Rotues

Khi Routing xử lý các URL request, nó sẽ cố gắng xử lý URL đề phù hợp với một route nào đó, với các điều kiện sau đây:

* Các routes patterns bạn định nghĩa hoặc các default route patterns, nếu có trong project.
* Thứ tự bạn thêm chúng vào Routes collection.
* Bất kỳ default value nào bạn định nghĩa cho một route.
* Bất kỳ hạn chế(validate) nào bạn định nghĩa cho một route.
* Bạn xác định một route để request tới một file vật lý.

Để tránh việc sai sót trong xử lý các request, bạn phải xem xét tất cả các điều kiện trên khi bạn định nghĩa routes. Bạn cũng nên tránh việc trùng lặp routes không cần thiết như ví dụ sau: Ví dụ: bạn có các rotues như sau:

* Route 1 được định nghĩa là: {controller}/{action}/{id}
* Route 2 được định nghĩa là: products/show/{id} Route sẽ không bao giờ phải xử lý bất kỳ một request nào từ URL vì routes 1 được định nghĩa trước, và các request phù hợp với route 2 sẽ luôn phù hợp với route 1, như: request http://server/application/products/show/bikes phù hợp với route 2 nhưng nó sẽ được xử lý bới route 1 với các giá trị:
* controller là product
* action là show
* id là bikes Default values sẽ được sử dụng nếu request không chứa parameter được định nghĩa. Bạn có thể xem xét ví dụ sau:
* Route 1: {report}/{year}/{month}, với default values là year và month.
* Route 2: {report}/{year}, với default value flà year. Route 2 sẽ không bao giờ được xử lý request, bởi vì với default là year, month thì request phù hợp với Route 2 sẽ luôn phù hợp với Rotue 1. Điều này có thể trái với mong muốn của bạn vì Route 1 là report theo tháng còn route 2 là report theo năm. Bạn có thể tránh được điều này bằng cách thêm vào một constaint, như là: annual/{report}/{year} và monthly/{report}/{year}/{month}.