Bài số 4

Định tuyến URL và điều phối hiển thị

Table of Contents

1 URL routing (Định tuyến URL)

1.1 Giới thiệu định tuyến URL

1.1.1 Hệ thống định tuyến trong ASP.NET MVC làm gì?

ASP.NET MVC Framework có một hệ thống định tuyến URL (URL Routing System) linh hoạt cho phép xác định các quy tắc ánh xạ địa chỉ URL bên trong ứng dụng. Một hệ thống định tuyến có 2 mục đích:

- Xây dựng một tập hợp các URL đi vào ứng dụng và định tuyến chúng tới các Controller và thực thi các phương thức Action để xử lý.
- Xây dựng các URL gửi đi mà có thể gọi ngược trở lại Controllers/Actions (ví dụ: form posts, liên kết <a href=“”> và các lời gọi AJAX)

Sử dụng các quy tắc ánh xạ URL để điều khiển URL đi vào và đi ra để tăng tính mềm dẻo cho việc lập trình ứng dụng, nghĩa là nếu muốn thay đổi cấu trúc URL (ví dụ /Catalog thành /Products) có thể thay đổi một tập hợp quy tắc ánh xạ mức ứng dụng mà không cần phải viết lại mã lập trình bên trong Controllers và Views.

1.1.2 Các quy tắc định tuyến URL mặc định trong ASP.NET MVC Web Application

Mặc định khi tạo ứng dụng với ASP.NET MVC Web Application trong Visual Studio sẽ tạo ra một ASP.NET Application class gọi là Global.asax chứa cấu hình các quy tắc định tuyến URL. Xây dựng các định tuyến thông qua phương thức RegisterRoutes(ReouteCollection routes) và khi ứng dụng bắt đầu, phương thức Application_Start() trong Global.asax.cs sẽ gọi RegisterRoutes để tạo ra bảng định tuyến.

Global.asax.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace BanHang
{
    // Note: For instructions on enabling IIS6 or IIS7 classic mode,
    // visit http://go.microsoft.com/?LinkId=9394801

    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);
        }
    }
}
```

Mặc định định tuyến URL trong ASP.NET MVC Framework có cấu trúc dạng: Controllers/ControllerAction/Id

Với ASP.NET MVC Web Application thì mặc định Controllers là HomeController, mặc định ControllerAction là Index, mặc định Id là rỗng. Nghĩa là khi gọi trang web được xây dựng thông qua template ASP.NET Web Application thì mặc định http://localhost/ tương đương với http://localhost/Home/Index/

Khi ứng dụng ASP.NET MVC Web Application nhận được một Url, MVC Framework sẽ định giá các quy tắc định tuyến trong tập hợp RouteTable.Routes để quyết định Controller nào sẽ điều khiển request.

MVC framework chọn Controller bằng cách định giá các quy tắc trong bảng định tuyến theo trật tự đã có sẵn.

1.2 Ví dụ định tuyến URL

Sử dụng ứng dụng BanHang dựa trên Framework ASP.NET MVC Web Application:

Tạo TimKiem URL

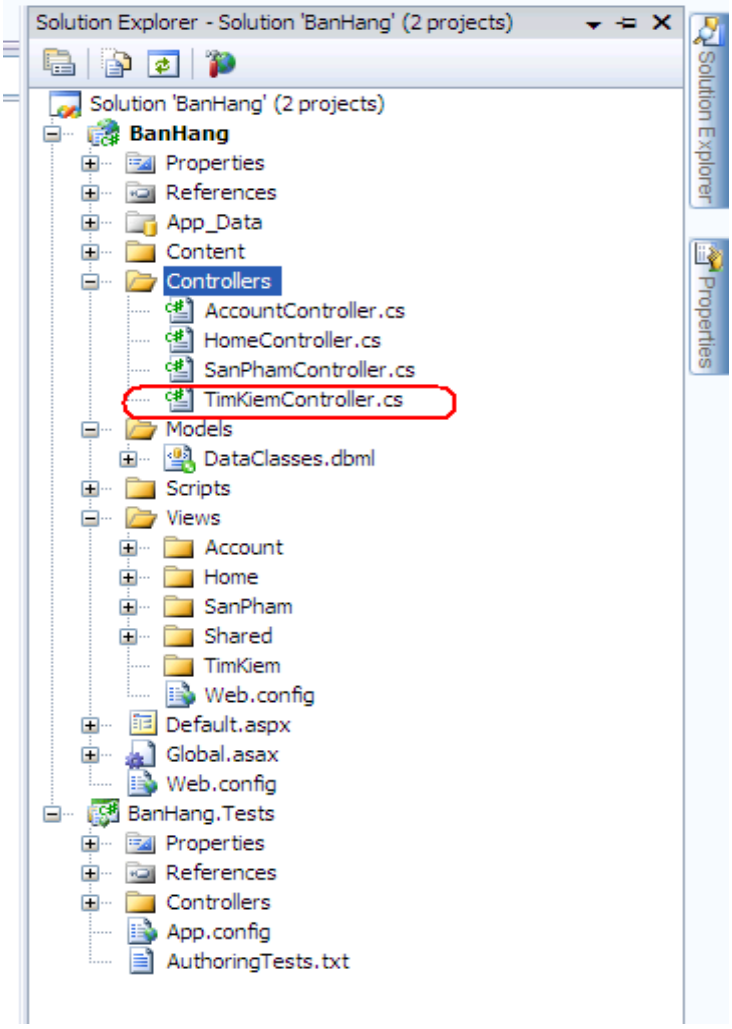

Figure 1. Tạo controller TimKiemController.cs

Có 2 action trong TimKiemController.cs: action Index() để hiển thị một trang search với một TextBox cho người dùng nhập từ khóa cần tìm, action Results để điều khiển khi yêu cầu tìm kiếm được xác định.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Mvc.Ajax;

namespace BanHang.Controllers
{
    public class TimKiemController : Controller
    {
        public ActionResult Index()
        {
            // Add action logic here
            return View();
        }

        public ActionResult Results(string query)
        {
            return View();
        }
    }
}
```

Trong Global.asax.cs một cách thức định tuyến mặc định. Theo quy tắc định tuyến mặc định thì khi yêu cầu một trang tìm kiếm, địa chỉ Url được gọi theo sẽ tương ứng với [controller]/[action]/[id] là /TimKiem/Results/[string query]. Cách dùng này không có vấn đề gì nhưng ta tìm hiểu một cách tùy biến định tuyến url để thay đổi thành /TimKiem/[string query]. Thêm vào trong Global.asax.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace BanHang
{
    // Note: For instructions on enabling IIS6 or IIS7 classic mode,
    // visit http://go.microsoft.com/?LinkId=9394801

    public class MvcApplication : System.Web.HttpApplication
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                "TimKiem",                                              // Route
name
                "TimKiem/{query}",                                // URL with parameters
                new { controller = "TimKiem", action = "Results" }  // Parameter
defaults
            );

            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);
        }
    }
}
```

Tạo 2 view hiển thị dữ liệu được điều khiển trong TimKiemController.cs là Index.aspx và Results.aspx

Index.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
AutoEventWireup="true" CodeBehind="Index.aspx.cs"
Inherits="BanHang.Views.TimKiem.Index" %>
<asp:Content ID="viewTimKiem" ContentPlaceHolderID="MainContent" runat="server">
    <form method="post" action="TimKiem/Search">
        <input type="text" id="txtTimKiem" name="txtTimKiem" />
        <input type="submit" value="Tìm Kiếm" />
    </form>
</asp:Content>
```

Result.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
AutoEventWireup="true" CodeBehind="Results.aspx.cs"
Inherits="BanHang.Views.TimKiem.Results" %>
<asp:Content ID="viewResults" ContentPlaceHolderID="MainContent" runat="server">
    Kết quả dữ liệu tìm kiếm được ở đây
</asp:Content>
```

Thêm vào Views\Shared\Site.master một tab tìm kiếm

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site.master.cs"
Inherits="BanHang.Views.Shared.Site" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <title><%= Html.Encode(ViewData["Title"]) %></title>
    <link href="../../Content/Site.css" rel="stylesheet" type="text/css" />
</head>

<body>
    <div class="page">

        <div id="header">
            <div id="title">
                <h1>My Sample MVC Application</h1>
            </div>

            <div id="logindisplay">
                <% Html.RenderPartial("LoginUserControl"); %>
            </div>

            <div id="menucontainer">

                <ul id="menu">
                    <li><%= Html.ActionLink("Home", "Index", "Home")%></li>
```

```
                    <li><%= Html.ActionLink("Sản phẩm", "Index", "SanPham")%></li>
                    <li><%= Html.ActionLink("Tìm kiếm", "Index", "TimKiem")%></li>
                    <li><%= Html.ActionLink("About Us", "About", "Home")%></li>
                </ul>

            </div>
        </div>

        <div id="main">
            <asp:ContentPlaceHolder ID="MainContent" runat="server" />

            <div id="footer">
                My Sample MVC Application &copy; Copyright 2008
            </div>
        </div>
    </div>
</body>
</html>
```

Kết quả thực thi chương trình (Figure 2)

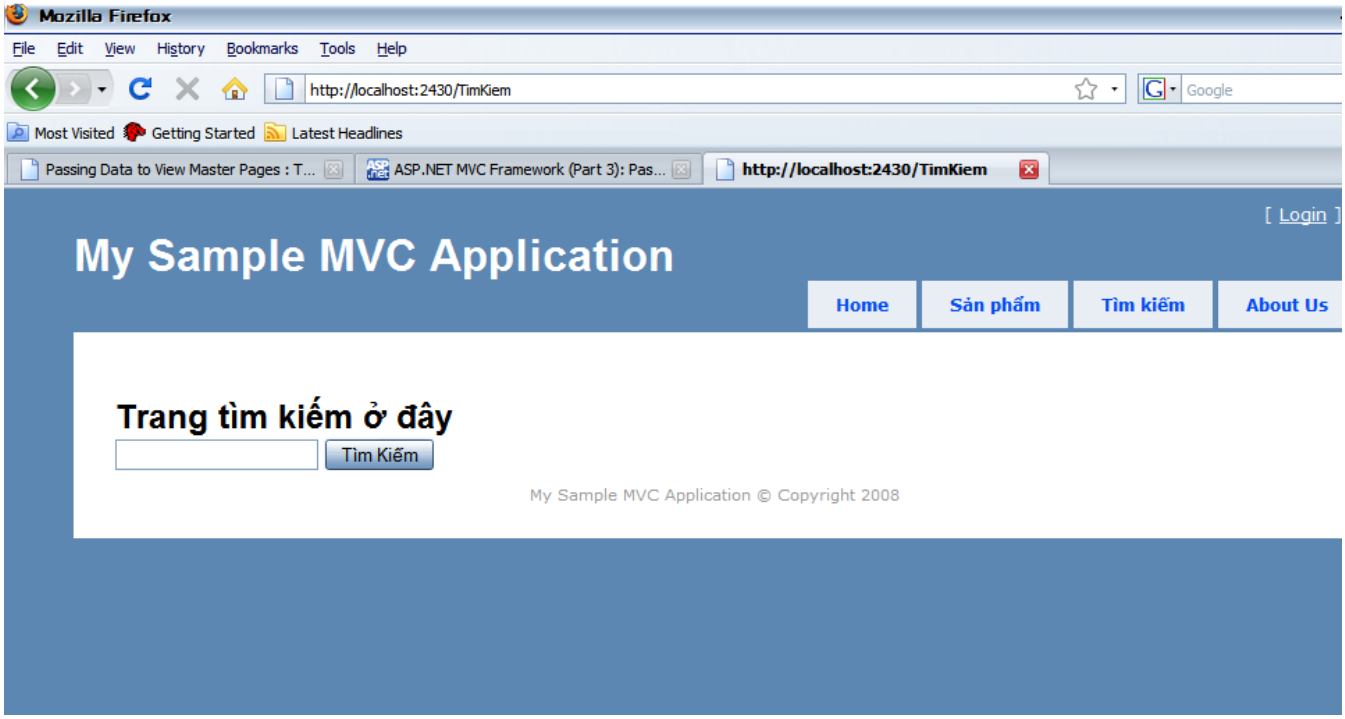

Figure 2 Thực hiện tìm kiếm với TimKiemController.cs

2 Điều phối hiển thị dữ liệu

ViewData là thành phần quan trọng trong việc hiển thị dữ liệu của ASP.NET MVC Framework. Mỗi Controller đều có một ViewData dictionary có thể dùng để đưa dữ liệu vào View . Để đưa dữ liệu vào ViewData dùng định dạng key/value (ví dụ ViewData[“Title’’] = “Sản phẩm”).

2.1 Điều phối hiển thị dữ liệu với ViewData Dictionary

SanPhamController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.Mvc;
using System.Web.Mvc.Ajax;
using BanHang.Models;

namespace BanHang.Controllers
{
    public class SanPhamController : Controller
    {
        DataClassesDataContext data = new DataClassesDataContext();

        public ActionResult Index()
        {
            // Add action logic here
            ViewData["Title"] = "Sản phẩm";

            return RedirectToAction("DanhMucLoaiSanPham");
        }

        public ActionResult DanhMucLoaiSanPham()
        {
            // Code cua ban o day
            ViewData["Title"] = "Danh mục loại sản phẩm";

            List<LoaiSanPham> lsp = data.LoaiSanPhams.ToList();

            return View("DanhMucLoaiSanPham", lsp);
        }

        public ActionResult DanhSachSanPham(string loaisanpham)
        {
            ViewData["Title"] = "Danh sách sản phẩm trong loại sản phẩm";

            List<SanPham> sp = data.LaySanPhamTuLoaiSanPham(loaisanpham);

            return View("DanhSachSanPham", sp);
        }


        public ActionResult ChiTietSanPham(int id)
        {
            ViewData["Title"] = "Chi tiết sản phẩm";

            SanPham ctsp = data.LaySanPhamTuID(id);

            return View("ChiTietSanPham", ctsp);
        }
    }
}
```

Có 2 cách tiếp cận việc hiển thị dữ liệu trong Views : cách 1 dùng trực tiếp <%= %> trong code, cách 2 sử dụng server controls để hiển thị dữ liệu

Thực thi view với cách dùng <%= %>

Views\SanPham\DanhSachSanPham.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
AutoEventWireup="true" CodeBehind="DanhSachSanPham.aspx.cs"
Inherits="BanHang.Views.SanPham.DanhSachSanPham" %>
<asp:Content ID="viewDanhSachSanPham" ContentPlaceHolderID="MainContent"
runat="server">
    <h1>Đây la`danh sách sản phẩm co´trong chuyên mục</h1>
    <ul>
    <% foreach (var sp in ViewData.Model)
       { %>
```

```
        <li>

            <%= Html.ActionLink(sp.TenSanPham , "ChiTietSanPham/" + sp.Id,
"SanPham") %>

        </li>

    <% } %>
    </ul>
</asp:Content>
```

Views\SanPham\DanhSachSanPham.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using BanHang.Models;

namespace BanHang.Views.SanPham
{
    public partial class DanhSachSanPham : ViewPage <List<BanHang.Models.SanPham>>
    {

    }
}
```

Thực thi view với server controls

Bind Data vào code -behide Views\SanPham\DanhSachSanPham.aspx.cs có thể dùng ViewPage hoặc từ điển ViewData, để hiển thị dùng một server control là <asp:DataList>. Vì không có <form runat="server"> như WebForm nên trong Views\SanPham\DanhSachSanPham.aspx không chứa một ViewState và cũng sẽ không phát sinh ID (kiểu như Ctrl00_listView), toàn thể trang chỉ chứa các định dạng HTML.

Views\SanPham\DanhSachSanPham.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
AutoEventWireup="true" CodeBehind="DanhSachSanPham.aspx.cs"
Inherits="BanHang.Views.SanPham.DanhSachSanPham" %>
<asp:Content ID="viewDanhSachSanPham" ContentPlaceHolderID="MainContent"
runat="server">
    <h1>Đây la`danh sách sản phẩm co´trong chuyên mục</h1>

    <asp:DataList ID="listView" runat="server">
        <ItemTemplate>

            <%= Html.ActionLink(Eval("TenSanPham") , "ChiTietSanPham/" +
Eval("Id"), "SanPham") %>

        </ItemTemplate>
    </asp:DataList>
</asp:Content>
```

Views\SanPham\DanhSachSanPham.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using BanHang.Models;

namespace BanHang.Views.SanPham
```

```
{
    public partial class DanhSachSanPham : ViewPage <List<BanHang.Models.SanPham>>
    {
        public void Page_Load()
        {
            listView.DataSource = ViewData["SanPham"];
            listView.DataBind();
        }
    }
}
```

2.2 Điều phối hiển thị dữ liệu với cách dùng Strongly Typed Classes

2.2.1 Lợi ích của việc dùng strongly typed

- Tránh được việc dùng chuỗi string để tra cứu các đối tượng và không phải mất thời gian kiểm tra biên dịch cho cả Controllers và Views
- Tránh được việc phải đưa dữ liệu vào ViewData dictionar y khi sử dụng ngôn ngữ strongly -typed dựa trên C#
- Nhận được code thông minh trong cả file .aspx và .aspx.cs như dùng với một class thực thụ của C#.
- Dùng lại được code trong khi tạo các unit test trong ứng dụng.

2.2.2 Tạo strongly-typed DuLieuDanhSachSanPham trong folder Models

DuLieuDanhSachSanPham.cs

```
using System;
using System.Data;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

namespace BanHang.Models
{
    public class DuLieuDanhSachSanPham
    {
```

```
        public string TenLoaiSanPham { get; set; }
        public List<Models.SanPham> SanPham { get; set; }
    }
}
```

Sửa đổi lại SanPhamController.cs cụ thể là thay đổi phương thức DanhSachSanPham

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Mvc.Ajax;
using BanHang.Models;

namespace BanHang.Controllers
{
    public class SanPhamController : Controller
    {
        DataClassesDataContext data = new DataClassesDataContext();

        public ActionResult Index()
        {
            // Add action logic here
            ViewData["Title"] = "Sản phẩm";

            return RedirectToAction("DanhMucLoaiSanPham");
        }

        public ActionResult DanhMucLoaiSanPham()
        {
            // Code cua ban o day
            ViewData["Title"] = "Danh mục loại sản phẩm";

            List<LoaiSanPham> lsp = data.LoaiSanPhams.ToList();

            return View("DanhMucLoaiSanPham", lsp);
        }

        public ActionResult DanhSachSanPham(string loaisanpham)
        {
            ViewData["Title"] = "Danh sách sản phẩm trong loại sản phẩm";

            //List<SanPham> sp = data.LaySanPhamTuLoaiSanPham(loaisanpham);
            //return View("DanhSachSanPham", sp);
            DuLieuDanhSachSanPham sp = new DuLieuDanhSachSanPham();
            ViewData.TenLoaiSanPham = loaisanpham;
            ViewData.SanPham = data.LaySanPhamTuLoaiSanPham(loaisanpham);

            return View("DanhSachSanPham", ViewData);
        }

        public ActionResult ChiTietSanPham(int id)
        {
            ViewData["Title"] = "Chi tiết sản phẩm";

            SanPham ctsp = data.LaySanPhamTuID(id);

            return View("ChiTietSanPham", ctsp);
        }
    }
}
```

2.2.3 Dùng ViewData dictionary với một đối tượng ViewData strongly typed

Trong Views\SanPham\DanhSachSanPham.aspx không cần thay đổi khi có sự thay đổi trong SanPhamController.cs vì lý do là đối tượng ViewData strongly typed đã được đưa tới từ ViewPage , đối tượng ViewData dictionary sẽ tự động sử dụng tìm đến các thuộc tính nằm trong ViewData strongly typed.

Dùng lớp cơ sở ViewPage<T> mang dữ liệu tới ViewData strongly typed

ASP.NET MVC Framework hỗ trợ một dictionary dựa trên lớp cơ sở ViewPage . ViewPage<T> (T là một kiểu của lớp ViewData mà Controllers chuyển tới Views) được thay đổi

Views\SanPham\DanhSachSanPham.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using BanHang.Models;

namespace BanHang.Views.SanPham
{
    public partial class DanhSachSanPham : ViewPage <DuLieuDanhSachSanPham>
    {
        public void Page_Load()
        {

        }
    }
}
```

3 Câu hỏi ôn tập

Hỏi: Định tuyến Url khi sử dụng Controllers khai báo phương thức ActionRes ults có chứa tham số , thực thi chương trình báo lỗi thì phải làm sao?

Đáp: Khi tạo phương thức trong Controllers có sử dụng tham số, có 2 trường hợp xảy ra: phương thức chứa tham số có thể nhận giá trị null hoặc phương thức c hứa tham số không cho phép nhập giá trị null . Với tham số không cho phép nhập giá trị null thì nếu gọi Url không chứa tham số thì sẽ báo lỗi. Có thể giải quyết bằng cách cho phép nhận giá trị null vào tham số hoặc bắt buộc khi gọi Url phải đưa tham số đầu vào.

Hỏi: Trong một ứng dụng MVC, liệu có tồn tại 1 mô hình dữ liệu Models, có n Controllers, mỗi Controllers điều khiển m Views không?

Đáp: Có. Thực chất việc tạo ra mô hình MVC chính là một cách thể hiện lập trình đa mục đích. Với một mô hình dữ liệu đã có, việc xây dựng các Controllers khác nhau cùng truy cập đến mô hình dữ liệu đã có là hoàn toàn thực hiện được. Mỗi Controllers sẽ có nhiều hơn m phương thức tương ứng với m Views.

4 Tài liệu tham khảo

http://asp.net/mvc

http://weblogs.asp.net/scottgu/archive/2007/12/03/asp-net-mvc-framework-part-2-url-routing.aspx

http://weblogs.asp.net/scottgu/archive/2007/12/06/asp-net-mvc-framework-part-3-passing-viewdata-from-controllers-to-views.aspx