**ĐỀ CƯƠNG ÔN THI**

1. **Khái niệm Session, vòng đời của nó. Viết ví dụ sử dụng Session trong MVC .Net**

- Khái niệm: Một session được hiểu như một phiên làm việc giữa client và server. Đối tượng Session là biến được tạo ra ở mỗi phiên làm việc của một người dùng, vậy là mỗi người đăng nhập vào sẽ có một session riêng và phiên làm việc của họ cũng có biến session riêng và được hủy khi phiên làm việc đó kết thúc và họ logout.

- Vòng đời: bắt đầu khi có request từ client, kết thúc khi người dùng đóng ứng dụng hoặc khi session hết thời gian (thuộc tính time out).

- Ví dụ sử dụng trong MVC .Net:

|  |
| --- |
| *Giả sử mình tạo một trang đăng nhập: dangnhap.aspx với giao diện bao gồm textbox lấy tên và pass word và nút “Gửi”*  *-->*  *Trong nút “Gửi” đoạn code sau:*  *protected void Submit\_Click(object sender, EventArgs e)*  *{*  *Session["ten"] = txtTen.Text;*  *Session["pass"] = txtPass.Text;*  *Response.Redirect("Nhan.aspx");*  *}*  *-->*  *Ở đây khai báo hai biến Session một là lấy tên, hai là lấy Pass và tiến hành gửi qua trang Nhận: Nhan.aspx*  *Ở trang Nhan.aspx bạn lấy thông tin như sau:*  *protected void Page\_Load(object sender, EventArgs e)*  *{*  *string ten = Session["ten"].ToString();*  *string pass = Session["pass"].ToString();*  *Response.Write("Chào bạn: " + ten + "<br />Pass: "+pass);*  *}*  *Sau đó bạn chạy trang Đăng nhập lên nhập thông tin và gửi sẽ nhận được thông tin ở trang nhận.* |

1. **Khái niệm Application, vòng đời của nó. Viết ví dụ sử dụng Application trong MVC .Net**

- Khái niệm: Đối tượng application dùng để quản lý tất cả các thông tin trong ứng dụng web, biến application được tạo ra có thể dùng ở mọi trang trong site và chỉ được hủy khi kết thúc ứng dụng (application tồn tại trong suốt quá trình sống của ứng dụng).

- Ví dụ:

|  |
| --- |
| *Giả sử mình làm một ứng dụng get Họ và Tên sử dụng đối tượng Application (một giao diện đơn giản bao gồm một text box lấy tên và nút gửi)*  *-->*  *Trong sự kiên click của nút gửi bạn sử dụng một biến application gán bằng tên sau đó chuyển hướng quá trang layTen.aspx.*  *protected void btGui\_Click(object sender, EventArgs e)*  *{*  *Application["Ten"] = txtTen.Text;*  *Response.Redirect("layTen.aspx");*  *}*  *Ở bên trang layTen.aspx bạn lấy giá trị biến Application đã tạo ra để hiển thị*  *protected void Page\_Load(object sender, EventArgs e)*  *{*  *string ten = Application["Ten"].ToString();*  *Response.Write("Tên Bạn Là: " + ten);*  *}* |

1. **Khái niệm Request. Ví dụ về cách sử dụng Request**

- Khái niệm: Request có thể hiểu nhanh là thông tin gửi từ client lên server. 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,... Chia làm hai nhóm chính là Nhóm Header và Nhóm chứa dữ liệu

- Ví dụ: Khi bạn lên trình duyệt browser 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, 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..

1. **Khái niệm Response. Ví dụ về cách sử dụng Response**

**-** Khái niệm: 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. Browser sẽ dựa vào các thông tin này để hiển thị trạng thái kết quả của request, còn mã HTML dùng để hiển thị giao diện của website. Nếu bạn nhập vào một URL không tồn tại thì thông tin của header sẽ không có gì.

1. **Các cách nhận dữ liệu đầu vào trong controller. Viết ví dụ trong trường hợp: dữ liệu truyền vào như là tham số của action.**

- Cách nhận dữ liệu: Controller thường xuyên làm công việc nhận dữ liệu đầu vào như là chuỗi query, giá trị từ form đưa lên hoặc là những tham số được hệ thống route parse từ incoming url. Chúng ta có 3 cách chính để truy xuất vào các dữ liệu loại này:

+ Chiết từ tập hợp các đối tượng context

+ Có dữ liệu truyền vào như là tham số của action

+ Gọi tính năng model binding một cách tường minh

- Ví dụ trường hợp nhận dữ liệu đầu vào như là tham số của action:

|  |
| --- |
| *Chúng ta có thể chiết các dữ liệu đầu vào bằng cách dùng các đối tượng context như đoạn mã sau*  ***public*** *ActionResult ShowWeatherForecast(){*  ***string*** *city = RouteData.Values["city"];*  *DateTime forDate = DateTime.Parse(Request.Form["forDate"]);*  *}*  *Tuy nhiên, chúng ta có thể dùng dạng ngắn gọn bằng cách truyền như tham số của action*  ***public*** *ActionResult ShowWeatherForecast(string city, DateTime forDate){*  *}* |

1. **Các cách nhận dữ liệu đầu vào trong controller. Viết ví dụ trong trường hợp: Lấy dữ liệu từ những đối tượng context**

- Cách nhận dữ liệu: Controller thường xuyên làm công việc nhận dữ liệu đầu vào như là chuỗi query, giá trị từ form đưa lên hoặc là những tham số được hệ thống route parse từ incoming url. Chúng ta có 3 cách chính để truy xuất vào các dữ liệu loại này:

+ Chiết từ tập hợp các đối tượng context

+ Có dữ liệu truyền vào như là tham số của action

+ Gọi tính năng model binding một cách tường minh

- Ví dụ trường hợp lấy dữ liệu từ những đối tượng context:

|  |
| --- |
| *Ví dụ:*  *public ActionResult RenameProduct() {*  *// Truy cập các thuộc tính khác nhau từ các đối tượng context*  *string userName = User.Identity.Name;*  *string serverName = Server.MachineName;*  *string clientIP = Request.UserHostAddress;*  *DateTime dateStamp = HttpContext.Timestamp;*  *AuditRequest(userName, serverName, clientIP, dateStamp, "Renaming product");*    *// Truy xuất dữ liệu đã đăng từ Request.Form*  *string oldProductName = Request.Form["OldName"];*  *string newProductName = Request.Form["NewName"];*  *bool result = AttemptProductRename(oldProductName, newProductName);*    *ViewData["RenameResult"] = result;*  *return View("ProductRenamed");*  *}* |

1. **Các cách nhận dữ liệu đầu vào trong controller. Viết ví dụ trong trường hợp: Gọi tính năng model binding một cách tường minh.**

- Cách nhận dữ liệu: Controller thường xuyên làm công việc nhận dữ liệu đầu vào như là chuỗi query, giá trị từ form đưa lên hoặc là những tham số được hệ thống route parse từ incoming url. Chúng ta có 3 cách chính để truy xuất vào các dữ liệu loại này:

+ Chiết từ tập hợp các đối tượng context

+ Có dữ liệu truyền vào như là tham số của action

+ Gọi tính năng model binding một cách tường minh

- Ví dụ trường hợp gọi tính năng model binding một cách tường minh:

|  |
| --- |
| *@model MvcApp.Models.Account*  *@{ ViewBag.Title = "Login"; }*  *<h2>Login</h2>*  *@using (Html.BeginForm()){*  *<p>Name: @Html.TextBoxFor(m => m.Name)</p>*  *<p>Password: @Html.PasswordFor(m => m.Password)</p>*  *<input type="submit" value="Login" />*  *}* |

1. **Các cách truyền dữ liệu đầu ra từ controller. Viết ví dụ: Truyền đối tượng bằng cách truyền nó như một tham số tới View**

- Các cách truyền dữ liệu đầu ra từ controller:

+ Sử dụng các ActionResult Types:

+ Truyền Data từ Action Method đến View

• Truyền đối tượng bằng cách truyền nó như một tham số tới View

• Truyền Data với ViewBag, ViewData

+

- Ví dụ truyền đối tượng bằng cách truyền nó như một tham số tới View:

|  |
| --- |
| *public ActionResult Index()*  *{*  *Account acc =* ***new*** *Account();*  *Acc.Name = ”admin”;*  *Acc.Password = ”123”;*  ***return*** *View(acc);*  *}* |

1. **Các cách truyền dữ liệu đầu ra từ controller. Viết ví dụ: Truyền Data với ViewBag**

- Các cách truyền dữ liệu đầu ra từ controller:

+ Sử dụng các ActionResult Types:

+ Truyền Data từ Action Method đến View

• Truyền đối tượng bằng cách truyền nó như một tham số tới View

• Truyền Data với View Bag:

+ Truyền bằng model

- Ví dụ truyền với ViewBag: ViewBag là thuộc tính động (dynamic), không yêu cầu chuyển kiểu với các kiểu dữ liệu phức tạp.

*public ActionResult Index()*

*{*

*ViewBag.Name = "Nhung Nguyen";*

*return View();*

*}*

1. **Khái niệm về Route (bộ định tuyến) trong MVC. Thêm các route mới**

- Khái niệm: Route (bộ định tuyến) trong MVC là các khai báo để ứng dụng MVC có thể biết được Action và Controller nào sẽ được gọi để xử lý yêu cầu từ phía người dùng.

Khác với trong webform, việc gọi các file aspx đường dẫn mang tính vật lý, thì các việc dùng route mang tính code và chúng ta có thể tùy biến các route để được các URL trông gọn gàng và mạch lạc.

Trong MVC4 , MVC5 thì việc định nghĩa Routed được khai báo trong lớp RouteConfig.cs (file này trong thư mục App\_Start)

Trong MVC2 , MVC3 thì được khai báo trong file Global.asax

- Thêm route mới: Ví dụ :

*Nếu muốn khi khởi chạy ứng dụng khi Action Index của Sample1Controller được gọi thì khai báo như sau :*

*routes.MapRoute(*

*name: "Default",*

*url: "{controller}/{action}/{id}",*

*defaults: new { controller = "Sample1”, action = “Index”, id = UrlParameter.Optional }*

*);*

*// khởi tạo mới 1 Route như sau :*

*routes.MapRoute(*

*"LogOn",*

*"Admin/{action}/{id}",*

*new { controller = "Account", action = "Index",id="" }*

*);*

*// khi người dùng nhập : /admin*

*thì Action Index của AccountController được gọi* .

1. **Vai trò của View trong MVC. Cách truyền dữ liệu: truyền bằng model từ Controller sang View**

- Vai trò View trong MVC: View được coi là một giao diện người dùng được sử dụng bởi khách truy cập trang web của bạn để nhìn thấy các dữ liệu. Các trang ASPX thường được sử dụng để hiển thị view trong các ứng dụng ASP.NET MVC.

- Cách truyền dữ liệu truyền bằng model từ Controller sang View: Model là một object bất kỳ. Model được truyền từ Controller sang View thông qua lệnh return View(model); Theo như trên, để truy xuất thông tin được truyền từ Controller ta sử dụng @Model, đây chính là đối tượng được truyền từ Controller thông qua lệnh return View(model);

**12**

Câu 12 : Khái niệm RAZOR ENGINE. Các quy tắc viết code với RAZOR ENGINE. Viết ví dụ theo yêu cầu

Trả lời:

1. Khái niệm RAZOR ENGINE

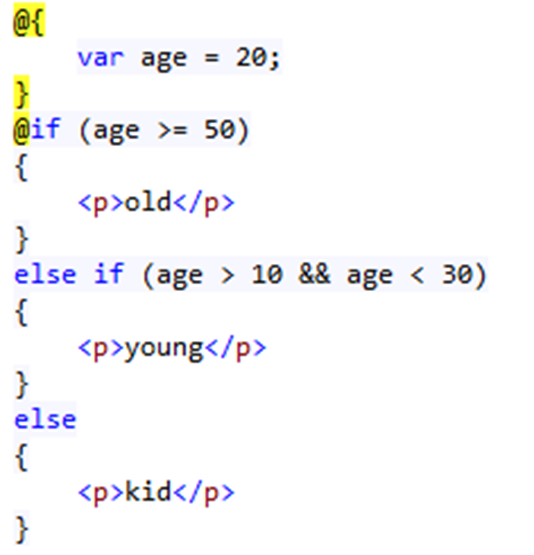
* Các phiên bản MVC3, MVC4, MVC5, ASP.net MVC có hỗ trợ một loại Engine mới đó là Razor View Engine, bản MVC trước đó và webform chỉ hỗ trợ aspx Engine.
* Mục đích của View Engine là sinh ra HTML bằng lập trình
* Những khai báo trong Razor” Các mã Razor, các thẻ html thuần, các HTML Helper. Thực chất Html Helper là các thư viện của Razor hỗ trợ lập trình MVC tạo ra các mã HTML.

2. Các quy tắc viết code với RAZOR ENGINE.

* Trong Razor có thể viết mã html thông thường như sau:

**<div> xin chao! </div>**

* Viết mã Razor có cấu trúc như mã C# được khai báo sau ký tự @

****

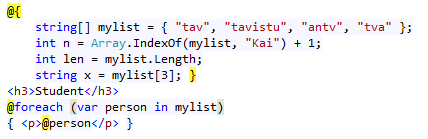
* Khối code được bao trong cặp @{...}

**@{**

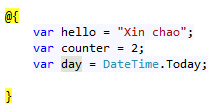
**string say=“Xin chao!”;**

**}**

* + Chuỗi hằng ký tự nằm trong cặp "..."



* + Các biến được khai báo bằng từ khóa var



* + C# files có đuôi mở rộng .cshtml

**13. Cách sử dụng HTML Helpers, viết ví dụ theo yêu cầu:**

- Giới thiệu

+ HTML Helper bao gồm các phương thức giúp tạo  
các thuộc tính HTML trên view.  
+ Tự động mã hóa

+ thuộc tính html là kiểu của System.Web.Mvc.HtmlHelper<T>

* Custom Helper Methods

+ Inline Html Helpers  
@helper ListingItems(string[] items){  
<ol> @foreach (string item in items){  
<li>@item</li>  
}  
</ol>  
}  
<h3>Programming Languages:</h3>  
@ListingItems(new string[] { "C", "C++", "C#" })  
<h3>Book List:</h3>  
@ListingItems(new string[] { "How to C", "how to C++",  
"how to C#" })

* Built-In Html Helpers:

**+** Standard Html Helpers

**+** Strongly Typed HTML Helpers

Strongly Typed HTML Helpers: Sinh ra các  
phần tử HTML cơ bản dựa trên các thuộc tính  
của model.  
▪ @Html.TextBoxFor  
▪ @Html.TextAreaFor  
▪ @Html.PasswordFor  
▪ @Html.HiddenFor  
▪ @Html.CheckBoxFor  
▪ @Html.RadioButtonFor  
▪ @Html.DropDownListFor  
▪ @Html.ListBoxFor,...

**+** Templated Helpers: sử dụng metadata và template . Metadata bao gồm thông tin về giá trị của model như tên, kiểu dữ liệu

html.display

html.editor

html.displayfor

html.editorfor

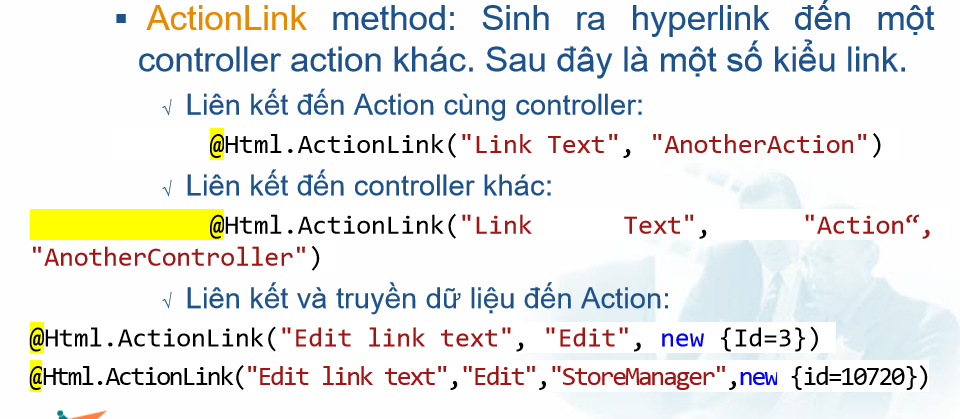
html.displayformodel

html.editorformodel

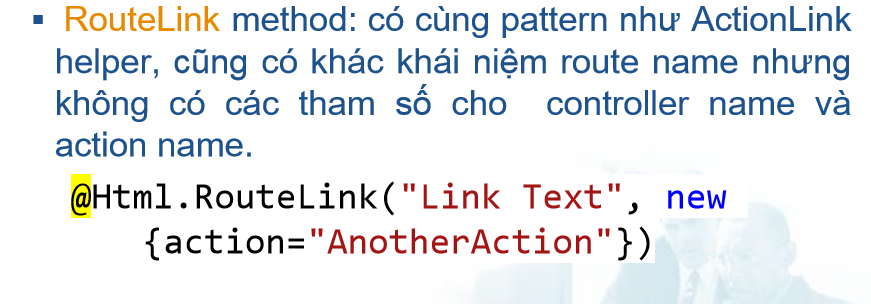
* Rendering Helpers:

**+** Html.ActionLink and Html.RouteLink

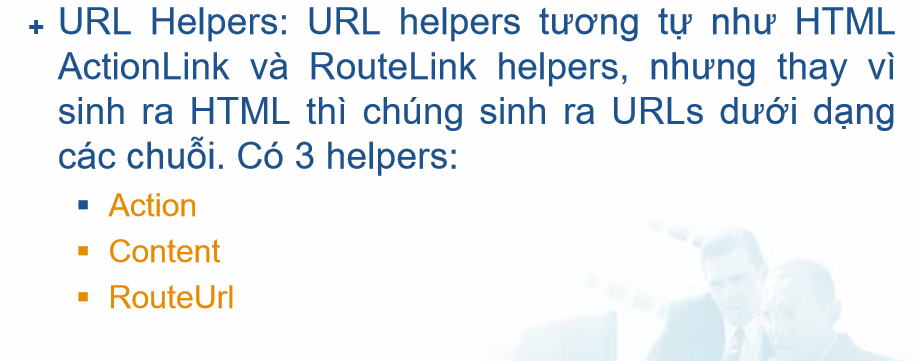
**Actionlink**



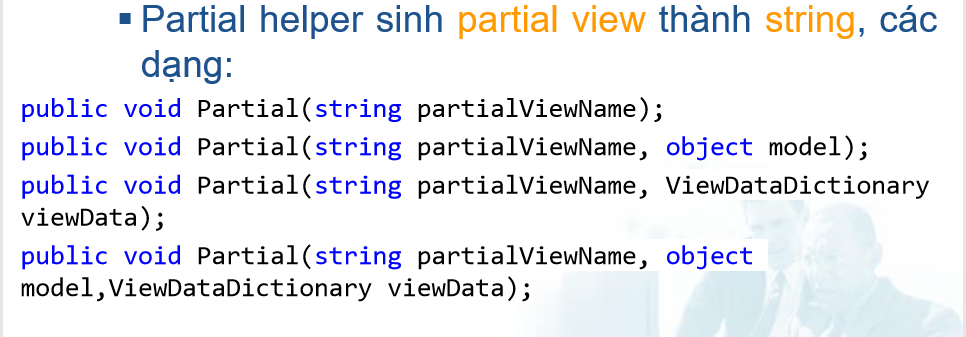
**Routerlink**

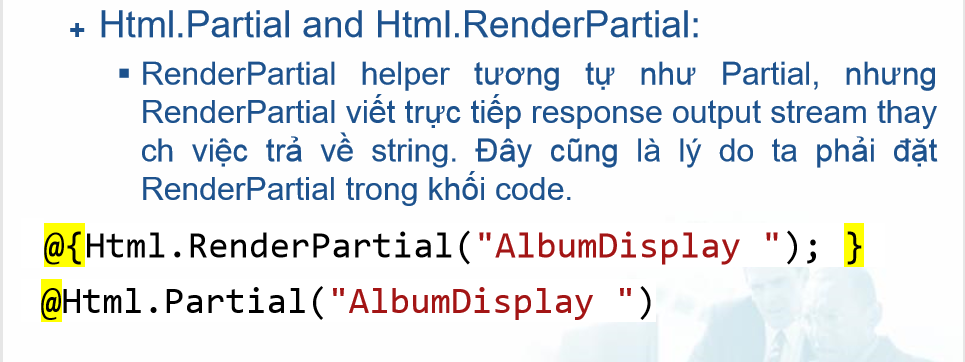


**+** URL Helpers:

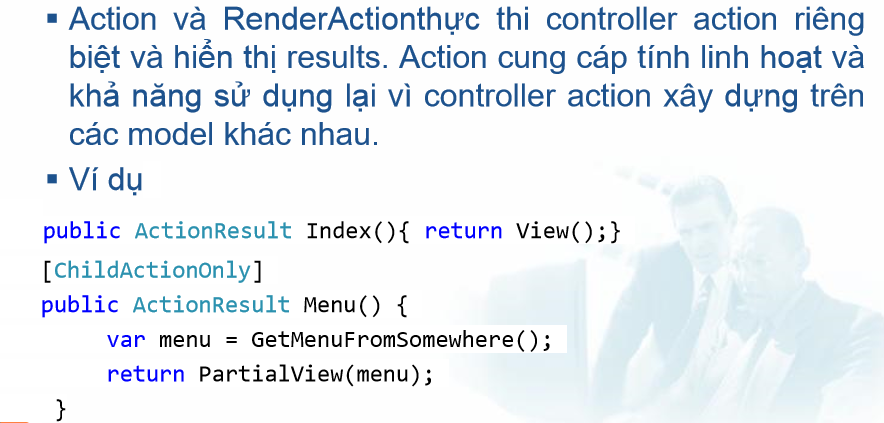


**+** Html.Partial and Html.RenderPartial:



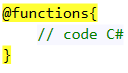


**+**  Html.Action and Html.RenderAction

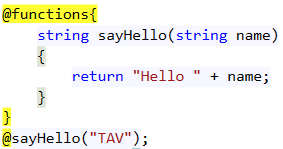


1. Cách sử dụng hàm trong HTML HELPERS. Viết ví dụ theo yêu cầu

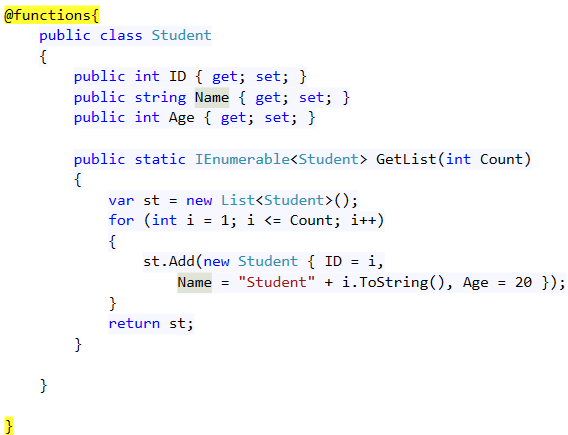
- Khai báo mã C# trong bổ từ khóa sau:

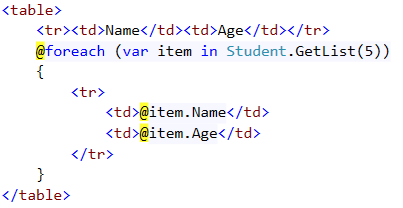


- Razor Code không hỗ trợ đầy đủ các kiểu dữ liệu hay thư viện .net, tuy nhiên ta có thể code mã C# để xây dựng đối tượng, phương thức trực tiếp trong đó.



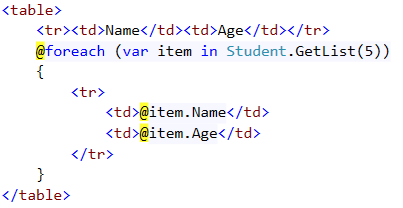
- Xây dựng đối tượng, phương thức trực tiếp trong đó.





- Ta có thể khai báo Student trong Models với namespace MVCDemo.Models , vậy ta có thể sử dụng ngay lớp đó với @using MVCDemo.Models;





- Có thể hiểu rằng chúng ta có thể tham chiếu tới cả những lớp bên ngoài ứng dụng (các lớp trong thư viện dll động)

- Như vậy các nhà phát triển có thể xây dựng kiến trúc phần mềm riêng cho mình dựa vào MVC .net

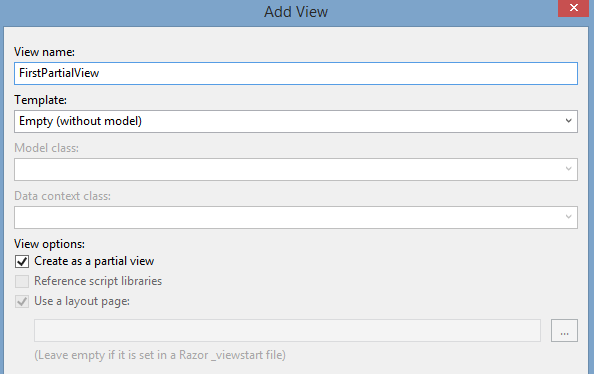
15. Sử dụng PartialView. Cách truyền vào PartialView nội dung theo một Model.

- PartialView là một loại thẻ đặc biệt, chứa các thẻ html thuần hoặc các thẻ html chứa model hiển thị dữ liệu.

- Tạo Partial

+ Trong Views/Login chọn

+ Add/View



- Nhập nội dung cho FirstPartialView



- Trong file Login.cshtml





Cách truyền vào PartialView nội dung theo một Model.

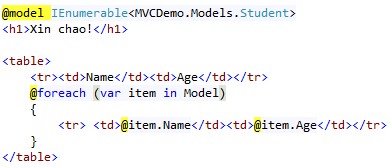
- Truyền vào PartialView nội dung theo một Model nào đó theo cú pháp



- Ví dụ: Trong Login.cshtml



- Trong FirstPartialView.cshtml



16. Khái niệm Layout trong MVC. Cách sử dụng template vào chương trình.

Trong một website các bạn sẽ thấy một page lúc nào cũng có đủ các thành phần bao gồm: header, footer, body ... Nếu các bạn để ý sẽ thấy phần header và phần footer của các page thuộc cùng website đều giống nhau.

Tuy nhiên theo cách thiết kế thông thường, cứ mỗi khi tạo ra một page mới chúng ta phải tạo lại phần header, footer. Mà 2 phần này thì nội dung giống nhau ở tất cả các page, do đó sẽ làm mất rất nhiều thời gian cho người thiết kế.

-Layout là các vùng được ngăn ra trong các Template. Các vùng đó có thể là header, footer, left, right,...

-Layout trong MVC đặt trong Views/Shared

-Trong Layout , các thành phần thẻ bắt buôc như html, body

-@RenderBody() là nơi đặt nôi dung kế thừa của các trang khác

-Một View muốn sử dụng layout thì khai báo như sau:



- Layout là một template hỗ trợ hiển thị cố định, các trang web khác kế thừa nội dung.

- Các đối tượng xử lý layout trong Razor: RenderBody, RenderPage, RenderSection, LayoutPage

- RenderBody: Nội dung trang kế thừa layout sẽ hiển thị đúng vào vị trí RenderBody

- RenderPage dùng để gọi nội dung từ một View khác

- RenderSection cần được khai báo trong các trang kế thừa

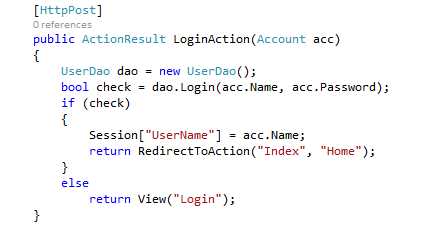
20. Sử dụng EF, truy vấn vào một bảng dữ liệu

EF là một ORM (Object Relational Mapper – Bộ ánh xạ đối tượng quan hệ) hỗ trợ phát triển các phần mềm ứng dụng hướng dữ liệu.Với EF, chúng ta có thể chuyển các đối tượng cơ sở dữ liệu, như các bảng (tables), thành các đối tượng .NET để chúng ta có thể dùng chúng trong truy vấn hay các ngữ cảnh kết buộc dữ liệu. Bên cạnh đó, EF cũng cho phép chúng ta thiết kế mô hình đối tượng trước rồi sau đó EF sẽ tạo ra cấu trúc dữ liệu cần thiết cho chúng ta. Với EF, chúng ta có thể làm việc dễ dàng với cơ sở dữ liệu mà không viết quá nhiều các dòng mã.

Như đã trình bày ở trên, EF sẽ giúp chúng ta ánh xạ các thành phần của cơ sở dữ liệu như các bảng, cột, quan hệ đến các đối tượng và các thuộc tính trong một mô hình đối tượng trong ứng dụng của chúng ta

* Khi sử dụng EF chúng ta có 3 lựa chọn đó là:
* Database First ( thiết kế database trước ) , sau đó dùng VS tạo lớp entity .Kỹ thuật này giống với kỹ thuật tạo lớp Entity dùng Linq to Sql .
* Model fist : Dùng VS tạo lớp Entity , sau đó từ Entity sinh ra database
* Code first : Tạo các lớp ứng dụng , và lớp Entity như các đối tượng C# ( vb.net ) thông thường. Khi ứng dụng chạy sẽ tự động tạo ra 1 CSDL tương ứng với các lớp .

Ví dụ truy vấn vào một bảng dữ liệu

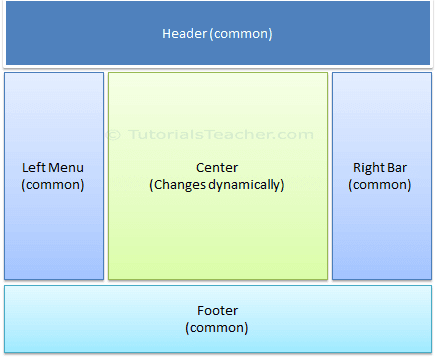


**Câu 17**

Một website có thể chứa các phần chung trong các giao diện người dùng mà vẫn giữ nguyên trong suốt ứng dụng chẳng hạn như logo, tiêu đề, thanh điều hướng bên trái,thanh điều hướng bên phải hoặc phần footer đó được gọi là layout web.

**ASP.NET MVC** giới thiệu **Layout view** là một view mà có thể dùng lại trong nhiều view khác để cung cấp một giao diện nhất quán trong nhiều trang của một site. Layout View giống như master page của ứng dụng ASP.NET Webform.

Ví dụ, một giao diện người dùng có thể chứa header, thanh menu trái, thanh menu phải và phần chân trang mà vẫn giống nhau trong tất cả các trang và chỉ có phần trung tâm thay đổi tự động theo từng trang trong website như hình dưới đây.

[](https://1.bp.blogspot.com/-zx6LVSpEsgo/WSA62FjwyxI/AAAAAAAACkc/-rGFfB9jeG0RwrxYVPCHu1HiGck6WeNcACLcB/s1600/layout-view-asp-net-mvc.png)

Khi tạo ứng dụng ASP.NET MVC, mặc định file layout có tên **\_Layout.cshtml** trong thư mục **Views/Shared.**  
  
File ***\_Layout.cshtml*** giống với view khác ngoại trừ việc trong Layout view có chứa 2 lệnh gọi method ***@RenderBody và @RenderSection.***

Layout chứa một và chỉ một lệnh **@RenderBody()** để giữ chỗ cho nội dung trong những view kế thừa layout.

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

Ví dụ

**\_Layout.cshtml** trong thư mục **Views/Shared.**

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<p>day la trang chu</p>

@RenderBody()

</body>

</html>

Trong Controller tạo 1 HomeController

public ActionResult Index()

{

return View();

}  
Tạo 1 View

@{

ViewBag.Title = "Index";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<p>goi den trang nay</p>

Kết quả

day la trang chu

goi den trang nay

**18. Cách sử dụng RenderSection. Lấy ví dụ**

RenderSection() hiển thị nội dung của 1 thành phần nhỏ trong layout. Layout không có hoặc nhiều lệnh *@RenderSection()* để giữ chỗ cho những thành phần khác được đánh dấu thông qua khối *@section tensection { }* trong view kế thừa layout.

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.*

***Ví dụ:***

RenderSection cần được khai báo trong các trang kế thừa

* Tại \_Layout.cshtml

<div class="page-title-area">

<div class="row align-items-center">

<div class="col-sm-6">

<div class="breadcrumbs-area clearfix">

<h4>@RenderSection("header", required: false)</h4>

</div>

</div>

</div>

</div>

* Tại trang kế thừa Index.cshtml

@model IEnumerable<Model.DTO.UserDTO>

@{

ViewBag.Title = "Người dùng";

Layout = "~/Areas/Admin/Views/Shared/\_Layout.cshtml";

}

@section header{

Quản lý người dùng

}

**Câu 19: Cách sử dụng RenderPage. Lấy ví dụ**

- Các đối tượng xử lý layout trong Razor: RenderBody, RenderPage, RenderSection, LayoutPage

- RenderPage dùng để gọi nội dung từ một View khác

- @RenderPage được sử dụng để gọi nội dung từ một View khác. Ví dụ: Khi ta có văn bản và ảnh quảng cáo và ta muốn sử dụng tất cả chúng trên trang của mình nhưng với điều kiện là các chương trình khuyến mãi chứa các văn bản và hình ảnh này phải được thay đổi thường xuyên.

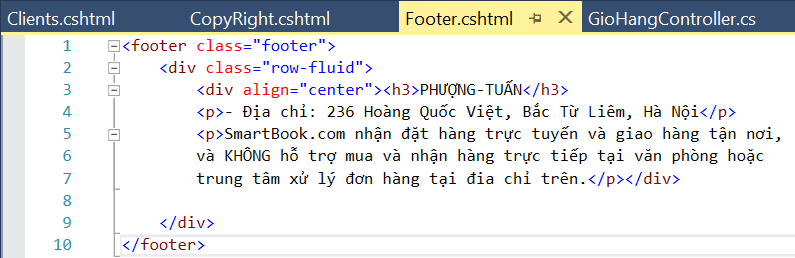
🡪 Ta chỉ cần tạo một trang chứa các văn bản và hình ảnh quảng cáo và sử dụng trang này bất cứ nơi nào ta muốn chèn. Khi trang web được hiển thị trong trình duyệt, nó sẽ hiển thị tất cả nội dung của trang đó. Khi ta muốn thay đổi văn bản quảng cáo, ta chỉ cần thay đổi nó ở một nơi và nó sẽ được thay đổi tương ứng ở tất cả các trang.

- Ví dụ:

+ B1: Tạo folder Common trong Views

+ B2: Tạo Empty Page với tên “Footer.cshtml”

Nội dung của trang “Footer.cshtml”:



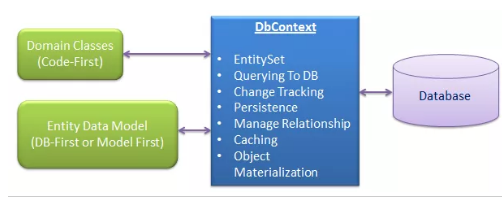
+ B3: Tại trang “Layout.cshtml”, gọi nội dung từ trang “Footer.cshtml” bằng RenderPage. Cú pháp:

**@RenderPage("~/Views/Common/Footer.cshtml")**



Câu 21: Vai trò của DbContext

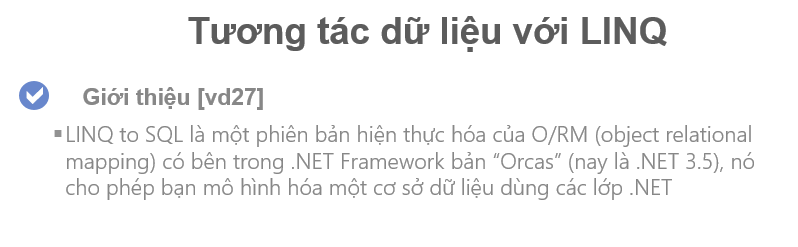
DbContext là một phần quan trọng của Entity Framework. Nó là một cầu nối giữa lớp domain hoặc thực thể và CSDL của bạn.



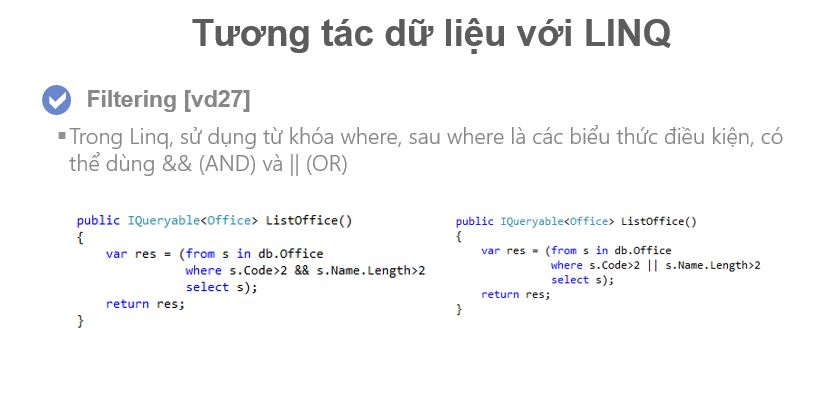
DbContext là lớp chính chịu trách nhiệm cho việc tương tác với dữ liệu như là đối tượng. DbContext chịu trách nhiệm cho các hoạt động sau:

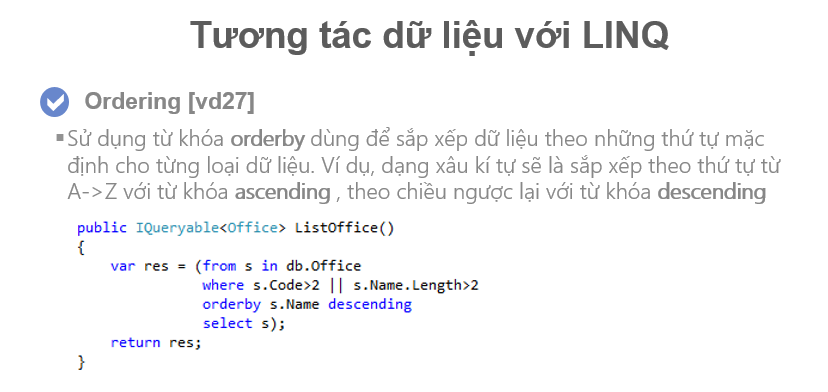
* **EntitySet:** DbContext chứa tập thực thể (DbSet<TEntity>) cho tất cả thực thể nối với những bảng của CSDL.
* **Querying:** DbContext chuyển đổi những truy vấn LINQ-to-Entities thành truy vấn SQL và gửi nó tới CSDL.
* **Change Tracking:** Nó giữ việc theo dõi những thay đổi xảy ra trong những thực thể sau khi nó đã truy vấn từ CSDL.
* **Persisting Data:** Nó cũng thực hiện các thao tác Insert, Update và Delete tới CSDL dựa trên những gì mà thực thể thể hiện.
* **Caching:** DbContext mặc định thực hiện caching mức đầu tiên. Nó lưu những thực thể đã được nhận suốt vòng đời của một lớp context.
* **Manage Relationship:** DbContext cũng quản lý những quan hệ sử dụng CSDL, MSL và SSDL trong phương pháp tiếp cận DB-First hoặc Model-First hoặc sử dụng fluent API trong phương pháp tiếp cận Code-First.
* **Object Materialization:** DbContext chuyển đổi bảng dữ liệu thô vào những đối tượng thực thể.

Câu 22.

****

****

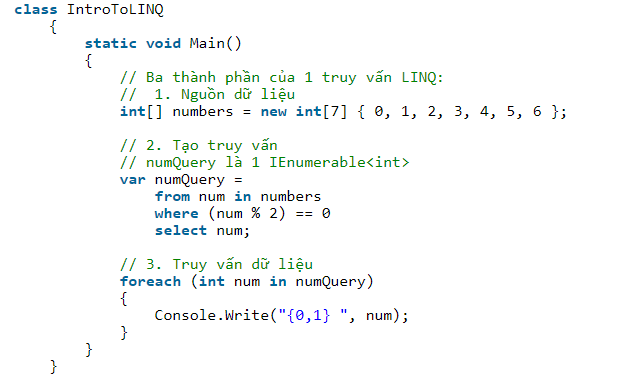
****

****

Ba thành phần của 1 hoạt động truy vấn  
Tất cả các hoạt động truy vấn [LINQ](https://www.dammio.com/glossary/linq) đều bao gồm 3 tác vụ:

* Kết nối với nguồn dữ liệu (data source)
* Tạo truy vấn
* Thực thi truy vấn

Ví dụ sau đây mô tả cách 3 tác vụ xảy ra khi thực hiện 1 hoạt động truy vấn. Để thuận tiện, ví dụ sử dụng 1 mảng số nguyên là nguồn dữ liệu; tuy nhiên bạn có thể dùng các dạng nguồn dữ liệu khác.

****

**25. Thực hiện truy vấn thuần SQL trong truy vấn 1 bảng với MVC**

**Trong file UserDAO.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Model.EF;

using PagedList;

using Model.DTO;

using System.Data.SqlClient;

namespace Model.DAO

{

public class UserDAO

{

public User FindID(int id)

{

string connectionString = "Data Source=DESKTOP-BH8VJ9Q;Initial Catalog=GirlyShop;Integrated Security=True";

var model = new User();

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand("", connection))

{

command.CommandText = "select \* from User where ID=@Id";

command.Parameters.AddWithValue("@Id", id);

SqlDataReader reader = command.ExecuteReader();

model.ID = id;

model.UserName = reader["UserName"].ToString();

model.UserName = reader["Password"].ToString();

}

return model;

}

}

}

**Trong file UserController:**

public ActionResult Details(int id)

{

var user = new UserDAO().FindID(id);

return View(user);

}

**Trong View Details:**

@model Model.EF.User

@{

ViewBag.Title = "Details";

}

<h2>Details</h2>

<div>

<h4>User</h4>

<hr />

<dl class="dl-horizontal">

<dt>

@Html.DisplayNameFor(model => model.UserName)

</dt>

<dd>

@Html.DisplayFor(model => model.UserName)

</dd>

</dl>

</div>