Câu 1:

**Session**

Một session hay còn gọi là một phiên làm việc. Trong khoa học máy tính, Nó đơn giản là cách giao tiếp giữa client (ở đây là trình duyệt web hoặc ứng dụng trên thiết bị của bạn) với server. Một session bắt đầu khi client gửi request đến sever, nó tồn tại xuyên suốt từ trang này đến trang khác trong ứng dụng và chỉ kết thúc khi hết thời gian timeout hoặc khi bạn đóng ứng dụng. Giá trị của session sẽ được lưu trong một tệp tin trên máy chủ. Chính vì điều này, nếu bạn dùng session một cách vô tội vạ (giống như mình đã từng đó) thì sẽ khiến cho máy chủ phải lưu rất nhiều. Đặc biệt nếu ứng dụng đó có đến vài triệu người dùng chẳng hạn thì ... điều đó thật là kinh khủng. Thông thường chúng ta chỉ nên lưu trữ những thông tin tạm thời trong session VD như: thông tin đăng nhập, thông tin các sản phẩm trong giỏ hàng (đối với các trang web thương mại điện tử)...

Với mỗi session sẽ được cấp phát một định danh duy nhất SessionID. Khi kết thúc một phiên làm việc và bắt đầu một phiên mới, dĩ nhiên bạn sẽ được cấp một SessionID khác với trước đó.

Bạn đã bao giờ đặt ra câu hỏi rằng "sau khi tạo ra một session được lưu trên mấy chủ, thì làm thế nào hệ thống biết được rằng session đó là của client nào chưa?". Easy thôi, với mỗi session được tạo ra, đồng thời chúng sẽ tạo ra một tệp tin cookie lưu trên trình duyệt của bannj ứng với session đó. Như vậy chỉ cần so sánh tệp tin cookie bên phía client được gửi lên sever và tệp session được lưu trên server là ra ngay ý mà (ahaha)

Câu 2: https://oktot.net/vongdoitrangasp/

Câu 3: của dung

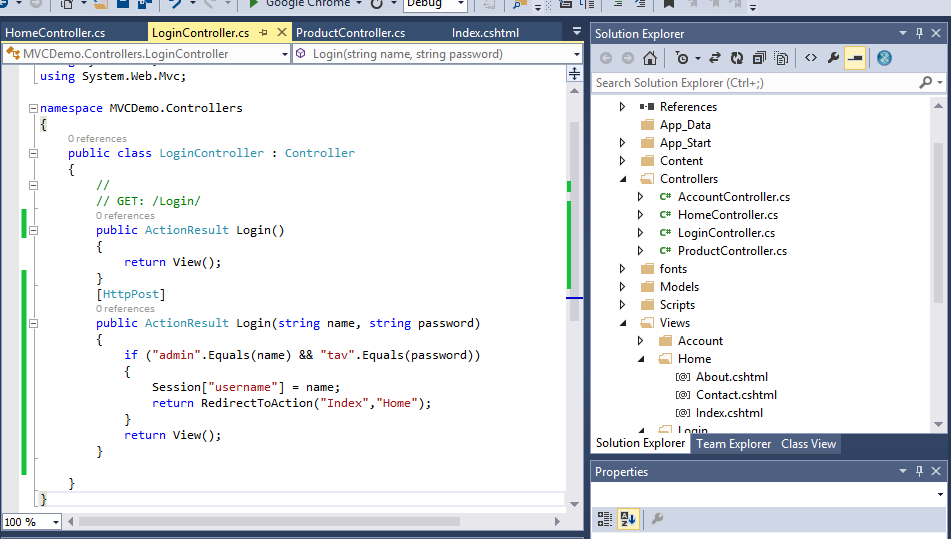
Câu 4: Của dung

Câu 5 : Cách nhận dữ liệu đầu vào trong controller

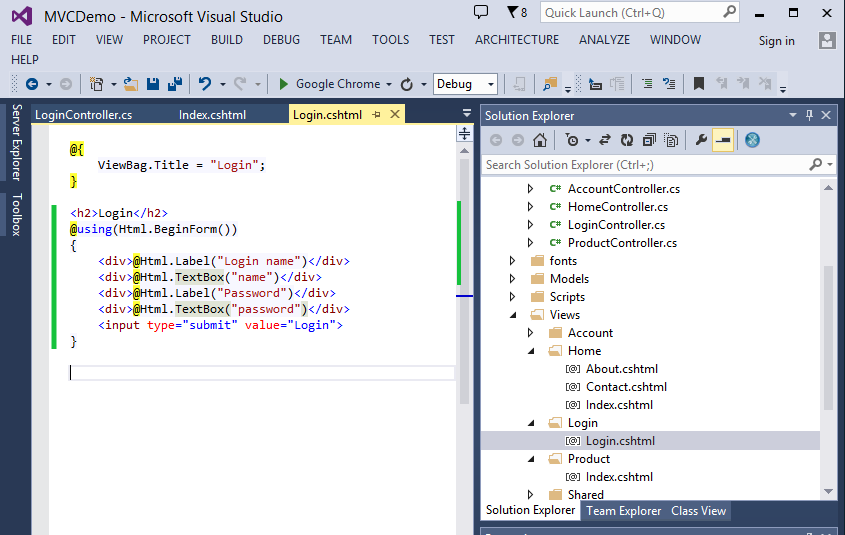
* Controller thường xuyên làm công việc nhận dữ liệu đầu vào như là chuỗi query, giá trị từ form đưa lên hoặc là những tham số được hệ thống route parse từ incoming url. Có 3 cách chính để truy xuất vào các dữ liệu loại này:
  + Có dữ liệu truyền vào như là tham số của action
  + Lấy dữ liệu từ những đối tượng context
  + Gọi tính năng model binding một cách tường minh

+ **Lấy dữ liệu truyền vào như là tham số của action**

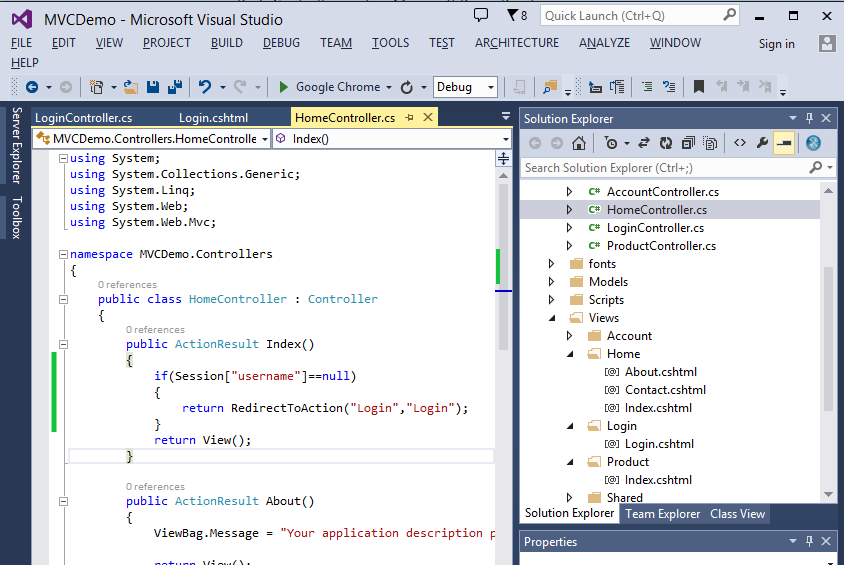
* Trong Controller tạo LoginController



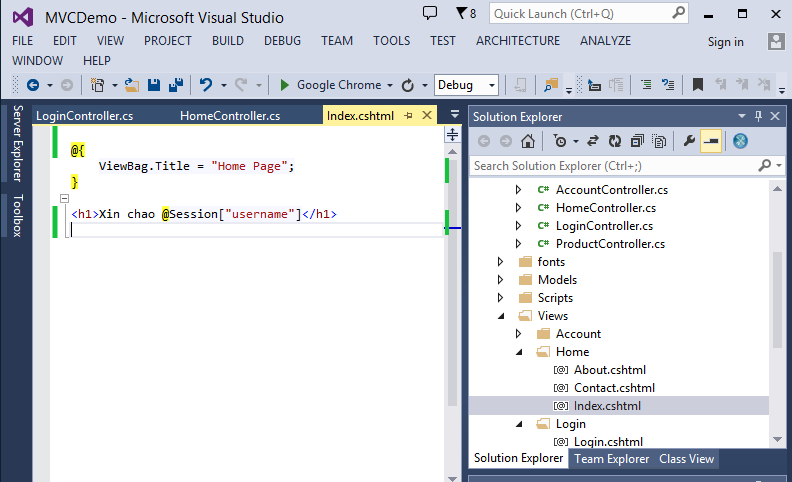
* Trong Views/Login tạo Login.cshtml



* Thay đổi code của HomeController

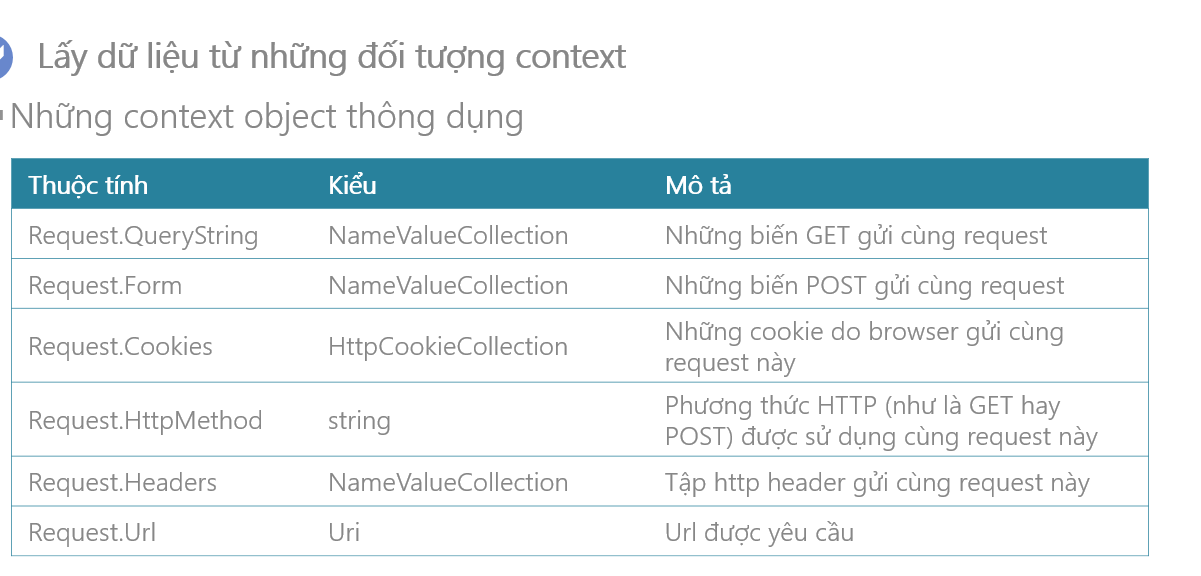


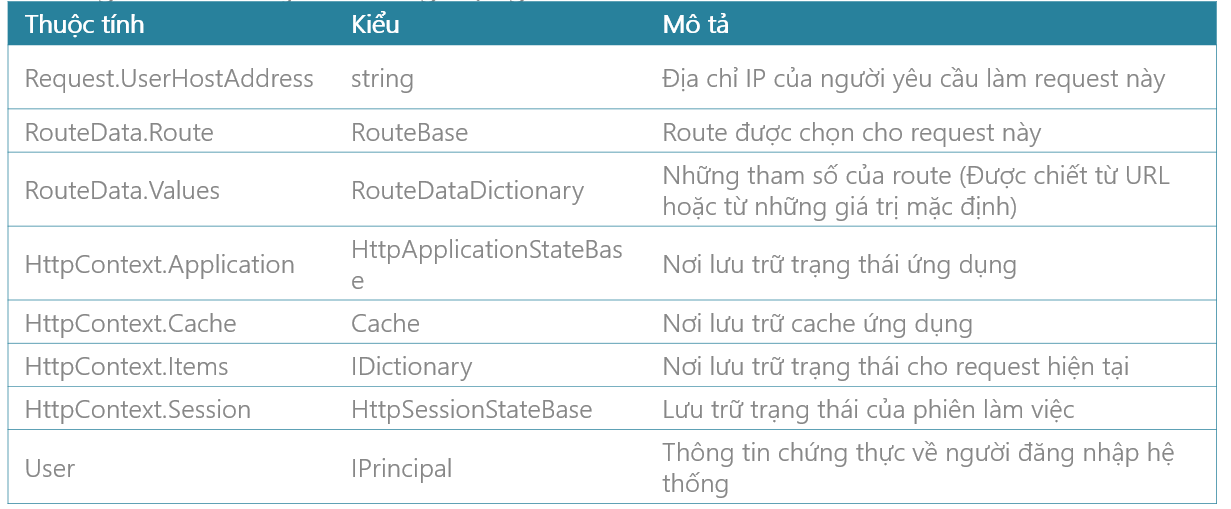
* Thay đổi code của Views/Home/Index.cshtml



Câu 6: **Nhận dữ liệu đầu vào trong controller**

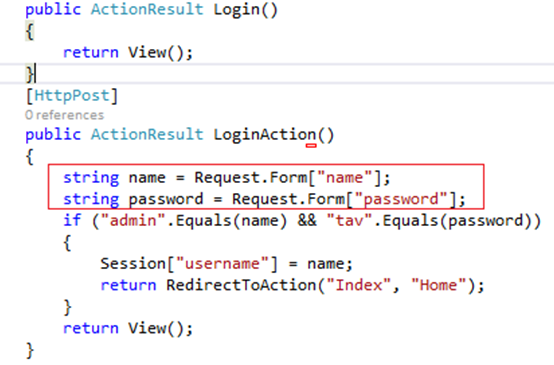
Lấy dữ liệu từ những đối tượng context



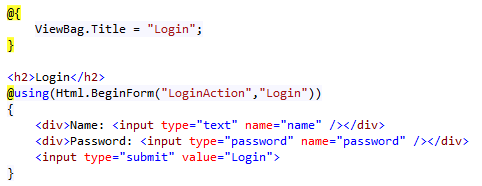


Ví dụ :

B1: Controller tạo LoginController

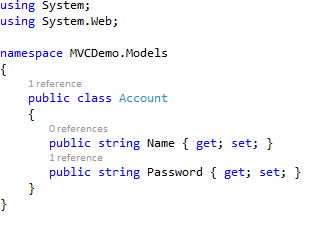


* B2: Views/Login/Login.cshtml

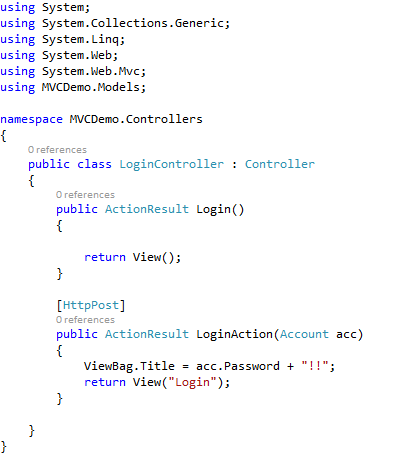


Câu 7: Lấy dữ liệu truyền vào thông qua Data Model Binding

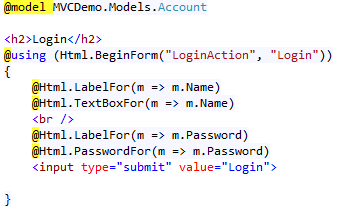
B1: Trong folder Models, thêm class Account:



B2: LoginAction trong LoginController



B3: Views/Login/Login.cshtml



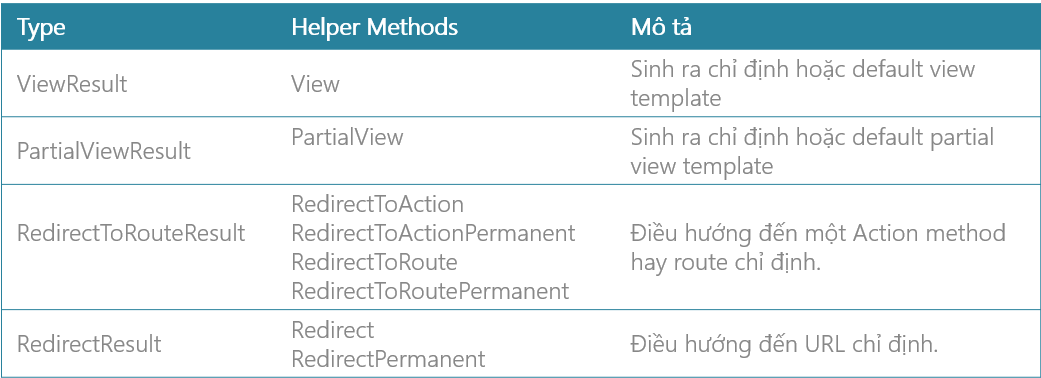
Câu 8: cách truyền dữ liệu đầu ra từ controller ( slide 22 chương 6)

Controller sau khi nhận dữ liệu thông qua Request, tiến hành xử lý, sẽ trả về dữ liệu thông qua một Response

* Có thể trả về nội dung thông qua Response.Write.
* Có thể điều hướng trang thông qua Response.Redirect

Mô hình MVC sử dụng đối tượng một đối tượng dẫn xuất từ class ActionResult thay cho đối tượng Response.

Các ActionResult Types :



Các cách:

* + - Cách 1: Truyền Data từ Action Method đến View :

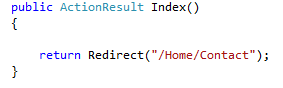
+ Truyền đối tượng bằng cách truyền nó như một tham số tới View

+ Truyền Data với ViewBag

* + - Cách 2: Điều hướng thông qua Redirections

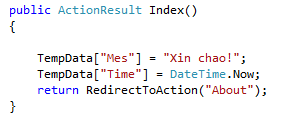
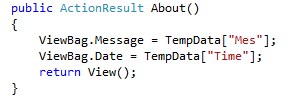
+ Điều hướng đến Literal URL:

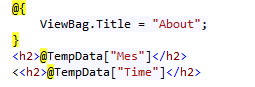
Để điều hướng trình duyệt ta gọi phương thức Redirect với tham số là URL cần được dẫn tới



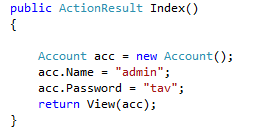
+ Điều hướng Routing System URL:

Sử dụng Routing System URL để sinh ra valid URLs dùng RedirectToRoute method, tại đó xác định controller, acction, id được điều hướng

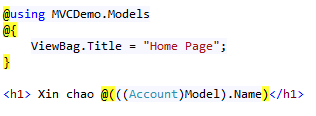
 



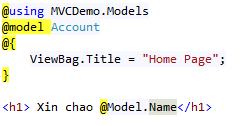
Ví dụ : Truyền đối tượng bằng cách truyền nó như một tham số tới View



* Phía View tiếp nhận. Để truy cập đối tượng này trong view sử dụng Razor Model keyword hoặc Razor model keyword

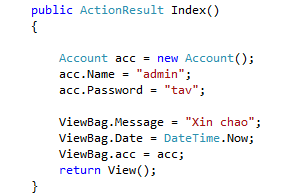
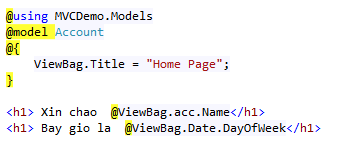


Hoặc



Câu 9: Các cách truyền dữ liệu đầu ra từ controller.

ví dụ: Truyền Data với ViewBag

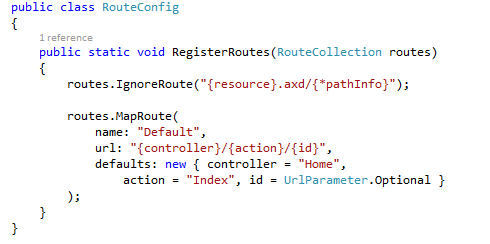
 

Câu 10: Khái niệm về Route (bộ định tuyến) trong MVC. Thêm các route mới(slide 31- chương 6)

* 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 bằng code và chúng ta có thể tùy biến các route để được các URL gọn gàng và mạch lạc.

Route so khớp với các requests đến với một một file trong file system và ánh xạ request đó tới controller action

Route được định nghĩa trong file RouteConfig.cs trong thư mục App\_Start



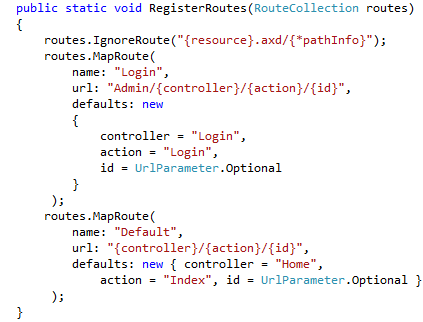
* Có thể định nghĩa nhiều Route, ở đây tạo sẵn một route có tên là Default .
* Home Controller, Action Index là giá trị mặc địn khi ứng dụng chạy .
* Việc gọi các Action trong Controller bất kỳ tuân theo cấu trúc :

**/TenController/Action/ThamSo**

* Với những Action không có tham số thì chỉ cần gọi : **/TenController/Action/**
* Với Action có tên là Index thì chỉ cần gọi : **/TenController**

-Thêm các route mới

+ Khi 1 yêu cầu từ phía client thì router sẽ tiến hành map từ trên xuống dưới, khi nào tìm được route có đủ các yêu cầu về tham số và tên thì dừng lại.



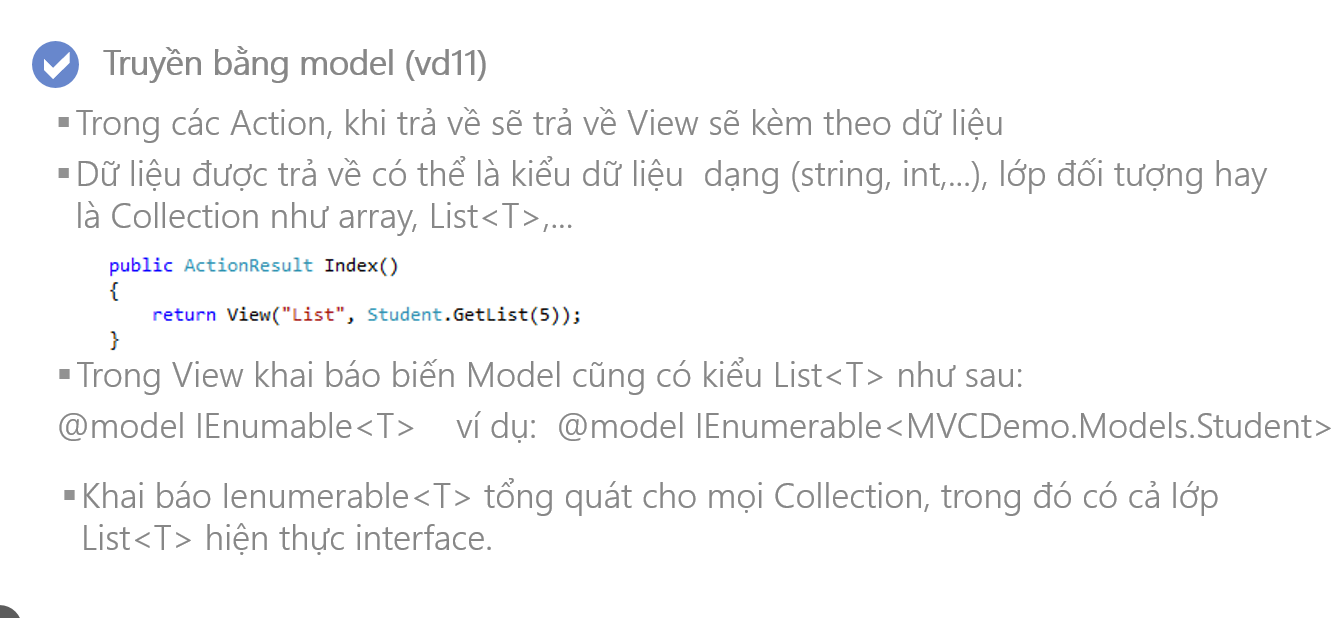
Câu 11: Vai trò của View trong MVC. Cách truyền dữ liệu: truyền bằng model từ Controller sang View

-Vai trò của View :

* View: có nhiệm vụ tiếp nhận dữ liệu từ controller và hiển thị nội dung sang các đoạn mã HTML.
* Trong mô hình MVC Controller thường cung cấp thông tin cho view bằng cách truyền dữ liệu. View chuyển dữ liệu này thành định dạng biểu diễn được cho người dùng.

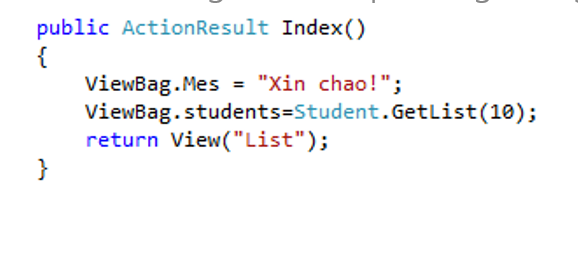
- Cách truyền dữ liệu: truyền bằng model từ Controller sang View :

Cách 1: Truyền bằng model



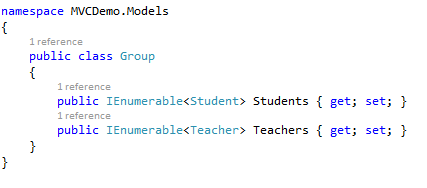
Cách 2: Truyền bằng ViewBag

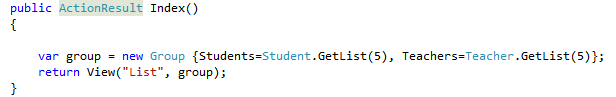
* Trong trường hợp Action trả về nhiều đối tượng Model khác thì ta dùng ViewBag
* ViewBag là đối tượng Dynamic, có thể gán cho chúng bất cứ dữ liệu nào, trong View có thể gọi trực tiếp chúng thông qua tên



Cách 3: Truyền bằng ViewModel

* Có thể dùng ViewModel thay thế ViewBag, giả sử ta muốn truyền 2 list về View





\*Ghi chú :

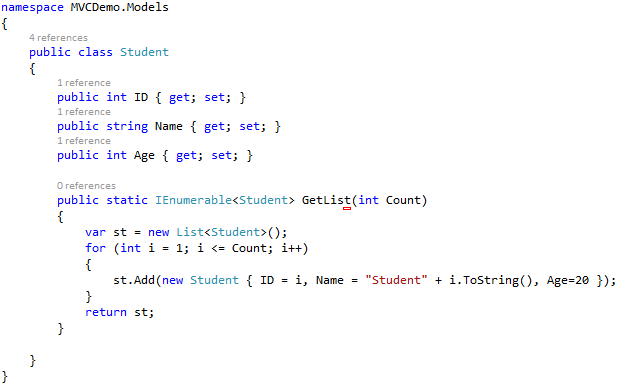
Có thể truyền dữ liệu đến view thông qua đối tượng ViewBag hoặc ViewData, tuy nhiên có một vấn đề khi sử dụng hai đối tượng này (tuy hai mà một) là ứng dụng sẽ không biết được các đối tượng có chứa những gì, và mã lệnh có đúng hay không cho đến khi ứng dụng chạy. Nếu lập trình sơ sót, khả năng sinh lỗi là rất cao.

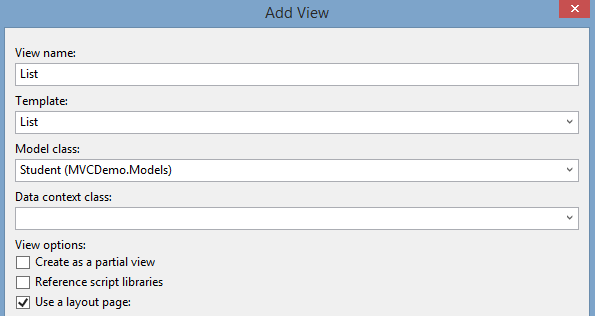
ASP.NET MVC cũng cung cấp khả năng để truyền các đối tượng có kiểu rõ ràng cho view.

Bằng cách sử dụng dòng @model bên trong tập tin view template, có thể quy định kiểu của đối model mà view sẽ nhận được, lúc đó view sẽ hiểu model được nhận có kiểu gì.

Ví dụ :

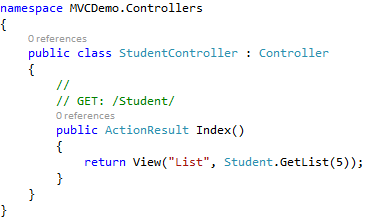
* Trong folder Models tạo class Student như sau:



* Tạo một View với các thông số sau:
* 



* Trong StudentController thay đổi như sau:



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

* 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.

\*Các quy tắc viết code với RAZOR ENGINE. Viết ví dụ theo yêu cầu

* 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!”;

}

<h2>@say</h2>

* + Chuỗi hằng ký tự nằm trong cặp "..."
  + Các biến được khai báo bằng từ khóa var
  + C# files có đuôi mở rộng .cshtml

Ví dụ :

**Một số vì dụ về Razor**

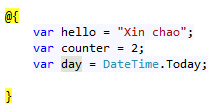
**▪ Các quy tắc cú pháp Razor chung cho C#  
🞦 Khối code được bao trong cặp @{...}  
🞦 Inline expressions (variables và functions) bắt  
đầu từ ký tự @  
🞦 Các câu lệnh kết bở dấu “;”  
🞦 Chuỗi hằng ký tự nằm trong cặp "..."  
🞦 Các biến được khai báo bằng từ khóa var  
🞦 C# files có đuôi mở rộng .cshtml**

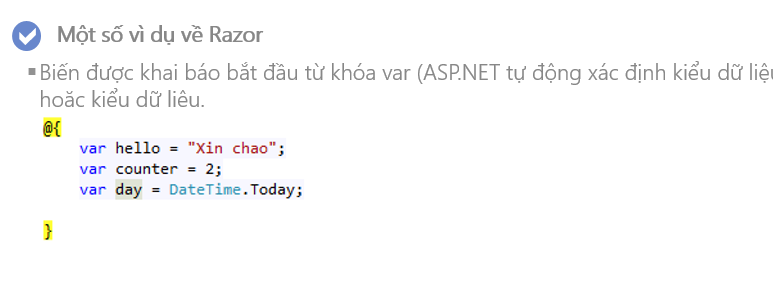
**▪ Các quy tắc Razor Syntax cho C#  
<h2>(1+2)=@(1+2)</h2>  
<h4>@Model.Title</h4>  
<span>@Html.Raw(@Model.Title)</span>  
@{ int x = 123;  
string y = "because."; }  
@if (showMessage) {  
<text>This is plain text</text> }**

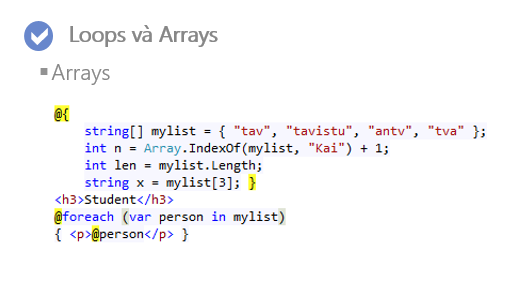
**🞦 Code Blocks:  
@foreach(dynamic a in ViewBag.ListSP ) {  
<tr><td>@a.MaSP</td>  
<td>@a.TenSP</td>  
<td>@a.GiaSP</td></tr> }**

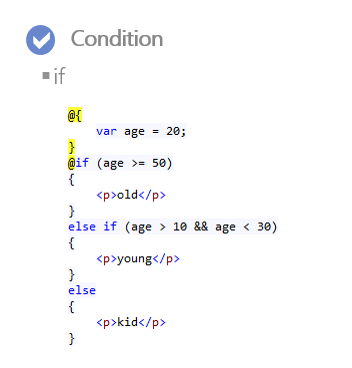
**▪ Razor - C# Variables:  
🞦 Biến dùng để lưu trữ dữ liệu  
🞦 Tên biến được đặt bắt đầu từ chữ cái  
🞦 Biến được khai báo bắt đầu từ khóa var (ASP.NET  
tự động xác định kiểu dữ liệu) hoặc kiểu dữ liệu.  
@{var greeting = "Welcome to W3Schools";  
var counter = 103;  
var today = DateTime.Today;  
// Using data types:  
string greeting = "Welcome to W3Schools";  
int counter = 103;  
DateTime today = DateTime.Today;}**

* Biến được khai báo bắt đầu từ khóa var (ASP.NET tự động xác định kiểu dữ liệu) hoặc kiểu dữ liệu.









Câu 13: Cách sử dụng HTML HELPERS. Viết ví dụ theo yêu cầu. ( slide 20 chương 7)

HTML Helper bao gồm các phương thức giúp tạo các thuộc tính HTML trên view.

HTML Helper là các thành phần sinh giao diện

web phù hợp buộc dữ liệu với model để duy trì

thông tin trên các thành phần đó.

•Đơn giản việc viết mã sinh giao diện

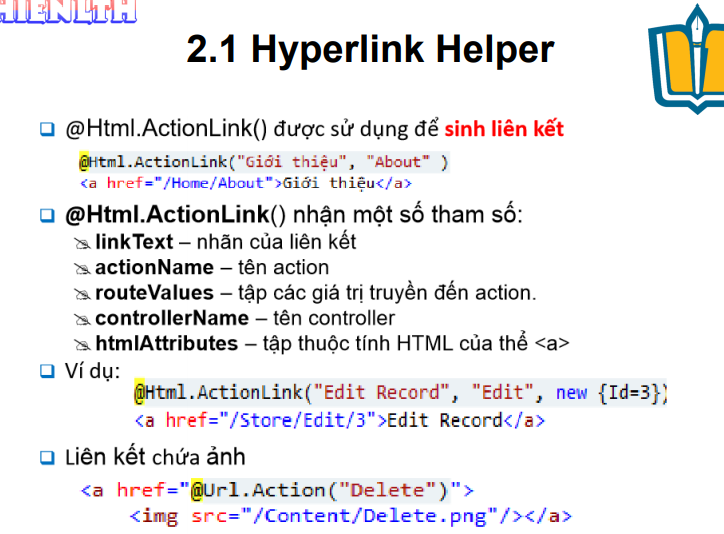
•Helper được chia làm 1 số nhóm

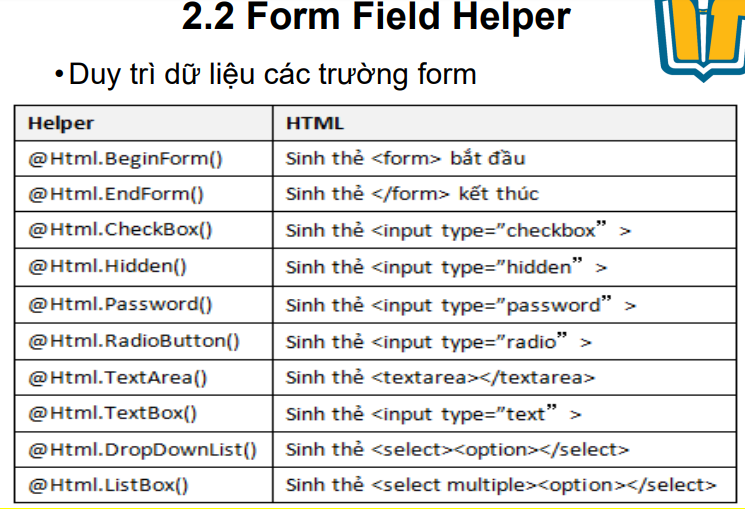
Liên kết

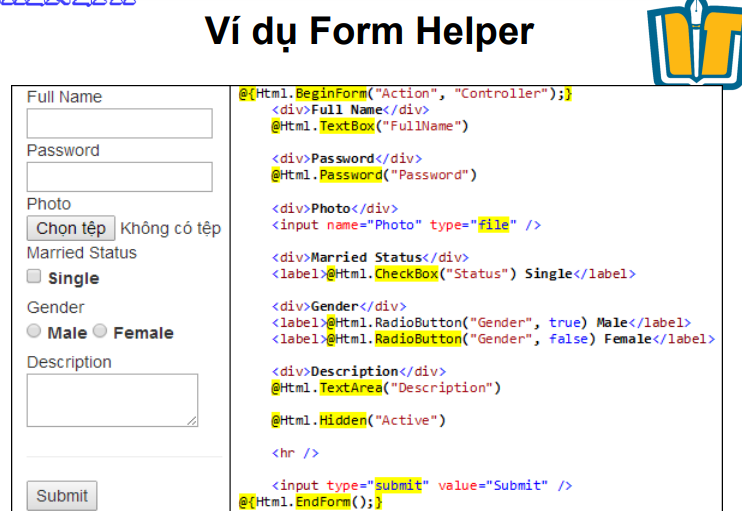
Form

Sinh giao diện từ model

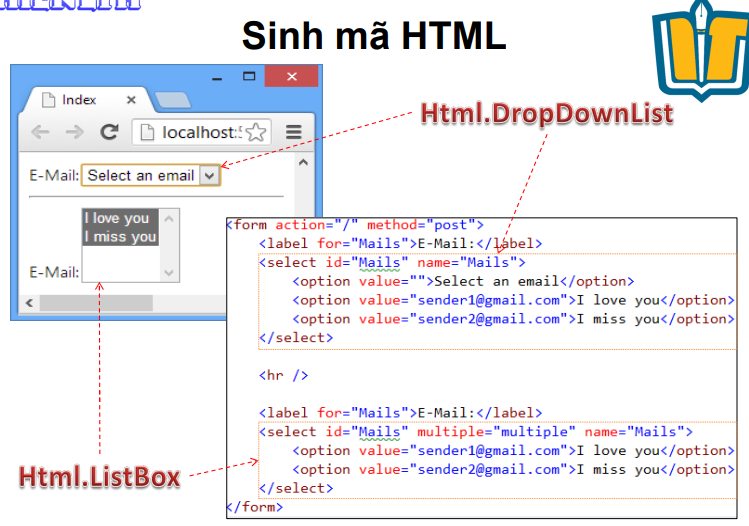
Kiểm lỗi

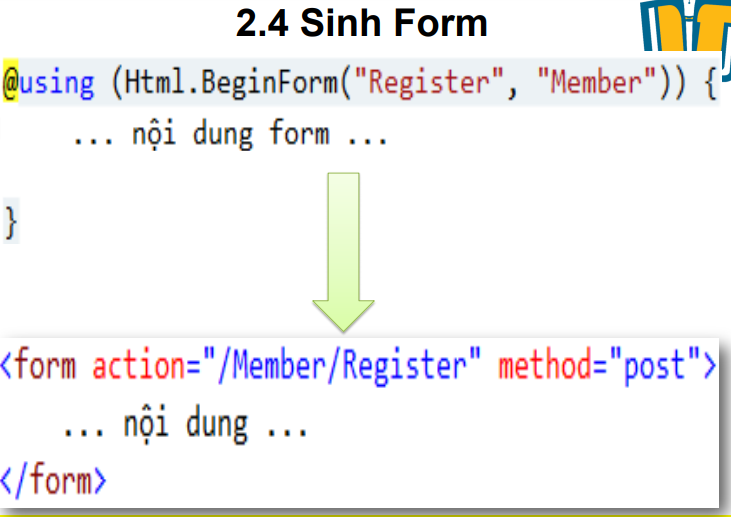


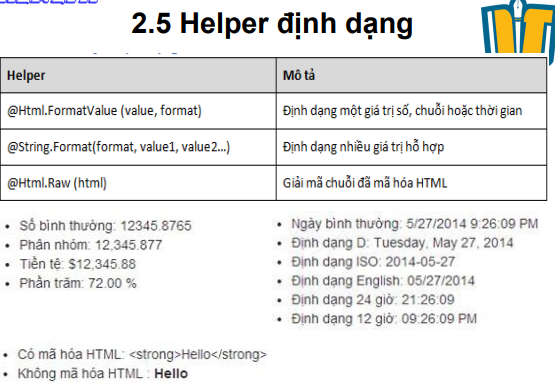


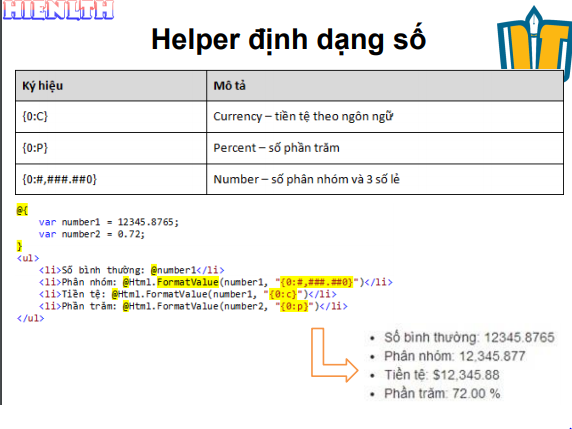






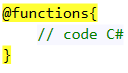




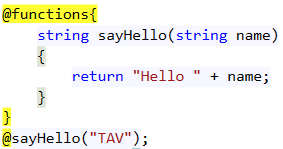


Câu 14: Cách sử dụng hàm trong HTML HELPERS. Viết ví dụ theo yêu cầu

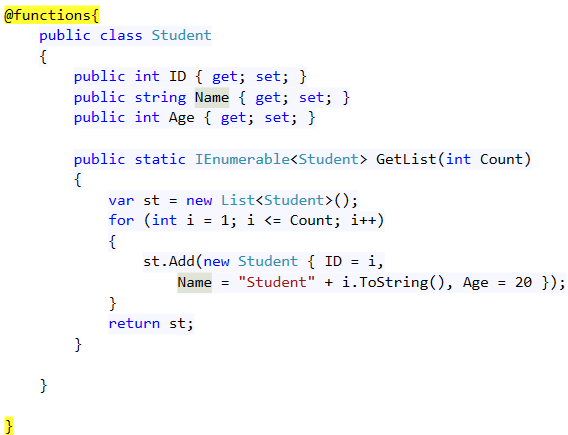
* Khai báo mã C# trong bổ từ khóa sau:

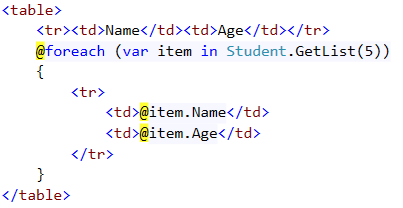


* Razor Code không hỗ trợ đầy đủ các kiểu dữ liệu hay thư viện .net, tuy nhiên ta có thể code mã C# để xây dựng **đối tượng**, **phương thức** trực tiếp trong đó.



* Xây dựng **đối tượng**, **phương thức** trực tiếp trong đó





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

Trả lời:

  - PartialView nó không phài là 1 View hoàn chỉnh, nó không thể đại diện cho 1 URL như kiểu Controller/ActionMethod có thể truy cập trực tiếp được. Mà nó được nhúng vào 1 View hoàn chỉnh. Nghĩa của PartialView nó đã là "Một phần của View"

Tác dụng của nó là để sử dụng lại được  trên nhiều view khác nhau

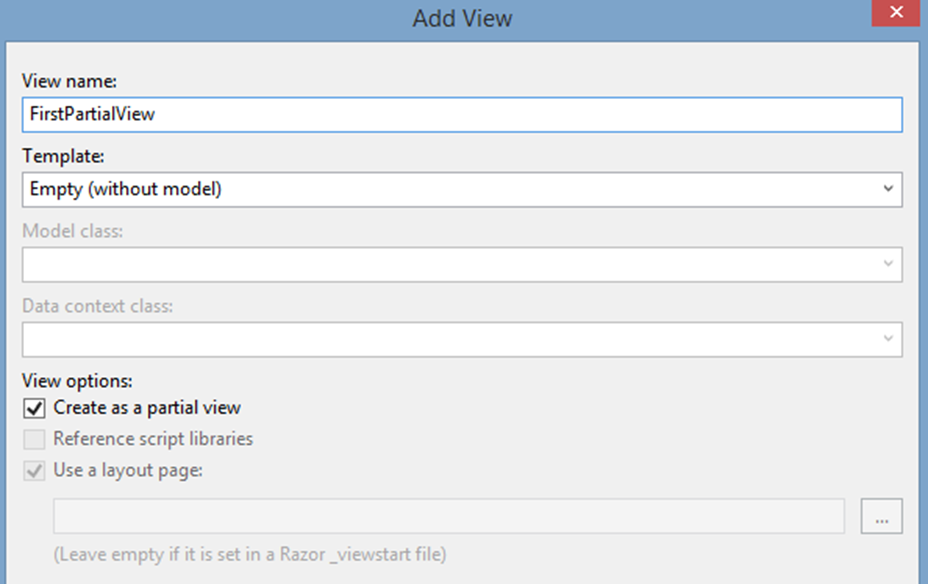
* Html.Partial, Html.RenderPartial dùng để gọi một PartialView.
* PartialView là một loại thẻ đặc biệt, chứa các thẻ html thuần hoặc các thẻ html chứa model hiển thị dữ liệu.
* Cấu trúc của HTML.Partial



Ví dụ :

**Tạo Partial**

* Trong Views/Login chọn
* Add/View



* Nhập nội dung cho FirstPartialView



* Trong file Login.cshtml

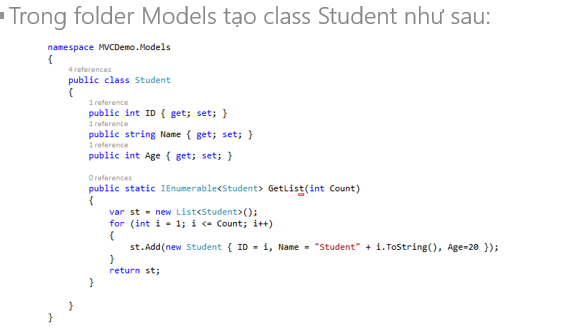


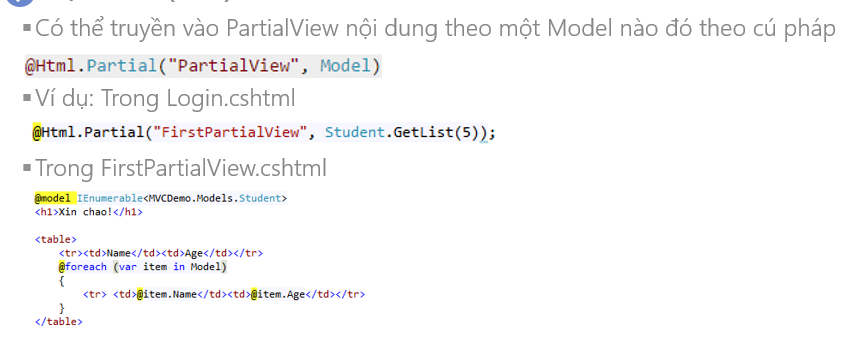


Ví dụ 2: Có thể truyền vào PảtialView nội dung theo một Model nào đó theo cú pháp:

@Html.Partial( “PartialView “ , Model )

Ví dụ :





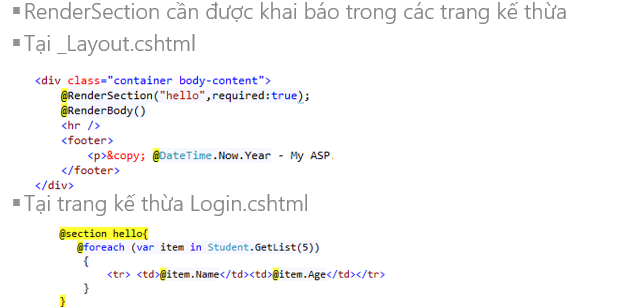
Câu 16 : Khái niệm Layout trong MVC. Cách sử dụng template vào chương trình.

* 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
* Tạo layout riêng:
* 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:

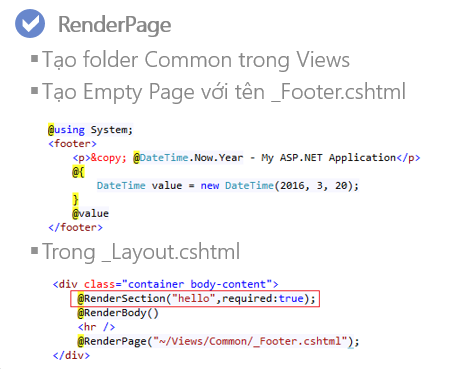


* Layout là một template hỗ trợ hiển thị cố định, các trang web khác kế thừa nội dung.
* Các đối tượng xử lý layout trong Razor: RenderBody, RenderPage, RenderSection, LayoutPage
* RenderBody: Nội dung trang kế thừa layout sẽ hiển thị đúng vào vị trí RenderBody
* RenderPage dùng để gọi nội dung từ một View khác
* RenderSection cần được khai báo trong các trang kế thừa

Ví dụ:



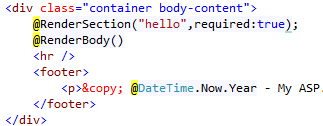
Ví dụ 2 :



Câu 17:

ví dụ: RenderBody: Nội dung trang kế thừa layout sẽ hiển thị đúng vào vị trí RenderBody()

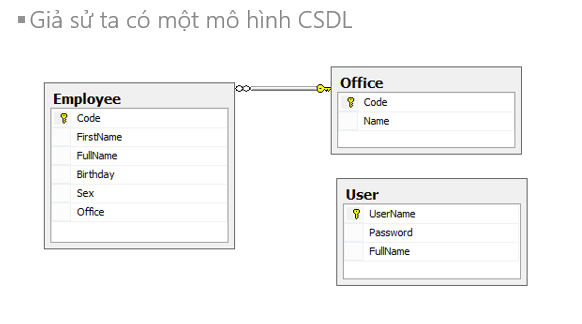
Tại \_Layout.cshtml

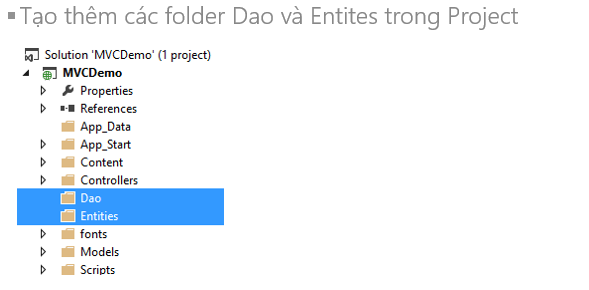


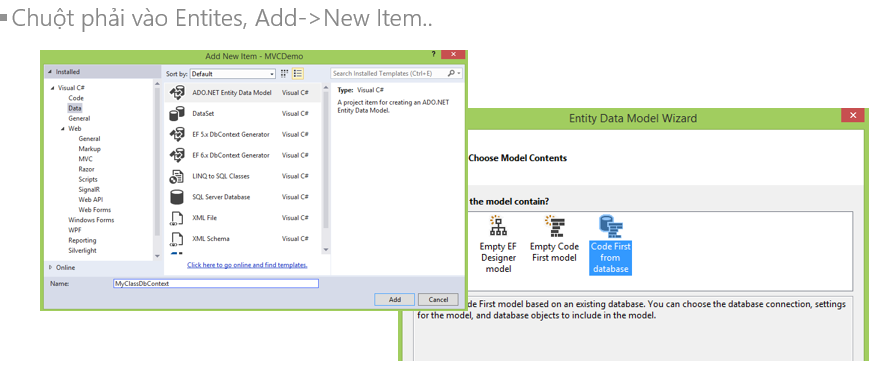
Một View muốn sử dụng layout thì khai báo như sau:

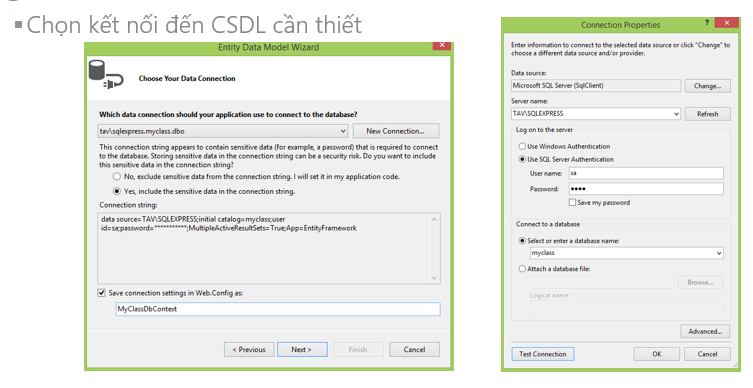


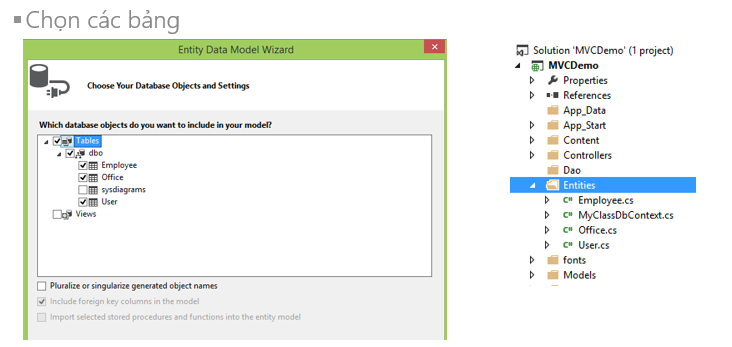
Câu 20 : Sử dụng EF, truy vấn một bảng dữ liệu.



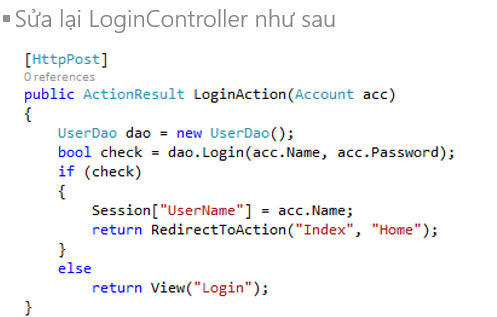


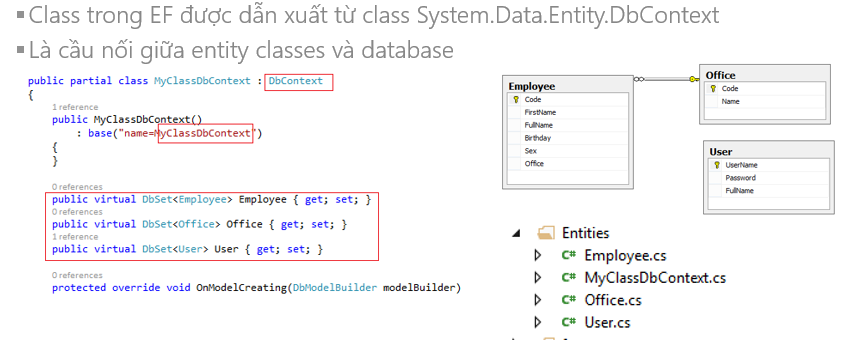


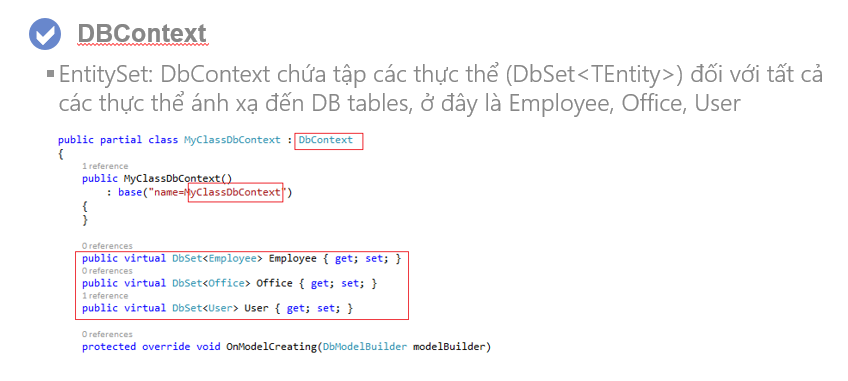




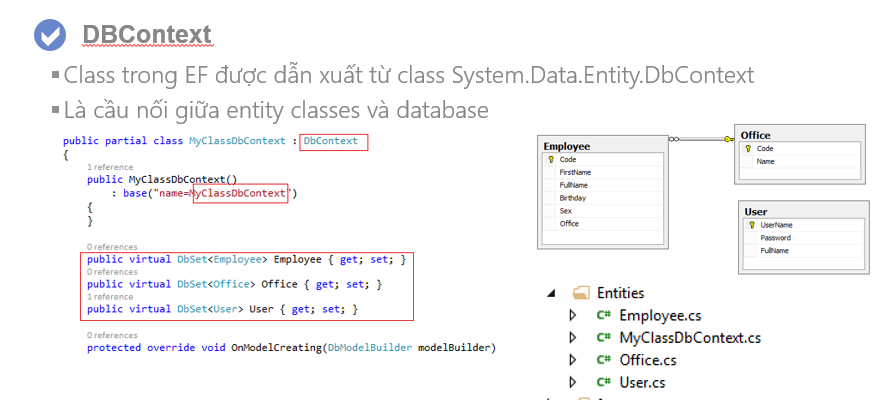


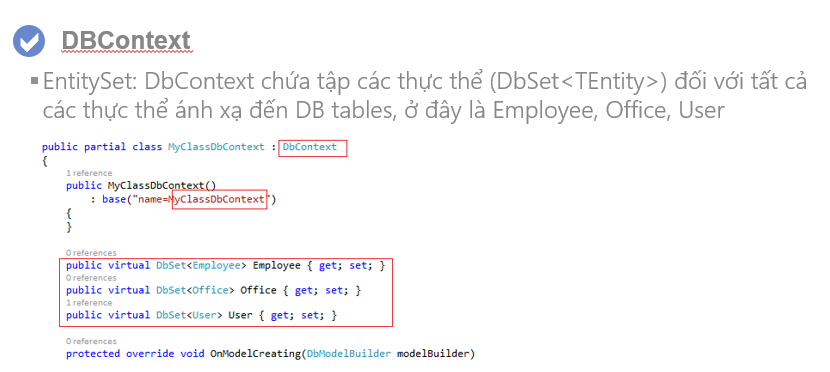


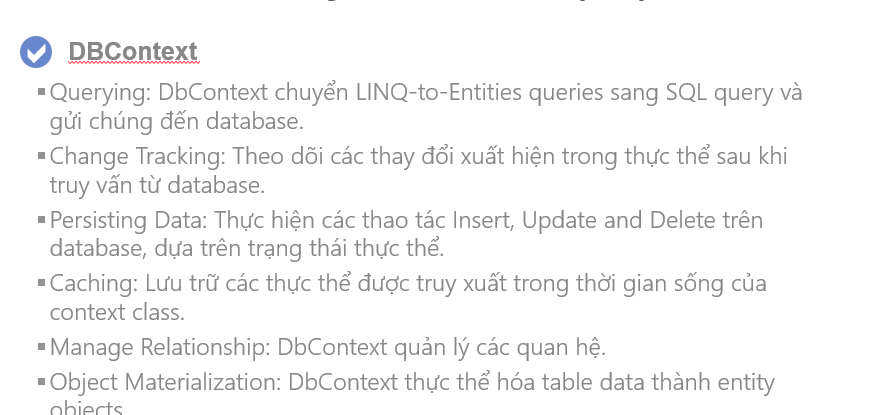




Câu 21: Vai trò của DCcontext







Câu 22: Thực hiện các tác vụ truy vấn với Linq.

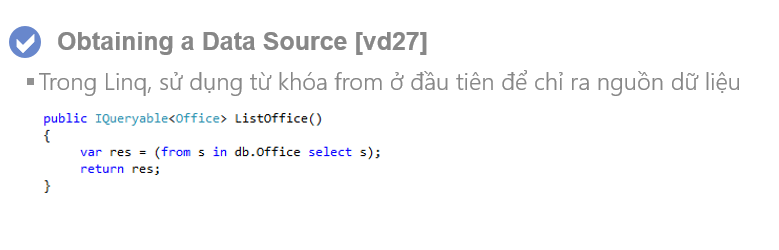
<https://www.dammio.com/2016/12/03/phan-1-linq-gioi-thieu-ve-linq?fbclid=IwAR2j7oLe_RiFz9VSPVVk2zg3rY5HzFilsCvozFVS0SXTCh8eofIT_s5rtrc>

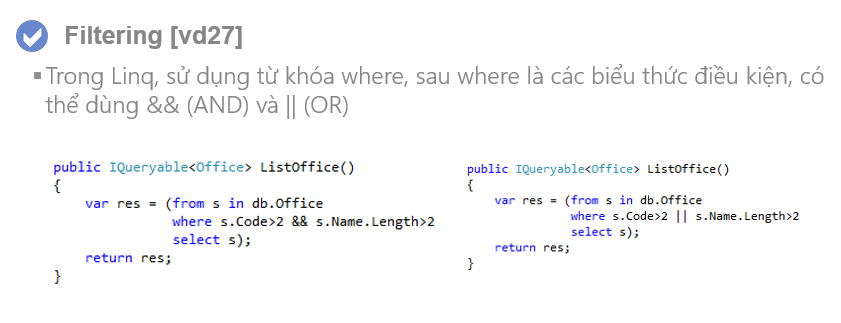
<https://www.dammio.com/2016/12/17/linq-phan-3-cac-truy-van-co-ban-trong-linq>

https://www.dammio.com/linq-co-ban

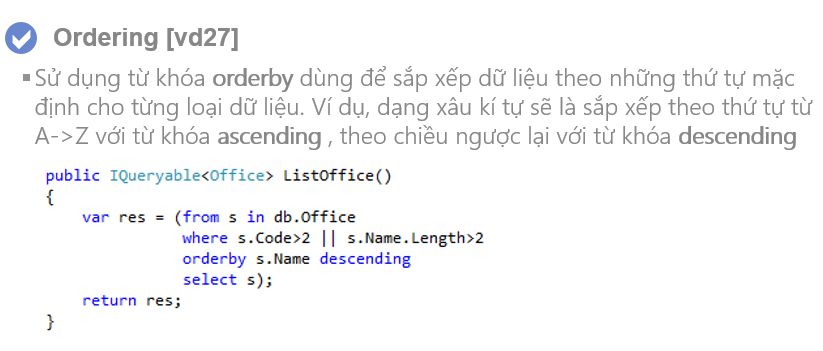
* LINQ to SQL là một phiên bản hiện thực hóa của O/RM (object relational mapping) có bên trong .NET Framework bản “Orcas” (nay là .NET 3.5), nó cho phép bạn mô hình hóa một cơ sở dữ liệu dùng các lớp .NET
* Các truy vấn cơ bản trong LINQ

1.Lấy một nguồn dữ liệu



2. Lọc dữ liệu (Filter) 

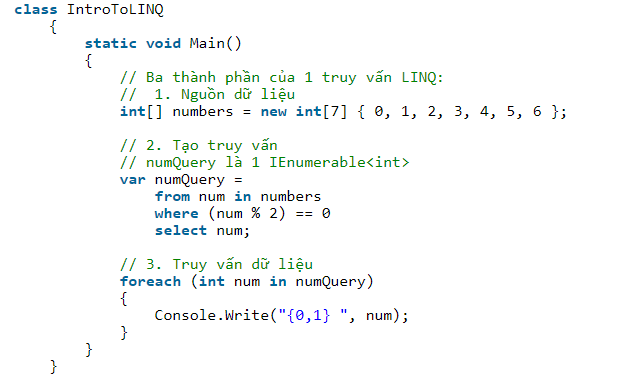
3. Sắp xếp (order)



**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.



Câu 23 :

Câu 24:

Câu 25: