

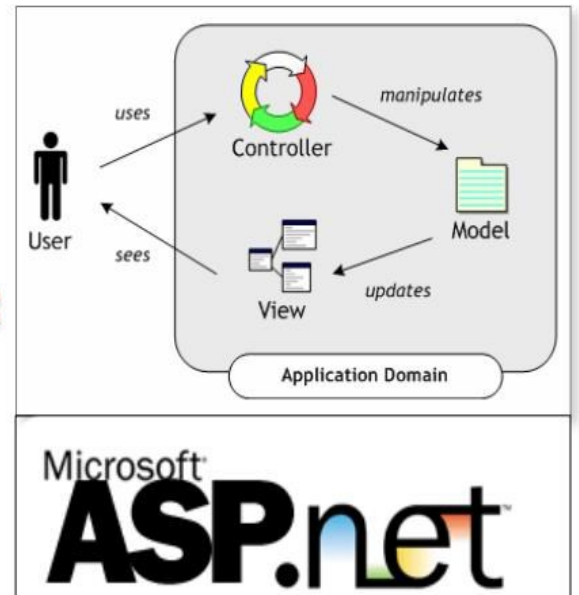
Bài 2. Tổng quan về ASP.Net MVC 5

- **Mục đích, yêu cầu:** Cung cấp cho sinh viên các khái niệm, đặc tính về ASP.Net MVC 5 Framework, so sánh sự khác biệt ASP.Net MVC 5 với Web Form, khái niệm về các đối tượng trong mô hình MVC.
- **Hình thức tổ chức dạy học:** Lý thuyết, trực tuyến + tự học
- **Thời gian:** Lý thuyết(online: 3) Tự học, tự nghiên cứu: 06
- **Nội dung:**

1. Giới thiệu về ASP.Net MVC 5	2
1.1. ASP.NET MVC là gì ?	2
1.2 MVC làm việc như thế nào?	3
1.3 Ưu & Khuyết điểm của MVC	3
1.4 Sự khác biệt với WebForm	4
1.5 Lợi ích web dựa trên mô hình MVC	4
1.6. Lịch sử phát triển ASP.Net MVC:	5
2. XÂY DỰNG ỨNG DỤNG WEB	5
2.1 Tạo mới Project ASP.Net MVC 5	5
2.2 Các thành phần trong Project ASP.Net MVC	7
2.3 Tạo mới một Controller	8
2.4 Điều hướng hiển thị:	12
2.5 Tạo mới một View	16
2.6 Thay đổi Layout Pages (giao diện của trang)	19

1. Giới thiệu về ASP.Net MVC 5

Microsoft
ASP.net MVC =



1.1. ASP.NET MVC là gì ?

✓ Controller

- Nhận yêu cầu từ user
- Xử lý và xây dựng model phù hợp
- Chuyển Model cho View

✓ **View:** Tiếp nhận Model từ Controller để sinh giao diện phù hợp

✓ **Model:** Chứa dữ liệu chia sẻ chung giữa Controller và View

Models

✓ Lưu trữ thông tin, trạng thái của các đối tượng, là 1 lớp được ánh xạ từ 1 bảng trong CSDL.

✓ Chứa tất cả các nghiệp vụ logic, phương thức xử lý, truy xuất database, các Class, hàm xử lý..

Ví dụ: lớp *Product* được sử dụng để mô tả dữ liệu từ bảng *Products*, bao gồm *ProductID*, *OrderDate*...

Views

✓ Chịu trách nhiệm hiển thị các thông tin lên cho người dùng thông qua giao diện.

- ✓ Chứa các đối tượng GUI(Textbox, images...).
- ✓ Các thông tin cần hiển thị được lấy từ thành phần **Models**.

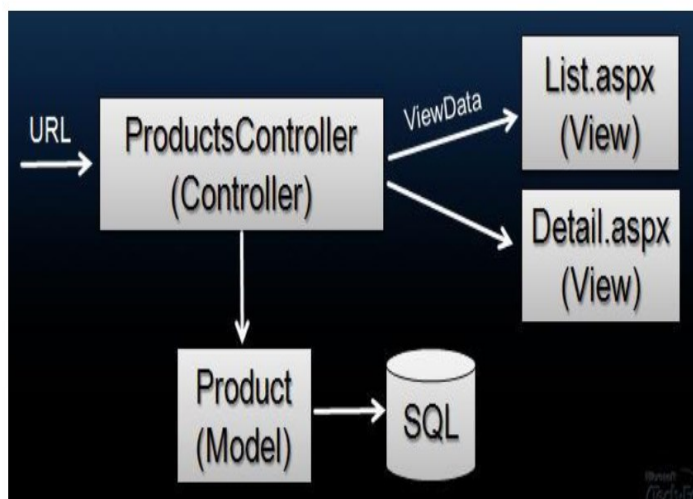
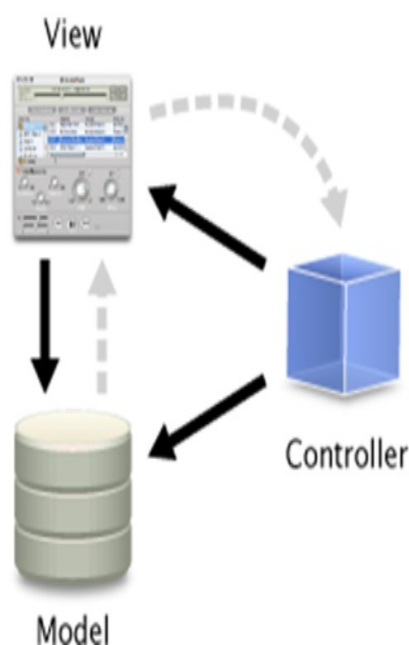
Ví dụ: Đối tượng *Product* có "Edit" view bao gồm các textboxes, các dropdowns và checkboxes để chỉnh sửa các thuộc tính của sản phẩm.

Controllers

- ✓ Xử lý các tác động về mặt giao diện, các thao tác đối với models, và chọn view để hiển thị ra màn hình.
- ✓ Điều hướng các yêu cầu từ người dùng và gọi phương thức xử lý.
- ✓ Trong MVC, view chỉ có tác dụng hiển thị giao diện, còn điều khiển vẫn do Controllers đảm trách.

1.2 MVC làm việc như thế nào?

- ✓ User tương tác với View, bằng cách click vào button, gửi yêu cầu đi.
- ✓ Controller nhận và điều hướng đến phương thức xử lý ở Model.
- ✓ Model nhận thông tin và thực thi các yêu cầu, View sẽ nhận kết quả từ Model và hiển thị lại cho người dùng.



1.3 Ưu & Nhược điểm của MVC

Ưu điểm:

- ✓ Thể hiện tính chuyên nghiệp trong lập trình, PTTK.

- ✓ Được chia thành các thành phần độc lập nên giúp phát triển ứng dụng nhanh, dễ nâng cấp, bảo trì..
- ✓ Ứng dụng tạo ra chạy ổn định trên Windows
- ✓ Đáp ứng nhiều loại thiết bị truy cập
- ✓ An toàn, Dễ tích hợp

Khuyết điểm:

- ✓ Đối với dự án nhỏ việc áp dụng mô hình MVC gây công kênh, tốn thời gian trong quá trình phát triển.
- ✓ Tốn thời gian trung chuyển dữ liệu của các thành phần

1.4 Sự khác biệt với WebForm

Tính năng	ASP.NET 2.0	ASP.NET MVC
Kiến trúc chương trình	Kiến trúc mô hình WebForm → Business → Database	Kiến trúc sử dụng việc phân chia chương trình thành Controllers, Models, Views
Cú pháp chương trình	Sử dụng cú pháp của webform, tất cả các sự kiện và controls do server quản lý	Các sự kiện được điều khiển bởi controllers, các controls không do server do server quản lý
Truy cập dữ liệu	Sử dụng hầu hết các công nghệ truy cập dữ liệu trong ứng dụng	Phần lớn dùng LINQ to SQL class để tạo mô hình truy cập đối tượng
Debug	Debug chương trình phải thực hiện tất cả bao gồm các lớp truy cập dữ liệu, sự hiển thị, điều khiển các controls	Debug có thể sử dụng các unit test kiểm tra các phương thức trong controller
Tốc độ phân tải	Tốc độ phân tải chậm khi trong trang có quá nhiều các controls vì ViewState quá lớn	Phân tải nhanh hơn do không phải quản lý ViewState để quản lý các control trong trang
Tương tác với javascript	Tương tác với javascript khó khăn vì các controls được điều khiển bởi server	Tương tác với javascript dễ dàng vì các đối tượng không do server quản lý điều khiển không khó
URL Address	Cấu trúc địa chỉ URL có dạng <filename>.aspx?&<các tham số>	Cấu trúc địa chỉ rành mạch theo dạng Controllers/Action/Id

1.5 Lợi ích web dựa trên mô hình MVC

- ✓ Dễ dàng quản lý sự phức tạp của ứng dụng bằng cách chia ứng dụng thành Model, View, Controller.
- ✓ Không sử dụng view state hoặc server-based form.

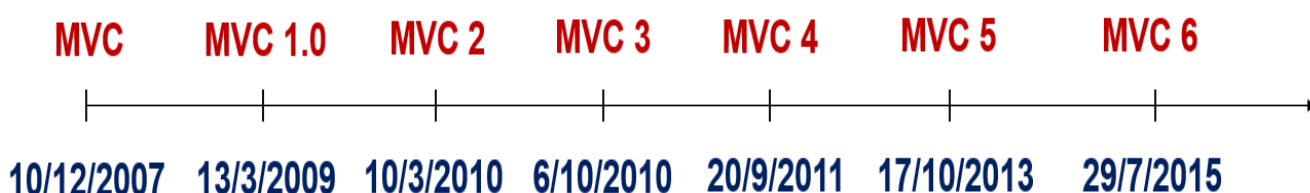
Điều này tốt cho những lập trình viên muốn quản lý hết các khía cạnh của một ứng dụng.

✓ Sử dụng mẫu Front Controller, mẫu này giúp quản lý các requests (yêu cầu) chỉ thông qua một Controller.

Hỗ trợ tốt hơn cho mô hình phát triển ứng dụng hướng kiểm thử (TDD)

✓ Hỗ trợ tốt cho các ứng dụng được xây dựng bởi những đội có nhiều lập trình viên và thiết kế mà vẫn quản lý được tính năng của ứng dụng.

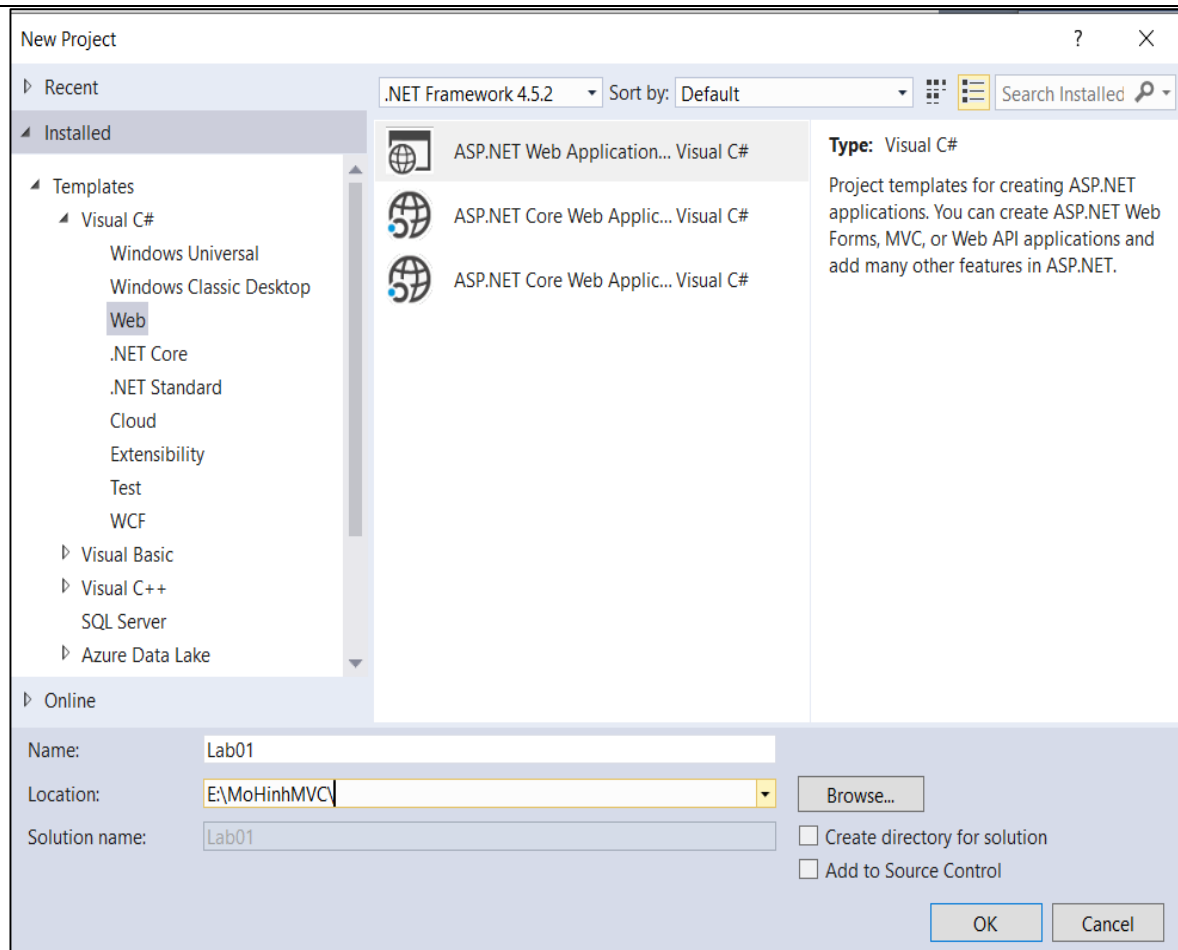
1.6. Lịch sử phát triển ASP.Net MVC:



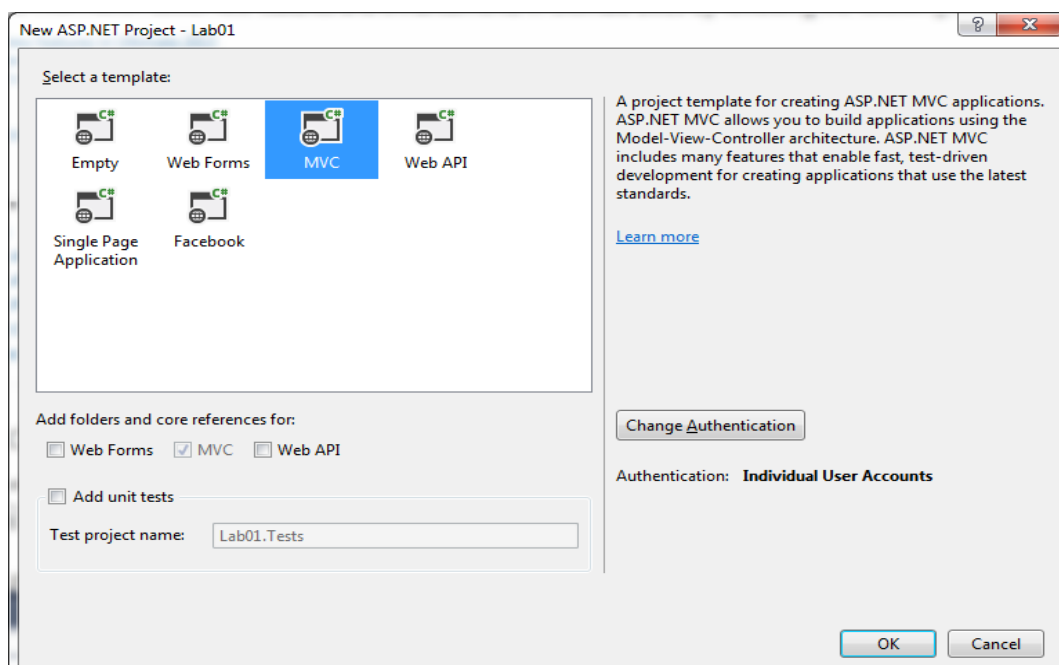
2. XÂY DỰNG ỨNG DỤNG WEB

2.1 Tạo mới Project ASP.Net MVC 5

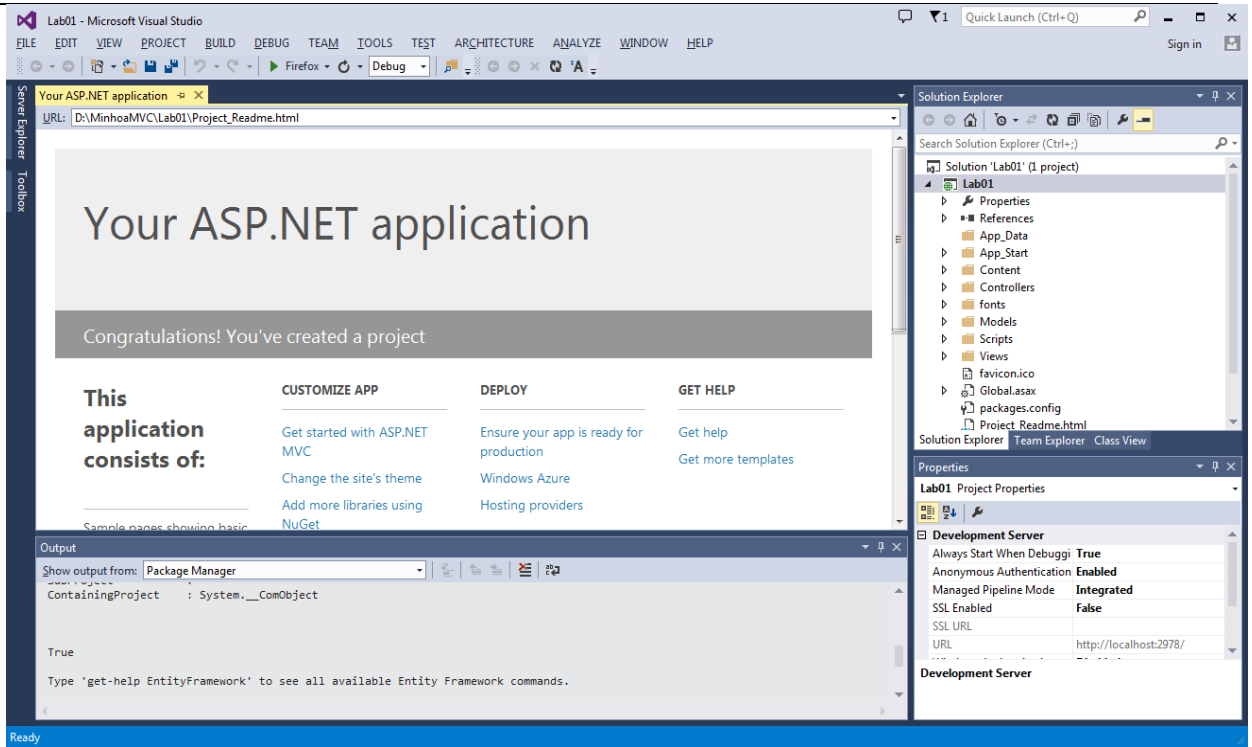
- ✓ Khởi động Visual Studio 2017
- ✓ Tạo mới 1 project: File -> New Project
 - Click vào New Project
 - Chọn Visual C# ở khung bên trái, rồi chọn Web
 - Chọn ASP.NET Web Application khung bên phải.
 - Đặt tên cho project là "**Lab01**" rồi click OK.



✓ Ở cửa sổ New ASP.NET Project, click MVC rồi click OK.



Visual Studio sử dụng một khuôn mẫu mặc định (default template) cho ASP.NET MVC Project vừa tạo, do đó sẽ có ngay một ứng dụng có thể chạy ngay. Đây là một project đơn giản, phù hợp để bắt đầu.



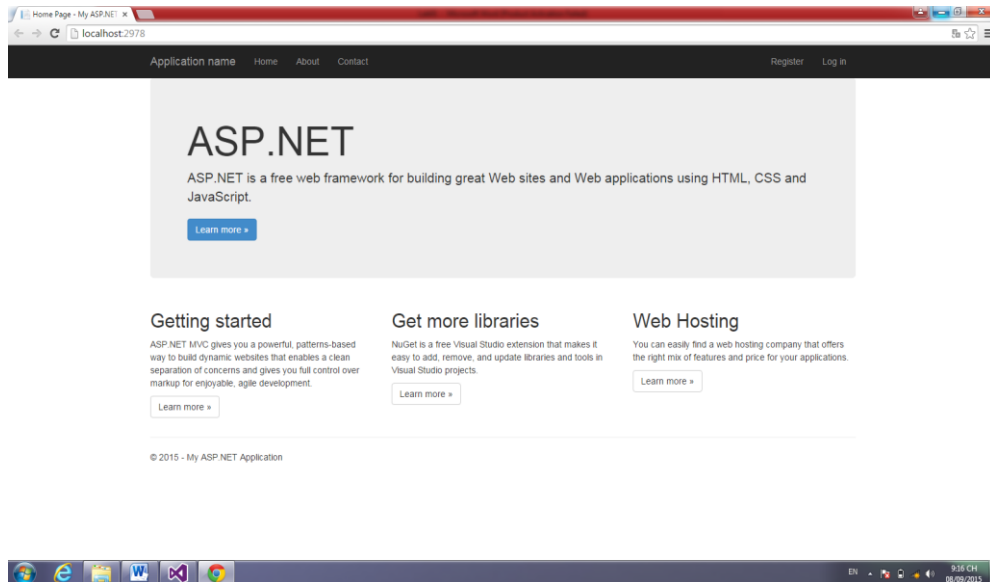
2.2 Các thành phần trong Project ASP.Net MVC

- **Properties:** Chứa các thuộc tính của project.
- **References:** Chứa các thư viện được sử dụng trong Project
- **App_Data:** Thư mục chứa file dữ liệu của Project nếu add cả file dữ liệu vào project
- **App_Start:** Thư mục chứa các file cấu hình khởi động và biên dịch của project. Chú ý đến 2 file
 - **FilterConfig.cs** dùng để khai báo các filter sử dụng trước khi thực hiện 1 hành động nào đó
 - **RouteConfig.cs** định nghĩa các routes trong project.
- **Content:** Thư mục chứa các file .CSS (dùng cho các view)
- **Controllers:** Thư mục chứa các file xxController.cs là các Controller
- **Models:** Thư mục chứa các file .cs là các Model gắn với các bảng trong CSDL.
- **Scripts:** Thư mục chứa các file .JS (dùng cho các view)
- **Views:** Thư mục chứa các view trong các folder, mỗi view là một file HTML với đuôi là .cshtml.

- **Shared:** Thư mục chứa các file HTML với đuôi là .cshtml dùng chung trong các view.

- **Global.aspx:** File chứa các khai báo chung sử dụng cho toàn bộ project (Biến toàn cục).
- **package.config:** File quản lý các package chúng ta cài vào
- **Web.config:** File quan trọng, định nghĩa các cài đặt hệ thống cho project.

Chạy thử bằng cách nhấn **F5** hoặc **Ctrl + F5** (chế độ không cần Debug) để xem kết quả: Visual Studio sẽ gọi một tiến trình là IIS để chạy ứng dụng. Sau đó sẽ gọi trình duyệt để duyệt vào ứng dụng. Lúc này, quan sát trên thanh địa chỉ của trình duyệt, sẽ thấy một địa chỉ có kiểu như sau: localhost:port.

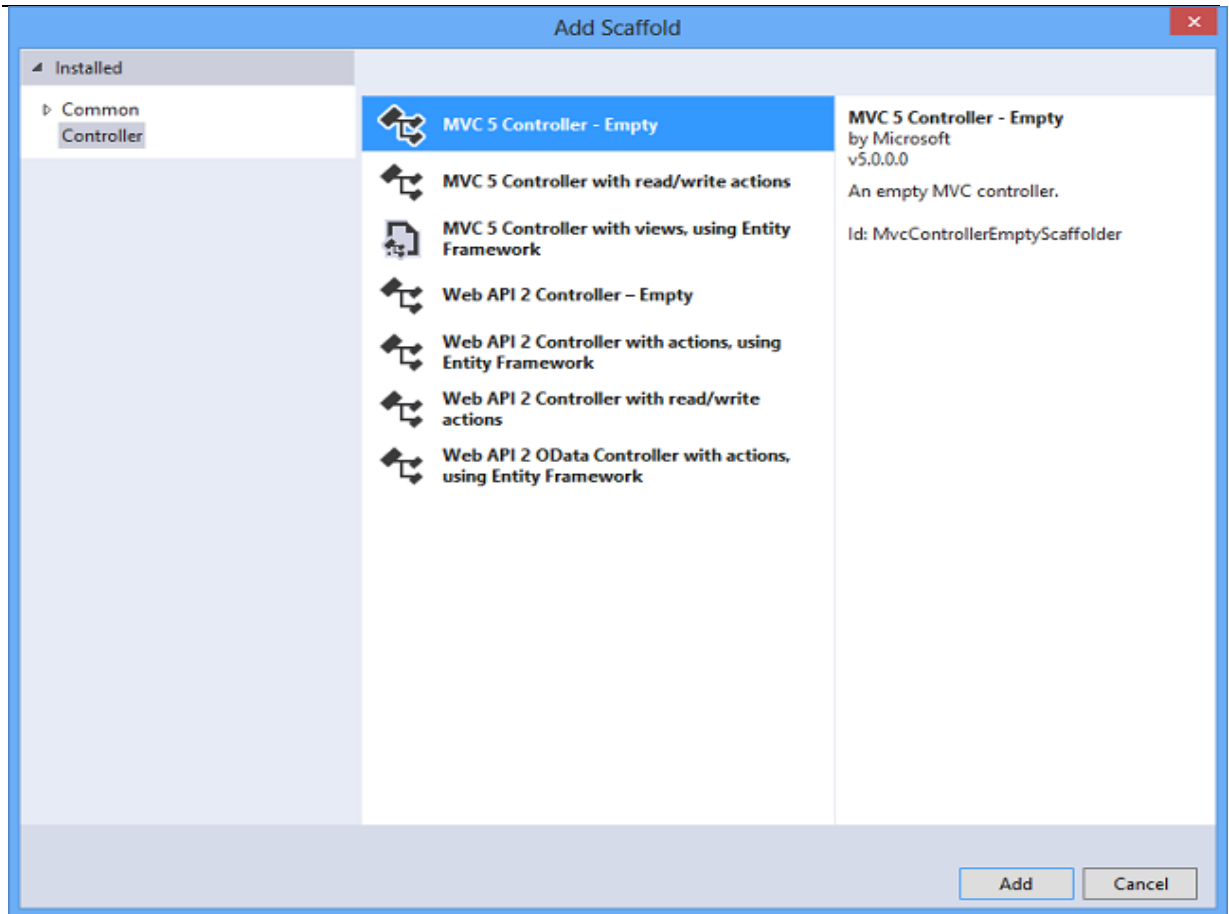


2.3 Tạo mới một Controller

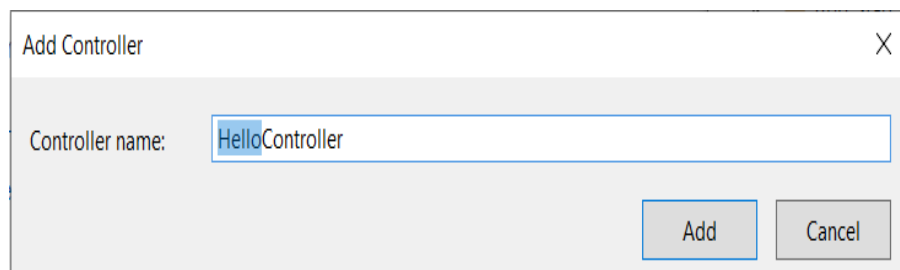
Bắt đầu tạo ra một lớp controller:

- ✓ Trong cửa sổ Solution Explorer, right-click thư mục *Controllers*
- ✓ Click Add,
- ✓ Chọn *Controller*

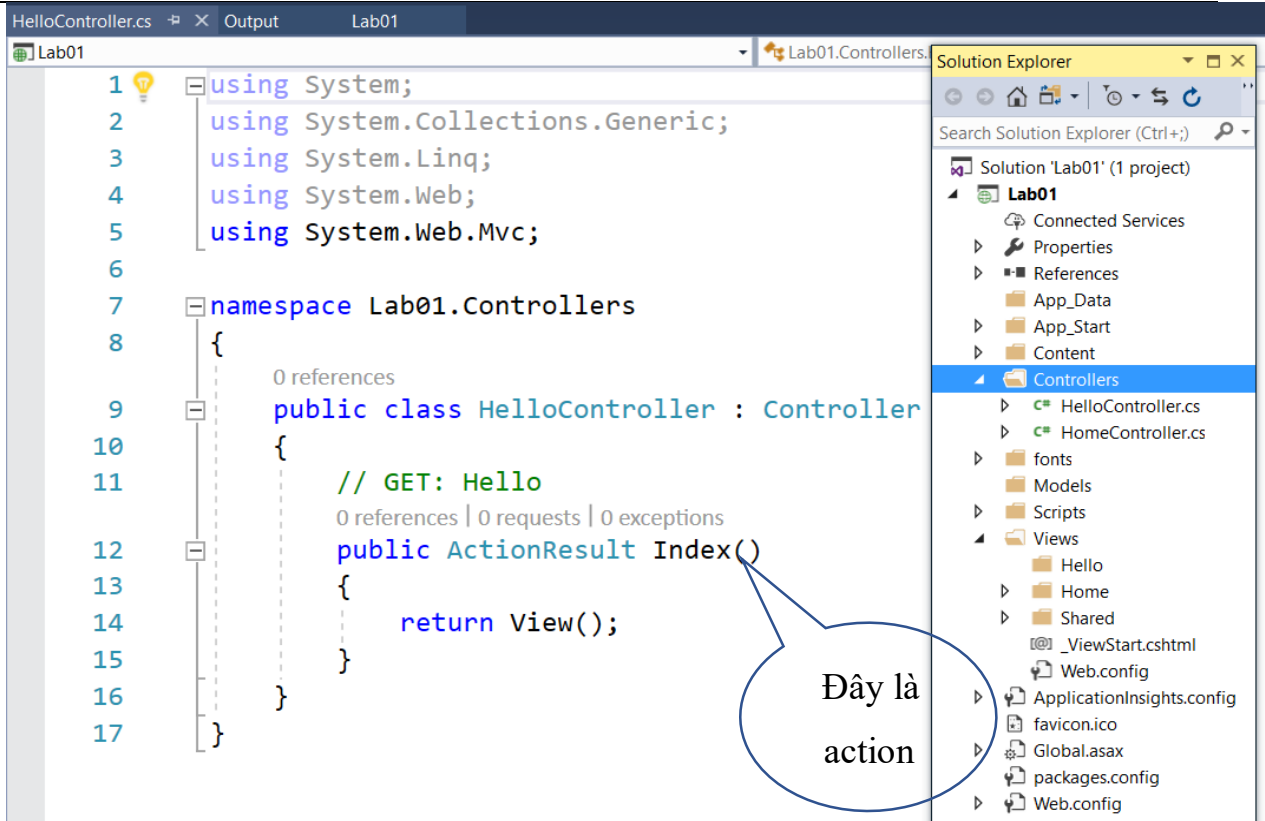
Trong cửa sổ Add Scaffold, click MVC 5 Controller - Empty, rồi click Add.



Đặt tên cho controller mới tạo là **"HelloController"** rồi click Add



Như vậy trong cửa sổ Solution Explorer sẽ có một file mới được tạo có tên là **HelloController.cs** và một thư mục mới có tên là **Views\Hello**. Mặc định controller mới tạo sẽ được mở sẵn trong IDE.



- **Action method** (Phương thức hành động)
 - Một controller có thể có nhiều action method để xử lý cho các yêu cầu cần thiết
 - Thường có ánh xạ one-to-one với các tương tác của người dùng (như: nhập URL vào cửa sổ trình duyệt, click chuột vào 1 đường link, submit một form, ...)
 - **Cú pháp**

```

public Kiểu_trả_về Tên_Action ( [ ThamSố ] )
{
    //Tập hợp lệnh xử lý
    //...
    return Giá_trị_trả_về;
}

```

Trong đó **Kiểu_trả_về** có thể là: **ActionResult** hoặc một kiểu được dẫn xuất từ **ActionResult**

- Hầu hết các action method trả về một thể hiện của một lớp được dẫn xuất từ **ActionResult**.

- Lớp **ActionResult** là lớp cơ sở cho tất cả các kết quả trả về của action.

Hãy thay nội dung đoạn code như bên dưới.

```
public class HelloController : Controller
```

```
{
```

```
public class HelloController : Controller
{
    // GET: /Hello/
    0 references | 0 requests | 0 exceptions
    public string Index()
    {
        return "Đây là phương thức Index, phương thức mặc định của Controller Hello.";
    }
    // GET: /Hello/ChaoMung/
    0 references | 0 requests | 0 exceptions
    public string ChaoMung()
    {
        return "Đây là phương thức ChaoMung nằm trong Controller Hello!";
    }
}
```

```
}
```

Giải thích đoạn mã trên:

- Phương thức **Index()** trả về kiểu string với giá trị là “Đây là phương thức Index, phương thức mặc định của **Controller Hello**.” Đây là phương thức mặc định của 1 Controller bất kỳ.
- Phương thức **ChaoMung()** cũng trả về kiểu string với giá trị “Đây là phương thức ChaoMung nằm trong Controller Hello!”

Ta có thể tạo nhiều phương thức thực thi ở tập tin **HelloController.cs** tùy ý.

Chạy thử bằng cách nhấn **F5** hoặc **Ctrl + F5** (chế độ không cần Debug) để xem kết quả.

Application name

ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

Giao diện website MVC mặc định

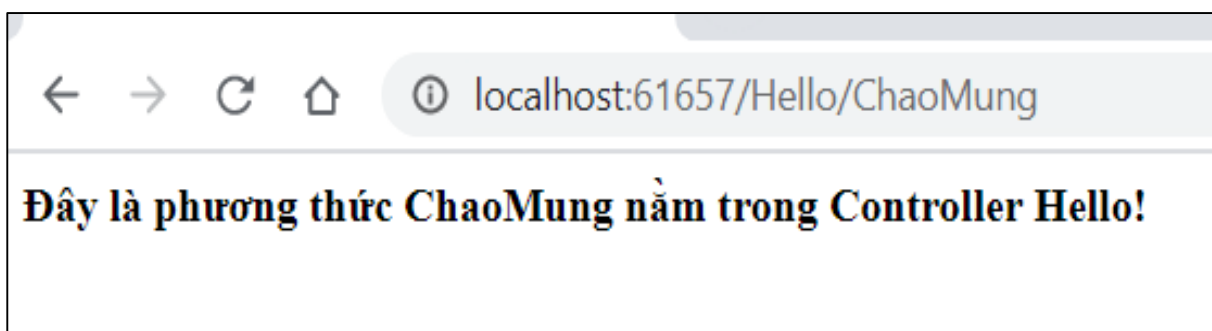
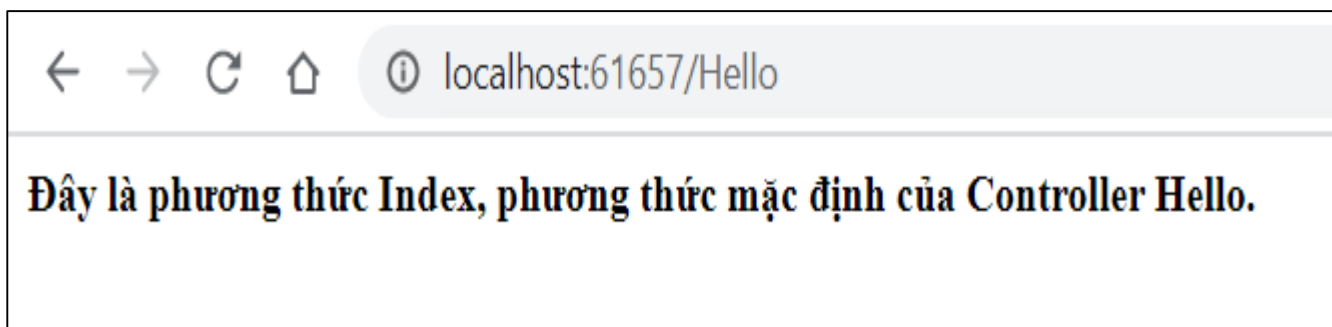
Ở trình duyệt, ta thử chạy 2 địa chỉ:

<http://localhost:xxxx/Hello/>

và <http://localhost:xxxx/Hello/ChaoMung>

để xem kết quả (với xxxx là số cổng tự động giao bởi server **IIS Express** của Visual Studio, bạn không cần quan tâm số cổng này).

Kết quả như hình sau:



2.4 Điều hướng hiển thị:

ASP.NET MVC sẽ gọi các **controller** khác nhau cùng với các phương thức tương ứng, điều này phụ thuộc vào các URL trên thanh địa chỉ của trình duyệt. Mặc định, như sau:

/[Controller]/[ActionName]/[Parameters].

Ta có thể thiết lập các định dạng điều hướng trong tập tin

App_Start/RouteConfig.cs

```

public class RouteConfig
{
    1 reference | 0 exceptions
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
        );
    }
}

```

Khi chạy một ứng dụng và nếu không chỉ định URL cụ thể thì sẽ lấy mặc định là **"Home" controller** và phương thức **"Index"**.

Trong đó, phần đầu của URL để xác định **controller** nào. Như vậy, **/Hello** sẽ ánh xạ đến lớp **HelloController**.

Phần thứ hai của URL để xác định phương thức nào sẽ thực thi. Như vậy **/Hello/Index** sẽ gọi phương thức **Index** của lớp **HelloController** để thực thi. Trong trường hợp, chỉ chỉ định **/Hello** thì có nghĩa là phương thức có tên **Index** sẽ được xem là mặc định sẽ thực thi.

Phần thứ ba của URL để xác định các tham số (Parameters) cung cấp cho phương thức (sẽ đề cập sau)

Ví dụ điều chỉnh code trong **App_Start/RouteConfig.cs** như sau:

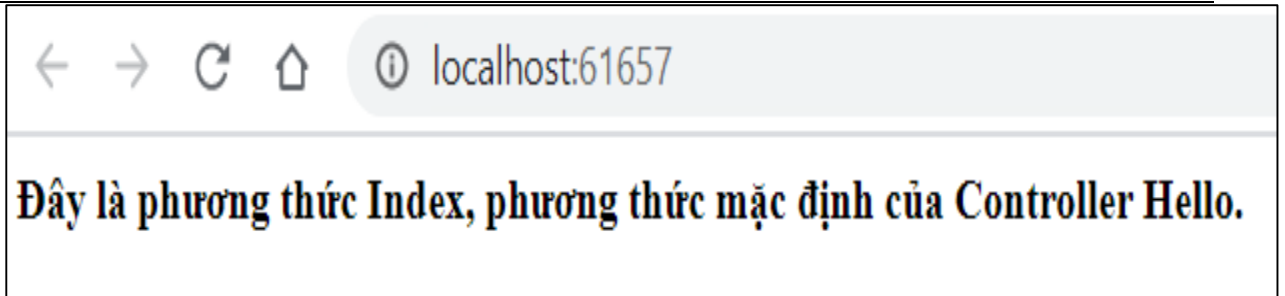
```

public class RouteConfig
{
    1 reference | 0 exceptions
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

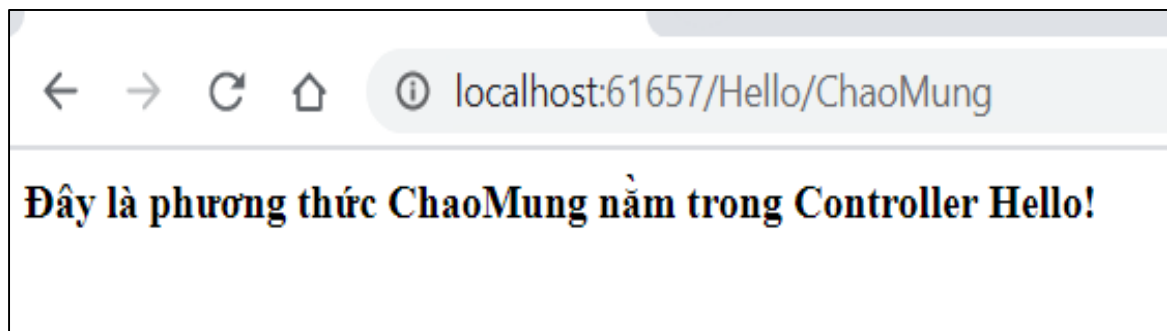
        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            // defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
            defaults: new { controller = "Hello", action = "Index", id = UrlParameter.Optional }
        );
    }
}

```

Chạy thử, kết quả sau khi điều hướng Controller:



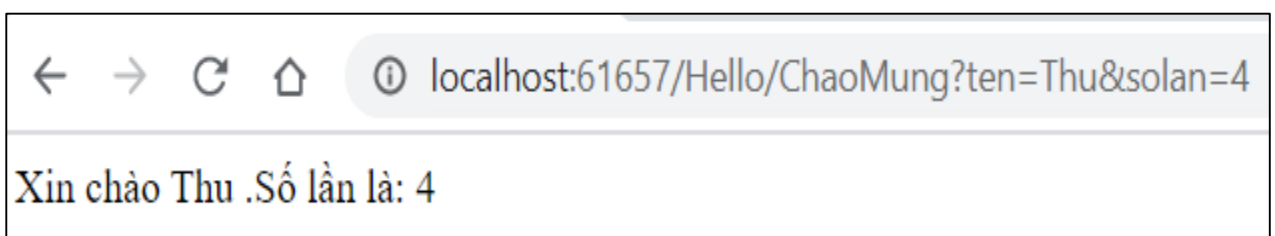
Duyệt đến URL ***http://localhost:61657/Hello/ChaoMung/***. Phương thức ***ChaoMung*** chạy và trả về là một chuỗi ***"Đây là phương thức Index,..."***. Mặc nhiên MVC đang ánh xạ tới ***/[Controller]/[ActionName]/[Parameters]***. Như vậy với URL này, ***controller*** là ***Hello*** và phương thức được thực hiện là ***ChaoMung***(không có sử dụng phần ***[Parameters]*** ở trong URL này).



Để sử dụng các tham số(*Parameters*), ta thay đổi code ở phương thức ***ChaoMung*** như sau:

```
// GET: /Hello/ChaoMung?ten=Thu&solan=4
0 references | 0 requests | 0 exceptions
public string ChaoMung(string ten, int solan=1)
{
    return HttpUtility.HtmlEncode("Xin chào " + ten + " .Số lần là: " + solan);
}
```

Chạy ứng dụng: **<http://localhost:61657/Hello/ChaoMung?ten=Thu&solan=4>**

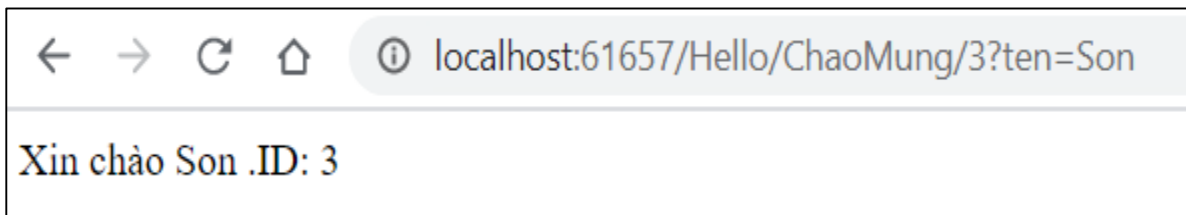


Như vậy ở ví dụ trên thì thành phần tham số (*Parameters*) theo cấu trúc vẫn chưa dùng, tham số ***ten*** và ***solan*** được dùng ở đây chỉ là tham số theo ***query strings***. Dấu ***?*** (question mark) trong URL là một phân ngăn cách để chỉ ra phía sau đó là

query strings. Dấu **&** để ngăn cách các cặp *query strings* với nhau. Ta tiếp tục cập nhật lại đoạn code với nội dung như sau:

```
// GET: /Hello/ChaoMung
0 references | 0 requests | 0 exceptions
public string ChaoMung(string ten, int id = 1)
{
    return HttpUtility.HtmlEncode("Xin chào " + ten + " .ID: " + id);
}
```

Chạy ứng dụng: ***http://localhost:61657/Hello/ChaoMung/3?ten=Son***



Lúc này thành phần thứ 3 trong URL ánh xạ đúng với parameter ID. Phương thức ***ChaoMung*** có chứa tham số (ID) đã đúng với phần đăng ký của phương thức ***RegisterRoutes***.

```
public class RouteConfig
{
    1 reference | 0 exceptions
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

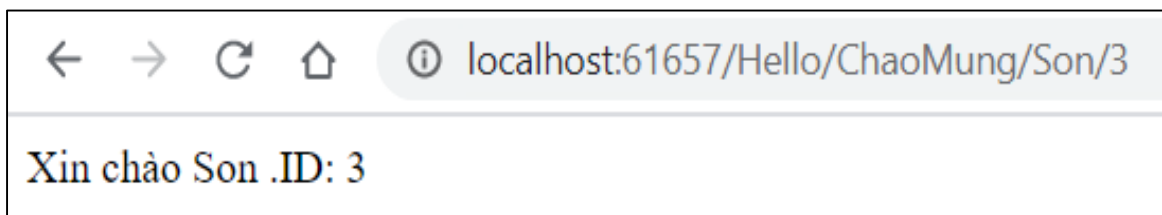
        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
        );
    }
}
```

Trong một ứng dụng ASP.NET MVC, đây là dạng điển hình trong việc truyền tham số (giống như tham số ID ở ví dụ trên), thay vì phải truyền tham số theo *query strings*, cũng có thể thêm vào cấu trúc cho ***ten*** và ***id*** ở phần parameters trong URL. Tại file ***App_Start\RouteConfig.cs***:

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        // defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
        defaults: new { controller = "Hello", action = "Index", id = UrlParameter.Optional }
    );
    routes.MapRoute(
        name: "Hello",
        url: "{controller}/{action}/{ten}/{id}"
    );
}
```

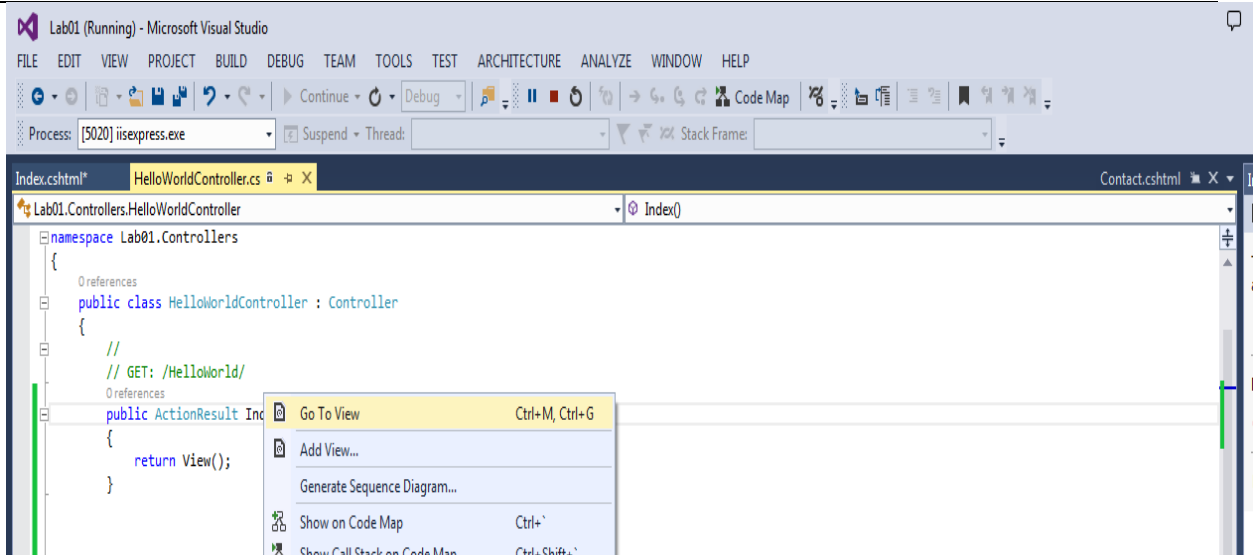
Chạy ứng dụng: ***http://localhost:61657/Hello/ChaoMung/Son/3***



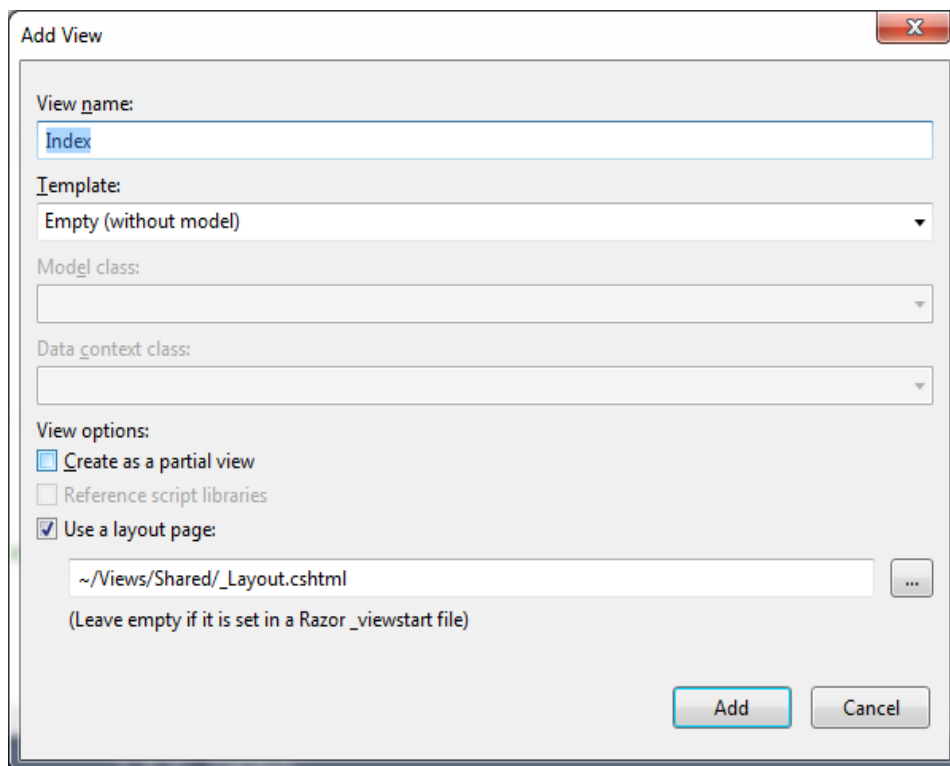
Đối với các ứng dụng MVC, các định tuyến mặc định sẽ hoạt động tốt hầu hết các trường hợp. Tuy nhiên, tùy vào các nhu cầu cụ thể, ta có thể thay đổi các định tuyến để phù hợp với các nhu cầu.

2.5 Tạo mới một View

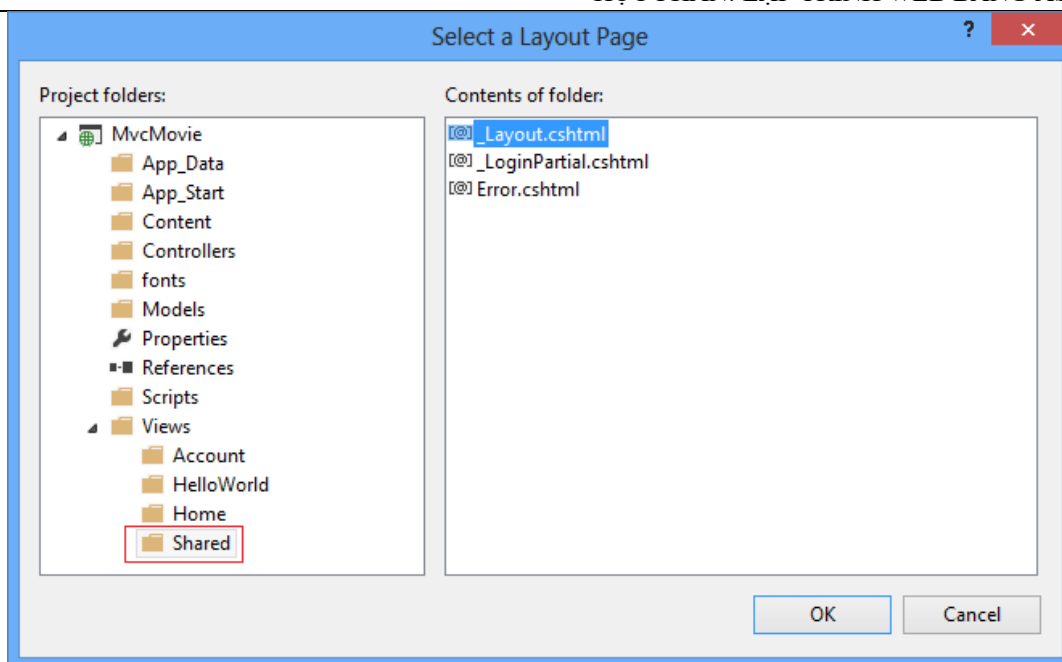
- ✓ Ta tiếp tục cập nhật lớp **HelloController** để sử dụng với hiển thị một file khuôn mẫu giao diện (View Template File) để hiểu rõ việc tạo ra một HTML trả về để hiển thị phía client (browser).
- ✓ Hiện tại thì phương thức **Index** trong lớp **controller**. Ta sẽ thay đổi phương thức **Index** để nó trả về một View object, và hiển thị nó: Right click lên tên phương thức, chọn **Add View**



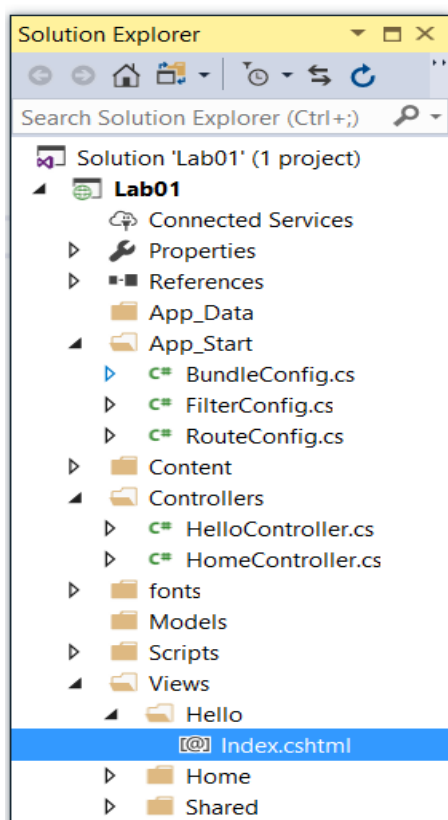
Tại cửa sổ **Add View**, gõ tên view là **Index**, Chọn Layout tại mục **Use a layout page**, rồi click OK.



Tại cửa sổ **Select a Layout Page**, chọn mặc định là **View/Shared/_Layout.cshtml** rồi click OK.



Tập tin `Views\Hello\Index.cshtml` được tạo như sau:

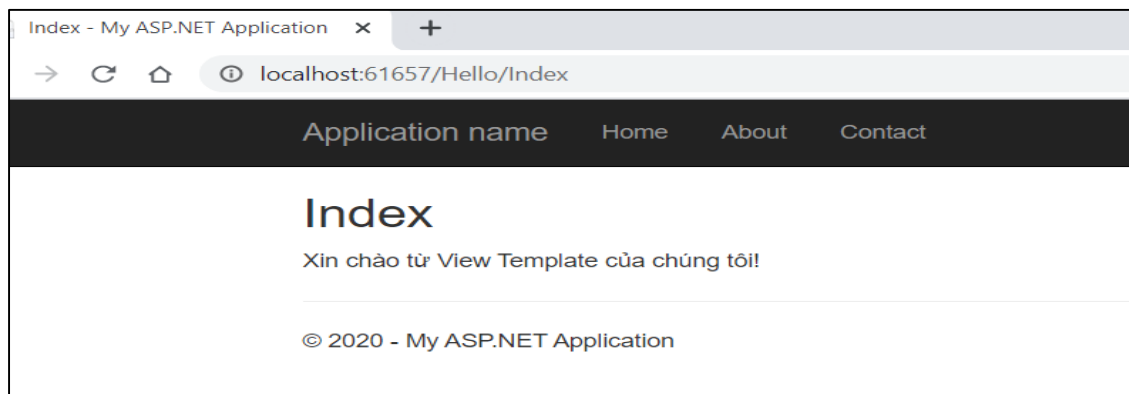


Và đoạn code Razor như sau:

```

Index.cshtml
1
2      @{
3          ViewBag.Title = "Index";
4          Layout = "~/Views/Shared/_Layout.cshtml";
5      }
6      <h2>Index</h2>
7      <p>Xin chào từ View Template của chúng tôi!</p>
  
```

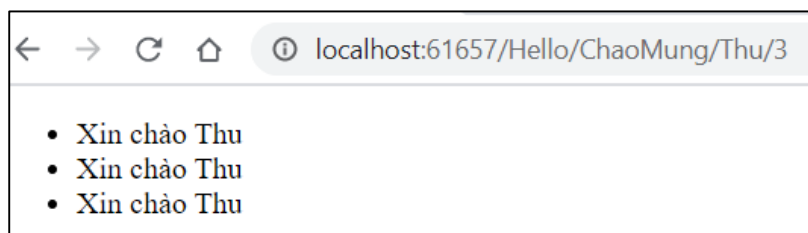
=>Chạy xem kết quả:



=> Quy trình hiển thị dữ liệu trên giao diện View như sau: Đầu tiên, một người dùng sẽ chạy đường dẫn **http://localhost:xxxx/Hello/Index**, server sẽ dò tìm và thực thi phương thức **Index()** trong tệp **HelloController.cs**. Phương thức **Index()** trả về View {return View()}. Vì vậy server sẽ thực thi tệp tin **Index.cshtml** nằm trong thư mục **Views/Hello** và hiển thị kết quả trên màn hình.

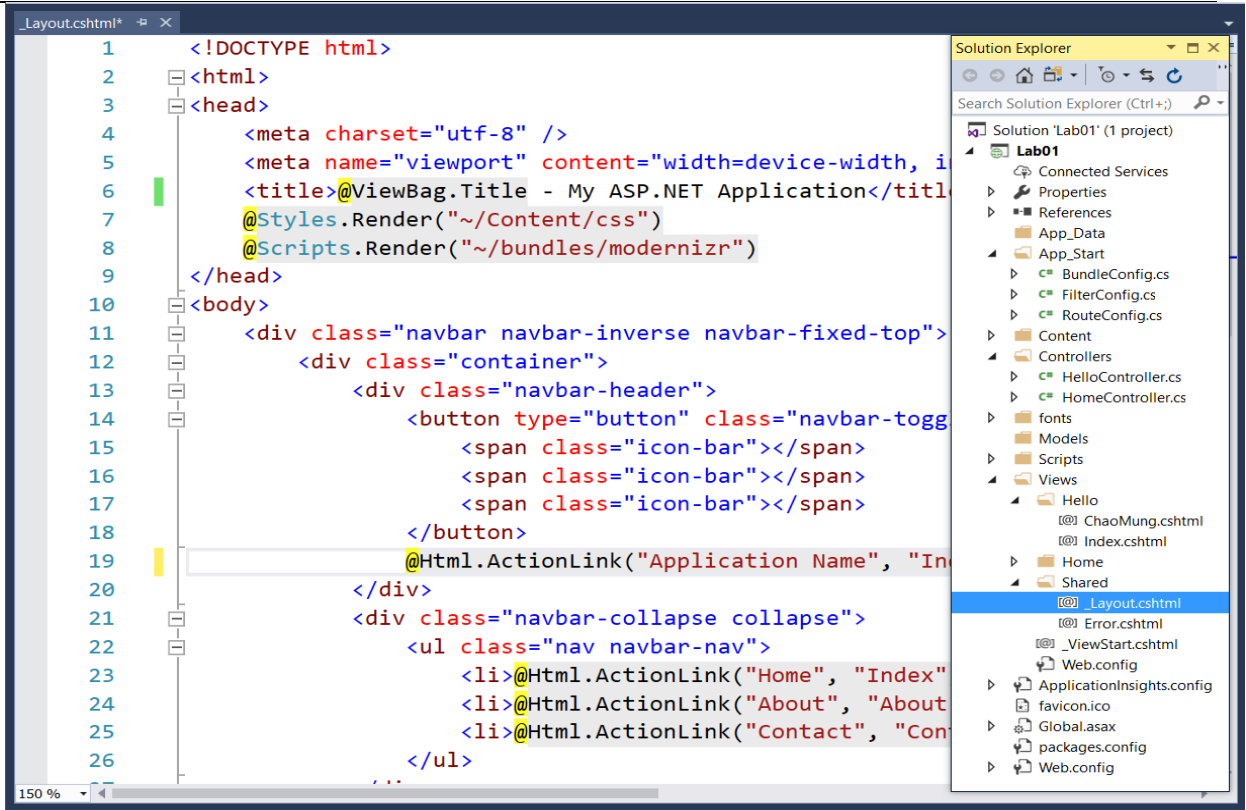
<http://localhost:61657/Hello/ChaoMung/Thu/3>

được kết quả là:



2.6 Thay đổi Layout Pages (giao diện của trang)

Vào thư mục **/Views/Shared** ở Solution Explorer và mở tệp tin **_Layout.cshtml**. Tệp tin này được gọi là **layout page** và nó nằm ở thư mục dùng chung mà các trang cùng sử dụng.



```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1">
6      <title>@ViewBag.Title - My ASP.NET Application</title>
7      @Styles.Render("~/Content/css")
8      @Scripts.Render("~/bundles/modernizr")
9  </head>
10 <body>
11     <div class="navbar navbar-inverse navbar-fixed-top">
12         <div class="container">
13             <div class="navbar-header">
14                 <button type="button" class="navbar-toggle">
15                     <span class="icon-bar"></span>
16                     <span class="icon-bar"></span>
17                     <span class="icon-bar"></span>
18                 </button>
19                 @Html.ActionLink("Application Name", "Index", "Home")
20             </div>
21             <div class="navbar-collapse collapse">
22                 <ul class="nav navbar-nav">
23                     <li>@Html.ActionLink("Home", "Index", "Home")</li>
24                     <li>@Html.ActionLink("About", "About", "Home")</li>
25                     <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
26                 </ul>

```

Các khuôn mẫu giao diện (Layout templates) cho phép chúng ta bố trí các thành phần giao diện của site trong cùng một vị trí và nó áp dụng cho tất cả các trang.

- **@RenderBody()** là một thành phần giữ chỗ để cho các trang hiển thị ở chính chỗ đó.
- **@Html.ActionLink** là cách tạo liên kết tới action trong một controller và thực thi action đó.

Cú pháp: **@Html.ActionLink (Text của link, Tên action, Tên controller)**

Ví dụ: **@Html.ActionLink("Home", "Index", "Home")**: là liên kết hiển thị trên trang web là “Home”, liên kết để thực thi action method có tên là **Index()** ở trong Controller có tên là “**Home**”

Ta sửa và thêm **ActionLink** như sau:

```
<body>
  <div class="navbar navbar-inverse navbar-fixed-top">
    <div class="container">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar">
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        @Html.ActionLink("Application Hello", "Index", "Hello", new { area = "" }, new { @class
      </div>
      <div class="navbar-collapse collapse">
        <ul class="nav navbar-nav">
          <li>@Html.ActionLink("Home", "Index", "Home")</li>
          <li>@Html.ActionLink("About", "About", "Home")</li>
          <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
          <li>@Html.ActionLink("Liên kết của tôi", "MyLink", "Home")</li>
        </ul>
      </div>
    </div>
  </div>
```

- Trong **HomeController.cs** thêm một Action method tên là **MyLink**

namespace Lab01.Controllers

```
{
    public class HomeController : Controller
    {
        .....

        public ActionResult MyLink()
        {
            ViewBag.Message = "Liên kết của tôi.";
            return View();
        }
    }
}
```

- Tạo một View có tên là **MyLink** cho action method **MyLink()** và sửa code của **MyLink.cshtml** như sau:

```
@{
    ViewBag.Title = "MyLink";
}
<h2>@ViewBag.Title.</h2>
<h3>@ViewBag.Message</h3>
<p>Đây là trang liên kết của tôi để test ActionLink trong ASP.NET MVC</p>
```

- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử kiểm tra hoạt động của Link “Liên kết của tôi”

