Lab 01:

TỔNG QUAN VỀ ASP.NET MVC 5

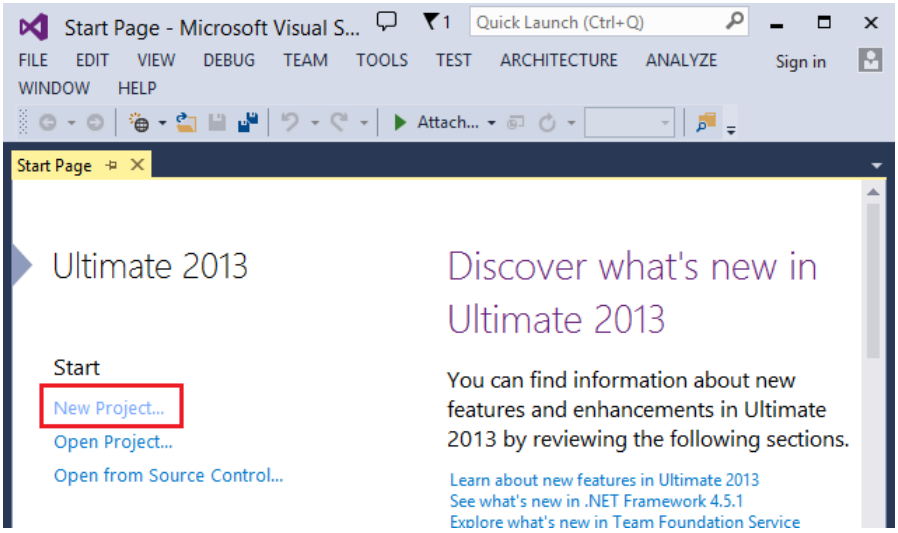
A. MỤC TIÊU:

- ✓ Hướng dẫn sinh viên làm quen với Visual Studio 2013 với ứng dụng Web ASP.Net MVC 5.
- ✓ Tạo mới Project MSP.net MVC 5, biên dịch project, thực thi project trên nhiều kích thước màn hình.
- ✓ Giới thiệu về mô hình MVC là gì?
- ✓ Tạo thêm vào project controller,
- ✓ Các cách truyền tham số trong Controller
- ✓ Tạo View
- ✓ Tạo model

B. NỘI DUNG:

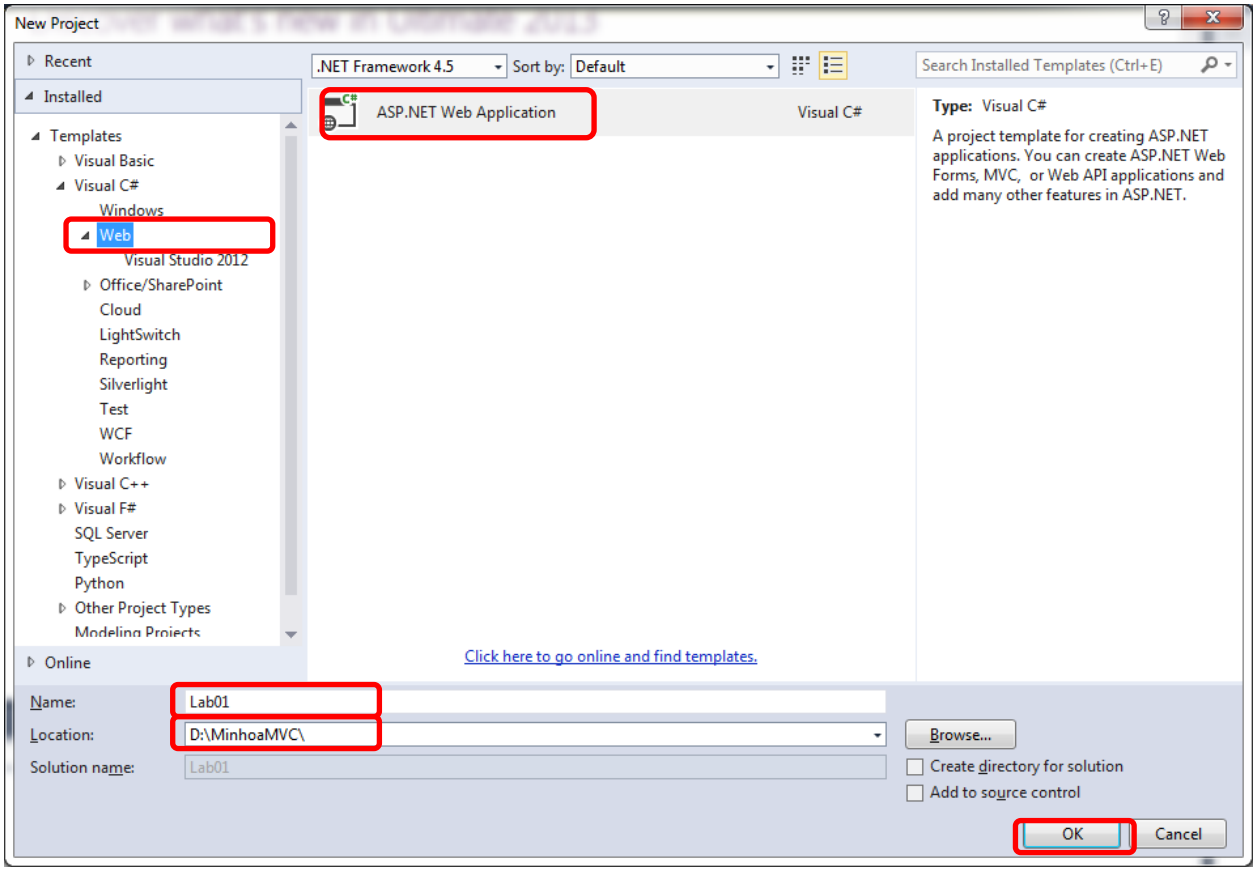
Bài tập 1: Tạo mới Project ASP.Net MVC 5

- ✓ Khởi động Visual Studio 2013.



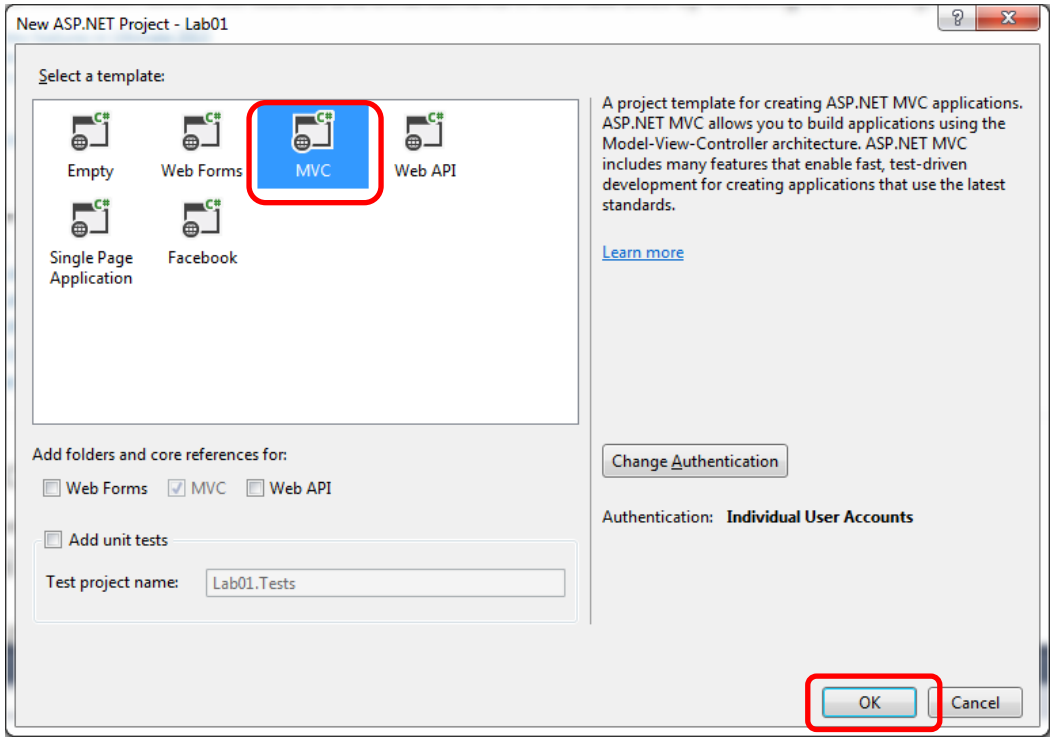
Hình 1: Màn hình khởi động Visual Studio 2013

- ✓ 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 "MvcMovie" rồi click **OK**.



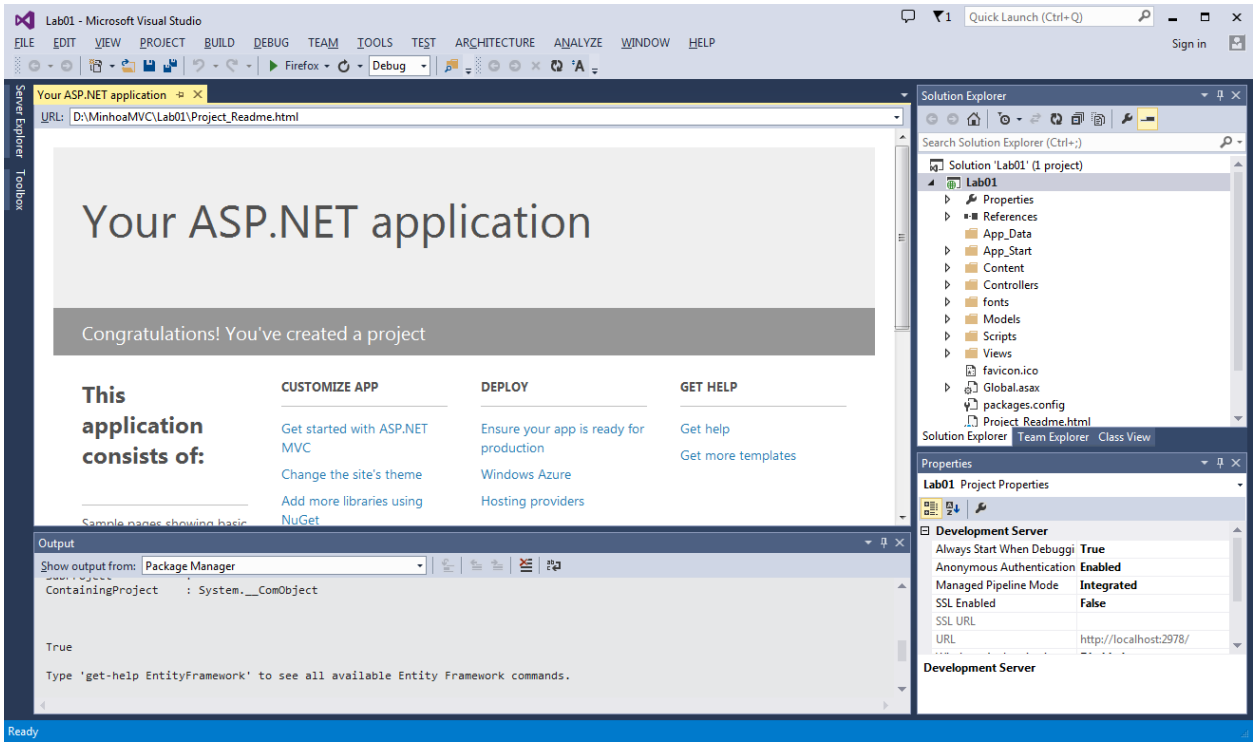
Hình 2: Màn hình tạo mới Project Website ASP.Net

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



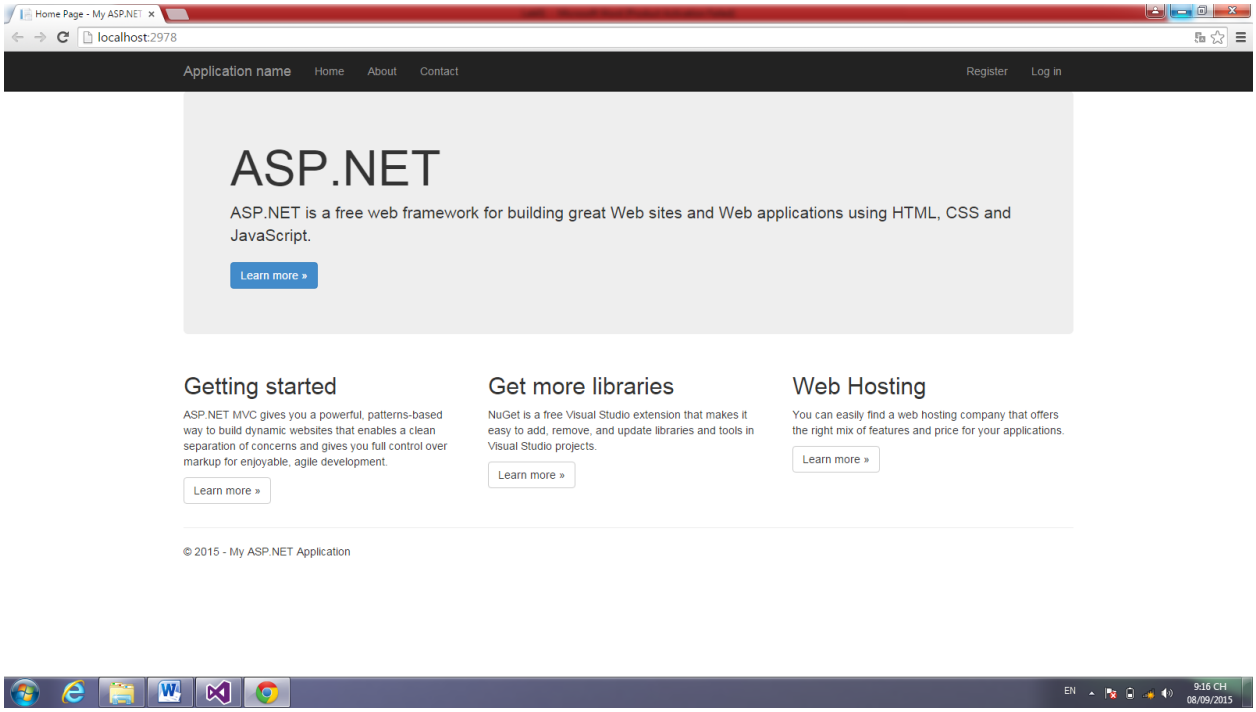
Hình 3: Màn hình chọn loại Project Website là MVC

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.



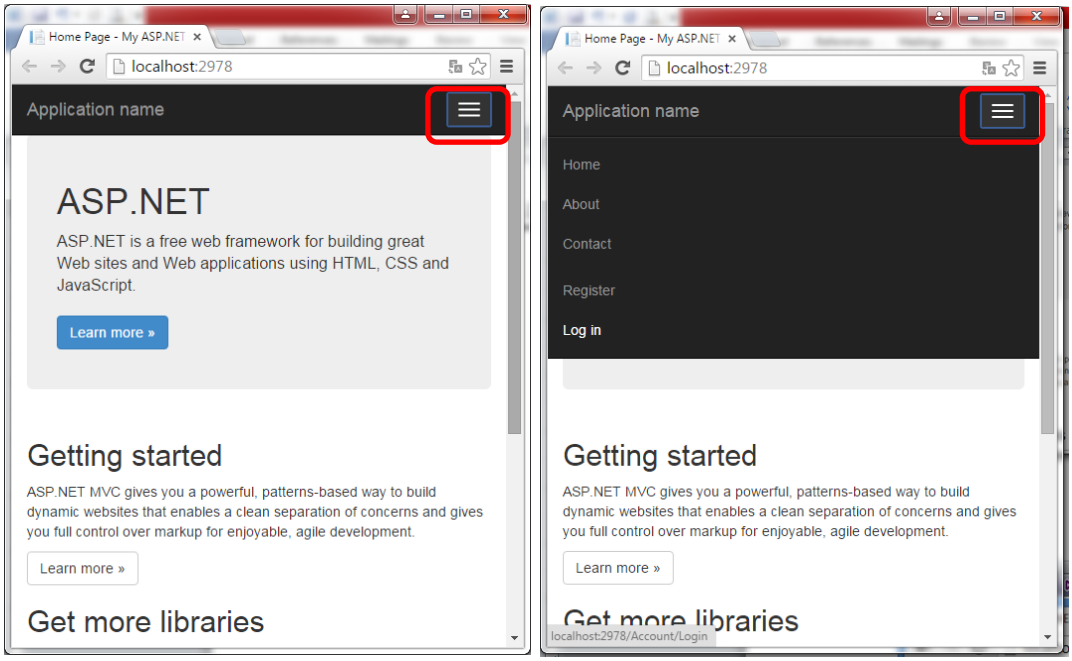
Hình 4: Màn hình Website MVC đã tạo

✓ Nhấn F5 để bắt đầu run với chế độ Debug (chế độ chạy có gỡ lỗi): 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.



Hình 5: Màn hình kết quả Website MVC

✓ Nếu ta thu nhỏ trình duyệt sẽ thấy giao diện có thay đổi, phù hợp với kích thước màn hình:



Hình 6: Màn hình kết quả Website MVC khi thay đổi kích thước màn hình

- ✓ Ứng dụng của chúng ta lúc này đã tạo sẵn chức năng Đăng Nhập (Login) và Đăng Ký (Register)….

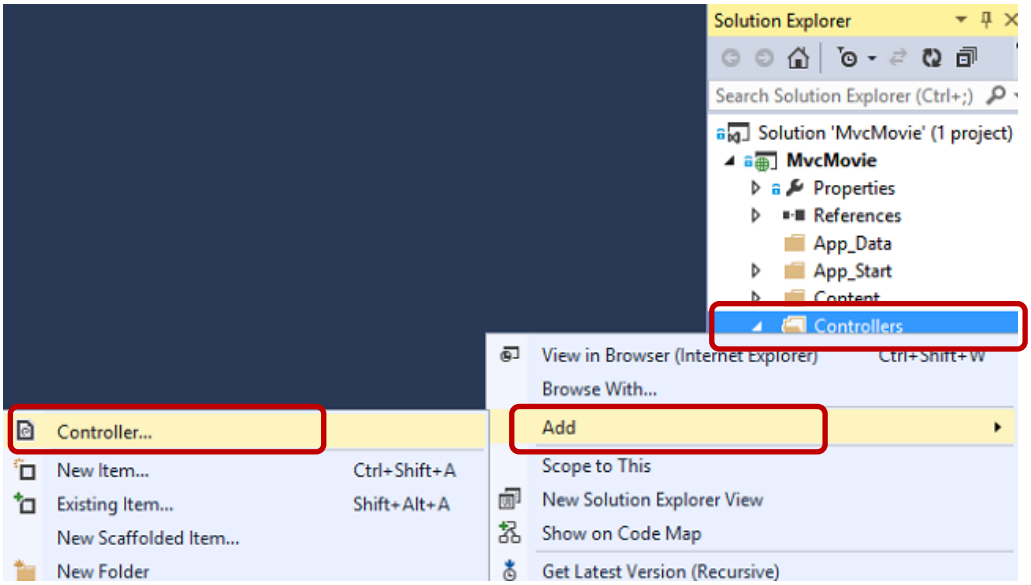
Bài tập 2: Tạo mới một Controller

MVC (*Model-View-Controller)* Là một mô hình phát triển ứng dụng có kiến trúc tốt, có khả năng kiểm thử tốt và dễ dàng vận hành và bảo trì. Một ứng dụng theo kiến trúc MVC gồm:

- **Models:** Là các lớp chức năng thể hiện cho việc xử lý dữ liệu và các vấn đề liên quan đến nghiệp vụ và các quy tắc ràng buộc dữ liệu. Đồng thời, nó còn có nhiệm vụ lưu trữ thông tin, trạng thái của các đối tượng vào CSDL.
- **Views:** Là thành phần chịu trách nhiệm hiển thị các giao diện dưới dạng html theo cơ chế phát sinh động của ứng dụng. Các thông tin cần hiển thị sẽ được lấy từ Models thông qua các Controllers.
- **Controllers:** Là các lớp thực hiện việc điều quản các yêu cầu từ trình duyệt, nhận dữ liệu từ Models và xác định việc hiển thị lên các Views.

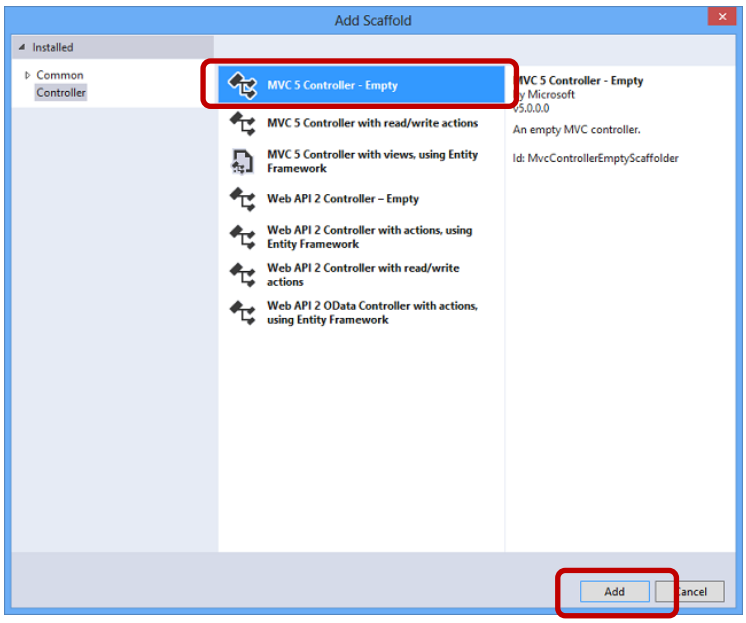
Bắt đầu tạo ra một lớp controller:

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



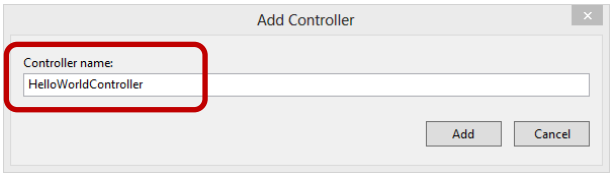
Hình 7: Màn hình tạo mới Controller

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



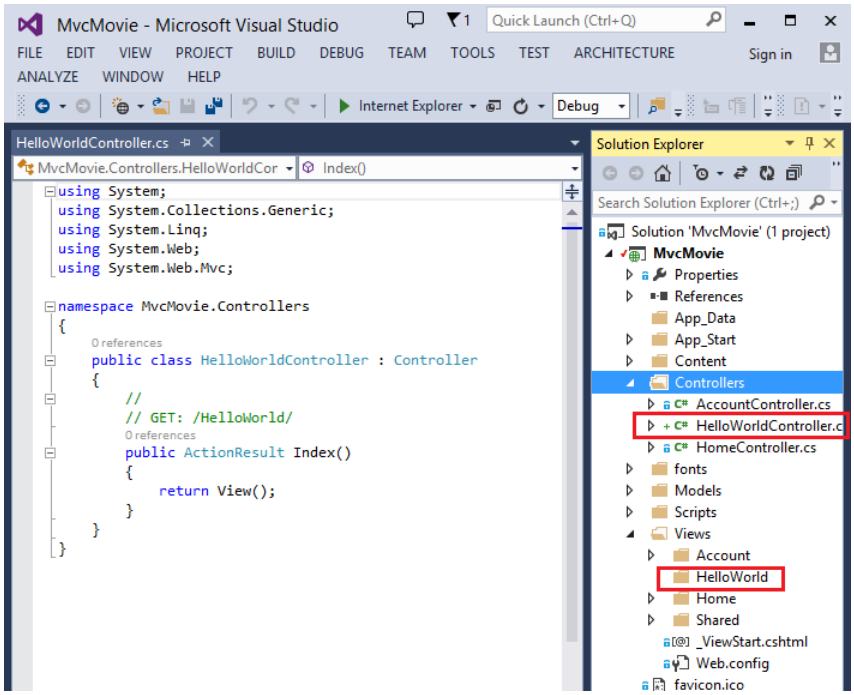
Hình 8: Màn hình chọn Controller cần tạo là rỗng

- Đặt tên cho controller mới tạo là "HelloWorldController" rồi click **Add**.



Hình 9: Màn hình đặt tên Controller

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



Hình 10: Màn hình kết quả Controller đã tạo

Hãy thay nội dung đoạn code như bên dưới.

```
public class HelloWorldController : Controller
{
    //
    // GET: /HelloWorld/
    public String Index()
    {
        return "This is my <b> default </b> action...";
    }
    // GET: /HelloWorld/Wellcome
    public String Wellcome()
    {
        return "this is the wellcome to action method ...";
    }
}
```

Hình 10: Màn hình Code minh họa cho Controller

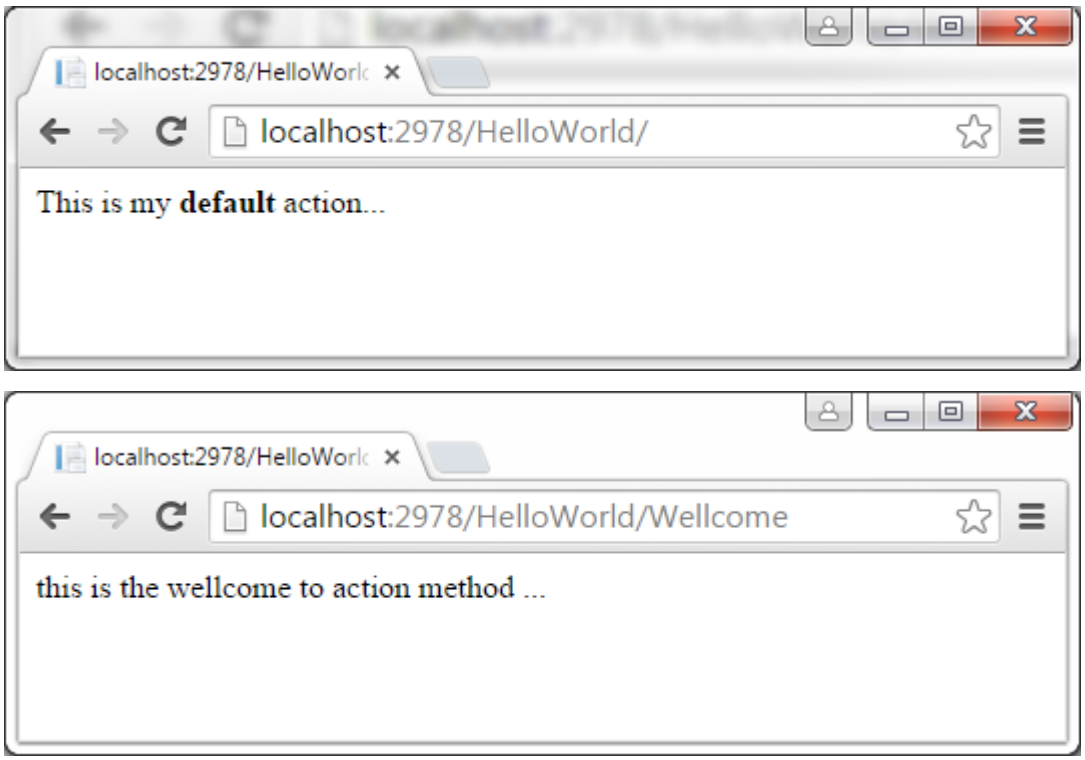
Các phương thức trong controller sẽ trả về một chuỗi theo dạng HTML. Controller có tên HelloWorldController có một phương thức với tên là Index.

Thử chạy thử xem bằng cách nhấn F5. Trong thanh địa chỉ ở trình duyệt ta gõ thêm " ví dụ là:

http://localhost:2978/HelloWorld/

http://localhost:2978/HelloWorld/wellcome

Kết quả như hình sau:



Hình 11: Màn hình kết quả chương trình sau khi tạo Controller

Điều hướng hiển thị:

ASP.NET MVC sẽ gọi các controller khác nhau cùng với các phương thức tương ứng, điều này phục 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 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 }
    );
}
```

Hình 12: Màn hình code điều hướng trong tập tin *App_Start/RouteConfig.cs*

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".

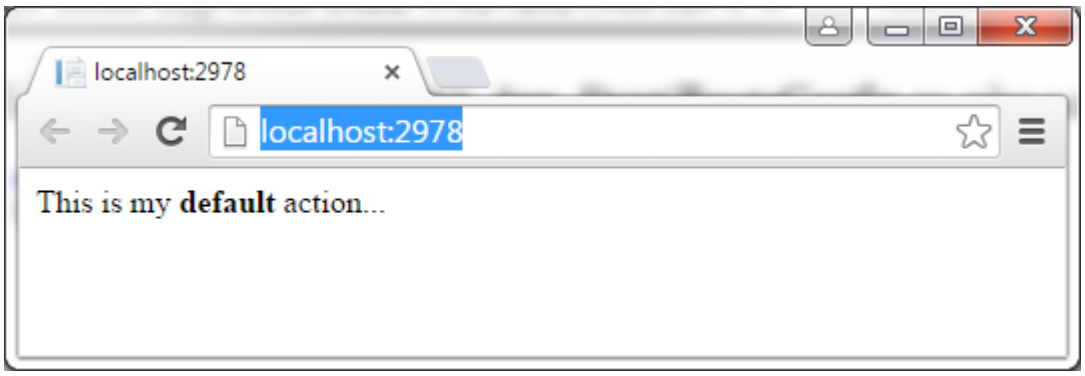
Trong đó, phần đầu của URL để xác định controller nào. Như vậy, */HelloWorld* sẽ ánh xạ đến lớp HelloWorldController.

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

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

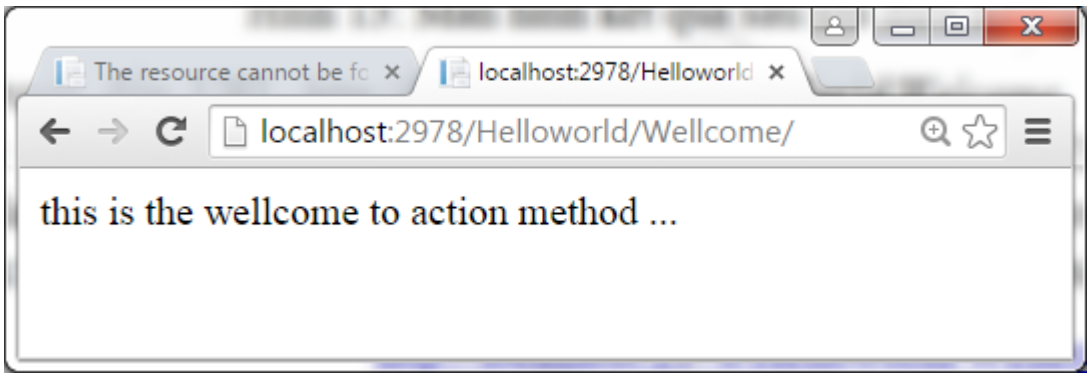
Ví dụ điều chỉnh code trong *App_Start/RouteConfig.cs như sau:*

```
public class RouteConfig
{
    1 reference
    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 = "HelloWorld", action = "Index", id = UrlParameter.Optional }
        );
    }
}
```



Hình 13: Màn hình kết quả sau khi điều hướng Controller

Duyệt đến URL http://localhost:2978/Helloworld/Wellcome/. Phương thức Welcome chạy và trả về là một chuỗi "This is the Welcome action method...". Mặc nhiên MVC đang ánh xạ tới /[Controller]/[ActionName]/[Parameters]. Như vậy với URL này, controller là HelloWorld và phương thức được thực hiện là Wellcome (không có sử dụng phần [Parameters] ở trong URL này).



Hình 13: Màn hình kết quả sau khi điều hướng Controller

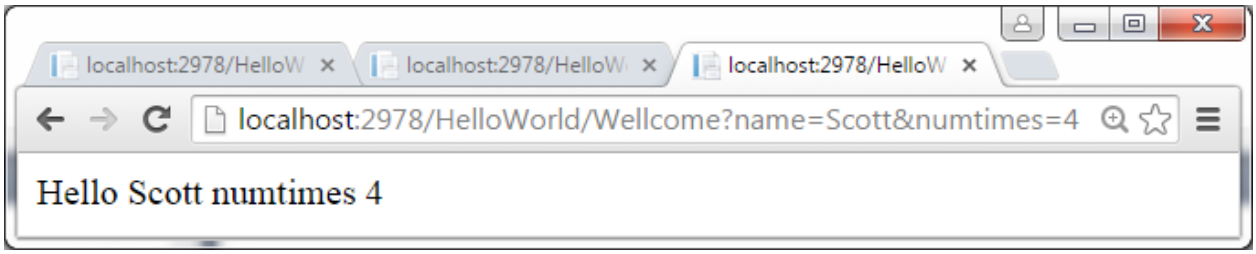
Hãy thử sửa code để thử duyệt với URL */HelloWorld/Welcome?name=Scott&numtimes=4* bằng cách thay đổi code ở phương thức Welcome với việc bổ sung 02 tham số như hình dưới.

```
// GET: /HelloWorld/Wellcome?name=Scott&numtimes=4
public String Wellcome(String name, int numtimes)
{
 return HttpUtility.HtmlEncode("Hello " + name + " numtimes " + numtimes);
}
```

Hình 14: Màn hình code bổ sung tham số cho điều hướng Controller

Chạy ứng dụng: *http://localhost:2978/HelloWorld/Wellcome?name=Scott&numtimes=4*

Thử thay đổi các giá trị cho tham số name và numtimes ở thanh địa chỉ . ASP.NET MVC model binding system sẽ tự động ánh xạ các parameters có tên trong query string ở URL với các tham số trong phương thức.



Hình 15: Màn hình kết quả sau khi bổ sung tham số cho điều hướng Controller

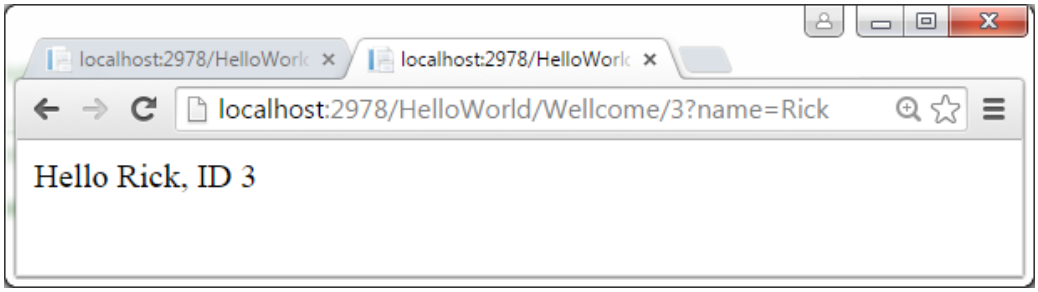
Như vậy ở ví dụ trên thì thành phần tham số (Parameters) theo cấu trúc vẫn chưa dùng, tham số name và numtimes được dùng ở đây chỉ là tham số theo query strings.

Dấu ? (question mark) trong URL là một phần ngăn cách để chỉ ra phía sau đó là query strings. Dấu & để ngăn cách các cặp query strings với nhau. Ta tiếp tục cập nhật lại đoạn code với nội dung như sau:

```
// GET: /HelloWorld/Wellcome
public String Wellcome(String name, int id=1)
{
  return HttpUtility.HtmlEncode("Hello " + name + ", ID: " + id);
}
```

Hình 16: Màn hình cập nhật code tham số cho điều hướng Controller

Chạy ứng dụng: *http://localhost:2978/HelloWorld/Wellcome/3?name=Rick*



Hình 17: Màn hình kết quả sau khi cập nhật code tham số cho điều hướng Controller

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

```
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 }
    );
}
```

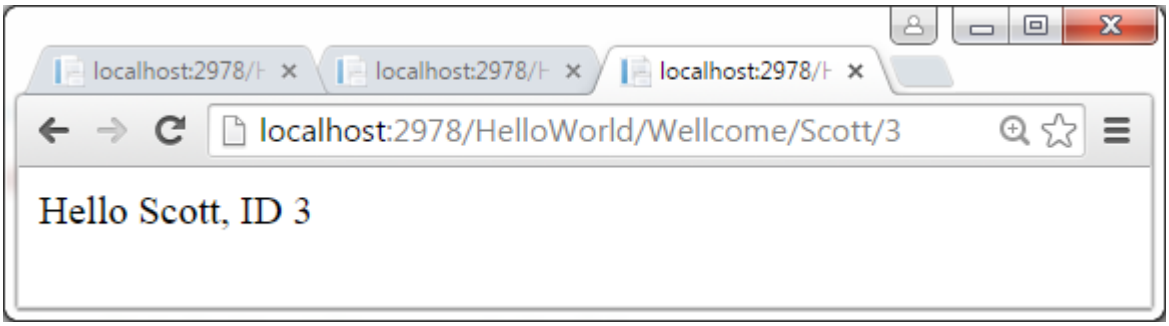
Hình 18: Màn hình của phương thức RegisterRoutes

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 name và numtimes ở phần parameters trong URL. Tại file *App_Start\RouteConfig.cs*:

```
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 = "Home", action = "Index", id =
                                                                //UrlParameter.Optional }
            defaults: new { controller = "Home", action = "Index", id =
                                                                UrlParameter.Optional }
        );

        routes.MapRoute (
            name: "Hello",
            url: "{controller}/{action}/{name}/{id}"
       );
     }

   }
```

Hình 19: Màn hình của phương thức RegisterRoutes

Chạy ứng dụng : http://localhost:2978/HelloWorld/Wellcome/Scott/3



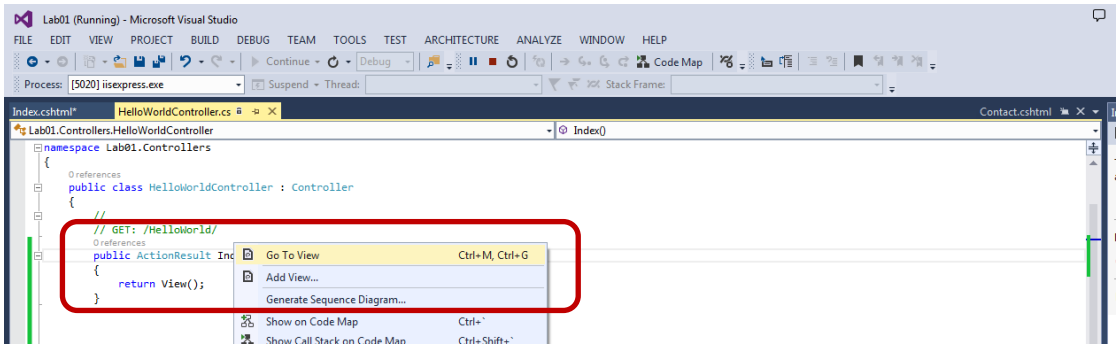
Hình 20: Màn hình kết quả truyền tham số qua ID

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.

Bài tập 3: Tạo mới một View

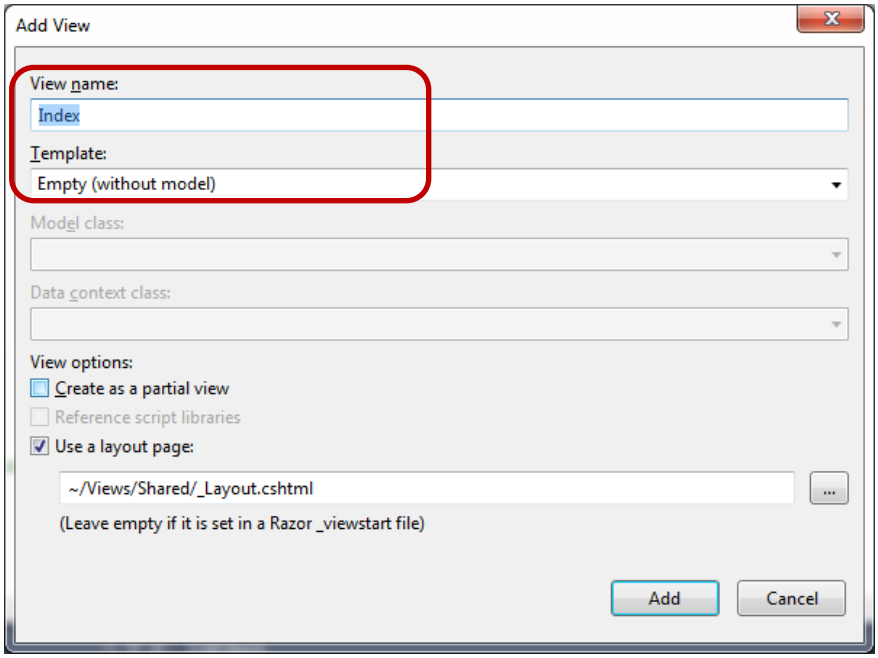
- ✓ Ta tiếp tục cập nhật lớp HelloWorldController để sử dụng với hiển thị một file khuôn mẫu giao diện (View Template File) để hiểu rõ việc tạo ra một HTML trả về để hiển thị phía client (browser).
- ✓ Hiện tại thì phương thức Index trong lớp controller. Ta sẽ thay đổi phương thức Index để nó trả về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

  Right click lên tên phương thức, chọn Add View



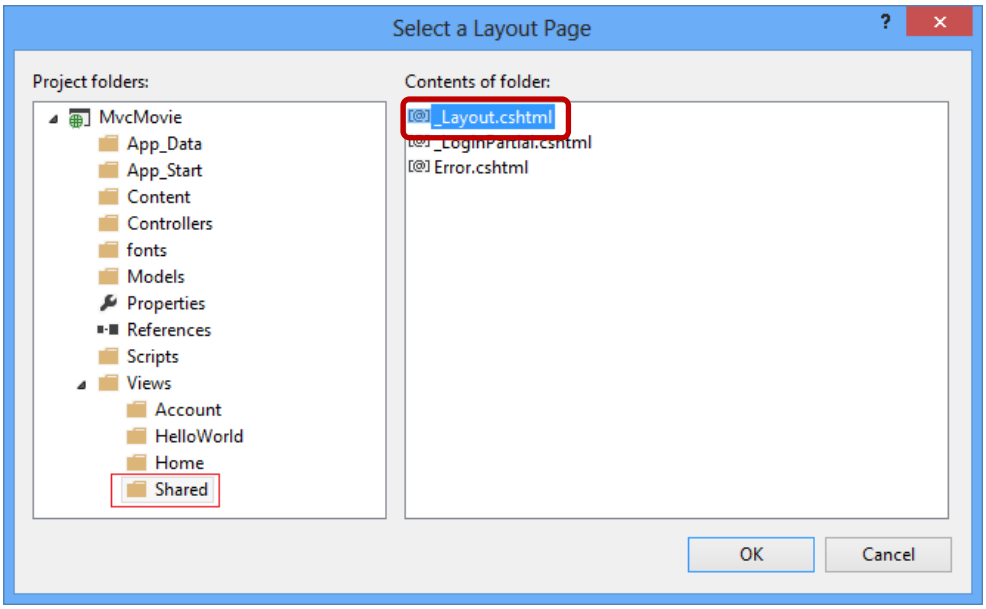
Hình 21: Màn hình tạo mới **MVC 5 View Page with (Layout Razor)**

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



Hình 22: Màn hình đặt tên View

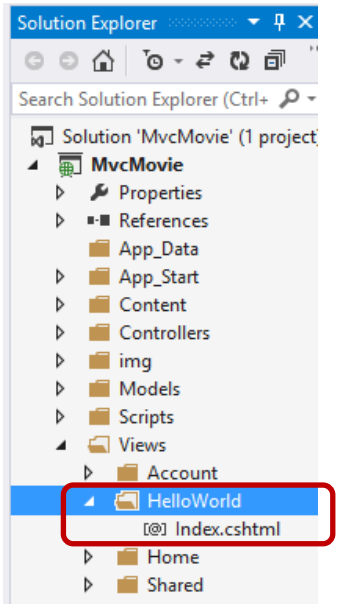
Tại cửa sổ **Select a Layout Page**, chọn mặc định là **_Layout.cshtml** rồi click **OK**.



Hình 23: Màn hình chọn Layout cho View

Ở cửa sổ bên trái thư mục *Views\Shared* được chọn (Nếu chúng ta có một giao diện tùy biến khác thì nó nằm trong thư mục khác, do đó có thể chọn cái khác đó)..

Tập tin *\Views\HelloWorld\Index.cshtml* được tạo như sau.



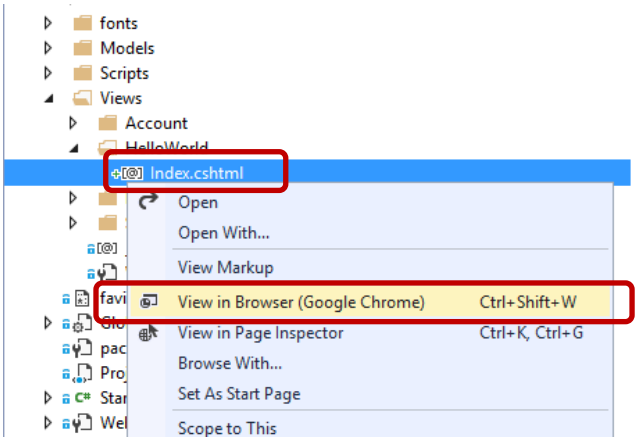
Hình 24: Màn hình kết quả View đã được tạo

Và đoạn code Razor như sau:

```
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Index</h2>

<p> Hello from our view Template</p>
```

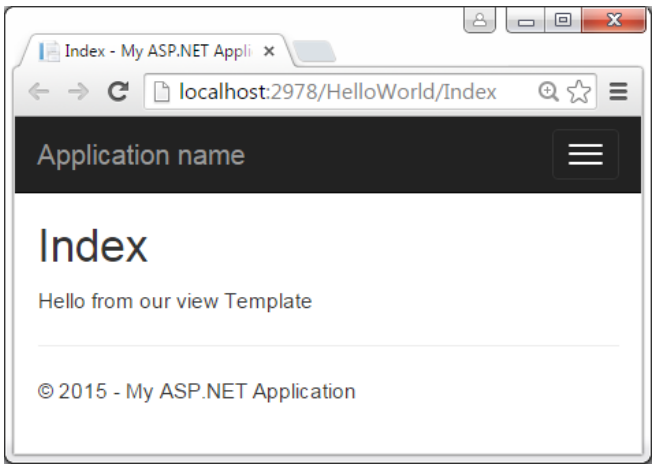
Hình 25: Màn hình code có sẵn của View được tạo

Right click tập tin *Index.cshtml* và chọn **View in Browser**.



Hình 26: Màn hình xem kết quả của view

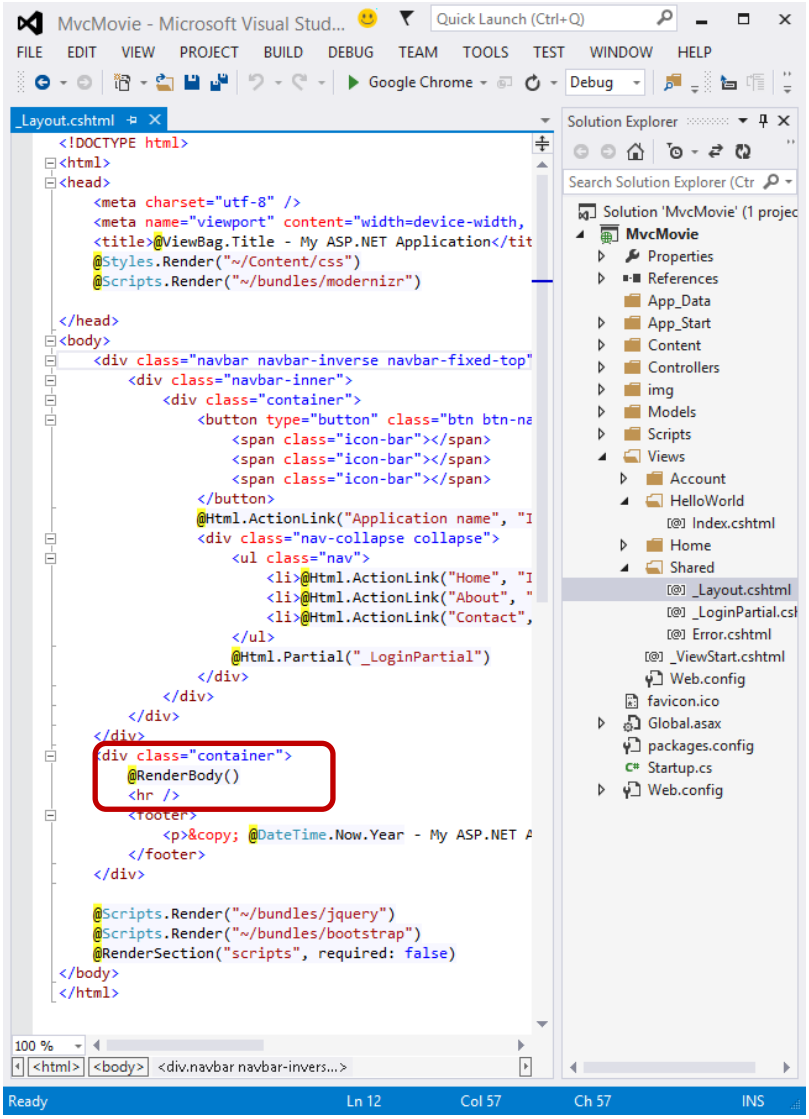
Kết quả như sau:



Hình 27: Màn hình kết quả của view

Bài tập 4: ***Thay đổi Views và Layout Pages (giao diện của trang)***

Trước tiên, thử thay đổi tên ứng dụng "Application name" ở liên kết trên cùng của trang. Phần này nó hiển thị hầu hết ở các trang. Vào thư mục */Views/Shared* ở **Solution Explorer** và mở tập tin *_Layout.cshtml*. Tập tin này được gọi là *layout page* và nó nằm ở thư mục dùng chung mà các trang cùng sử dụng.



Hình 28: Màn hình trang _Layout.cshtml

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.

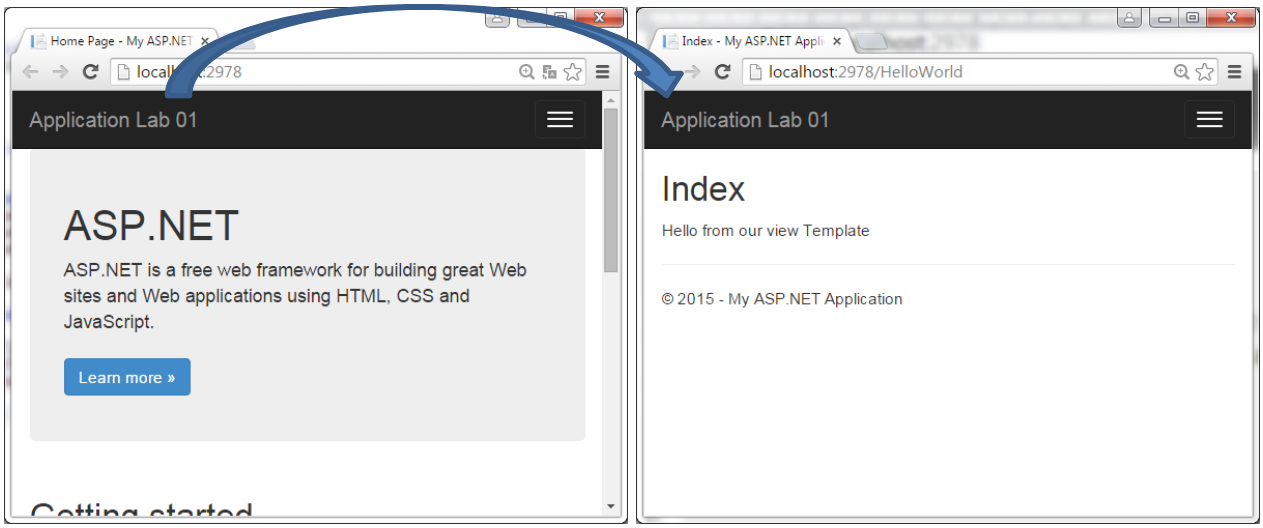
Tìm đến dòng có chữ @RenderBody(). RenderBody là một thành phần giữ chỗ để cho các trang hiển thị ở chính chỗ đó.

Ví dụ, nếu chúng ta có liên kết **About**, thì *Views\Home\About.cshtml* view được rendered bên trong phương thức RenderBody.

Chúng ta sẽ thay đổi nội dung của phần tử title bằng cách thay đổi liên kết ActionLink trong layout template từ chuỗi "Application name" sang chuỗi "HelloWorld" và controller từ Home sang HelloWorld. Sau khi xong ta sẽ có như hình sau:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application HelloWorld</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<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 HelloWorld", "Index", "HelloWorld",
                                                    null, new { @class = "navbar-brand" })
            </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>
                </ul>
                @Html.Partial("_LoginPartial")
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>
    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```

Hình 29: Màn hình trang _Layout.cshtml



Hình 30: Màn hình từ trang Home chuyển sang trang HelloWorld

----------Hết Lab 01---------