CÔNG NGHỆ WEB

CONTROLLER

NỘI DUNG

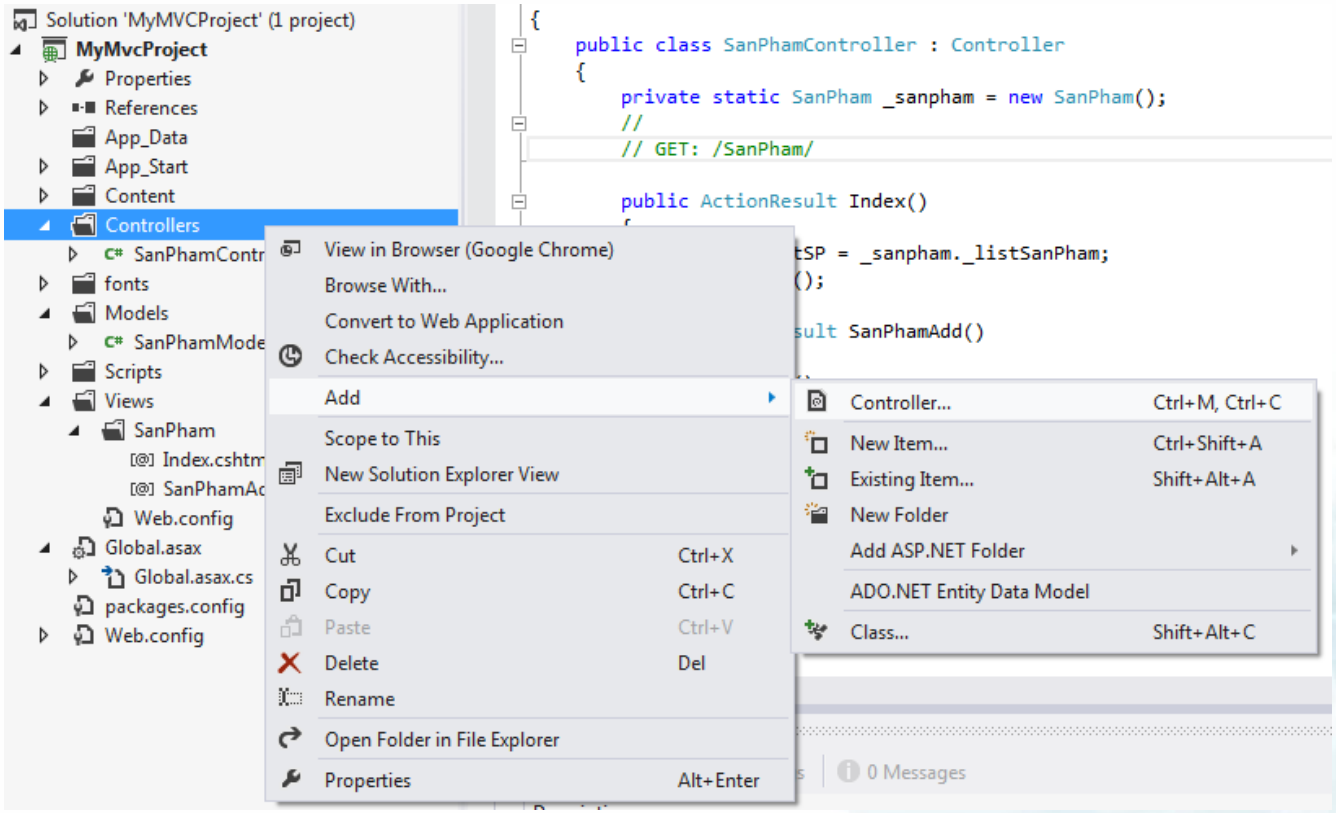
- Giới thiệu
- Tạo Controller
- Nhận Input
- Cung cấp Output
- Định tuyến

GIỚI THIỆU CONTROLLER

- Controller chịu trách nhiệm tương tác giữa models và views.
  - Tạo các biến đến model trong đáp ứng input của người dùng
  - Liên quan đến các luồng trong ứng dụng, làm việc với các dữ liệu đi vào, và cung cấp các dữ liệu đi ra view liên quan.
- URL báo routing mechanism là controller class được thể hiện và action method được gọi. Sau đó controller's method quyết định view sử dụng, tiếp theo view đó lại renders ra HTML.

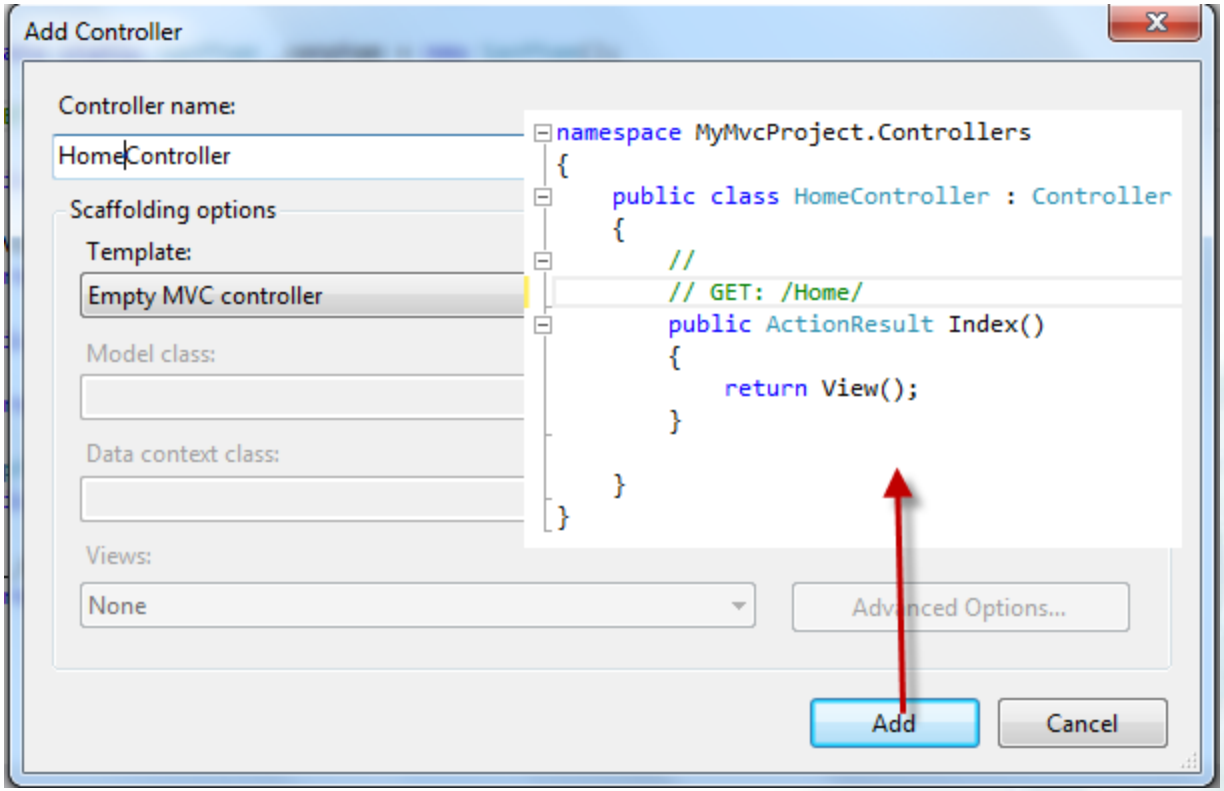
TẠO CONTROLLER

- Right-clicked vào Controllers folder→ Add→ Controller



TẠO CONTROLLER

- Right-clicked vào Controllers folder→ Add→ Controller



NHẬN INPUT

- Controllers thường phải truy cập dữ liệu đến như query string values, form values, và parameters phân tích từ URL đến bằng hệ thống routing system. Có 3 cách chính để truy cập dữ liệu này:
  - Từ tập context objects.
  - Data truyền như là parameters tới action method.
  - Data Model Binding (Gọi “framework’s model binding feature”)

NHẬN INPUT

- Nhận dữ liệu từ các đối tượng Context:
  - Controller cung cấp các thuộc tính truy cập trực tiếp các đối tượng sau:
    - public HttpContextBase *HttpContext* { get; }
    - public HttpRequestBase *Request* { get; }
    - public HttpResponseBase *Response* { get; }
    - public RouteData *RouteData* { get; }

NHẬN INPUT

- Getting Data from Context Objects:
  - Các thuộc tính Context Objects hay sử dụng

Property	Type	Mô tả
Request.QueryString	NameValueCollection	GET variables gửi với request này
Request.Form	NameValueCollection	POST variables gửi với request này.
Request.Cookies	HttpCookieCollection	Cookies gửi bởi trình duyệt cùng với request này.
Request.HttpMethod	string	HTTP method (GET or POST) sử dụng cho request này.
Request.Url	Uri	URL requested
Request.UserHostAddress	string	IP address của người dùng

NHẬN INPUT

- Getting Data from Context Objects:

Property	Type	Mô tả
RouteData.Route	RouteBase	Thực thể RouteTable.Routes của request này.
HttpContext.Application	HttpApplicationStateBase	Lưu trữ Application state
HttpContext.Cache	Cache	Lưu trữ Application cache
HttpContext.Items	IDictionary	Lưu trữ State của current request
HttpContext.Session	HttpSessionStateBase	Lưu trữ State cho visitor's session
User	IPrincipal	Thông tin Authentication về người dùng logged
TempData	TempDataDictionary	Temporary data items lưu trữ cho current user

- Getting Data from Context Objects:
  - Ví dụ:

View: Login.cshtml

```
1. @using (Html.BeginForm("LoginResult", "Account")){
2.    <p>Name: <input type="text" name="name" /></p>
3.    <p>Password: <input type="password" name="password" /></p>
4.    <input type="submit" value="Login" />
5. }
```

NHẬN INPUT

- Getting Data from Context Objects:
  - Ví dụ: Controller:

```
public ActionResult Login(){
    return View();
}
 [HttpPost]
 public ActionResult LoginResult(){
     string name = Request.Form["name"];
     string password = Request.Form["password"];
     if (name == "huongct" && password == "pass")
         ViewBag.Message = "Succeeded";
     else
         ViewBag.Message = "Failed";
     return View();
 }
```

- Getting Data from Context Objects:
  - Ví dụ:

```
public ActionResult RenameProduct() {
  // Access various properties from context objects
  string userName = User.Identity.Name;
  string serverName = Server.MachineName;
  string clientIP = Request.UserHostAddress;
  DateTime dateStamp = HttpContext.Timestamp;
  AuditRequest(userName, serverName, clientIP, dateStamp, "Renaming product");
  // Retrieve posted data from Request.Form
  string oldProductName = Request.Form["OldName"];
  string newProductName = Request.Form["NewName"];
  bool result = AttemptProductRename(oldProductName, newProductName);

  ViewData["RenameResult"] = result;
  return View("ProductRenamed");
}
```

NHẬN INPUT

- Using Action Method Parameters:
  - Truy cập dữ liệu sử dụng "*Request.Form*" or "*Request.QueryString*" không phải là ý tưởng hay vì ta phải nhập "manually" tên và hay lỗi.
  - Action method có thể nhận các tham số tương tự như việc sử dụng "manually" các đối tượng context.

NHẬN INPUT

- Using Action Method Parameters:
  - Cách tốt hơn là sử dụng parameters của action method. Nếu name của parameter khớp với key name của tập các đối tượng sau, data được truyền một cách tự động.
    - Request.Form
    - Request.QueryString
    - Request.Files
    - RouteData.Values

- Using Action Method Parameters:
  - Tham số lựa chọn và tham số bắt buộc:
    - Tham số lựa chọn: Nếu MVC Framework không tìm thấy giá trị cho tham số kiểu tham chiếu như kiểu string, object,... Action method vẫn được gọi nhưng được truyền với giá trị null. Để chuyển sang tham số bắt buộc khì cần viết code đầu Action Method từ chuỗi giá trị null (đảm bảo không nhận giá trị null) ví dụ throw một ArgumentNullException.

NHẬN INPUT

- Using Action Method Parameters:
  - Tham số lựa chọn và tham số bắt buộc:
    - Tham số bắt buộc: Nếu MVC Framework không tìm thấy giá trị cho tham số kiểu tham trị như kiểu int, double,... Thì xuất hiện ngoại lệ và Action method không được gọi... Để chuyển sang tham số lựa chọn:
      - Ta chỉ định giá trị mặc định,
      - Hoặc chuyển sang kiểu tham số kiểu nullable type (như int? or DateTime?)

- Using Action Method Parameters:
  - Ví dụ:

```
1.  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.  }
```

- Using Action Method Parameters:
  - Ví dụ:

```
public ActionResult ShowWeatherForecast() {
    string city =
                    (string)RouteData.Values["city"];
    DateTime forDate =
          DateTime.Parse(Request.Form["forDate"]);
   // ... implement weather forecast here ...
    return View();
}
```

- Using Action Method Parameters:
  - Ví dụ:

```
public ActionResult ShowWeatherForecast(string
city, DateTime forDate) {
    // ... implement weather forecast here ...
    return View();
}
```

NHẬN INPUT

- Using Action Method Parameters:
  - Ví dụ:

```
public ActionResult Search(string query = "all",
int page = 1)
{
    // ...process request...
    return View();
}
```

NHẬN INPUT

- Data Model Binding:
  - Khi làm việc với strongly-typed views, ta có cách truy cập dữ liệu tốt hơn thông qua Data Model Binding.
  - Ví dụ:

```
1.  namespace MvcApp.Models{
2.    public class Account {
3.      public string Name { get; set; }
4.      public string Password { get; set; }
5.    }
6.  }
```

- Data Model Binding:
  - Ví dụ:

```
1.  @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.  }
```

- Data Model Binding:
  - Ví dụ:

```
public class AccountController : Controller{
    public ActionResult Login(){
        Account acc = new Account { Name = “huongct" };
        return View(acc);
    }
    [HttpPost]
    public ActionResult Login(Account acc){
        if (acc.Name == “huongct" && acc.Password == “pass")
            ViewBag.Message = "Succeeded";
        else
            ViewBag.Message = "Failed";
        return View("LoginResult");
    }
}
```

CUNG CẤP OUTPUT

- Giới thiệu:
  - Một controller sau khi kết thúc tiến trình Request thường phải sinh ra một Response.
  - Ví dụ tạo HTML response gửi cho client sử dụng phương thức Response.Write, điều hướng sử dụng Response.Redirect

CUNG CẤP OUTPUT

- Giới thiệu:

```
public class DerivedController : Controller {
    //
    // GET: /Derived/
        public ActionResult Index(){
             return View();
         }
         public void ProduceOutput()
         {  if (Server.MachineName == "CHUHUONG"){
                 Response.Redirect("/SanPham/Index");
             }
             else              {
                 Response.Write("Controller:Derived, Action: ProduceOutput");
             }
         }
    }
```

CUNG CẤP OUTPUT

- Giới thiệu:
  - Mô hình MVC sử dụng đối tượng một đối tượng dẫn xuất từ class ActionResult thay cho đối tượng Response.
  - MVC Framework chứa một số built-in action result types được dẫn xuất từ ActionResult, và chúng có các phương thức helper thuận tiện trong class Controller.
  - Các ActionResult Types:

CUNG CẤP OUTPUT

- Giới thiệu:
  - Action methods thường có ánh xạ 1-1 với các tương tác người dùng.
  - Hầu hết action methods trả về thể hiện của class được dẫn xuất từ lớp ActionResult.
  - Lớp **ActionResult** là cơ sở cho tất cả action results. Có nhiều kiểu action result dựa trên tác vụ của action method đang thực hiện.
  - Ta có thể tạo action methods trả về một đối tượng của bất kỳ kiểu nào như string, integer, boolean value. Các kiểu này được gói trong kiểu ActionResult trước khi rendered cho luồng response .

CUNG CẤP OUTPUT

- Giới thiệu:
  - Sau là một số kiểu built-in ActionResult:

Type	Helper Methods	Mô tả
ViewResult	View	Sinh ra chỉ định hoặc default view template
PartialViewResult	PartialView	Sinh ra chỉ định hoặc default partial view template
RedirectToRoute Result	RedirectToAction RedirectToActionPermanent RedirectToRoute RedirectToRoutePermanent	Điều hướng đến một Action method hay route chỉ định.

CUNG CẤP OUTPUT

- Giới thiệu:

Type	Helper Methods	Mô tả
RedirectResult	Redirect RedirectPermanent	Điều hướng đến URL chỉ định.
ContentResult	Content	Trả về kiểu nội dung người dùng định nghĩa.
JsonResult	Json	Trả về đối serialized JSON
JavaScriptResult	JavaScript	Trả về script có thể thực thi ở client.
FileResult	File	Trả về binary output viết vào response.
EmptyResult	(None)	Không gì (void).

CUNG CẤP OUTPUT

- Trả về HTML bằng cách sinh View:

```
public class ExampleController : Controller
    {   //
        // GET: /Example/
        public ActionResult Index()
        {
            return View("HomePage");
        }
    }
```

→ Chỉ định view muốn render là “HomePage” bằng sử dụng phương thức View

CUNG CẤP OUTPUT

- Trả về HTML bằng cách sinh View:

```
public class ExampleController : Controller
    {   //
        // GET: /Example/
        public ActionResult Index()
        {
            return View();
        }
    }
```

→ Chỉ định view muốn render là Index (trùng tên Action method, tên thực sự của view xác định bởi RouteData.Values["action"]).

CUNG CẤP OUTPUT

- Truyền Data từ Action Method đến View :
  - Cung cấp View Model Object: Gửi một đối tượng đến view bằng cách truyền nó như một một tham số tới View method.

```
public ActionResult Index(){
    DateTime date = DateTime.Now;
    return View(date);
}
```

→ Đối tượng DateTime được truyền như view model. Để truy cập đối tượng ngày trong view sử dụng Razor Model keyword.

CUNG CẤP OUTPUT

- Truyền Data từ Action Method đến View :
  - Ví dụ: Truy cập đối tượng trên view

```
@Model
@{     ViewBag.Title = "Index";}
<h2>Index</h2>
The day is: @(((DateTime)Model).DayOfWeek)
```

Hoặc:

CUNG CẤP OUTPUT

- Truyền Data từ Action Method đến View :
  - Ví dụ: Truy cập đối tượng trên view

```
@model DateTime
@{    ViewBag.Title = "Index";}
<h2>Index</h2>
The day is: @Model.DayOfWeek
```

Chú ý: Khi sử dụng Razor model keyword sử dụng lowercase m khi chỉ định kiểu của model và uppercase M khi được giá trị của model.

CUNG CẤP OUTPUT

- Truyền Data từ Action Method đến View :
  - Truyền Data với View Bag: Có thể truyền data qua dynamic object và truy cập chúng trên view:

```
public ViewResult Index()
{   ViewBag.Message = "Hello";
    ViewBag.Date = DateTime.Now;

    return View();
}
```

CUNG CẤP OUTPUT

- Truyền Data từ Action Method đến View :
  - + Truyền Data với View Bag:
    - Đọc View Bag trên View

```
<h2>Index</h2>
@{ ViewBag.Title = "Index";}
<h2>Index</h2>
The day is: @ViewBag.Date.DayOfWeek
<p/> The message is: @ViewBag.Message
```

    - Visual Studio không cung cấp IntelliSense cho qua dynamic object bao gồm cả View Bag nên lỗi dạng '@ViewBag.Date.DayOfWeek.ohh.ohh' sinh ra khi view được rendered.

CUNG CẤP OUTPUT

- Thực hiện các Redirections :
  - Điều hướng đến Literal URL: Để điều hướng trình duyệt ta gọi phương thức Redirect và trả về thể hiện của class RedirectResult

```
public ActionResult Index()
{   DateTime date = DateTime.Now
    return View(date);
}
public RedirectResult Redirect()
{
  return Redirect("/Example/Index");
}
```

CUNG CẤP OUTPUT

- Thực hiện các Redirections :
  - Điều hướng Routing System URL: Sử dụng Routing System URL để sinh ra valid URLs dùng RedirectToRoute method và trả về thể hiện của class RedirectToRouteResult

```
public RedirectToRouteResult Redirect()
{   return RedirectToRoute(new
    {   controller = "Example",
        action = "Index",
        ID = "MyID"});
}
```

CUNG CẤP OUTPUT

- Thực hiện các Redirections :
  - Bảo toàn data qua Redirections: TempData tương tự như Session data, ngoại trừ giá trị của TempData được đánh dấu xóa khi chúng được đọc và được removed khi request được xử lý.
    - Ví dụ:

```
public RedirectToRouteResult RedirectToRoute()
{       TempData["Message"] = "Hello";
        TempData["Date"] = DateTime.Now;
        return RedirectToAction("Index");
}
```

CUNG CẤP OUTPUT

- Thực hiện các Redirections :
  - Bảo toàn data qua Redirections:
    - Ví dụ: Ta có thể đọc giá trị TempData quay về target action method, và sau đó gửi đến view

```
public ViewResult Index()
 {    ViewBag.Message = TempData["Message"];
      ViewBag.Date = TempData["Date"];
      return View();
  }
```

CUNG CẤP OUTPUT

- Thực hiện các Redirections :
  - Bảo toàn data qua Redirections:
    - Ví dụ: Hoặc đưa thẳng giá trị này qua view

```
@{    ViewBag.Title = "Index"; }
<h2>Index</h2>
The day is:
      @(((DateTime)TempData["Date"]).DayOfWeek)
<p /> The message is: @TempData["Message"]
```

CUNG CẤP OUTPUT

- Thực hiện các Redirections :
  - + Bảo toàn data qua Redirections:
    - Ta có thể nhận giá trị TempData mà không đánh dấu gỡ bỏ bằng phương thức Peek

```
DateTime time = (DateTime)TempData.Peek("Date");
```

    - Ta có bảo toàn một giá trị nếu không sẽ bị xóa bằng Keep method

```
TempData.Keep("Date");
```

ROUTE

- 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.

ROUTE

- Tạo và đăng ký định tuyến đơn giản:
  - Các routes được định nghĩa trong file RouteConfig.cs nằm trong thư mục App_Start:

```
public class RouteConfig
{    public static void RegisterRoutes(RouteCollection routes)
     {   routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
         routes.MapRoute(
               name: "Default",
               url: "{controller}/{action}/{id}",
               defaults: new { controller = "SanPham", action =
"Index", id = UrlParameter.Optional });
        }
   }
```

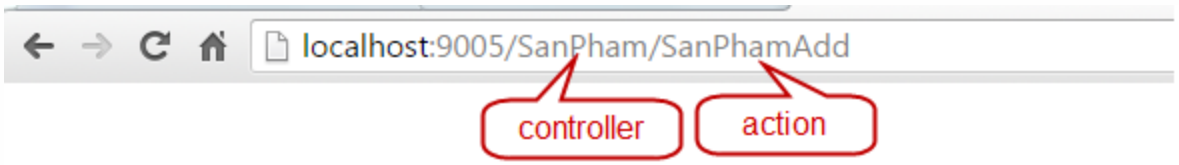
ROUTE

- Tạo và đăng ký định tuyến đơn giản:
  - Phương thức static RegisterRoutes được gọi trong tệp Global.asax.cs:

```
protected void Application_Start(
{   AreaRegistration.RegisterAllAreas();
    WebApiConfig.Register(GlobalConfiguration.Configuration);
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
}
```

ROUTE

- Tạo và đăng ký định tuyến đơn giản:




```
http://mysite.com/Admin/Index
```

First Segment Second Segment

- Tạo và đăng ký định tuyến đơn giản:

Request URL	Segment Variables
http://mysite.com/Admin/Index	controller = Admin action = Index
http://mysite.com/Index/Admin	controller = Index action = Admin
http://mysite.com/Apples/Oranges	controller = Apples action = Oranges
http://mysite.com/Admin	No match—too few segments
http://mysite.com/Admin/Index/Soccer	No match—too many segments

ROUTE

- Sử dụng Static URL Segments: Giả sử muốn URL chứa Public dạng

  http://mydomain.com/Public/Home/Index

  Đăng ký như sau:

```
public class RouteConfig {
    public static void RegisterRoutes(RouteCollection routes) {

        routes.MapRoute("MyRoute", "{controller}/{action}",
            new { controller = "Home", action = "Index" });

        routes.MapRoute("", "Public/{controller}/{action}",
           new { controller = "Home", action = "Index" });
    }
}
```

ROUTE

- Sử dụng Static URL Segments: Giả sử muốn URL chứa Public dạng

  http://mydomain.com/Public/Home/Index

  Đăng ký như sau:

```
public class RouteConfig {
    public static void RegisterRoutes(RouteCollection routes) {

        routes.MapRoute("MyRoute", "{controller}/{action}",
            new { controller = "Home", action = "Index" });

        routes.MapRoute("", "Public/{controller}/{action}",
           new { controller = "Home", action = "Index" });
    }
}
```

ROUTE

- Sử dụng Static URL Segments: Hoặc trộn lẫn

  http://mydomain.com/XHome/Index

  Đăng ký như sau:

```
routes.MapRoute("", "X{controller}/{action}");
```

ROUTE

- Định nghĩa Segment Variables: MVC Framework không giới hạn Segment Variables, ta có thể định nghĩa như ví dụ:

```
public class RouteConfig {
    public static void RegisterRoutes(RouteCollection routes) {

        routes.MapRoute("MyRoute", "{controller}/{action}/{id}",
            new { controller = "Home", action = "Index",
                id = "DefaultId" });
    }
}
```

  + URL pattern định nghĩa standard controller, action variables và custom variable id.

- Định nghĩa Segment Variables:
  - Truy cập Segment Variables:
    - Sử dụng thuộc tính RouteData.Values để truy cập.

```
public class HomeController : Controller {

    public ActionResult Index() {
        ViewBag.Controller = "Home";
        ViewBag.Action = "Index";
        return View("ActionName");
    }

    public ActionResult CustomVariable() {
        ViewBag.Controller = "Home";
        ViewBag.Action = "CustomVariable";
        ViewBag.CustomVariable = RouteData.Values["id"];
        return View();
    }
}
```

- Định nghĩa Segment Variables:
  - 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.

```
public ActionResult CustomVariable(string id) {
    ViewBag.Controller = "Home";
    ViewBag.Action = "CustomVariable";
    ViewBag.CustomVariable = id;
    return View();
}
```

ROUTE

- 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:

```
public class RouteConfig {
    public static void RegisterRoutes(RouteCollection routes) {

        routes.MapRoute("MyRoute", "{controller}/{action}/{id}",
            new { controller = "Home", action = "Index",
                id = UrlParameter.Optional });
    }
}
```

- Định nghĩa Optional URL Segments:

Segments	Example URL	Maps To
0	mydomain.com	controller = Home action = Index
1	mydomain.com/Customer	controller = Customer action = Index
2	mydomain.com/Customer/List	controller = Customer action = List
3	mydomain.com/Customer/List/All	controller = Customer action = List id = All
4	mydomain.com/Customer/List/All/Delete	No match—too many segments

- Ví dụ:

```
  public static void RegisterRoutes(RouteCollection routes){
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
        routes.MapRoute( name: "Cart", url: "gio-hang/{id}",
          defaults: new { controller = "Cart", action = "Index", id =
UrlParameter.Optional },
          namespaces: new[] { "GioHangMVC.Controllers" }
      );
      routes.MapRoute(
              name: "Default",
              url: "{controller}/{action}/{id}",
              defaults: new { controller = "SanPham", action = "Index", id =
UrlParameter.Optional }
          );
       }
```

THẢO LUẬN – CÂU HỎI