

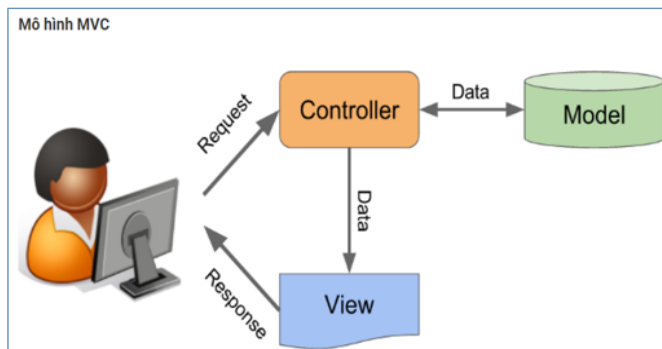
Bài 2. Tổng quan về ASP.Net MVC 5

1. GIỚI THIỆU VỀ ASP.NET MVC

GIỚI THIỆU VỀ ASP.NET MVC

❖ ASP.NET MVC là gì ?

Microsoft
ASP.net MVC =

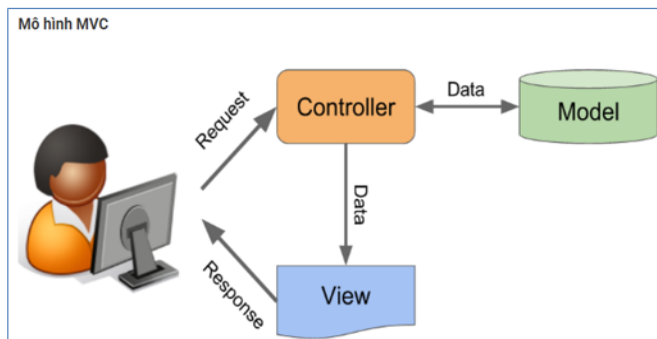


- ASP.Net MVC là một framework web được phát triển bởi [Microsoft](#), thực thi mô hình MVC.
- MVC là viết tắt của cụm từ “**Model-View-Controller**”.
- MVC là một **mẫu kiến trúc phần mềm**.
- MVC chia thành ba phần được kết nối với nhau và mỗi thành phần đều có một **nhiệm vụ riêng** của nó và **độc lập** với các thành phần khác.

GIỚI THIỆU VỀ ASP.NET MVC

❖ ASP.NET MVC là gì ?

Microsoft
ASP.net MVC =



- **Model (dữ liệu):** Quản lý, xử lý các dữ liệu.
- **View (giao diện):** Hiển thị dữ liệu cho người dùng.
- **Controller (bộ điều khiển):** Điều khiển sự tương tác của hai thành phần **Model** và **View**.

GIỚI THIỆU VỀ ASP.NET MVC

❖ Mô hình MVC khác mô hình 3 tầng

Mô hình 3 tầng gồm:

- Tầng 1 là tầng truy cập dữ liệu
 - Tầng 2 là tầng xử lý logic
 - Tầng 3 là tầng giao diện => **V**
- | => **M**

GIỚI THIỆU VỀ ASP.NET MVC

❖ So sánh ASP.NET Web Forms và ASP.NET MVC

ASP.NET WEB FORMS	ASP.NET MVC
Sử dụng mô hình lập trình sự kiện.	Sử dụng mô hình MVC trong xử lý các yêu cầu
Mỗi trang (.aspx) đều có tập tin mã lệnh điều khiển tương ứng (aspx.cs), không tách rời.	Mã lệnh được phân chia rõ ràng theo mô hình MVC, giúp cho việc nâng cấp, bảo trì được dễ dàng, thuận tiện.
Tất cả thông tin trạng thái đều được lưu giữ trong Viewstate. Do đó, dung lượng Web Forms thường khá lớn.	Không lưu giữ thông tin trạng thái, do đó dung lượng nhẹ, tốc độ ứng dụng web MVC nhanh hơn Web Form
Không cần nhiều kiến thức về HTML, CSS và Javascript do đã có sẵn Toolbox	Nắm vững kiến thức về HTML, CSS và Javascript để biết triển khai front-end, xây dựng các xử lý và kết hợp các thành phần với back-end cho hiệu quả.

GIỚI THIỆU VỀ ASP.NET MVC

❖ So sánh ASP.NET Web Forms và ASP.NET MVC

Tóm lại:

ASP.NET WEB FORMS	ASP.NET MVC
Do không cần biết chi tiết về HTML, CSS và Javascript nên ASP WebForms phù hợp để triển khai nhanh các ứng dụng web có ngân sách giới hạn, các ứng dụng web nội bộ.	Với các ứng dụng web lớn, cần nhiều nhân sự, nhiều xử lý ở các mức front-end và back-end, cần sự phối hợp hiệu quả giữa các xử lý, linh động có khả năng dễ dàng thay đổi cho phù hợp với thực tế, ta nên sử dụng ASP.NET MVC để việc quản lý cũng như bảo trì, nâng cấp sau này sẽ dễ dàng và thuận tiện hơn nhiều.

GIỚI THIỆU VỀ ASP.NET MVC

❖ Lịch sử phát triển ASP.Net MVC



- Tháng 06/2016, **ASP.NET Core** ra đời:
 - Chạy trên các HĐH Windows, Linux và OS.
 - Phiên bản mã nguồn mở (open source)

2. XÂY DỰNG ỨNG DỤNG WEB

2.1 Tạo mới Project ASP.Net MVC 5

- ✓ Khởi động Visual Studio 2019
- ✓ Tạo mới 1 project: Create a new project
 - Chọn ASP.NET Web Application khung bên phải, chọn Next
 - Đặt tên cho project là "**Lab01**" rồi click Create.

Configure your new project

ASP.NET Web Application (.NET Framework) C# Windows Cloud Web

Project name
Lab01

Location
E:\Demo-ASP-MVC\

Solution name ⓘ
Lab01

☐ Place solution and project in the same directory

Framework
.NET Framework 4.7.2

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

Create a new ASP.NET Web Application

Empty
An empty project template for creating ASP.NET applications. This template does not have any content in it.

Web Forms
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

MVC
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Web API
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

Authentication
No Authentication
[Change](#)

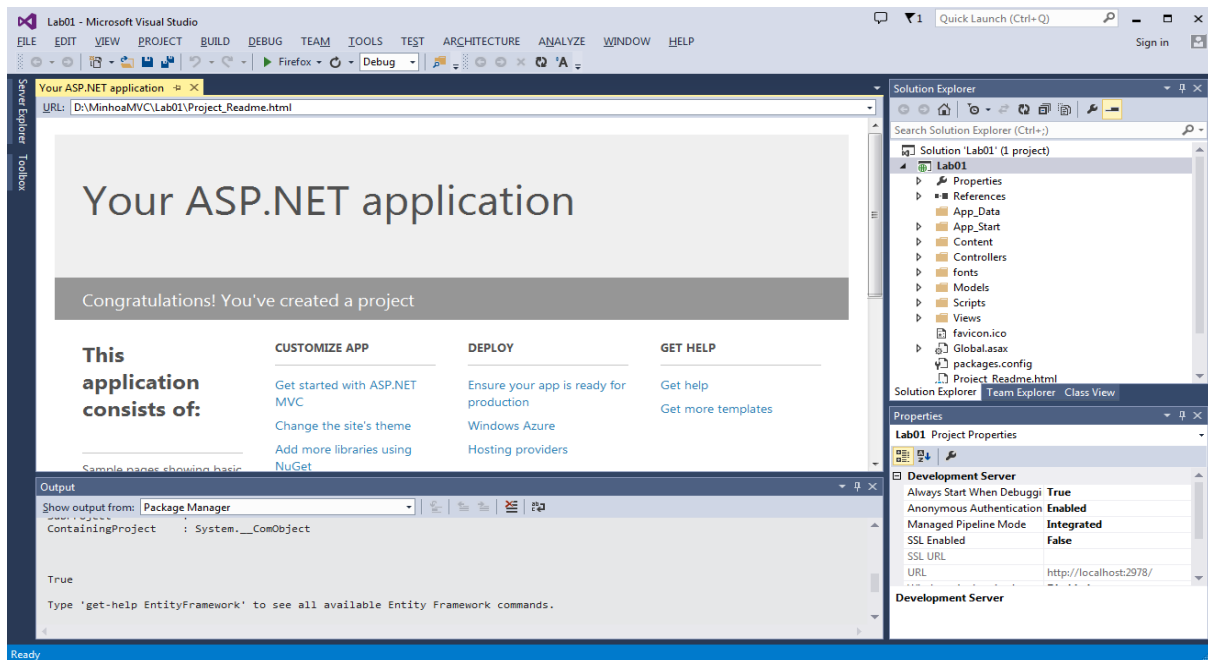
Add folders & core references
☐ Web Forms
☒ MVC
☐ Web API

Advanced
☒ Configure for HTTPS
☐ Docker support
(Requires [Docker Desktop](#))
☐ Also create a project for unit tests

[Back](#) [Create](#)

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.

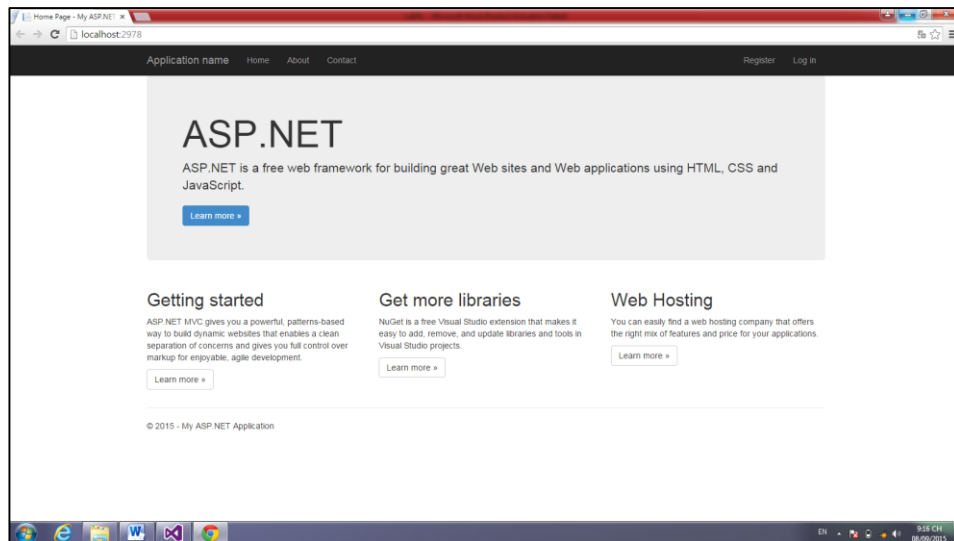


2.2 Các thành phần trong Project ASP.Net MVC

- **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.

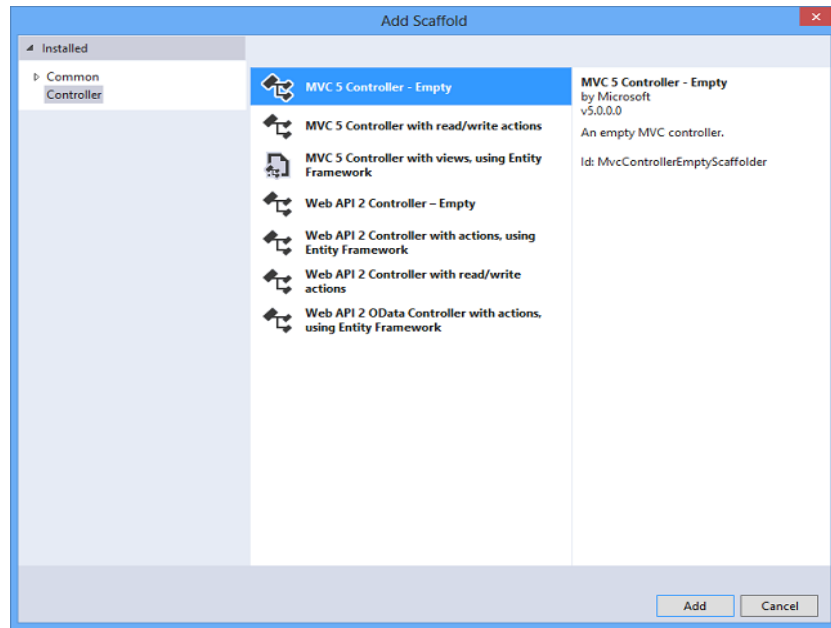


2.3 Tạo mới một Controller

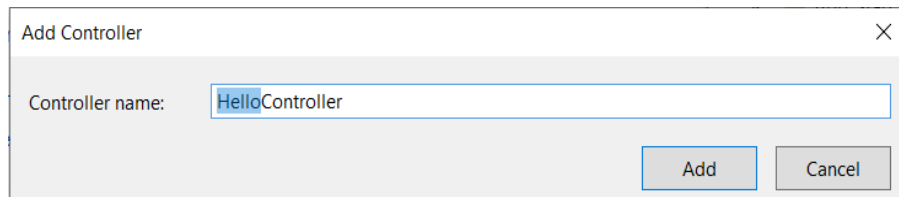
Bắt đầu tạo ra một lớp controller:

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

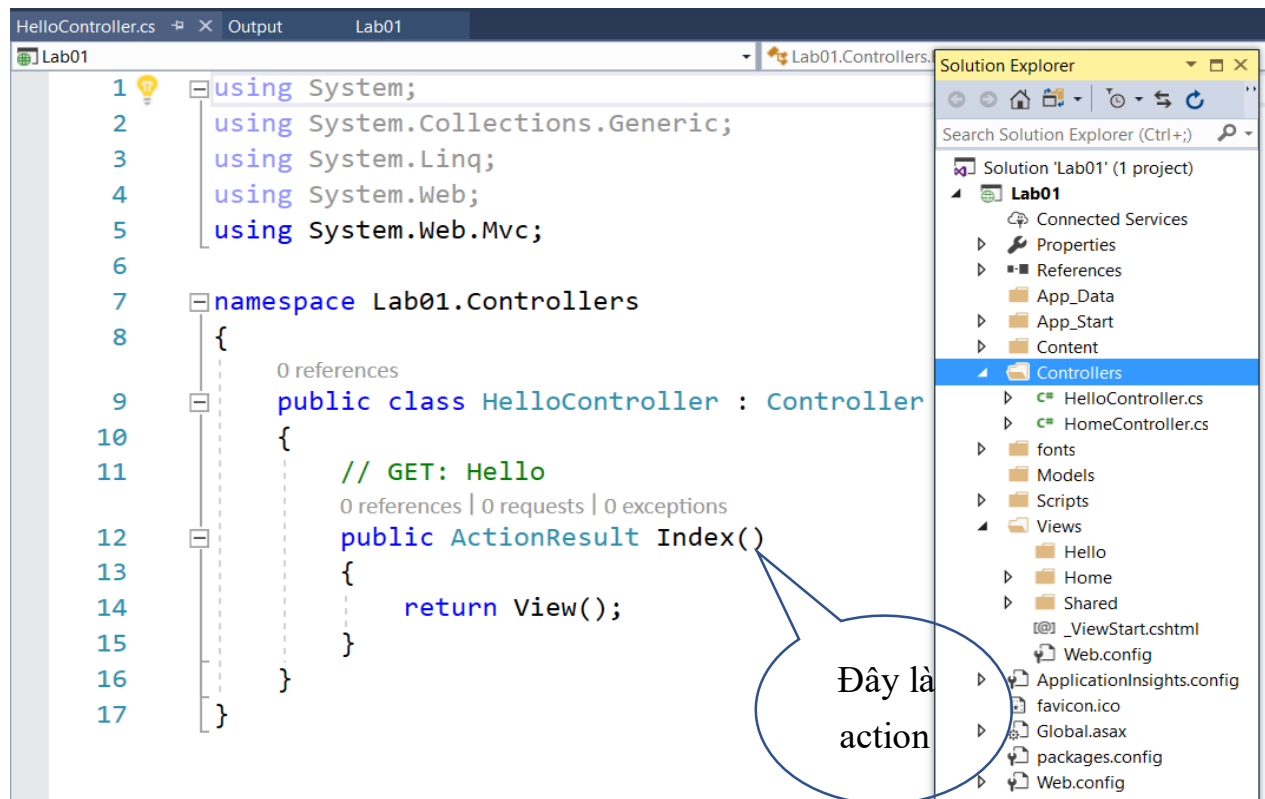
Trong cửa sổ Add Scaffold, click MVC 5 Controller - Empty, rồi click Add.



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**.
- **ActionResult** là lớp trừu tượng, 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.



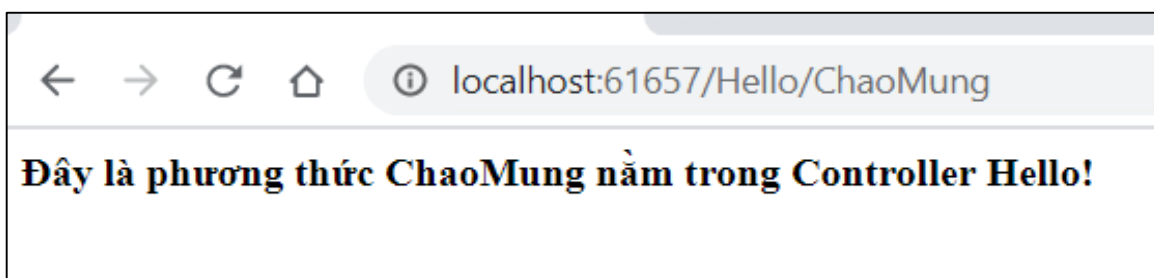
Ở trình duyệt, ta thử chạy 2 địa chỉ:

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

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

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

Kết quả như hình sau:



2.4 Điều hướng (định tuyến) 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:

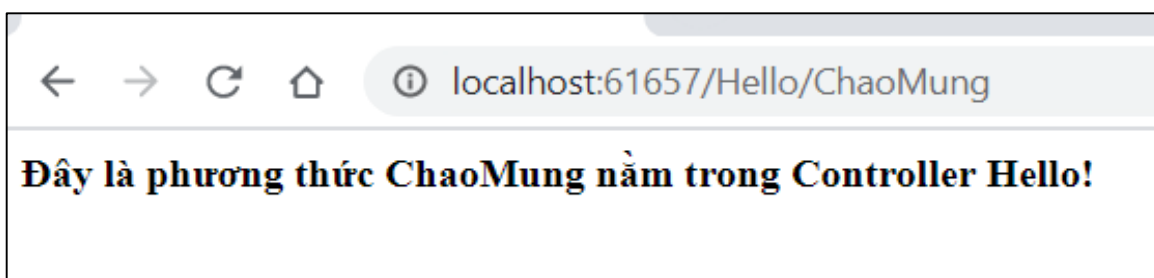
```
public class RouteConfig
{
    1 reference | 0 exceptions
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        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*), trong *HelloController.cs* ta tạo các phương thức sau:

+ Tạo phương thức *MaNV* như sau:

```
public string MaNV(int id)
{
    return "Mã nhân viên là: " + id;
}
```

Chạy ứng dụng:



Ví dụ trên thì thành phần tham số (*Parameters*) theo cấu trúc mặc định vẫn chưa dùng, tham số *id* được dùng ở đây chỉ là tham số theo *query strings*. Dấu ? (question mark) trong URL là một phần ngăn cách để chỉ ra phía sau đó là

query strings.

Chạy ứng dụng với tham số theo cấu trúc mặc định:

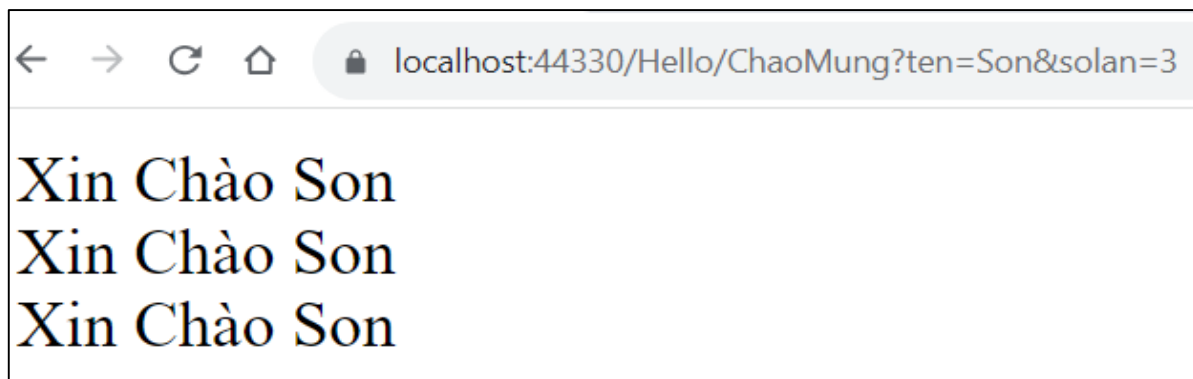
```
url: "{controller}/{action}/{id}"
```



+ Ta sửa lại phương thức **ChaoMung** như sau:

```
public string ChaoMung(string ten, int solan)
{
    string str = "";
    for (int i = 0; i < solan; i++)
    {
        str = str + "Xin Chào " + ten + "<br/>";
    }
    return str;
}
```

Chạy ứng dụng:

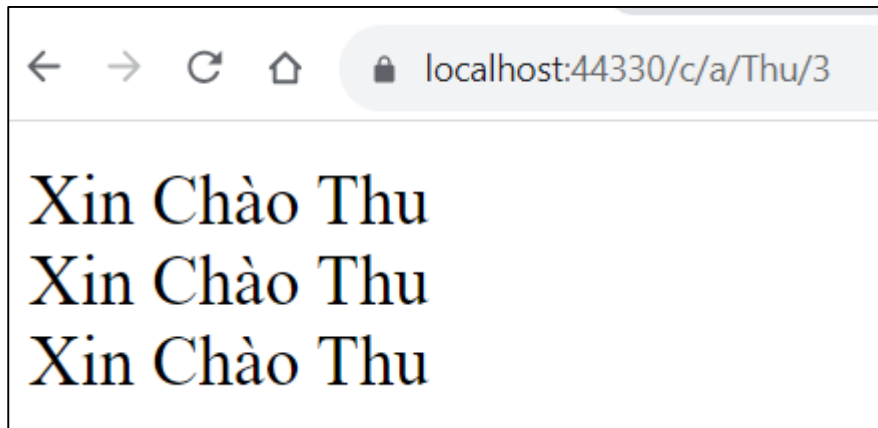


Ta có thể tự định nghĩa thêm cấu trúc định tuyến trong file **RouteConfig.cs** như sau:

```
public class RouteConfig
{
    1 reference
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
        routes.MapRoute(
            name: "defaults",
            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: "Hello1",
            url: "{c}/{a}/{ten}/{solan}",
            //defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
            defaults: new { controller = "Hello", action = "ChaoMung" }
        );
    }
}
```

Chạy ứng dụng:



Ta thấy URL ngắn gọn và tham số không phải viết rõ ràng.

Nhận xét: mỗi khi cấu hình một **action method**, ta phải vào file **RouteConfig.cs** để chỉnh sửa lại và khi đổi tên **controller/method** thì các cấu hình này không thay đổi theo. Để khắc phục nhược điểm này, ta có thể sử dụng định tuyến theo thuộc tính (attribute routing) để xác định đường đi.

Trong file **RouteConfig.cs** ta thêm lệnh sau:

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

        routes.MapRoute(
            name: "defaults",
```

Tạo thêm phương thức **ChaoMungV2**, thêm định tuyến ngay ở phía trên phương thức:

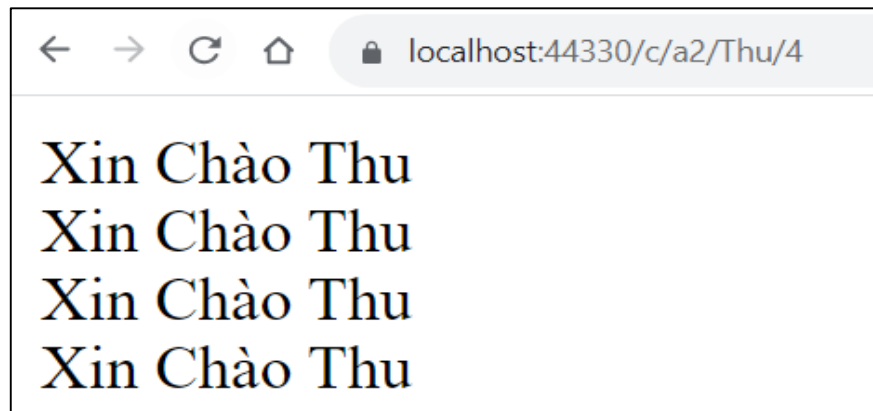
```
public string ChaoMung(string ten, int solan)
{
    string str = "";
    for (int i = 0; i < solan; i++)
    {
        str = str + "Xin Chào " + ten + "<br/>";
    }
    return str;
}
```

```
[Route("{c}/{a2}/{ten}/{solan}")]
```

0 references

```
public string ChaoMungV2(string ten, int solan)
{
    string str = "";
    for (int i = 0; i < solan; i++)
    {
        str = str + "Xin Chào " + ten + "<br/>";
    }
    return str;
}
```

Chạy ứng dụng:



Có thể gán giá trị mặc định cho tham số:

```
[Route("{c}/{a2}/{ten=Minh}/{solan=3}")]
0 references
public string ChaoMungV2(string ten, int solan)
{
    string str = "";
    for (int i = 0; i < solan; i++)
    {
        str = str + "Xin Chào " + ten + "<br/>";
    }
    return str;
}
```

Chạy ứng dụng:



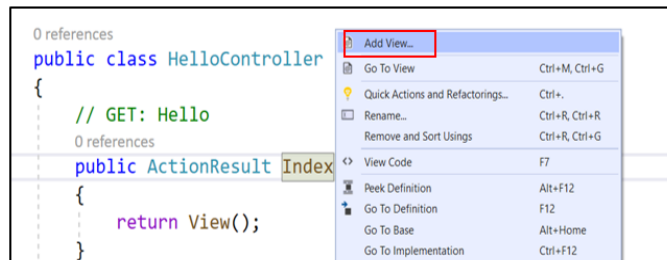
Đối với các ứng dụng MVC, các định tuyến mặc định sẽ hoạt động tốt hầu hết các trường hợp. Tuy nhiên, tùy vào các nhu cầu cụ thể, ta có thể thay đổi các định tuyến để phù hợp với các nhu cầu.

2.5 Tạo mới một View

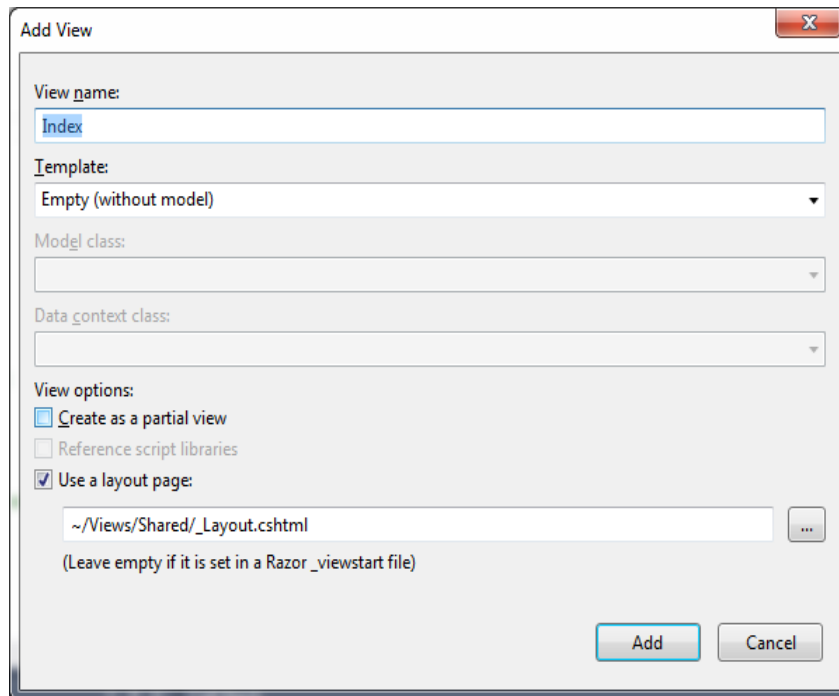
✓ Ta tiếp tục cập nhật lớp **HelloController** sử dụng với hiển thị một file khuôn mẫu giao diện (View Template File) để hiểu rõ việc tạo ra một HTML trả về hiển thị phía client (browser).

✓ 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:

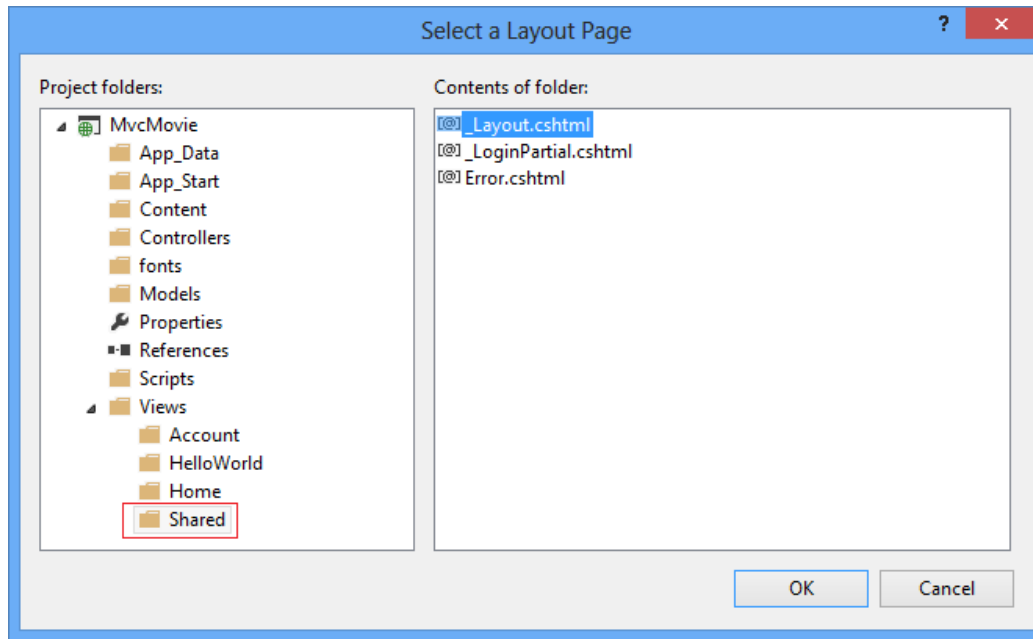
Right click lên tên phương thức, chọn **Add View**



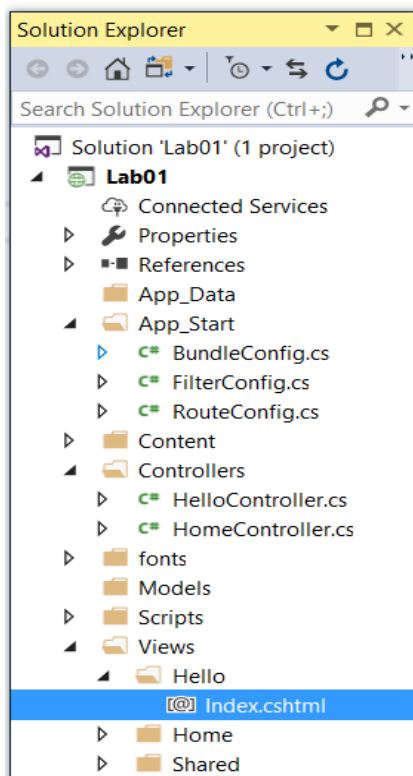
Tại cửa sổ **Add View**, để tên view mặc định là **Index**, Chọn Layout tại mục **Use a layout page**, rồi click OK.



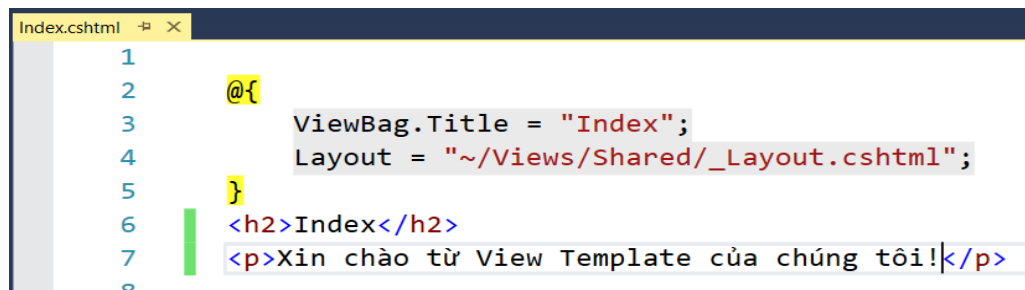
Tại cửa sổ **Select a Layout Page**, chọn mặc định là **View/Shared/_Layout.cshtml** rồi click OK.



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



Và đoạn code Razor như sau:

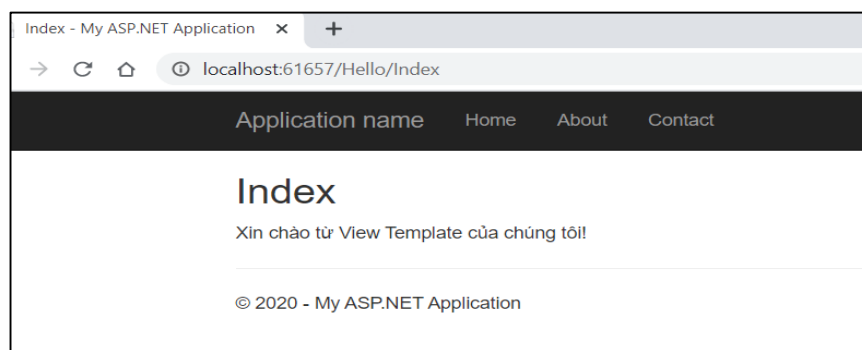


```
1
2 @{
3     ViewBag.Title = "Index";
4     Layout = "~/Views/Shared/_Layout.cshtml";
5 }
6 <h2>Index</h2>
7 <p>Xin chào từ View Template của chúng tôi!</p>
```

=>Chạy xem kết quả:

Cách 1: kích chuột phải vào cửa sổ code của view rồi chọn **View in Browser** để xem view trên trình duyệt

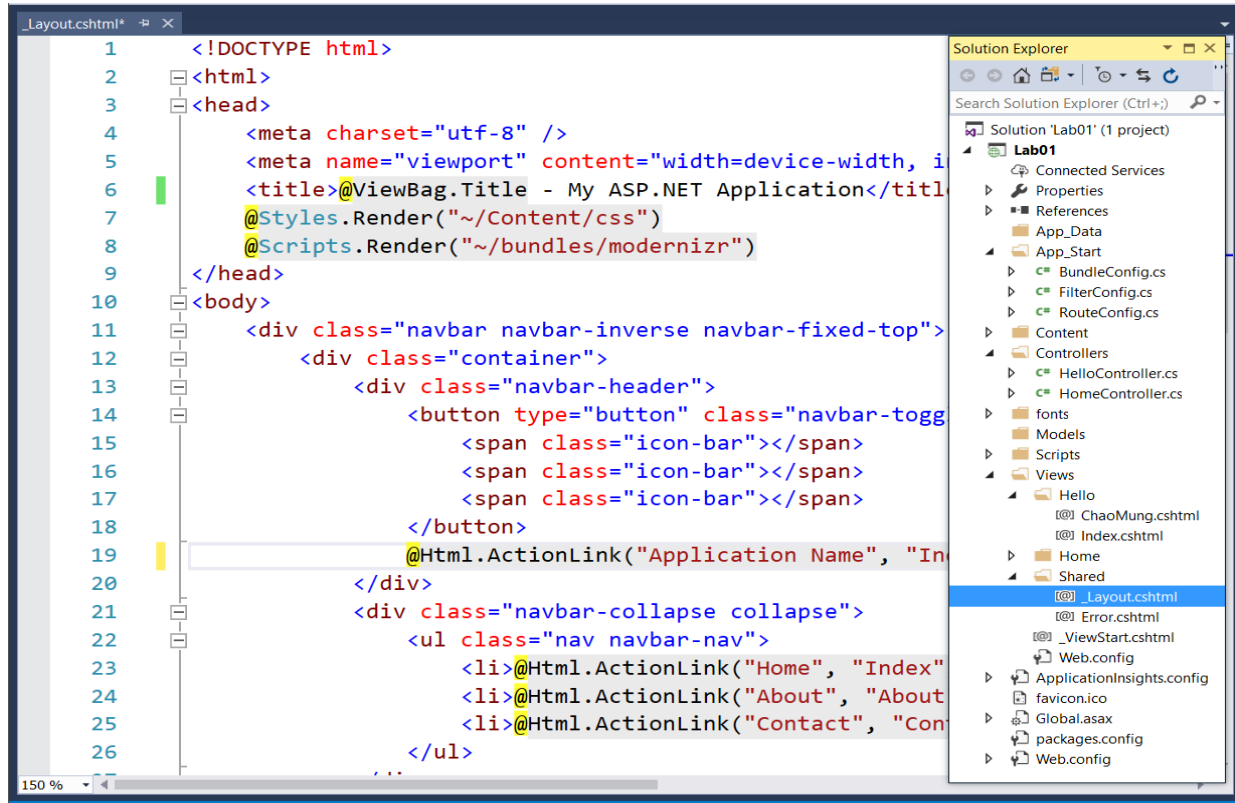
Cách 2: Ấn phím F5 (hoặc Ctrl+F5) và nhập URL với action method tương ứng



=> 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.

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.



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-collapse">
          <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 = "application-hello" })
      </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

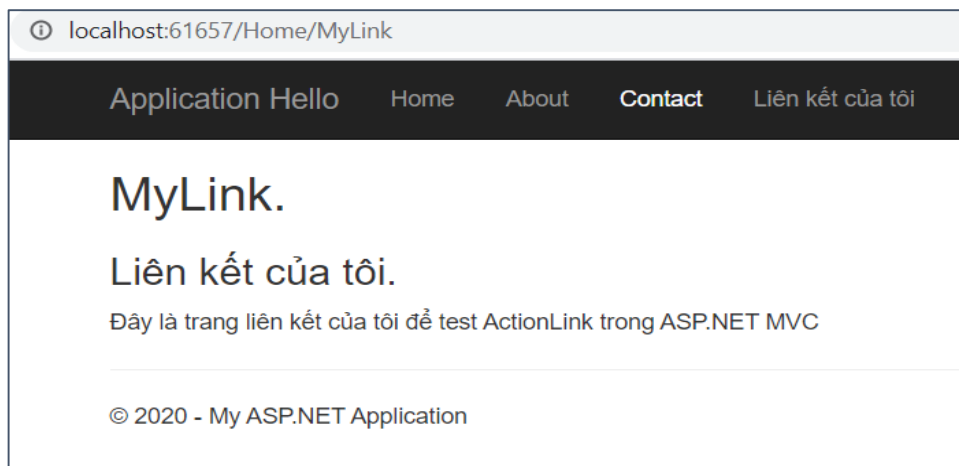
```
{
  public class HomeController : Controller
  {
    .....

    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**”



7. Lớp trừu tượng ActionResult

ActionResult là lớp trừu tượng, dùng để trả về dữ liệu cho client ở các định dạng khác nhau.

Để minh họa các kiểu của **ActionResult**, chúng ta xét các ví dụ sau:

Thêm Action Method **Display** trong **HomeController.cs**

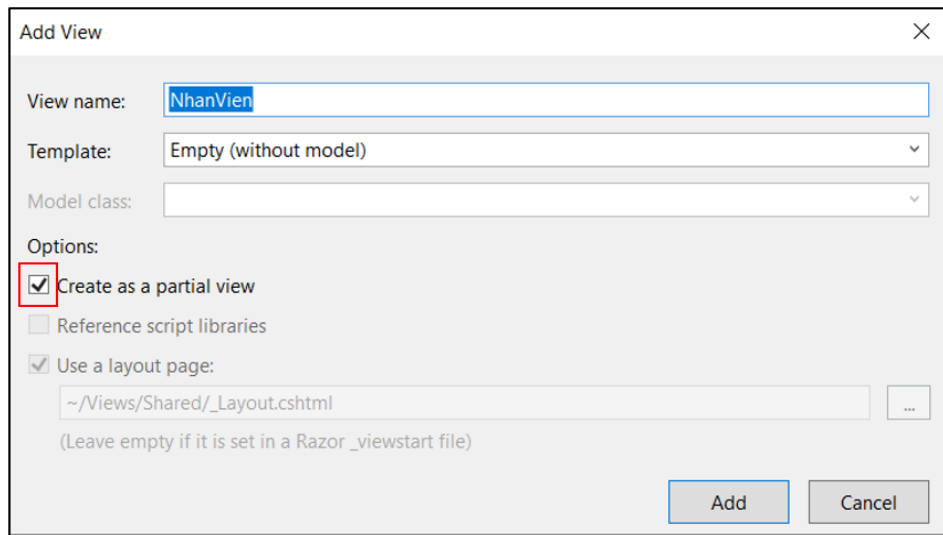
- ❖ Trả về **PartialView**: là chế độ xem đặc biệt hiển thị 1 phần View, PartialView có thể được sử dụng lại ở nhiều View, giúp giảm trùng lặp mã.

```
public ActionResult Display()
{
    return PartialView("NhanVien");
}
```

- Thêm Link tới Action Method **Display** trong **_Layout.cshtml**

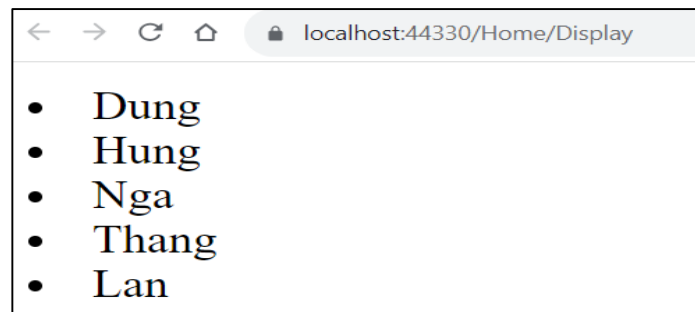
```
<div class="navbar-collapse collapse">
  <ul class="nav navbar-nav">
    <li>@Html.ActionLink("Home", "Index", "Home")</li>
    <li>@Html.ActionLink("About", "About", "Home")</li>
    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
    <li>@Html.ActionLink("Liên kết của tôi", "MyLink", "Home")</li>
    <li>@Html.ActionLink("Display", "Display", "Home")</li>
  </ul>
</div>
```

- Tạo **PartialView("NhanVien")** //Click chuột phải vào tên Action **Display**, chọn **Add View**



```
@{
    string[] names = { "Dung", "Hung", "Nga", "Thang", "Lan" };
    foreach (var item in names)
    { <li> @item </li> }
}
```

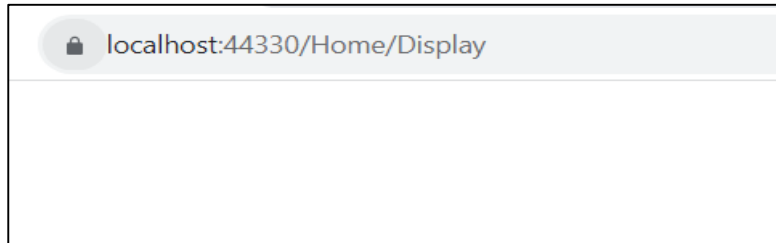
- Chạy chương trình:



❖ Trả về một View trống

```
public ActionResult Display()
{
    //return PartialView("NhanVien");
    return new EmptyResult();
}
```

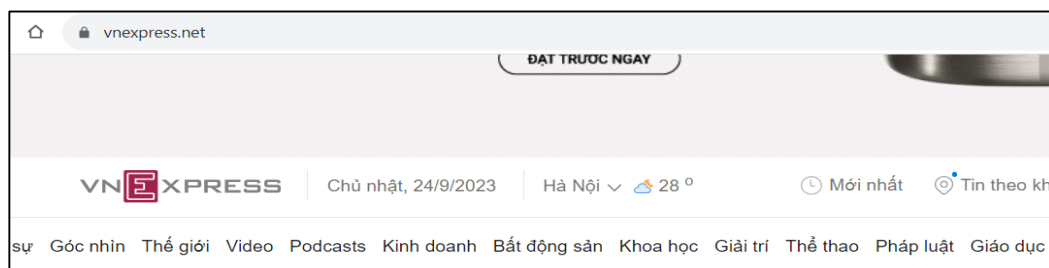
- Chạy chương trình:



❖ Chuyển hướng tới một URL

```
public ActionResult Display()
{
    //return PartialView("NhanVien");
    //return new EmptyResult();
    return Redirect("http://vnexpress.net");
}
```

- Chạy chương trình:



❖ Chuyển hướng tới một Action Method khác

```
public ActionResult Display()
{
    //return PartialView("NhanVien");
    //return new EmptyResult();
    //return Redirect("http://vnexpress.net");
    return RedirectToAction("ChaoMung", "Hello");
}
```

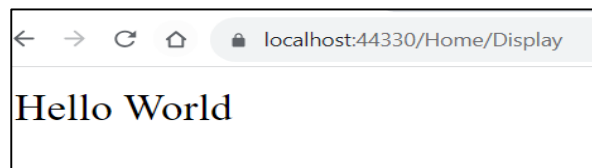
- Chạy chương trình:



❖ Trả về luồng dữ liệu mà không yêu cầu 1 View


```
public ActionResult Display()
{
    //return PartialView("NhanVien");
    //return new EmptyResult();
    //return Redirect("http://vnexpress.net");
    //return RedirectToAction("ChaoMung", "Hello");
    return Content("Hello World");
}
```

- Chạy chương trình:



❖ Trả về một file cho Client

```
public ActionResult Display()
{
    //return PartialView("NhanVien");
    //return new EmptyResult();
    //return Redirect("http://vnexpress.net");
    //return RedirectToAction("ChaoMung", "Hello");
    // return Content("Hello World");
    return new FilePathResult(@"d:\Baitap.pdf", "application/pdf");
}
```

- Chạy chương trình:

