**NỘI DUNG ÔN TẬP**

1. **Lý thuyết**
2. Nêu các cách truyền dữ liệu đến Controller? Lấy ví dụ minh họa?

\*Action Method Parameters

public ActionResult LoginResult(string name, string password)  
2. {  
3. if (name == “huongct" && password == “pass")  
4. ViewBag.Message = "Succeeded";  
5. else  
6. ViewBag.Message = "Failed";  
7. return View();  
8. }

\*Data Model Binding:

@model MvcApp.Models.Account  
2. @{ ViewBag.Title = "Login"; }  
3. <h2>Login</h2>  
4. @using (Html.BeginForm()){  
5. <p>Name: @Html.TextBoxFor(m => m.Name)</p>  
6. <p>Password: @Html.PasswordFor(m => m.Password)</p>  
7. <input type="submit" value="Login" />  
8. }

\*

Việc truyền dữ liệu từ View vào Controller có thể dùng bằng cách gửi qua URL dạng POST hoặc GET.  
Và nó không phải là hoàn toàn gửi qua bằng mỗi phương thức URL, nó còn có dạng Form, Cookies, Session...

Bạn có thể truyền dữ liệu từ controller lên view theo 3 cách :   
- Cách 1 : Bạn dùng ViewBag(MVC3),ViewData(MVC2) để truyền dữ liệu từ controller lên view, ViewBag là 1 Object Dynamic chính vì vậy bạn có thể gán cho ViewBag = 1 Object,1 String, hay 1 Integer đều được.   
Cú pháp và cách sử dụng bạn có thể tham khảo [Here](http://weblogs.asp.net/hajan/archive/2010/12/11/viewbag-dynamic-in-asp-net-mvc-3-rc-2.aspx)  
- Cách 2 : Bạn có thể truyền dữ liệu từ controller lên view thông qua Các Action

- Cách 3 : bạn có thể truyền và lấy dữ liệu từ view thông qua Ajax

6/slide6

1. Controller cung cấp các kiểu output nào? Lấy ví dụ?

\*ViewResult: (View) Sinh ra chỉ định hoặc default view template

\*PartialViewResult: (PartialView) Sinh ra chỉ định hoặc default partial view template

\*RedirectToRouteResult: (RedirectToAction, RedirectToActionPermanent, RedirectToRoute, RedirectToRoutePermanent) Điều hướng tới 1 action method hay route chỉ định

\*RedirectResult: (Redirect, RedirectPermanent) Điều hướng đến URL chỉ định

\*ContentResult: (Content) Trả về kiểu nội dung người dung định nghĩa

\*JsonResult: (Json) Trả về đối serialized JSON

\*JavascripResult: (JavaScript) Trả về script có thể thực thi ở client

\*FileResult: (File) Trả về binary output viết vào response

\*EmptyResult: (none) k có gì

<http://gockinhnghiem.com/2012/01/25/output-du-lieu-trong-mvc/>

24/slide6

1. Trình bày định tuyến - Route? Ví dụ?

Giới thiệu: Định tuyến trong ASP.NET MVC  
framework phục vụ các mục đích sau:  
🞦 Nó so khớp với các requests đến với một một file  
trong file system và ánh xạ request đó tới  
controller action.  
🞦 Xây dựng cấu trúc URLs tương ứng controller  
actions.  
🞦 URL rewriting tập trung ánh xạ một URL sang một  
URL khác. Route tập trung ánh xạ một URL đến  
một nguồn tài nguyên.

Các routes được định nghĩa trong file  
RouteConfig.cs nằm trong thư mục App\_Start

Các routes được định nghĩa trong file  
RouteConfig.cs nằm trong thư mục App\_Start:

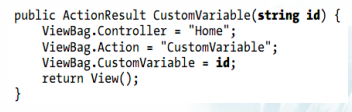
Sử dụng Static URL Segments: Giả sử muốn  
URL chứa Public dạng  
<http://mydomain.com/Public/Home/Index>

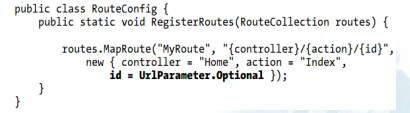
Routes.maproute(“”, “Public/{Controller}/{action}”, new{controller = “home”, action = “Index”})

Sử dụng Static URL Segments: Hoặc trộn lẫn  
<http://mydomain.com/XHome/Index>

Routes.maproute(“”, “X{Controller}/{action}”)

Truy cập Segment Variables:  
▪ Sử dụng Custom Variables như là tham số Action  
Method: Nếu định nghĩa tham số cùng tên khớp với URL  
pattern variables, MVC Framework sẽ truyền các giá trị được chứa  
trong URL như là các tham số cho action method



Định nghĩa Optional URL Segments:  
🞦 Optional URL segment là segment mà có nó hay  
không cũng được, ta không cần định nghĩa giá trị  
mặc định co segment này. Để chỉ định ta dùng chỉ  
định UrlParameter.Optional: 

43/slide6

1. Nêu các quy tắc trong view? Lấy dụ minh họa?

Trong controller chỉ cần gọi return View() để  
render the *view* mà không cần chỉ định tên  
view? Folder View được cấu tạo:  
🞦 Mỗi controller có một thư mục trùng tên trong thư  
mục view /Views/ControllerName (Không chứa hậu  
tố controller).  
🞦 Mỗi controller folder trên chứa một view file cho mỗi  
action method có tên trùng

Ta có thể “override” các quy tắc này bằng cách chỉ  
định tên view muốn được render.  
🞦 Chỉ định view khác nhưng trong cùng một thư mục:  
public ActionResult Index(){  
return View("NotIndex");  
}  
🞦 Chỉ định View khác khác cả thư mục:  
public ActionResult Index(){  
return  
View("~/Views/Example/Index.cshtml");  
}

Slide7

1. Trình bày ViewBag và ViewData? Ví dụ?

Giống nhau giữa ViewBag & ViewData:  
▪ Giúp duy trì data khi chuyển từ controller đến  
view.  
▪ Sử dụng truyền data từ controller đến view  
tương ứng.  
▪ “Short life” nghĩa là giá trị thành null khi  
redirection xuất hiện.

🞦 Khác nhau ViewBag & ViewData:  
▪ ViewData là dictionary của các đối tượng được dẫn  
xuất từ class ViewDataDictionary và được truy cập  
bằng cách sử dụng strings như là keys.  
▪ ViewBag là một dynamic property và là ưu điểm của  
các đặc tính động mới trong C# 4.0.  
▪ ViewData yêu cầu chuyển đổi kiểu dữ liệu  
(typecasting) và kiểm tra null values để tránh lỗi.  
▪ ViewBag không yêu cầu typecasting cho complex data  
type

14/slide7

1. Nêu cách khai báo biến, khối code trong Razor C#? Lấy ví dụ minh họa?

▪ 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;}

27/slide7

1. Nêu vai trò của Inline Html Helpers? Ví dụ?

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

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#" })

Slide7

1. Nêu các hàm Html Helpers chuẩn? Ví dụ?

* **Custom Helper Methods:**

**+** Inline Html Helpers

* **Built-In Html Helpers:**

**+** Standard Html Helpers

**+** Strongly Typed HTML Helpers

**+** Templated Helpers:

* **Rendering Helpers:**

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

**+** URL Helpers:

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

**+** 🞦 Html.Action and Html.RenderAction:

Slide7

1. Nêu các hàm Strongly Typed HTML Helpers? Ví dụ?

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,...

14,15/slide7

1. Trình bày Html.ActionLink và Html.RouteLink? Ví dụ?

▪ ActionLink method: Sinh ra hyperlink đến một  
controller action khác. Sau đây là một số kiểu link.  
√ Liên kết đến Action cùng controller:  
@Html.ActionLink("Link Text", "AnotherAction")  
√ Liên kết đến controller khác:  
@Html.ActionLink("Link Text", "Action“,  
"AnotherController")  
√ Liên kết và truyền dữ liệu đến Action:  
@Html.ActionLink("Edit link text", "Edit", new {Id=3})  
@Html.ActionLink("Edit link text","Edit","StoreManager",new {id=10720})

▪ RouteLink method: có cùng pattern như ActionLink  
helper, cũng có khác khái niệm route name nhưng  
không có các tham số cho controller name và  
action name.  
@Html.RouteLink("Link Text", new  
{action="AnotherAction"})

Slide7

1. Trình bày ý nghĩa, vai trò của Partial View? Ví dụ?

🞦 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.

Slide7

1. Trình bày Html.Partial và Html.RenderPartial? Ví dụ?

▪Partial helper sinh partial view thành string, các  
dạng:  
public void Partial(string partialViewName);  
public void Partial(string partialViewName, object model);  
public void Partial(string partialViewName, ViewDataDictionary  
viewData);  
public void Partial(string partialViewName, object  
model,ViewDataDictionary viewData);

▪RenderPartial helper tương tự như Partial, nhưng  
RenderPartial viết trực tiếp response output stream thay  
ch việc trả về string. Đây cũng là lý do ta phải đặt  
RenderPartial trong khối code.  
@{Html.RenderPartial("AlbumDisplay "); }  
@Html.Partial("AlbumDisplay ")

Slide7

1. Trình bày ý nghĩa, vai trò và các sử dụng Sections? Ví dụ?

Sử dụng tạo sections mà nội dung sẽ đc chèn vào layout tại các vị trí chỉ định

🞦Tạo các Section trong Layout:  
<body>  
@RenderSection("Header")  
<div class="layout">  
This is part of the layout  
</div>  
@RenderBody()  
<div class="layout">  
This is part of the layout  
</div>  
@RenderSection("Footer")  
</body>

🞦Sử dụng Sections trong View:  
@section Header {  
<div class="view">  
@foreach (string str in new [] {"Home", "List", "Edit"}){  
@Html.ActionLink(str, str, null, new { style = "margin: 5px" })  
}  
</div>  
} <  
div class="view">  
This is a list of fruit names:  
@foreach (string name in Model) {  
<span><b>@name</b></span>  
}  
</div>  
@section Footer {  
<div class="view">  
This is the footer  
</div>  
}

🞦RenderSection (“Name”) helper method chỉ  
định vị trí sẽ được chèn vào trên view.  
🞦@section Name: Sử dụng trên mỗi view để  
định Section chỉ định  
🞦Phần của view không chứa trong mỗi section  
được chèn vào trong tại vị trí RenderBody  
helper.

🞦Kiểm tra Section:  
@if (IsSectionDefined("Footer")) {  
@RenderSection("Footer")  
} else {  
<h4>This is the default footer</h4>  
}  
🞦Tạo Section dạng lựa chọn:  
@RenderSection("scripts", required:false)

Slide7

1. Trình bày ý nghĩa, vai trò và các sử dụng Child actions? Ví dụ?

🞦Là các action methods được gọi trong view.  
Nó giúp tránh lặp lại controller logic mà muốn  
lặp lại nhiều chỗ trên ứng dụng.  
🞦Child actions là các actions như partial views  
để view (return PartialView();).  
🞦Tạo Child Action:  
[ChildActionOnly]  
public ActionResult Time(){  
return PartialView(DateTime.Now);  
}

🞦Sử dụng Child Action:  
▪Cùng controller  
@{Html.RenderAction("HeaderCart");}  
@Html.Action("HeaderCart")  
▪Khác controller”  
@{Html.RenderAction("HeaderCart", "Home");}  
@Html.Action("HeaderCart", "Home")

Slide7

1. Trình bày Entity framework? Có các cách tiếp cận nào và trình bày cách tiếp cận đó?

▪Entity framework là Object/Relational  
Mapping (O/RM) framework:  
🞦Là sự phát triển nâng cao của ADO.NET  
cho tự động truy cập và lưu trữ dữ liệu trong  
database.  
🞦Làm việc với dữ liệu quan hệ như “domainspecific objects”  
🞦Có thể truy vấn sử dụng LINQ, sau đó truy  
xuất và thao tác dữ liệu như kiểu đối tượng

▪Entity framework cung cấp 3 cách tiếp  
cận:  
🞦Code First  
🞦Model First  
🞦Database first  
▪Code First  
🞦Viết các classes trước  
🞦Sau tạo database từ classes đó.

▪Model First  
🞦Tạo Entities, relationships, và các kế thừa  
trực tiếp từ cửa sổ thiết kế  
🞦Sau sinh database từ model này

▪Database first  
🞦Tạo EDM, context và entity classes từ  
database đã tồn tại

5/slide9

1. Trình bày cách thêm, sửa, xóa, truy vấn dữ liệu sử dụng mô hình Entity Framework?

5/slide9

1. Nêu cách sử dụng phương thức SqlQuery và ExecuteSqlCommand? Lấy ví dụ minh họa?

▪Sử dụng phương thức SqlQuery và  
ExecuteSqlCommand : Để thực thi các lệnh Sql.  
🞇<DataContext>.DataBase.SqlQuery  
🞇<DataContext>.<EntityName>.SqlQuery  
🞇<DataContext>.DataBase. ExecuteSqlCommand  
Chú ý: SqlQuery cho phép trả về các thực thể từ  
database (SELECT) còn ExecuteSqlCommand chỉ thực  
hiện câu lệnh và trả về trạng thái của code từ Database  
(INSERT, UPDATE, DELETE)

▪Sử dụng phương thức SqlQuery và  
ExecuteSqlCommand  
public List<SACH> GetDanhSach(string MaTL){  
SqlParameter[] idParam={  
new SqlParameter {ParameterName = "MaTL", Value = MaTL },  
new SqlParameter{ParameterName = "Ten", Value = MaTL }};  
var courseList = context.SACHes.SqlQuery("GetSach @MaTL,  
@Ten", idParam).ToList<SACH>();  
return courseList;  
}

public List<SACH> GetDanhSach(string MaTL){  
SqlParameter[] idParam={  
new SqlParameter {ParameterName = "MaTL", Value = MaTL },  
new SqlParameter{ParameterName = "Ten", Value = MaTL }};  
**var courseList = context.SACHes.SqlQuery("GetSach @MaTL,  
@Ten", idParam).ToList<SACH>();**return courseList;  
}

35/slide9

1. Trình bày ý nghĩa, vai trò của các classs DBContext, DBSet?

12/slide9

1. Nêu các cách kiểm tra tính hợp lệ của dữ liệu trong Model? Ví dụ?

40/slide9

1. Trình bày các ký hiệu Validation Annotations?

🞦 [Required]: Giá trị không được rỗng hoặc là  
các chuỗi chỉ có các spaces. Nếu muốn  
whitespace là hợp lệ, sử dụng  
[Required(AllowEmptyStrings = true)].  
[Required]  
public string UserName { get; set; }

🞦 [StringLength(n)]: Chỉ định độ dài lớn nhất của dữ  
liệu. Ta có thể chỉ định độ dài tối thiểu sử dụng  
thuộc tính MinimumLength  
[StringLength(30)]  
public string UserName { get; set; }  
[StringLength(100, ErrorMessage = "The {0} must be at  
least {2} characters long.", MinimumLength = 6)]  
public string Password { get; set; }

🞦 [RegularExpression("pattern")]: Chỉ định dữ liệu  
phải đúng khuôn định dạng.  
[RegularExpression(@"[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-  
]+\.[A-Za-z]{2,4}")]  
public string Email { get; set; }  
🞦 [Range(n,m)] : Chỉ định dữ liệu kiểu số phải nằm  
trong khoảng. Ta có thể chỉ định biên một phía  
bằng cách sử dụng hằng số MinValue, MaxValue,  
ví dụ [Range(int.MinValue, 50)]  
[Range(0,10)]  
public int Marked { get; set; }

🞦 [Compare("MyOtherProperty")]: Hai thuộc tính bắt  
buộc phải có cùng giá trị.  
[Required]  
public string Password { get; set; }  
[Display(Name = "Confirm password")]  
[Compare("Password", ErrorMessage = "The password and  
confirmation password do not match.")]  
public string ConfirmPassword { get; set; }

🞦 Chú ý:  
▪ Sử dụng ErrorMessage property để chỉ định „custom  
error message‟.  
▪ DataType attribute cung cấp thông tin về kiểu dữ liệu tại  
thời điểm runtime.  
▪ Display attribute: Hiển thị tên cho thuộc tính model.  
[DataType(DataType.Password)]  
[Display(Name = "Confirm password")]  
[Compare("Password", ErrorMessage = "The password and  
confirmation password do not match.")]  
public string ConfirmPassword { get; set; }

43/slide9

1. **Bài tập lớn môn học: Hoàn thiện bài tập lớn môn học, đảm bảo các chức năng cơ bản của ứng dụng và mỗi sinh viên đảm bảo các kỹ năng sau:**
2. Xây dựng được trang hiển thị danh sách sản phẩm hoặc bài tin
3. Từ trang hiển thị danh sách sản phẩm/bài tin cho phép chọn hiển thị chi tiết sản phẩm/ bài tin (45’)
4. Xây dựng được các trang thực hiện thao tác quản trị (hiển thị, thêm, sửa, xóa) một bảng trong Cơ sở dữ liệu (Chú ý dùng được với bảng có upload file ảnh)
5. Xây dựng được trang login và thực hiện phân quyền
6. Nắm bắt kỹ thuật giỏ hàng và hoàn thiện giỏ hàng.
7. Nắm bắt kỹ thuật thanh toán và thực hiện thanh toán.
8. Xây dựng dịch vụ web và sử dụng dịch vụ web.