**Chương 2:**

# TỔNG QUAN VỀ ASP.NET MVC 4

## 2.1 TỔNG QUAN

ASP.NET MVC 4 là một framework để xây dựng các ứng dụng web có khả năng mở rộng dựa trên các tiêu chuẩn bằng cách sử dụng các mẫu thiết kế mô hình và sức mạnh của framework ASP.NET mới. Framework 4 tập trung vào phát triển ứng dụng web trên điện thoại di động dễ dàng hơn.

Khi tạo một dự án mới ASP.NET MVC 4 có một ứng dụng dự án mẫu điện thoại di động để chúng ta có thể sử dụng cho việc xây dựng một ứng dụng chuyên dụng cho các thiết bị di động. Ngoài ra, ASP.NET MVC 4 tích hợp với gói điện thoại di động thông qua một gói NuGet jQuery.Mobile.MVC.

jQuery Mobile là một framework dựa trên nền HTML5 để phát triển các ứng dụng web tương thích với tất cả các nền tảng thiết bị di động phổ biến, bao gồm cả Windows Phone, iPhone, Android v.v. Tuy nhiên, nếu chúng ta cần chuyên môn hóa cho nhiều loại thiết bị khác nhau thì ASP.NET MVC 4 cũng cho phép chúng ta tạo các View đặc thù khác nhau cho các thiết bị khác nhau và cung cấp tối ưu hóa thiết bị cụ thể.

Với chương này sẽ bắt đầu với kiểu dự án MVC ASP.NET 4 "Internet Application" để tạo một ứng dụng Photo Gallery. Dần dần sẽ nâng cấp ứng dụng bằng cách sử dụng jQuery Mobile và tính năng mới của ASP.NET MVC 4 để làm tương thích với các thiết bị di động khác nhau và các trình duyệt web của máy tính.

Chúng ta cũng sẽ học cách viết code mới với ASP.NET MVC 4 để dễ dàng hơn cho việc viết các phương thức hành động bất đồng bộ bởi sự hỗ trợ của các loại trả về ActionResult.

### 2.1.1 Mục tiêu

Trong chương này, chúng ta sẽ học cách:

- ✓ Tận dụng lợi thế cải tiến của các dự án ASP.NET MVC mẫu, bao gồm cả các mẫu dự án ứng dụng trên điện thoại di động mới
- ✓ Sử dụng thuộc tính viewport của HTML5 và truy vấn media của CSS để cải thiện hiển thị trên các thiết bị di động
- ✓ Sử dụng jQuery Mobile để đẩy nhanh tiến độ đối với việc tối ưu hóa xây dựng giao diện cảm ứng.
- ✓ Tạo các giao diện cho các thiết bị di động cụ thể.

- ✓ Sử dụng các thành phần chuyển đổi View để chuyển đổi giữa các View của điện thoại di động và máy tính để bàn trong ứng dụng
- ✓ Tạo bộ điều khiển bất đồng bộ sử dụng hỗ trợ công việc

### 2.1.2 Điều kiện tiên quyết

Chúng ta phải hoàn thành các bài sau đây:

- ✓ Cài đặt Microsoft Visual Studio Express 2012 for Web
- ✓ Cài đặt thiết bị giả lập Windows Phone Emulator (bao gồm trong Windows Phone 7.1.1 SDK )
- ✓ Tùy chọn - WebMatrix 3 với Electric Plum Simulator iPhone mở rộng (chỉ dành cho bài 3, được sử dụng để duyệt qua các ứng dụng web với giả lập iPhone)

### 2.1.3 Các bài tập

[1]. Tạo mới dự án ASP.NET MVC 4 kiểu Internet Application

[2]. Tạo ứng dụng web Photo Gallery

[3]. Bổ sung sự hỗ trợ thêm cho các thiết bị di động

[4]. Sử dụng điều khiển bất đồng bộ

Lưu ý: Mỗi bài được đi kèm bởi một thư mục End chứa các giải pháp hoàn thiện.

Chúng ta có thể sử dụng giải pháp này như một hướng dẫn nếu chúng ta cần giúp đỡ thêm trong khi thực hiện dự án.

## 2.2 Bài 1: Dự án ASP.NET MVC 4 Internet Application

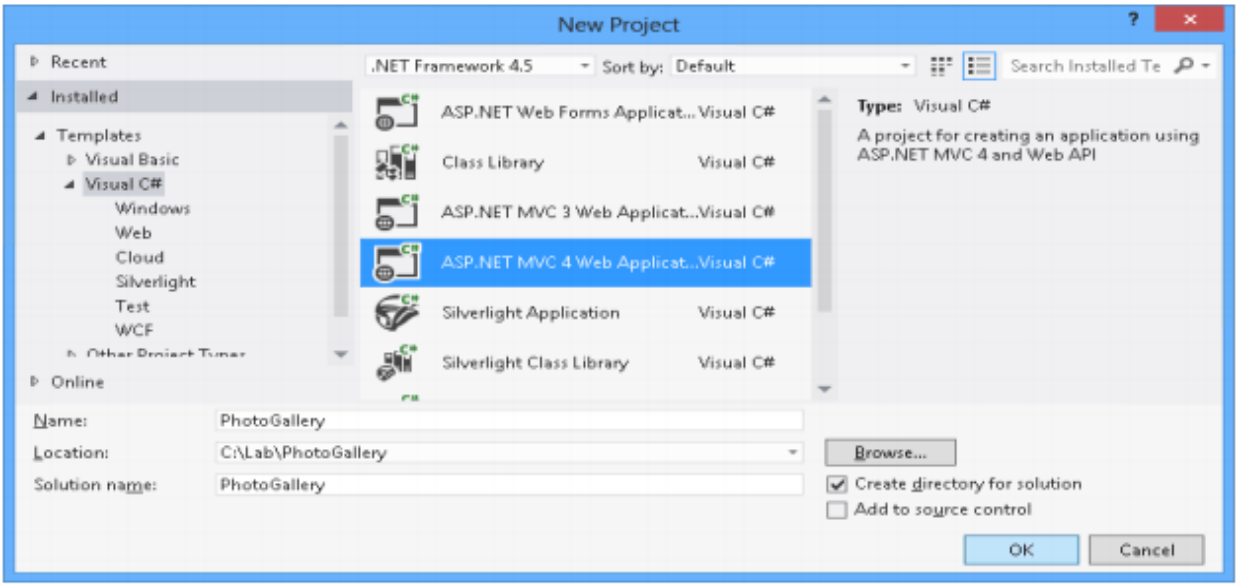
Trong bài tập này, chúng ta sẽ khám phá những cải tiến trong mẫu dự án ASP.NET MVC 4. Ngoài mẫu ứng dụng Internet, đã hiện diện trong MVC 3, phiên bản này hiện nay bao gồm một mẫu riêng biệt cho các ứng dụng điện thoại di động. Trước tiên, chúng ta sẽ xem xét một số tính năng liên quan của mỗi mẫu. Sau đó, chúng ta sẽ làm việc trên các trang được sinh ra của các phương pháp tiếp cận khác nhau.

### 2.2.1 Nhiệm vụ 1: Khám phá mẫu dự án Internet Application

[1]. Mở Visual Studio .

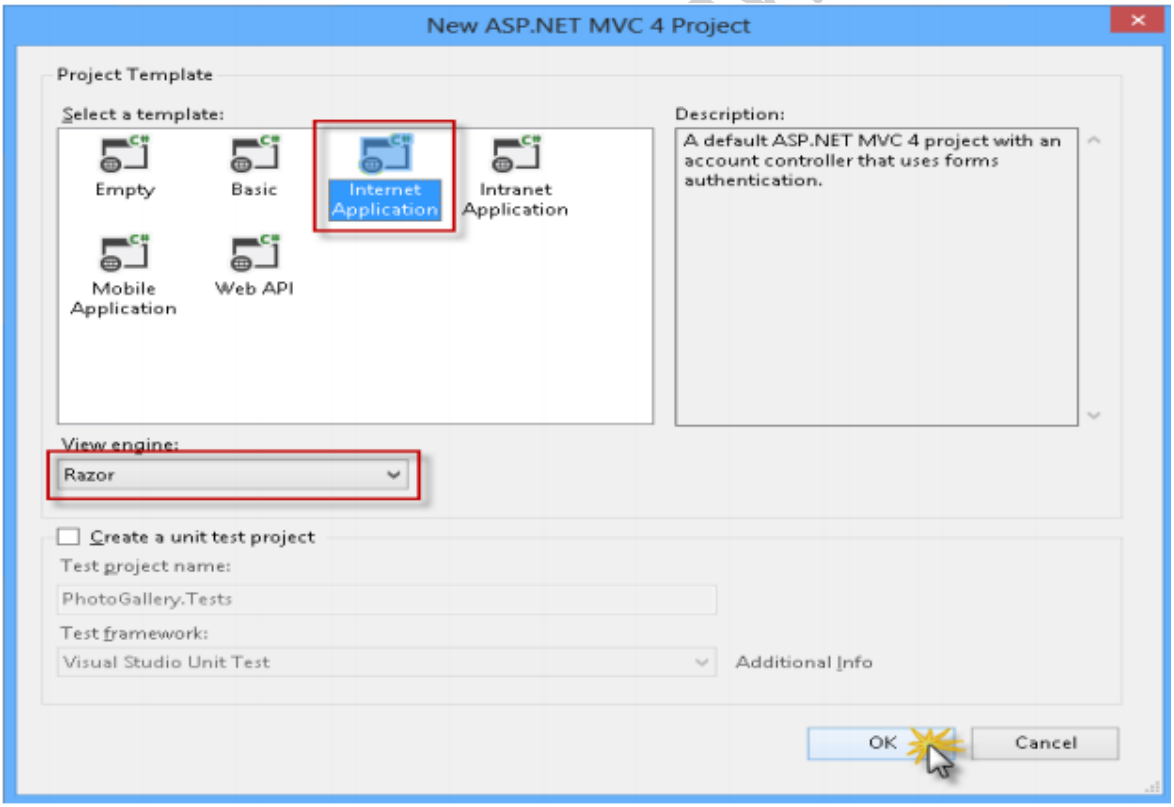
[2]. Chọn File | New | Project menu.

Trong hộp thoại New Project , chọn mẫu C # Visual | Web trên bảng bên trái, và chọn ASP.NET MVC 4 Web Application. Đặt tên cho dự án là PhotoGallery , chọn một thư mục (hoặc để mặc định) và nhấn OK .




Tạo một dự án mới

[3]. Trong hộp thoại New ASP.NET MVC Project 4, chọn mẫu dự án Internet Application và nhấn vào OK . Hãy chắc chắn rằng chúng ta đã chọn Razor cho View của chúng ta.

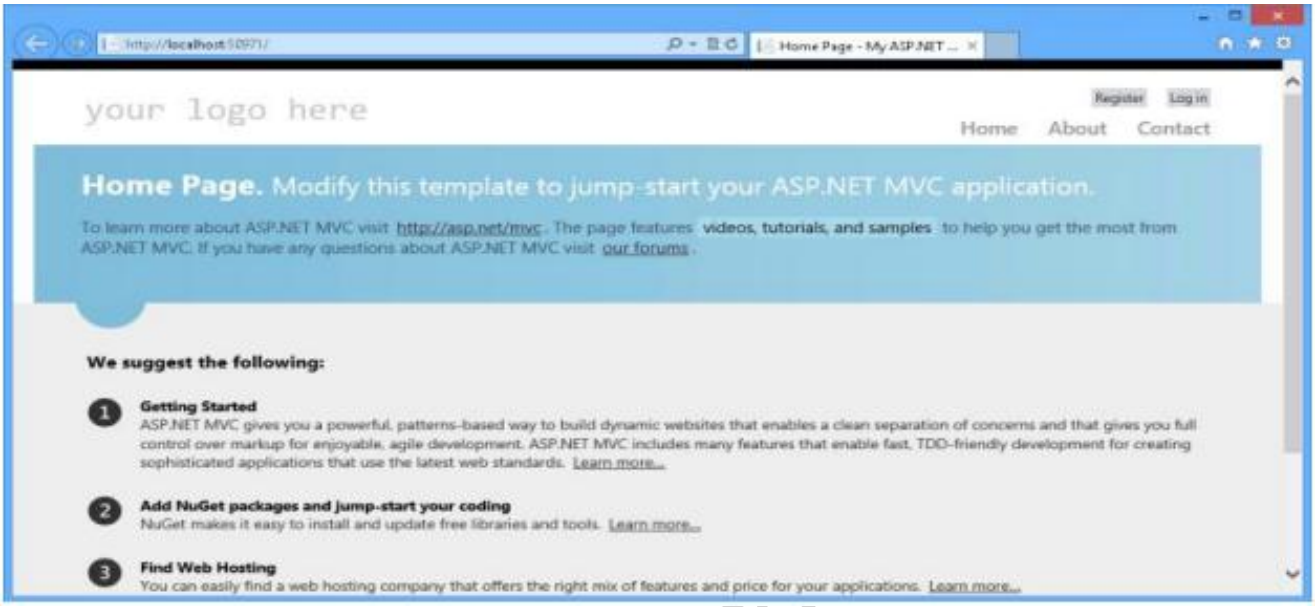



Tạo một ứng dụng ASP.NET MVC mới Internet 4

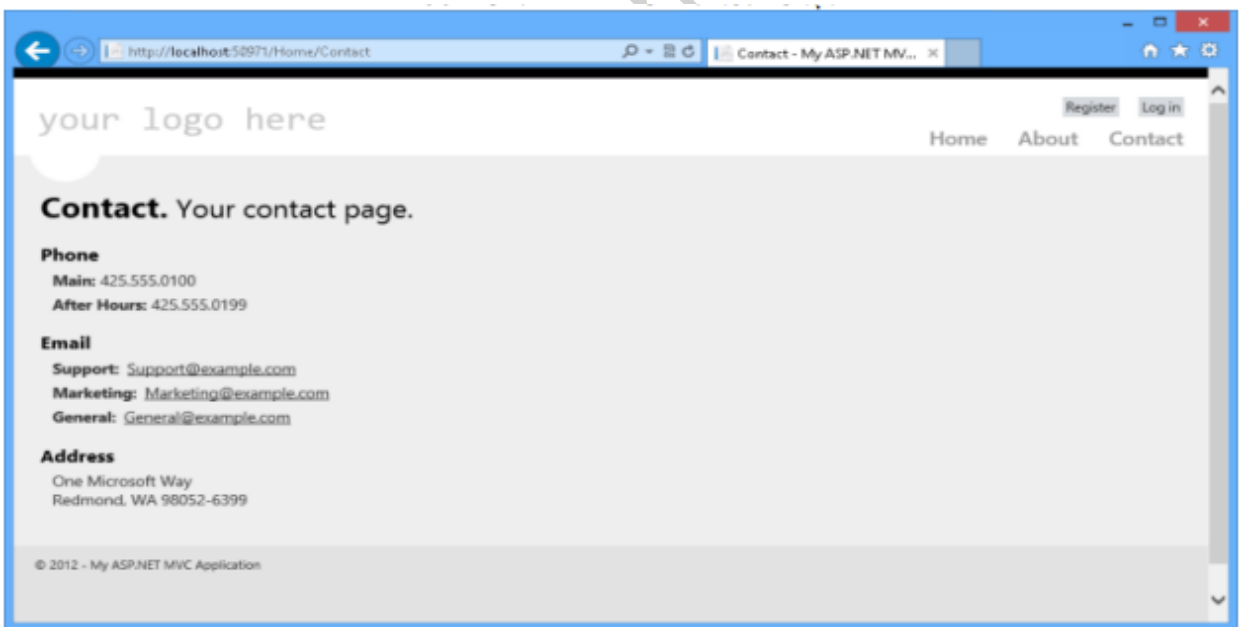
***Lưu ý:*** cú pháp Razor đã được giới thiệu trong ASP.NET MVC 3. Mục tiêu của nó là để giảm thiểu số lượng các ký tự và số lần bấm phím cần thiết trong một tập tin, cho

phép viết mã nhanh. Razor dựa trên kỹ năng lập trình C#/VB và đưa ra cú pháp đánh dấu mẫu cho phép công việc xây dựng HTML tuyệt vời.

[4]. Nhấn F5 để chạy giải pháp và xem các trang mẫu. Chúng ta có thể kiểm tra các tính năng sau: Các mẫu đã được đổi mới, cung cấp style hiện đại hơn.




Mẫu ASP.NET MVC 4 thiết kế lại



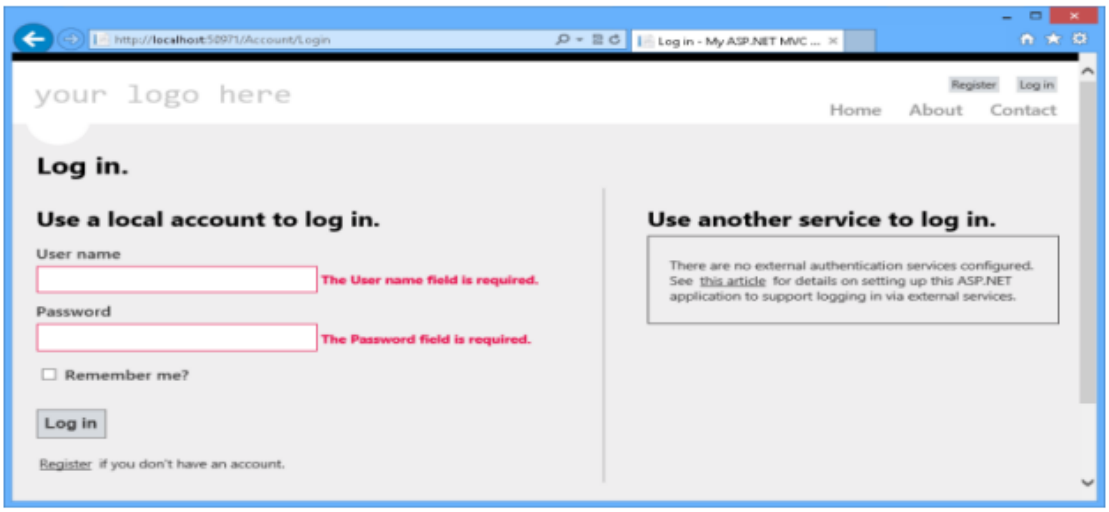

Trang liên hệ mới

**Hiển thị thích nghi**

Kiểm tra thay đổi kích thước cửa sổ trình duyệt và theo dõi cách bố trí trang tự động điều chỉnh kích thước cửa sổ mới. Các mẫu này sử dụng kỹ thuật dựng hình thích ứng để làm đúng trong cả hai nền tảng máy tính để bàn và điện thoại di động mà không có bất kỳ tùy chỉnh.


ASP.NET MVC 4 dự án mẫu trong kích thước trình duyệt khác nhau Giao diện người dùng phong phú hơn với JavaScript Một sự cải tiến đối với các dự án mẫu mặc định là sử dụng JavaScript để cung cấp tương tác với JavaScript nhiều hơn.

Các liên kết Đăng nhập và Đăng ký sử dụng theo mẫu này cụ thể sử dụng validations của jQuery để xác nhận các trường đầu vào từ phía khách hàng.




jQuery Validation

***Lưu ý:*** Có 2 phần đăng nhập, trong phần đầu tiên, chúng ta có thể đăng nhập bằng cách sử dụng một tài khoản đã được đăng ký từ trang web và trong phần thứ hai, chúng ta có thể đăng nhập bằng cách sử dụng một dịch vụ xác thực khác như google (theo mặc định bị vô hiệu hóa).

[5]. Đóng trình duyệt để kết thúc trình gỡ lỗi và trở về Visual Studio.

[6]. Mở tập tin AuthConfig.cs nằm dưới thư mục App_Start.

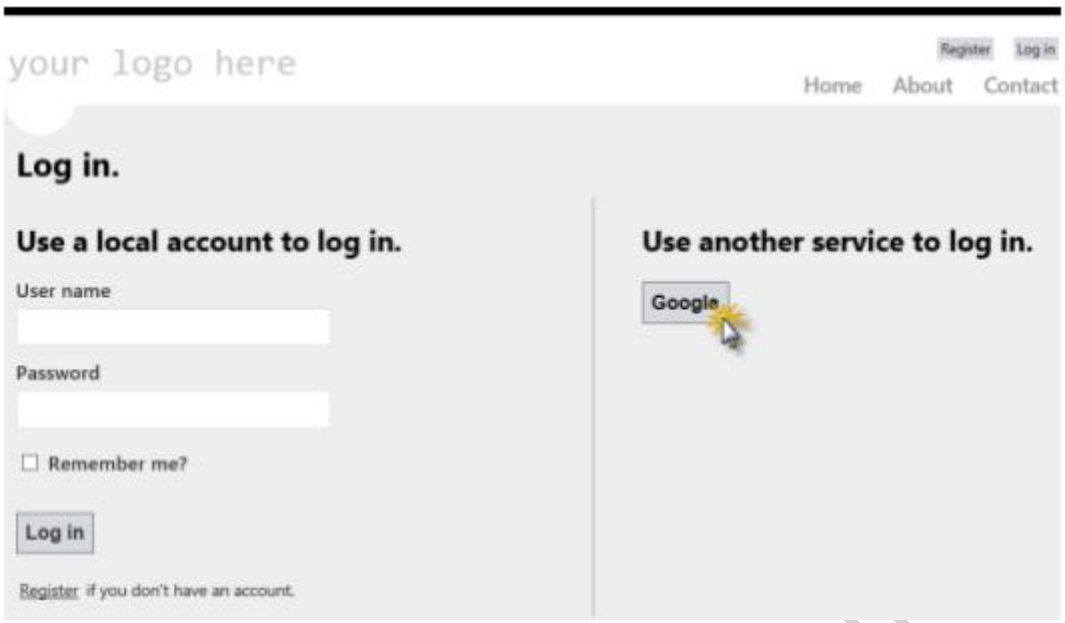
[7]. Bỏ ghi chú của dòng cuối cùng để đăng ký được với Google đối với khách hàng xác thực bởi OAuth.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Web.WebPages.OAuth;
using MvcMusicStore.Models;

namespace MvcMusicStore
{
    public static class AuthConfig
    {
        public static void RegisterAuth()
        {
            //OAuthWebSecurity.RegisterMicrosoftClient(
            //    clientId: "",
            //    clientSecret: "");

            //OAuthWebSecurity.RegisterTwitterClient(
            //    consumerKey: "",
            //    consumerSecret: "");

            //OAuthWebSecurity.RegisterFacebookClient(
            //    appId: "",
            //    appSecret: "");

            OAuthWebSecurity.RegisterGoogleClient();
        }
    }
}
```

***Lưu ý:*** Thông báo chúng ta có thể dễ dàng cho phép xác thực bằng cách sử dụng bất kỳ dịch vụ OpenID hoặc OAuth như Facebook, Twitter, Microsoft, vv

[8]. Nhấn F5 để chạy giải pháp và điều hướng đến trang đăng nhập.

[9]. Chọn dịch vụ Google để đăng nhập.

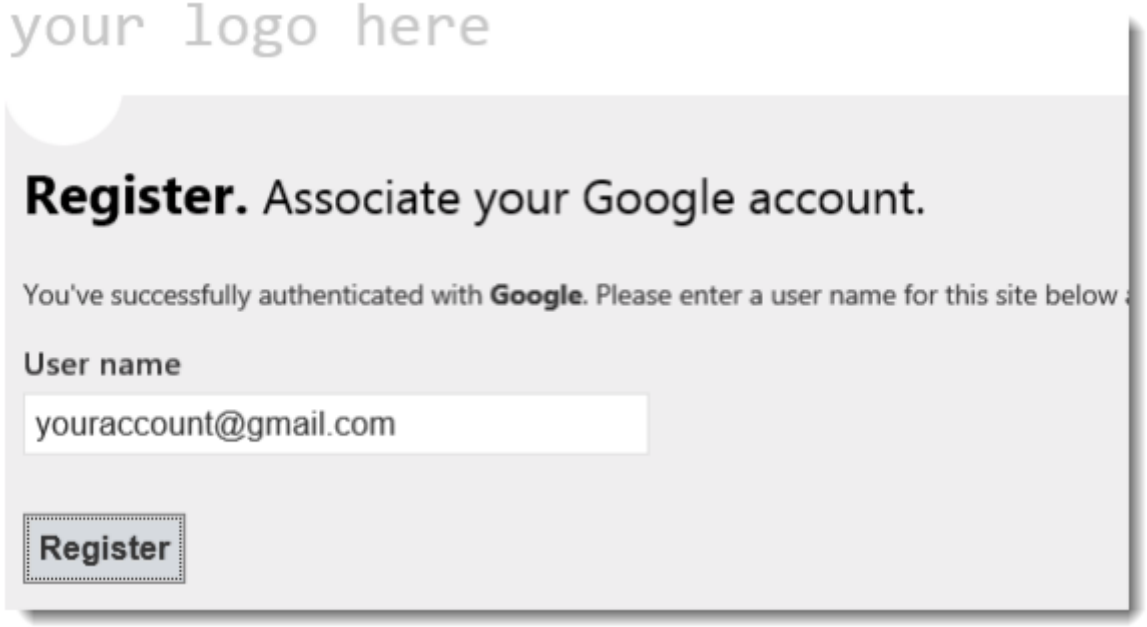



Chọn đăng nhập dịch vụ

[10]. Đăng nhập bằng cách sử dụng tài khoản Google của chúng ta.

[11]. Cho phép website (localhost) để lấy thông tin từ tài khoản Google.

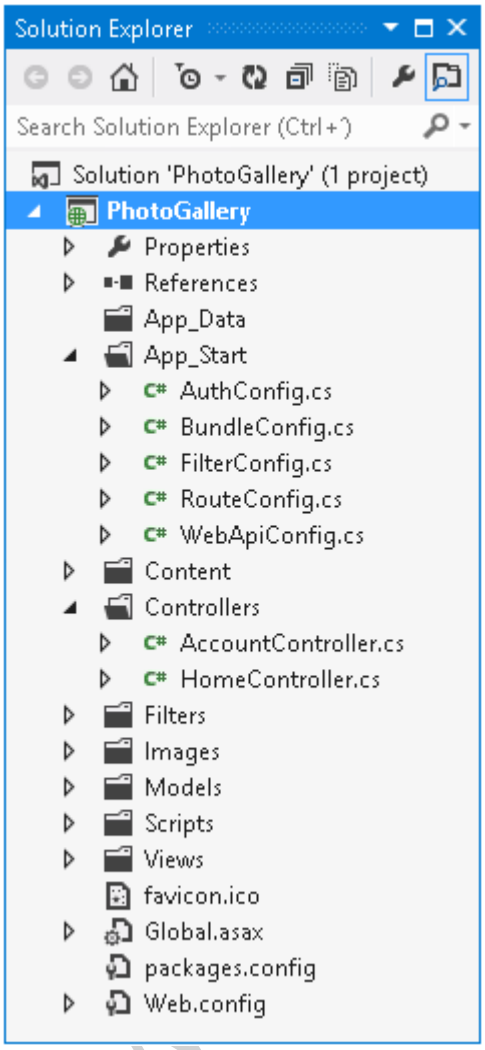
[12]. Cuối cùng, chúng ta sẽ phải đăng ký trong website để kết hợp với tài khoản Google.




Liên kết tài khoản Google của chúng ta

[13]. Đóng trình duyệt để ngăn chặn các trình gỡ lỗi và trở về Visual Studio.

[14]. Bây giờ khám phá giải pháp để kiểm tra một số tính năng mới khác được giới thiệu bởi ASP.NET MVC 4 trong dự án mẫu.




ASP.NET MVC 4 Internet Ứng dụng Dự án mẫu

✓ **HTML 5 Markup**

Duyệt các View mẫu để tìm ra các đánh dấu chủ đề mới.

```
<header>
    <div class="content-wrapper">
        <div class="float-left">
            <p class="site-title">@Html.ActionLink("your logo here", "Index", "Home")</p>
        </div>
        <div class="float-right">
            <section id="login">
                @Html.Partial("_LoginPartial")
            </section>
            <nav>
                <ul id="menu">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                </ul>
            </nav>
        </div>
    </div>
</header>
```

Mẫu mới đánh dấu theo Razor và HTML5 (About.cshtml).

✓ **Cập nhật thư viện JavaScript**

Mẫu mặc định của ASP.NET MVC 4 hiện nay bao gồm KnockoutJS, JavaScript MVVM framework cho phép chúng ta tạo ra các ứng dụng web phong phú và đáp ứng cao bằng cách sử dụng JavaScript và HTML. Giống như trong MVC3, jQuery và các thư viện jQuery UI cũng được bao gồm trong ASP.NET MVC 4.

Lưu ý: Chúng ta có thể nhận được thêm thông tin về thư viện KnockOutJS trong liên kết này:http://learn.knockoutjs.com/ . Ngoài ra, chúng ta có thể tìm hiểu về jQuery và jQuery UI tại http://docs.jquery.com/.

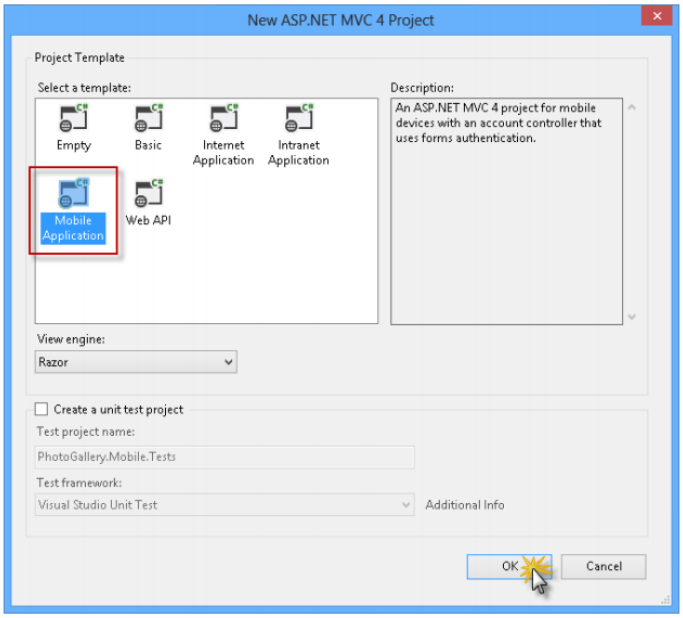
### 2.2.1 Nhiệm vụ 2: Khám phá mẫu dự án Mobile Application

ASP.NET MVC 4 tạo điều kiện cho việc phát triển trang web cho các trình duyệt di động và máy tính bảng. Mẫu này có cấu trúc ứng dụng tương tự như các mẫu ứng dụng Internet, nhưng style của nó được sửa đổi cho đúng với các thiết bị di động có cảm ứng.

[1]. Chọn File | New | Project menu.

Trong hộp thoại New Project , chọn mẫu C # Visual | Web trên phần panel bên trái, và chọn ASP.NET MVC Web Application 4, Đặt tên dự án là PhotoGallery.Mobile , chọn một thư mục lưu trữ (hoặc để mặc định), chọn "Add Solution "và nhấn OK .

[2]. Trong hộp thoại ASP.NET MVC Project 4, chọn mẫu dự án ứng dụng di động và bấm vào OK . Hãy chắc chắn rằng chúng ta đã chọn Razor cho các View.



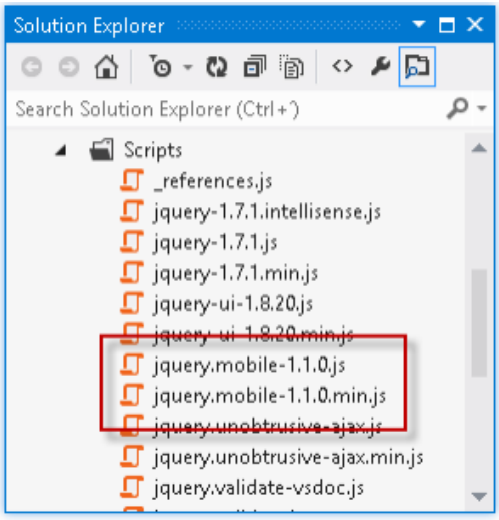

Tạo một ASP.NET MVC mới 4 ứng dụng di động

[3]. Bây giờ chúng ta có thể khám phá các giải pháp và kiểm tra một số các tính năng mới được giới thiệu bởi giải pháp trong mẫu ASP.NET MVC 4 Mobile Application:

✓ **Thư viện jQuery Mobile**

Các ứng dụng dự án mẫu điện thoại di động bao gồm các thư viện jQuery Mobile, là một thư viện mã nguồn mở có khả năng tương thích trình duyệt di động. jQuery Mobile áp dụng tăng cường tiến bộ đối với các trình duyệt di động có hỗ trợ CSS và JavaScript. Tăng cường tiến bộ cho phép tất cả các trình duyệt hiển thị nội dung cơ bản của một trang web có thể hiển thị được nội dung phong phú.

Các tập tin JavaScript và CSS của jQuery giúp trình duyệt di động phù hợp với nội dung trên màn hình mà không cần thực hiện bất kỳ thay đổi trong đánh dấu trang.




Thư viện di động jQuery trong dự án mẫu

✓ **Đánh dấu dựa trên HTML5**

```
@using (Html.BeginForm()) {
    @Html.ValidationSummary()

    <ul data-role="listview" data-inset="true">
        <li data-role="list-divider">Details</li>

        <li data-role="fieldcontain">
            @Html.LabelFor(m => m.UserName)
            @Html.TextBoxFor(m => m.UserName)
        </li>

        <li data-role="fieldcontain">
            @Html.LabelFor(m => m.Password)
            @Html.PasswordFor(m => m.Password)
        </li>

        <li data-role="fieldcontain">
            <ul data-role="listview" data-inset="true">
                <li data-role="list-divider">Navigation</li>
                <li>@Html.ActionLink("About", "About", "Home")</li>
                <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
            </ul>
            <input type="submit" value="Log in" />
        </li>
    </ul>
}
```

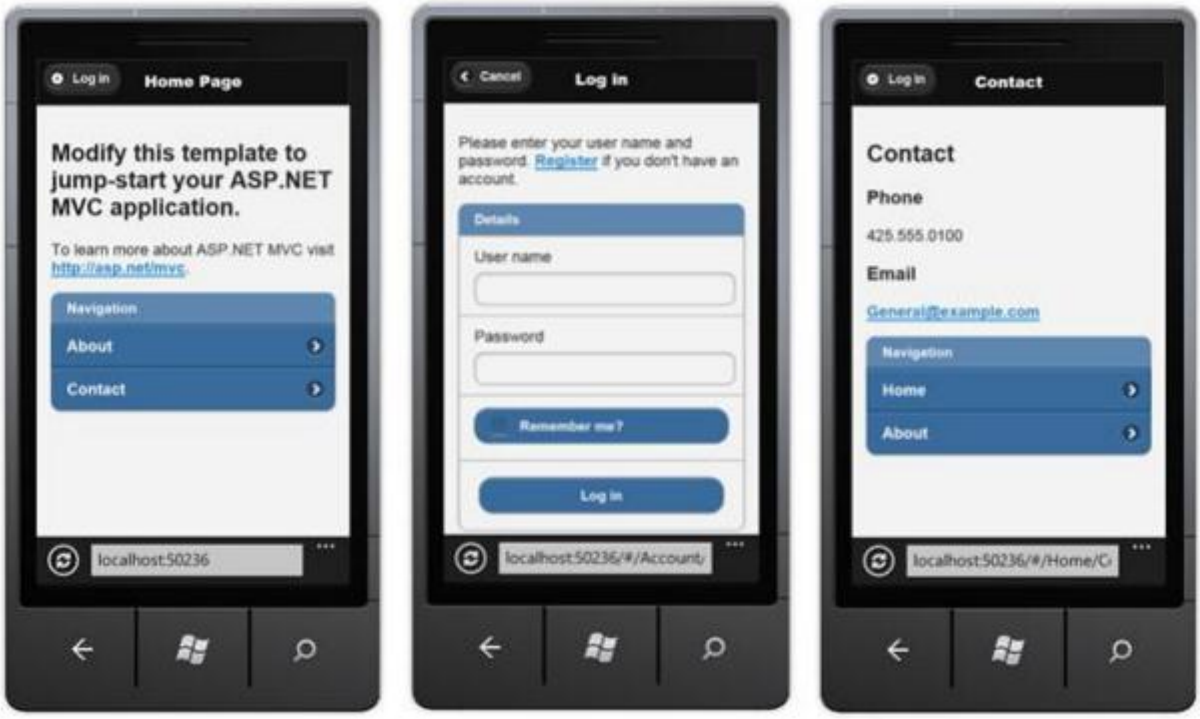
Mẫu Mobile Application cách sử dụng HTML5(Login.cshtml và index.cshtml)

[4]. Nhấn F5 để chạy các giải pháp.

[5]. Mở Windows Phone 7 Emulator .

Trong màn hình bắt đầu điện thoại, mở Internet Explorer. Kiểm tra URL máy tính để bàn đang duyệt và duyệt đến URL đó từ điện thoại (ví dụ như http://localhost:[PortNumber]/).

Bây giờ chúng ta có thể đến trang login hoặc trang about. Chú ý rằng style của trang web dựa trên ứng dụng Metro mới đối với điện thoại di động. Dự án ASP.NET MVC 4 được hiển thị đúng trên các thiết bị di động, đảm bảo tất cả các thành phần của trang có thể nhìn thấy và kích hoạt. Chú ý rằng các liên kết trên tiêu đề đủ lớn để bấm được.




Các trang của dự án mẫu trong thiết bị di động

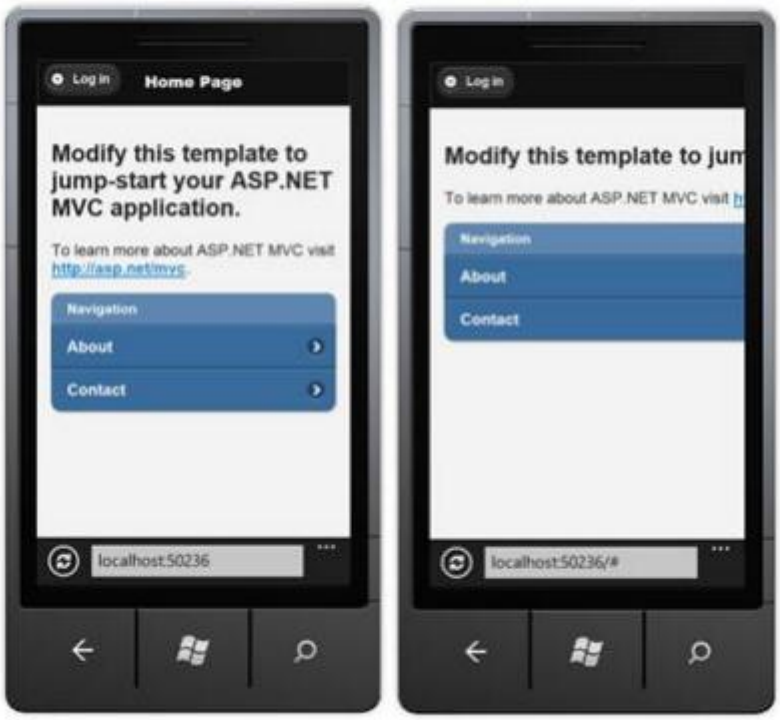
Mẫu mới cũng sử dụng thẻ meta Viewport . Hầu hết các trình duyệt di động xác định chiều rộng cửa sổ cho các trình duyệt ảo (viewport) cái mà lớn hơn chiều rộng thực tế của các thiết bị di động.

Điều này cho phép trình duyệt di động hiển thị toàn bộ trang web bên trong màn hình hiển thị ảo. Thẻ meta Viewport cho phép các nhà phát triển web thiết lập chiều rộng, chiều cao và quy mô của khu vực trình duyệt trên các thiết bị di động . Mẫu ASP.NET MVC 4 Mobile Application thiết lập View với chiều rộng thiết bị ("width=device-width") trong Layout ( Views/Shared_Layout.cshtml ), do đó tất cả các trang sẽ có

View của chúng với chiều rộng màn hình điện thoại. Chú ý rằng thẻ meta Viewport sẽ không thay đổi View của trình duyệt mặc định.

Mở _Layout.cshtml , nằm trong thư mục Views | Shared, và hủy bỏ thẻ meta Viewport. Chạy ứng dụng và kiểm tra sự khác biệt.

```
...
<meta charset="utf-8" />
<title>@ViewBag.Title</title>
@* <meta name="viewport" content="width=device-width" /> *@
...
```




*Các trang web sau khi bỏ thẻ meta viewport*

Các trang web sau khi bỏ thẻ meta viewport

Trong Visual Studio, nhấn SHIFT + F5 để gỡ lỗi các ứng dụng.

Phục hồi lại thẻ meta viewport.

```
...
<meta charset="utf-8" />
<title>@ViewBag.Title</title>
<meta name="viewport" content="width=device-width" />
...
```

### 2.2.2 Nhiệm vụ 3: Sử dụng sinh giao diện thích ứng

Trong nhiệm vụ này, chúng ta sẽ học một phương pháp khác để làm cho trang web hiển thị một cách chính xác trên các thiết bị di động và trình duyệt web cùng một lúc mà không cần tuỳ biến bất kỳ điều gì. Chúng ta đã sử dụng thẻ meta Viewport

với mục đích tương tự. Bây giờ chúng ta sẽ đáp ứng một phương pháp mạnh mẽ khác: sinh giao diện thích ứng.

Sinh giao diện thích ứng là một kỹ thuật sử dụng CSS3 truy vấn đa phương tiện để tùy chỉnh style áp dụng cho trang. Truy vấn đa phương tiện định nghĩa các điều kiện bên trong bảng định kiểu (nhóm các CSS theo một điều kiện nhất định). Chỉ khi điều kiện là đúng sự thật, style mới được áp dụng cho các đối tượng đã kê khai.

Sự linh hoạt được cung cấp bởi kỹ thuật sinh giao diện thích ứng cho phép tùy chỉnh bất kỳ thứ gì để hiển thị các trang web trên các thiết bị khác nhau. Chúng ta có thể định nghĩa nhiều style chúng ta muốn trên một style duy nhất mà không cần viết mã logic để lựa chọn style. Vì vậy, đó là cách rất gọn gàng thích ứng kiểu trang vì nó làm giảm số lượng mã trùng lặp và logic cho mục đích sinh giao diện. Mặt khác, băng thông tiêu thụ sẽ tăng khi kích thước của file CSS tăng.

Bằng cách sử dụng kỹ thuật sinh giao diện thích ứng, trang web của chúng ta sẽ được hiển thị đúng, bất kể của trình duyệt. Tuy nhiên, chúng ta nên xem xét tăng thêm tải băng thông.

***Lưu ý:*** Các định dạng cơ bản của một truy vấn đa phương tiện là: @media [Scope: all | handheld | print | projection | screen] ([property:value] and ... [property:value])

Ví dụ truy vấn đa phương tiện: ***>@media all and (max-width: 1000px) and (min-width: 700px) {}***: Đối với các màn hình có độ phân giải từ 700px đến 1000px.

***@media screen and (min-width: 400px) and (max-width: 700px) { ... }:*** Chỉ dành cho màn hình. Độ phân giải phải từ 400 đến 700px.

***@media handheld and (min-width: 20em), screen and (min-width: 20em) { ... }:*** Đối với các thiết bị cầm tay (điện thoại di động và các thiết bị) và màn hình. Chiều rộng tối thiểu phải lớn hơn 20em.

Chúng ta có thể tìm thêm thông tin về việc này trên trang web W3C .

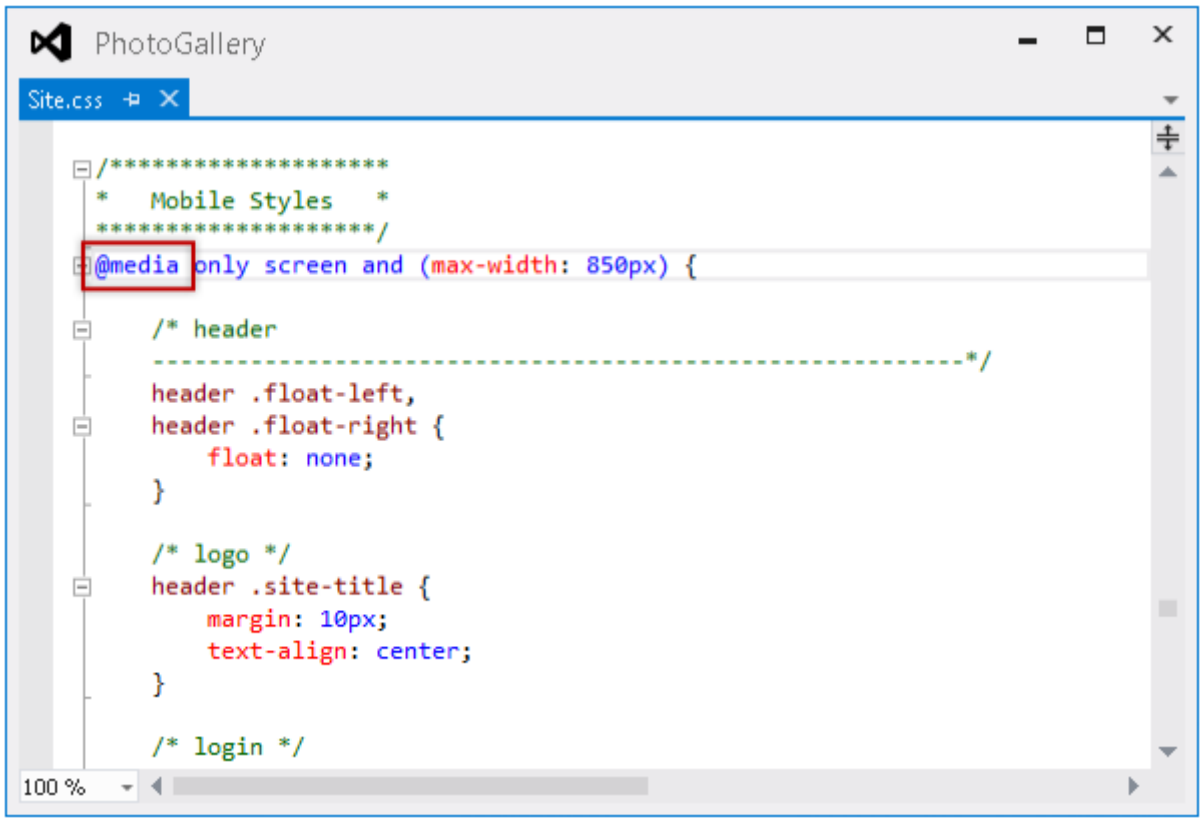
Bây giờ chúng ta sẽ tìm hiểu "sinh giao diện thích ứng" làm việc thế nào, việc cải thiện khả năng đọc của mẫu website mặc định của MVC ASP.NET 4.

[1]. Mở giải pháp PhotoGallery.sln đã tạo tại nhiệm vụ 1 và chọn dự án PhotoGallery. Nhấn F5 để chạy giải pháp.

[2]. Thay đổi độ rộng của trình duyệt, thiết lập cửa sổ bằng một nửa hoặc nhỏ hơn 1/4 kích thước ban đầu của nó. Chú ý những gì sẽ xảy ra với các mục trong tiêu đề: Một số thành phần sẽ không xuất hiện trong vùng hiển thị của tiêu đề.

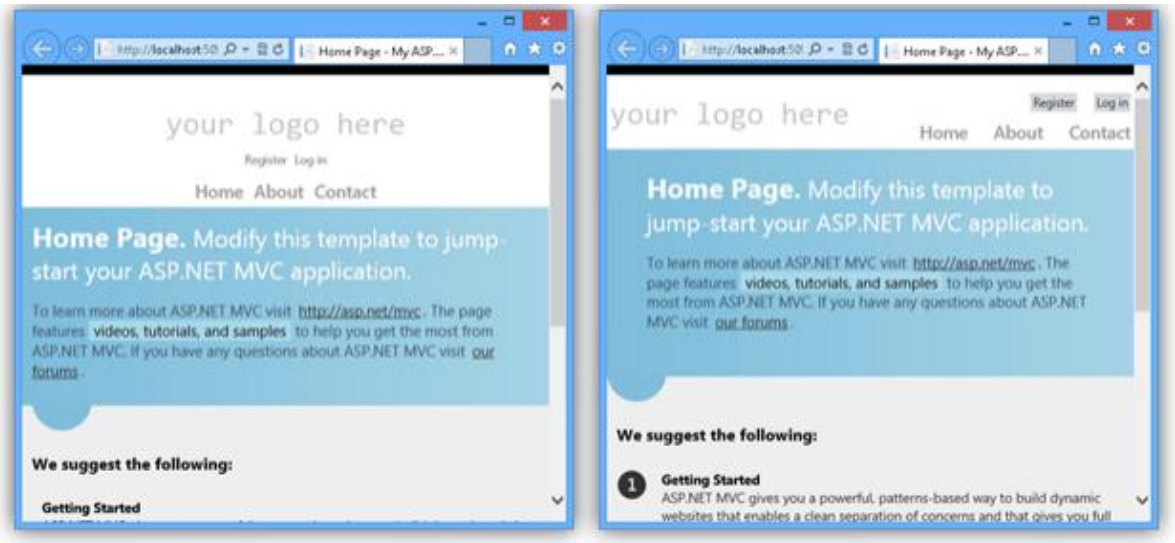
[3]. Mở tập tin Site.css từ Solution Explorer của Visual Studio tại thư mục Content. Nhấn Ctrl + F để mở tích hợp tìm kiếm Visual Studio và nhập @media để tìm vị trí xuất hiện của nó.

[4]. Điều kiện truy vấn @media được định nghĩa theo công việc mẫu hoạt động theo cách này: Khi kích thước cửa sổ của trình duyệt dưới 850px thì các quy tắc CSS áp dụng là những định nghĩa bên trong khối media này.




Định vị các truy vấn media

[5]. Thay thế giá trị thuộc tính max-width trong 850px với 10px , để vô hiệu hóa các rendering thích ứng và nhấn Ctrl + S để lưu thay đổi. Quay trở lại trình duyệt và nhấn CTRL + F5 để làm mới trang với những thay đổi chúng ta đã thực hiện. Chú ý sự khác biệt trong cả hai trang khi điều chỉnh độ rộng của cửa sổ.




Trang trái áp dụng @media còn trang phải thì không

Bây giờ, chúng ta hãy xem những gì xảy ra trên các thiết bị di động:


Trang trái áp dụng @media còn trang phải thì không

Mặc dù chúng ta sẽ nhận thấy rằng những thay đổi khi trang được kết xuất trong một trình duyệt web không phải là rất đáng kể, khi sử dụng một thiết bị di động, sự khác biệt trở nên rõ ràng hơn. Ở phía bên trái của hình ảnh, chúng ta có thể thấy rằng phong cách tùy chỉnh cải thiện khả năng đọc.

Sinh giao diện thích ứng có thể được sử dụng trong nhiều tình huống khác để dễ dàng áp dụng style có điều kiện đến website và giải quyết các vấn đề style phổ biến với cách tiếp cận gọn gàng.

Viewport meta tag và CSS truy vấn media không phải là danh riêng cho ASP.NET MVC 4, vì vậy chúng ta có thể tận dụng lợi thế của các tính năng này trong bất kỳ ứng dụng web nào.

6. Trong Visual Studio, nhấn SHIFT + F5 để gỡ lỗi các ứng dụng.

### 2.3 Bài 2: Tạo ứng dụng Web Photo Gallery

Trong bài tập này, chúng ta sẽ làm việc trên ứng dụng Photo Gallery để hiển thị hình ảnh. Chúng ta sẽ bắt đầu với mẫu dự án ASP.NET MVC 4 sau đó chúng ta sẽ thêm một tính năng để lấy hình ảnh từ một dịch vụ và hiển thị chúng trong trang chủ.

Trong bài tập sau, chúng ta sẽ cập nhật giải pháp để nâng cao cách nó được hiển thị trên các thiết bị di động.

#### 2.3.1 Nhiệm vụ 1: Tạo một dịch vụ Photo giả

Trong nhiệm vụ này, chúng ta sẽ tạo ra một mô hình dịch vụ ảnh để lấy nội dung sẽ được hiển thị trong thư viện ảnh. Để làm điều này, chúng ta sẽ thêm một Controller mới chỉ đơn giản là sẽ trả về một tập tin JSON chứa dữ liệu của mỗi bức ảnh.

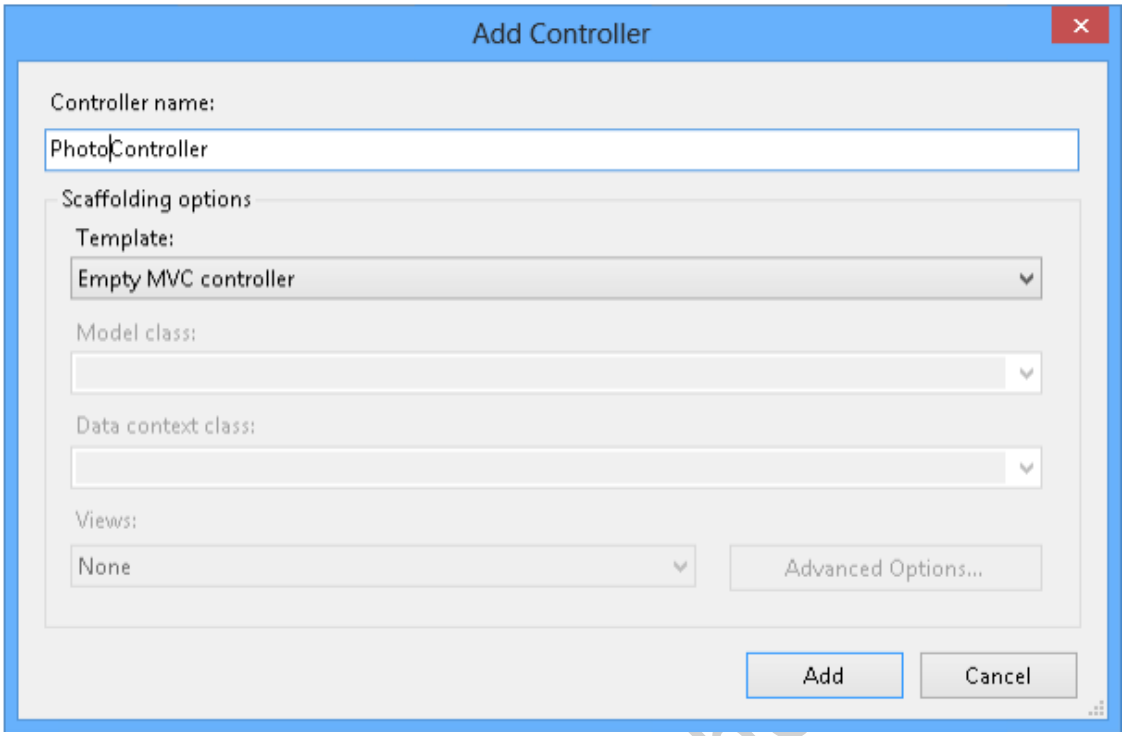
[1]. Mở Visual Studio nếu nó chưa được mở.

[2]. Chọn File | New | Project menu. Trong hộp thoại New Project , chọn mẫu C # Visual | Web trên phần panel bên trái, và chọn ASP.NET MVC Web Application 4. Đặt tên dự án PhotoGallery, chọn một thư mục lưu trữ (hoặc để mặc định) và nhấn OK . Ngoài ra, chúng ta có thể tiếp tục làm việc với dự án này đã được tạo ra trước đó.

[3]. Trong hộp thoại New ASP.NET MVC 4 Project , chọn mẫu dự án Internet Application và nhấn vào OK . Hãy chắc chắn rằng chúng ta đã chọn Razor cho các View của dự án.

[4]. Trong Solution Explorer , kích chuột phải vào thư mục App_Data của dự án của chúng ta, và chọn Add | Existing Item . Duyệt thư mục Source/Assets/App_Data và chọn thêm tập tin Photos.json vào dự án.

[5]. Tạo một bộ điều khiển mới với tên PhotoController . Để làm điều này, kích chuột phải vào trên thư mục Controllers > Add > Controller. Hoàn thành tên cho Controller và chọn mẫu Empty MVC controller sau đó nhấp vào Add .

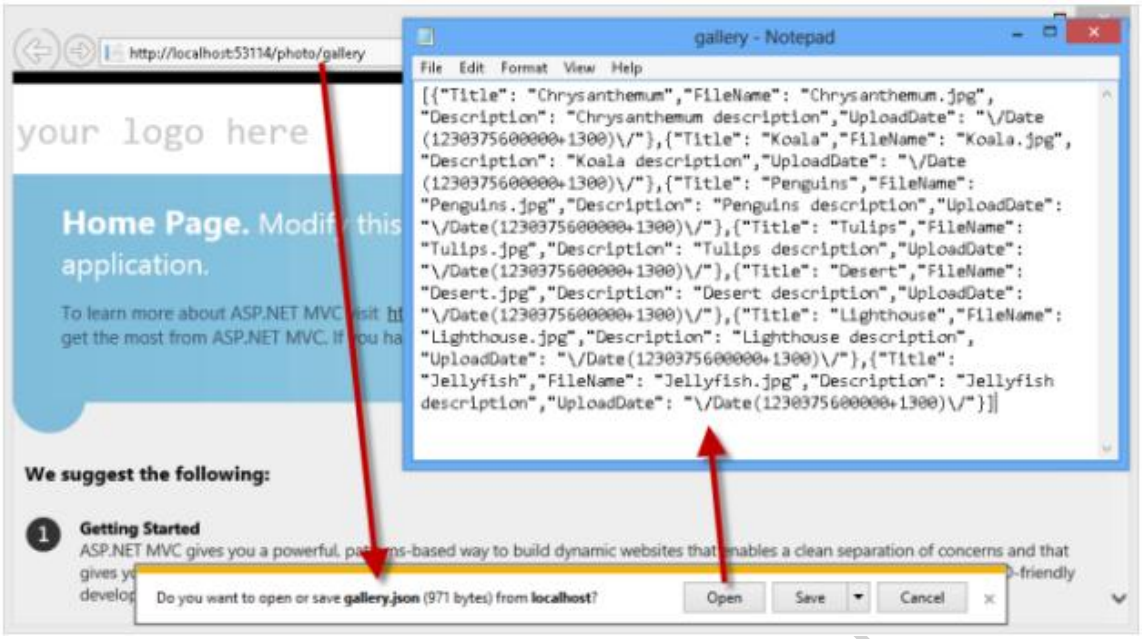



Thêm PhotoController

[6]. Thay thế phương thức Index thành Gallery hành động và trả về nội ung từ tập tin JSON gần đây chúng ta đã thêm vào dự án.

```
public class PhotoController : Controller
{
  public ActionResult Gallery()
  {
    return this.File("~/App_Data/Photos.json", "application/json");
  }
}
```

[7]. Nhấn F5 để chạy giải pháp và sau đó duyệt đến URL sau đây để kiểm tra dịch vụ ảnh: http://localhost:[port]/photo/gallery ([port] giá trị phụ thuộc vào port hiện tại nơi mà ứng dụng đã được đưa ra). URL này cần lấy nội dung của tập tin Photos.json.




Kiểm tra các dịch vụ hình ảnh

Trong thực tế thực hiện, chúng ta có thể sử dụng ASP.NET Web API để thực hiện các dịch vụ thư viện hình ảnh. ASP.NET Web API là một framework giúp dễ dàng xây dựng dịch vụ HTTP phục vụ nhiều dạng client khác nhau bao gồm cả trình duyệt và các thiết bị di động. ASP.NET Web API là một nền tảng lý tưởng cho việc xây dựng các ứng dụng RESTful trên NET Framework..

### 2.3.2 Nhiệm vụ 2: Hiển thị Photo Gallery

Trong nhiệm vụ này, chúng ta sẽ cập nhật trang Home để hiển thị bộ sưu tập ảnh bằng cách sử dụng dịch vụ giả chúng ta tạo ra trong nhiệm vụ đầu tiên của bài tập này.

Chúng ta sẽ thêm tập tin mô hình và cập nhật các View của thư viện ảnh.

[1]. Trong Visual Studio, nhấn SHIFT + F5 để gỡ lỗi các ứng dụng.

[2. Tạo lớp Phôt trong thư mục Models. Để làm điều này, kích chuột phải vào thư mục Models, chọn và nhấp vào Class . Sau đó, đặt tên cho Photo.cs và nhấp vào Add.

[3]. Thêm các thành viên sau đây vào lớp Photo.

```
public class Photo
{
    public string Title { get; set; }

    public string FileName { get; set; }

    public string Description { get; set; }

    public DateTime UploadDate { get; set; }
}
```

[4]. Mở tập tin HomeController.cs từ thư mục Controllers .

[5]. Thêm bằng cách sử dụng các báo cáo sau đây.

```
using System.Net.Http;
using System.Web.Script.Serialization;
using Newtonsoft.Json;
using PhotoGallery.Models;
```

[6]. Cập nhật hành động Index sử dụng HttpClient để lấy dữ liệu thư viện ảnh và sau đó sử dụng JavaScriptSerializer để tuần tự hóa và truyền cho View.

```
public ActionResult Index()
{
    var client = new HttpClient();
    var response = client.GetAsync(Url.Action("gallery", "photo", null, Request.Url.Scheme)).Result;
    var value = response.Content.ReadAsStringAsync().Result;

    var result = JsonConvert.DeserializeObject<List<Photo>>(value);

    return View(result);
}
```

[7]. Mở tập tin Index.cshtml nằm dưới thư mục Views/Home và thay thế tất cả các nội dung với mã sau đây. Mã này duyệt qua tất cả các hình ảnh lấy từ dịch vụ này và hiển thị chúng trong danh sách có thứ tự.

```
@model List<PhotoGallery.Models.Photo>
@{
    ViewBag.Title = "Photo Gallery";
}

<ul class="thumbnails">
    @foreach (var photo in Model)
    {
        <li class="item">
            <a href="@Url.Content("~/photos/" + photo.FileName)">
                <img alt="@photo.Description" src="@Url.Content("~/photos/" + photo.FileName)"
class="thumbnail-border" width="180" />
            </a>
                <span class="image-overlay">@photo.Title</span>
        </li>
    }
</ul>
```

[8]. Trong Solution Explorer , kích chuột phải vào thư mục Content của dự án của chúng ta, và chọn Add | Existing Item . Duyệt đến thư mục Source/Assets/Content của bài lab này và thêm tập tin Site.css. Chúng ta sẽ phải xác nhận thay thế của nó. Nếu chúng ta có tập tin Site.css mở, chúng ta sẽ phải xác nhận để tải lại tập tin này.

[9]. Mở File Explorer và sao chép toàn bộ thư mục Photo nằm dưới Source/Assets của bài lab này vào thư mục gốc của dự án của chúng ta trong Solution Explorer.

[10]. Chạy ứng dụng. Chúng ta sẽ thấy trang chủ hiển thị các bức ảnh trong bộ sưu tập.


Photo Gallery

[11]. Trong Visual Studio, nhấn SHIFT + F5 để gỡ lỗi ứng dụng.

### 2.4 Bài 3: Bổ sung hỗ trợ cho thiết bị di động

Một trong những cập nhật quan trọng trong ASP.NET MVC 4 là sự hỗ trợ cho phát triển điện thoại di động. Trong bài tập này, chúng ta sẽ khám phá tính năng mới ASP.NET MVC 4 cho các ứng dụng điện thoại di động bằng cách mở rộng giải pháp PhotoGallery mà chúng ta đã tạo ra trong bài tập trước.

#### 2.4.1 Nhiệm vụ 1: Cài Mobile jQuery cho dự án ASP.NET MVC 4

[1]. Mở giải pháp Begin nằm ở thư mục/Source/Ex3-MobileSupport/Begin. Nếu không, chúng ta có thể tiếp tục sử dụng giải pháp End thu được bằng cách hoàn thành bài tập trước.

1) Nếu chúng ta đã mở giải pháp Begin, chúng ta sẽ cần phải tải về một số gói NuGet trước khi tiếp tục. Để làm điều này, hãy nhấp vào menu Project và chọn Manage NuGet Packages.

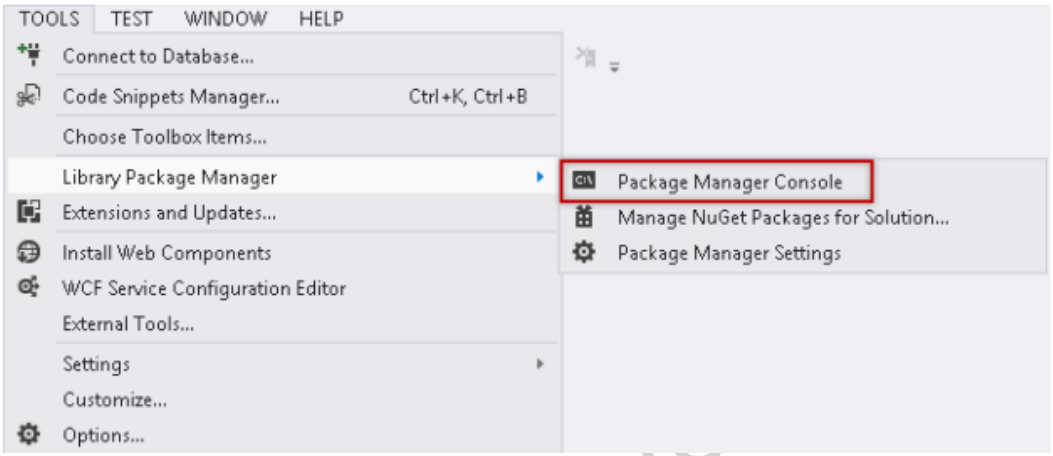
2) Trong hộp thoại Manage NuGet Packages, nhấp vào Restore để tải về các gói bị thiếu.

3) Cuối cùng, dịch giải pháp bằng cách nhấp vào Build | Build Solution .

Lưu ý: Một trong những lợi thế của việc sử dụng NuGet là chúng ta không cần phải kèm theo tất cả các thư viện trong dự án của chúng ta nên giảm quy mô dự án.

Với Power Tools NuGet, bằng cách xác định các phiên bản gói trong file Packages.config, chúng ta có thể tải về tất cả các thư viện cần lần đầu tiên chúng ta chạy các dự án. Đây là lý do tại sao chúng ta sẽ phải thực hiện các bước sau khi chúng ta mở một giải pháp hiện có từ bài lab.

[2]. Mở Package Manager Console bằng cách nhấn vào Tools > Package Manager Library > Package Manager Console .




Mở NuGet Package Manager Console

[3]. Trong Console Package Manager chạy lệnh sau để cài đặt các gói jQuery.Mobile.MVC .

jQuery Mobile là một thư viện mã nguồn mở để xây dựng tối ưu hóa cho giao diện web cảm ứng. NuGet jQuery.Mobile.MVC gói bao gồm những cái hỗ trợ để sử dụng jQuery Mobile với một ứng dụng ASP.NET MVC 4.

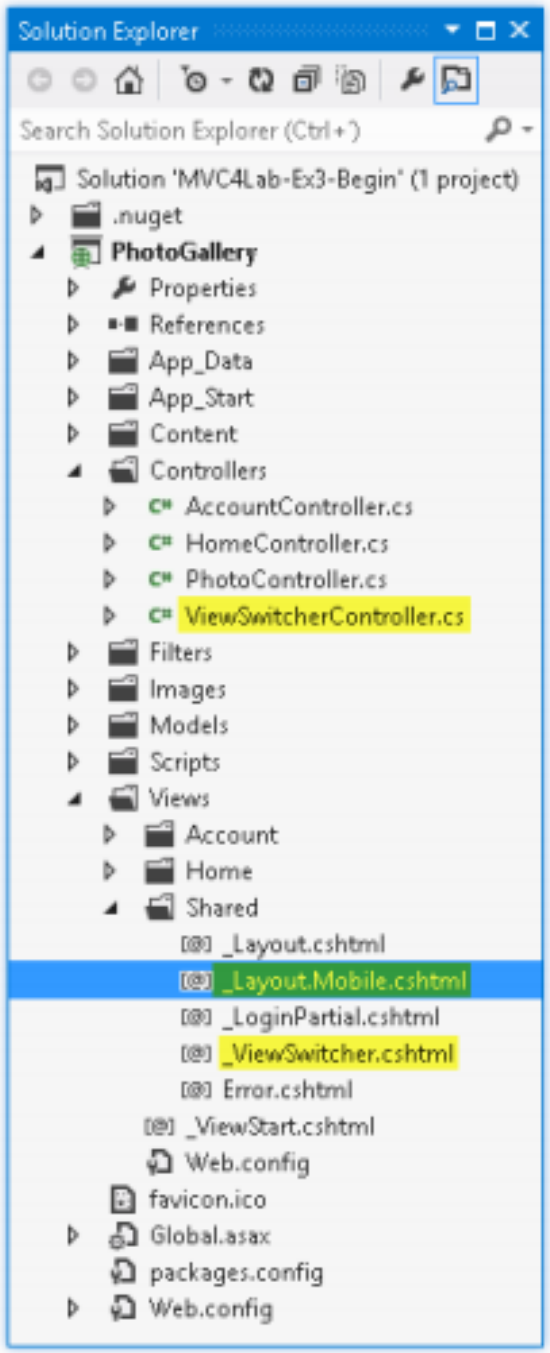
Lưu ý: Bằng cách chạy lệnh sau đây, chúng ta sẽ được tải về thư viện jQuery.Mobile.MVC từ Nuget.

***Install-Package jQuery.Mobile.MVC***

Lệnh này sẽ cài đặt jQuery Mobile và các tập tin helper, bao gồm:

Views/Shared/_Layout.Mobile.cshtml : là một layout cho điện thoại di động được tối ưu hóa cho màn hình nhỏ. Khi trang web nhận được một yêu cầu từ một trình duyệt di động, nó sẽ thay thế layput ban đầu (_Layout.cshtml) với layout này.

Một bộ chuyển View: trong Views/Shared/_ViewSwitcher.cshtml và một Controller ViewSwitcherController.cs . Thành phần này sẽ hiển thị một liên kết trên các trình duyệt di động cho phép người dùng chuyển sang phiên bản máy tính để bàn của trang.




Dự án Photo Gallery với sự hỗ trợ điện thoại di động

Đăng ký gói điện thoại di động. Để làm điều này, mở tập tin Global.asax.cs và thêm dòng sau đây.

```
protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();

    WebApiConfig.Register(GlobalConfiguration.Configuration);
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);

    BundleMobileConfig.RegisterBundles(BundleTable.Bundles);
    AuthConfig.RegisterAuth();
}
```

[4]. Chạy ứng dụng bằng cách sử dụng một trình duyệt web máy tính để bàn.

[5]. Mở Windows Phone 7 Emulator, nằm trong Start Menu | All Programs | Windows Phone . SDK 7.1 | Windows Phone Emulator.

Trong màn hình bắt đầu điện thoại, mở Internet Explorer. Kiểm tra URL nơi mà các ứng dụng bắt đầu và duyệt đến URL đó với trình duyệt điện thoại (ví dụ: http://localhost: [PortNumber]/).Chúng ta sẽ nhận thấy rằng ứng dụng của chúng ta sẽ trông khác với Windows PhoneEmulator khi jQuery.Mobile.MVC đã tạo ra tài sản mới trong dự án của chúng ta vì các View đã được tối ưu hóa cho các thiết bị di động. Chú ý thông báo ở phía trên cùng của điện thoại, hiển thị các liên kết chuyển sang xem máy tính để bàn. Ngoài _Layout.Mobile.cshtml ra có có một vài layout khác cũng được tạo ra.

Lưu ý: Cho đến nay, không có liên kết để trở lại View di động. Nó sẽ được bổ sung trong các phiên bản sau này.


Giao diện của trang chủ Photo Gallery

[6]. Trong Visual Studio, nhấn SHIFT + F5 để gỡ lỗi các ứng dụng.

### 2.4.2 Nhiệm vụ 2: Tạo View cho điện thoại di động

Trong nhiệm vụ này, chúng ta sẽ tạo ra một phiên bản View cho điện thoại di động Index.cshtml có nội dung phù hợp xuất hiện tốt hơn trong các thiết bị di động.

[1]. Sao chép Views/Home/Index.cshtml và dán để tạo ra một bản sao, đổi tên tập tin mới để Index.Mobile.cshtml .

[2]. Mở Index.Mobile.cshtml mới tạo ra và thay thế thẻ <ul> hiện có với mã sau đây. Bằng cách này, chúng ta sẽ cập nhật thẻ <ul> với chú thích dữ liệu của jQuery Mobile để sử dụng các mẫu giao diện điện thoại di động từ jQuery.

```
<ul data-role="listview" data-inset="true" data-filter="true">
```

**Lưu ý:** Thuộc tính data-role thiết lập giá trị listview sẽ sinh danh sách bằng cách sử dụng style listview.

Thuộc tính data-inset thiết lập giá trị true sẽ hiển thị danh sách được bo tròn góc.

Thuộc tính data-filter thiết lập giá trị true sẽ tạo ra một hộp tìm kiếm.

Chúng ta có thể tìm hiểu thêm về các qui ước các thuộc tính Mobile trong jQuery ở tài liệu tại website: http://jquerymobile.com/demos/1.1.1/

[3]. Nhấn CTRL + S để lưu thay đổi.

[4]. Chuyển sang Windows Phone Emulator và làm tươi trang web. Chú ý cái "nhìn và cảm nhận" mới của danh sách thư viện ảnh, cũng như các hộp tìm kiếm mới nằm trên đầu trang. Sau đó, gõ một từ vào hộp tìm kiếm (ví dụ, Tulips ) để bắt đầu tìm kiếm trong bộ sưu tập ảnh.


Gallery sử dụng listview style có bộ lọc

***Tóm tắt:*** chúng ta đã sử dụng cách thức Mobile hóa View để tạo ra một bản sao của View Index với hậu tố "Mobile". Hậu tố này báo cho ASP.NET MVC 4 biết để sinh giao diện từ View này khi yêu cầu truy xuất từ thiết bị di động. Ngoài ra, chúng ta đã cập nhật phiên bản di động của View Index để tận dụng sức mạnh của jQuery nhằm tăng cường cho trang web có giao diện đẹp đẽ trên các thiết bị di động.

[5]. Quay trở lại Visual Studio và mở Site.Mobile.css đặt trong thư mục Content.

[6]. Sửa chữa vị trí của tiêu đề hình ảnh làm cho nó hiển thị ở phía bên phải của hình ảnh. Để làm điều này, thêm đoạn mã sau vào tập tin Site.Mobile.css

```
.ui-li .ui-btn-inner a.ui-link-inherit, .ui-li-static.ui-li {
    padding: 0px !important;
}

li.item span.image-overlay
{
    position:relative;
    left:100px;
    top:-40px;
    height:0px;
    display:block;
}
```

[7]. Nhấn CTRL + S để lưu thay đổi.

[8]. Chuyển về Windows Phone Emulator và làm mới trang web. Chú ý tiêu đề ảnh đúng vị trí.


Tiêu đề vị trí ở phía bên phải của hình ảnh

### 2.4.3 Nhiệm vụ 3: jQuery Mobile Themes

Mỗi layout và widget trong jQuery Mobile được thiết kế xung quanh một framework CSS hướng đối tượng mới làm cho nó có thể áp dụng thống nhất một mẫu thiết kế hoàn chỉnh đến các trang web của ứng dụng.

Theme mặc định của jQuery Mobile bao gồm 5 swatches được đặt trưng bởi các chữ cái (a, b, c, d, e) để tham chiếu nhanh.

Trong nhiệm vụ này, chúng ta sẽ cập nhật các layout điện thoại di động để sử dụng một chủ đề khác hơn so với mặc định.

[1]. Chuyển về Visual Studio.

[2]. Mở _Layout.Mobile.cshtml trong Views/Shared .

[3]. Tìm các thẻ div có thuộc tính data-role=”page” và cập nhật thuộc tính datatheme=”e”.

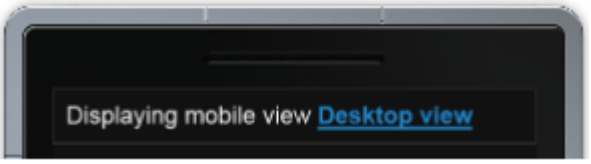
[4]. Nhấn CTRL + S để lưu thay đổi.

[5]. Làm mới trang web trong Windows Phone Emulator và nhận thấy chương trình màu sắc mới.


Layout điện thoại di động với một màu sắc khác

### 2.4.4 Nhiệm vụ 4: View-Switcher và ghi đè trình duyệt

Một quy ước đối với các trang web tối ưu trên điện thoại di động là thêm một liên kết để chuyển sang phiên bản Desktop. Gói JQuery.Mobile.MVC bao gồm một viewswitcher với mục đích này được sử dụng trong View _Layout.Mobile.cshtml.



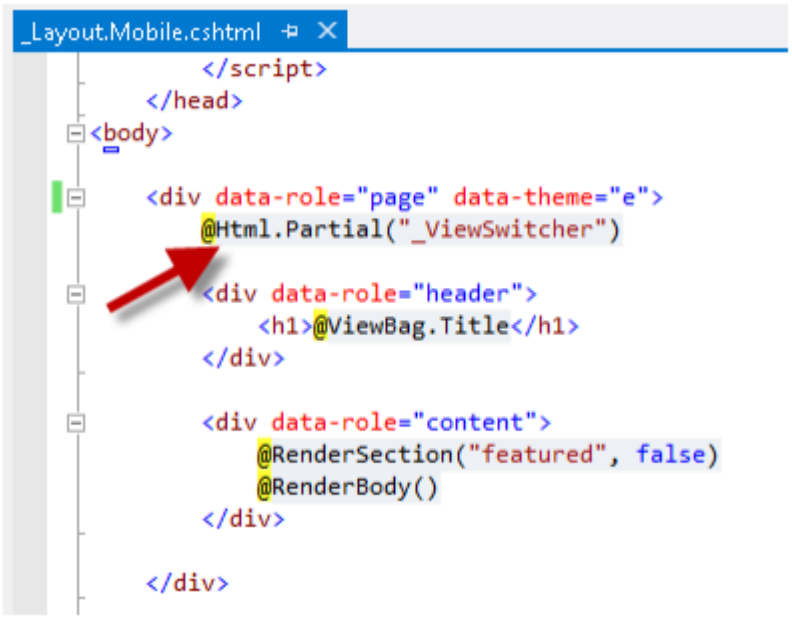

Liên kết để chuyển sang View trên Desktop

Bộ chuyển mạch sử dụng tính năng mới được gọi là ghi đè trình duyệt . Tính năng này cho phép ứng dụng của chúng ta đối xử các yêu cầu đến từ các trình duyệt khác nhau.

Trong nhiệm vụ này, chúng ta sẽ khám phá thực hiện mẫu của một bộ chuyển mạch được bổ sung bởi jQuery.Mobile.MVC và tính năng ghi đè trình duyệt mới trong ASP.NET MVC 4.

[1]. Chuyển về Visual Studio.

[2]. Mở _Layout.Mobile.cshtml trong Views/Shared và chú ý bộ chuyển view được tham chiếu như là một partial view.




Layout điện thoại di động sử dụng View-Switcher

[3]. Mở partial view _ViewSwitcher.cshtml.

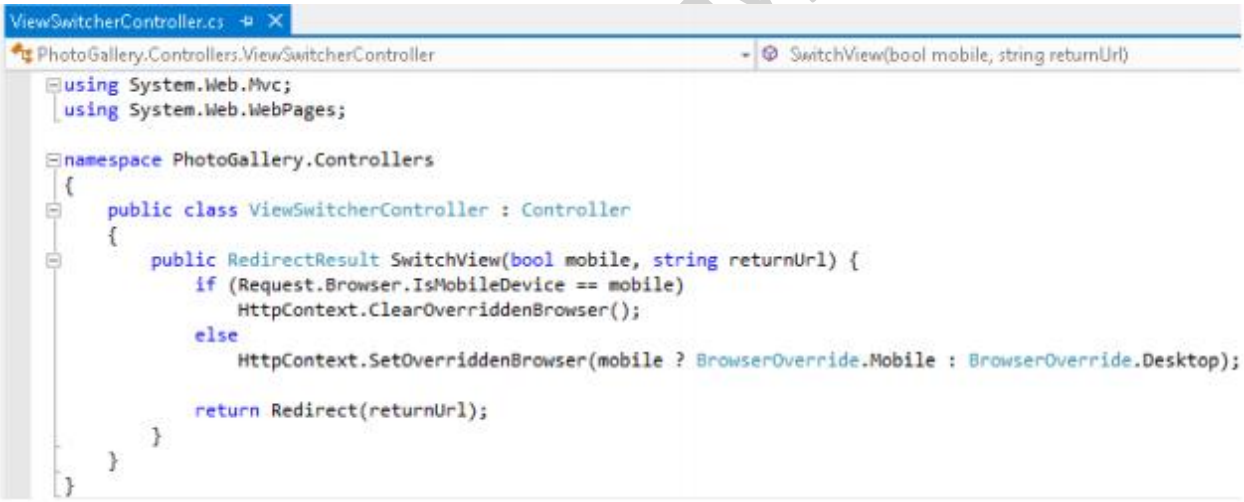
Partial View sử dụng phương thức mới ViewContext.HttpContext.GetOverriddenBrowser() để xác định nguồn gốc yêu cầu web và hiển thị các liên kết tương ứng để chuyển đổi giữa View của Desktop hay Mobile.

Phương thức GetOverridenBrowser() trả về HttpBrowserCapabilitiesBase tương ứng với trình duyệt hiện đang gửi yêu cầu. Chúng ta có thể sử dụng giá trị này để nhận được các đặc tính như IsMobileDevice .


Partial View ViewSwitcher

[4]. Mở lớp ViewSwitcherController.cs nằm trong thư mục Controllers. Chúng ta thấy rằng phương thức hành động SwitchView() được gọi bởi liên kết trong ViewSwitcher và chú ý đến các phương thức mới của lớp HttpContext.

- ✓ HttpContext.ClearOverridenBrowser() loại bỏ tất cả các trình duyệt bị ghi đè.
- ✓ HttpContext.SetOverridenBrowser() ghi đè lên giá trị của trình duyệt đang yêu cầu.




ViewSwitcherController

Ghi đè trình duyệt là tính năng cốt lõi của ASP.NET MVC 4 có sẵn ngay cả khi chúng ta không cài đặt các gói jQuery.Mobile.MVC. Tuy nhiên, tính năng này chỉ ảnh hưởng đến View, Layout, và Partial View mà không ảnh hưởng đến bất kỳ các tính năng phụ thuộc vào đối tượng Request.Browser.

### 2.4.5 Nhiệm vụ 5: Thêm View-Switcher vào View Desktop

Trong nhiệm vụ này, chúng ta sẽ cập nhật Layout máy tính để bàn để bao gồm bộ chuyển view. Điều này cho phép người sử dụng điện thoại di động quay trở lại View điện thoại di động khi đang xem trên trình duyệt của máy tính để bàn.

[1]. Làm mới trang web trong Windows Phone Emulator .

[2]. Nhấp vào liên kết ở trên cùng của thư viện hình View Desktop . Chú ý không có bộ chuyển viêw trong giao diện máy tính để bàn cho phép chúng ta quay trở lại View điện thoại di động.

[3]. Quay trở lại Visual Studio và mở _Layout.cshtml.

[4]. Tìm phần đăng nhập và chèn một lời gọi đên Partial View _ViewSwitcher dưới View _LogOnPartial. Sau đó, nhấn Ctrl + S để lưu thay đổi.

```
<div class="float-right">
    <section id="login">
        @Html.Partial("_LogOnPartial")

        @Html.Partial("_ViewSwitcher")
    </section>
    <nav>
```

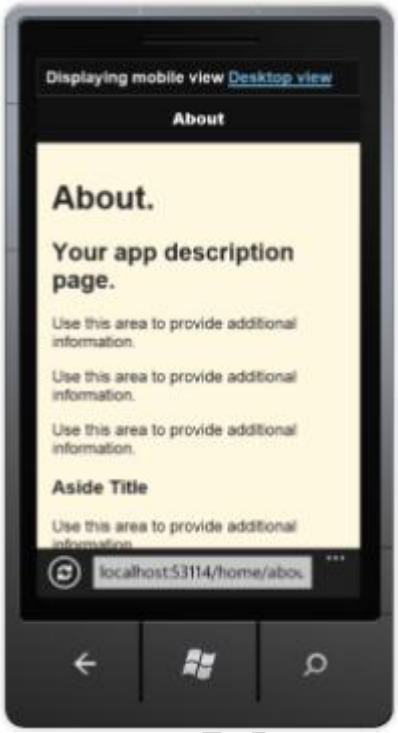
[5]. Nhấn CTRL + S để lưu thay đổi.

[6]. Làm mới trang trong Emulator Windows Phone và nhấp đúp vào màn hình để phóng to. Chú ý rằng trang chủ hiện nay cho thấy Mobile View liên kết chuyển từ điện thoại di động đến máy tính để bàn.


View Switcher hiển thị trong Desktop View

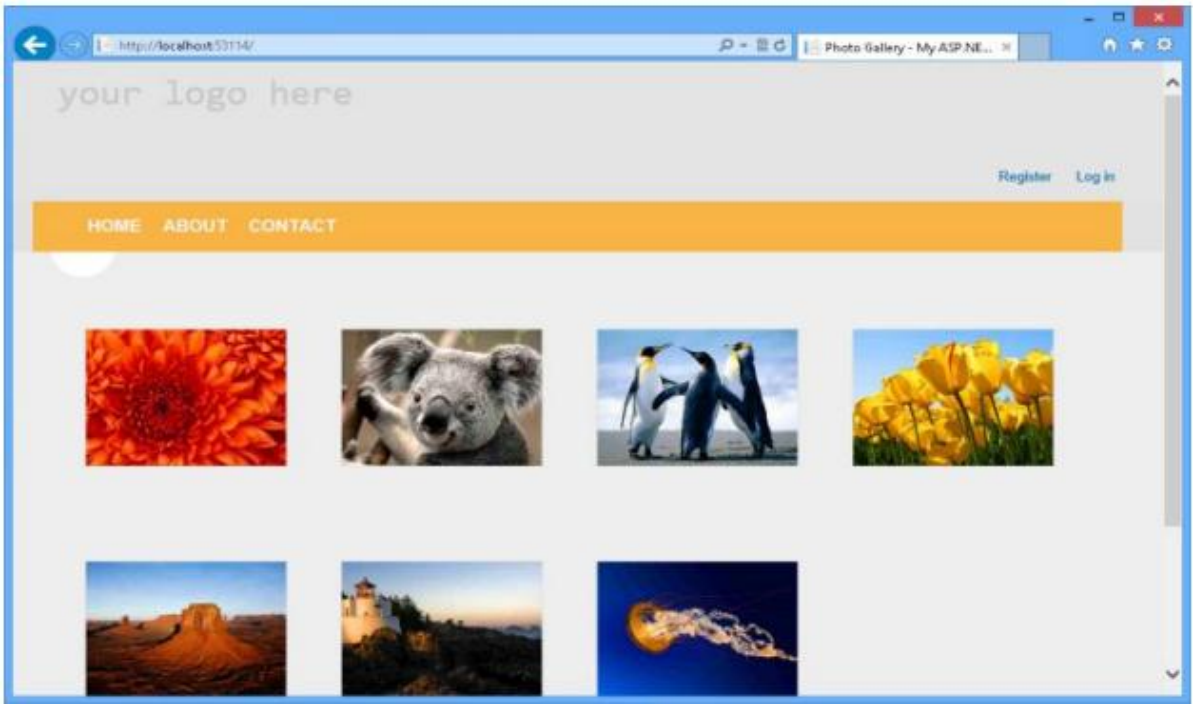
[7]. Chuyển sang View Mobile lần nữa và duyệt đến trang About

(http://localhost[port]/Home/About). Chú ý rằng, ngay cả khi chúng ta không tạo ra View About.Mobile.cshtml, trang About cũng được hiển thị bằng cách sử dụng layout điện thoại di động (_Layout.Mobile.cshtml).




Trang About

[8]. Cuối cùng, mở trang web trong một trình duyệt web máy tính để bàn. Chú ý rằng không có một chút cập nhật nào trước đó ảnh hưởng đến View của máy tính để bàn.




Xem trên máy tính để bàn

### 2.4.6 Nhiệm vụ 6: Tạo chế độ hiển thị mới

Tính năng chế độ hiển thị mới để ứng dụng chọn views tùy thuộc vào trình duyệt gửi yêu cầu. Ví dụ, nếu một trình duyệt máy tính để bàn yêu cầu trang chủ thì ứng dụng sẽ trả lại Views/Home/Index.cshtml. Sau đó, nếu một trình duyệt di động yêu cầu trang chủ thì ứng dụng sẽ trả lại Views/Home/Index.Mobile.cshtml.

Trong nhiệm vụ này, chúng ta sẽ tạo ra một Layout tùy chỉnh cho thiết bị iPhone và chúng ta sẽ giả lập yêu cầu từ thiết bị iPhone. Để làm điều này, chúng ta có thể sử dụng một trình giả lập iPhone emulator/simulator (như Electric Mobile Simulator) hoặc một trình duyệt với các tiện ích thay đổi user-agent.

[1]. Trong Visual Studio, nhấn SHIFT + F5 để gỡ lỗi các ứng dụng.

[2]. Mở Global.asax.cs và thêm lệnh using sau đây vào.

using System.Web.WebPages;

[3]. Tăng cường thêm mã sau vào phương thức Application_Start.

```
protected void Application_Start()
{
    // ...

    DisplayModeProvider.Instance.Modes.Insert(0, new DefaultDisplayMode("iPhone")
    {
        ContextCondition = context =>
            context.Request.UserAgent != null &&
            context.Request.UserAgent.IndexOf("iPhone", StringComparison.OrdinalIgnoreCase) >= 0
    });
}
```

Chúng ta đã đăng ký mới DefaultDisplayMode có tên là "iPhone" , ở bên trong danh sách tĩnh

DisplayModeProvider.Instance.Modes. Danh sách này sẽ được sử dụng để so khớp với mỗi yêu cầu gửi đến. Nếu yêu cầu gửi đến có chứa chuỗi "iPhone", ASP.NET MVC sẽ tìm thấy những View có tên chứa hậu tố "iPhone". Tham số 0 cho biết là chế độ mới. View này cụ thể hơn so với ".mobile" chung chung phù hợp với các yêu cầu từ các thiết bị di động.

Sau khi mã này chạy, khi một trình duyệt iPhone tạo ra một yêu cầu, ứng dụng của chúng ta sẽ sử dụng Views/Shared/_Layout.iPhone.cshtml mà chúng ta sẽ tạo ra trong các bước tiếp theo.

***Lưu ý:*** Trường hợp này chỉ để minh họa yêu cầu dành cho iPhone đã được đơn giản hóa cho mục đích giới thiệu và có thể đạt như mong đợi cho tất cả các chuỗi user-agent iPhone (ví dụ phân biệt hoa - thường).

[1]. Tạo một bản sao của tập tin _Layout.Mobile.cshtml trong Views/Shared và đổi tên bản sao thành _Layout.iPhone.csthml .

[2]. Mở _Layout.iPhone.csthml chúng ta đã tạo ở bước trước.

[3]. Tìm các phần tử div> với thuộc tính data-role=”page” và thay đổi thuộc tính datatheme=" a ".

```
<body>
    <div data-role="page" data-theme="a">
        @Html.Partial("_ViewSwitcher")
...
```

Bây giờ chúng ta có 3 Layout trong ứng dụng ASP.NET MVC 4 của chúng ta:

a) _Layout.cshtml : layout mặc định được sử dụng cho các trình duyệt máy tính để bàn.

b) _Layout.mobile.cshtml : layout mặc định được sử dụng cho các thiết bị di động.

c) _Layout.iPhone.cshtml : layout cụ thể cho thiết bị iPhone, bằng cách sử dụng các màu sắc khác nhau để phân biệt với _Layout.mobile.cshtml.

[4]. Nhấn F5 để chạy các ứng dụng và duyệt các trang web trong Windows Phone Emulator .

[5]. Mở một mô phỏng iPhone và duyệt đến trang web. Chú ý rằng mỗi điện thoại bằng cách sử dụng các mẫu cụ thể.


Sử dụng các View khác nhau cho mỗi thiết bị di động

## 2.5 Bài 4: Sử dụng Controller bất đồng bộ

Microsoft NET Framework 4.5 giới thiệu các tính năng ngôn ngữ mới trong C# và Visual Basic để cung cấp một nền tảng mới cho lập trình NET bất đồng bộ. Nền tảng mới này làm cho lập trình bất đồng bộ tương tự và đơn giản như lập trình đồng bộ.

Chúng ta đã có thể viết các phương thức hành động bất đồng bộ trong ASP.NET MVC 4 sử dụng lớp AsyncController. Chúng ta có thể sử dụng các phương thức hành động bất đồng bộ thực thi trong thời gian dài. Điều này tránh sự ngăn chặn các máy chủ Web khỏi thực hiện các công việc trong khi yêu cầu đang được xử lý. Lớp AsyncController thường được sử dụng cho các lời gọi dịch vụ Web thực hiện trong thời gian dài.

Bài tập này giải thích các vấn đề cơ bản của hoạt động không đồng bộ trong ASP.NET MVC 4. Nếu chúng ta muốn hiểu sâu hơn, chúng ta có thể đọc các bài viết sau đây:http://msdn.microsoft.com/en-us/library/ee728598%28v=vs.100%29.aspx

### 2.5.1 Nhiệm vụ 1: Cài đặt một điều khiển không đồng bộ

[1]. Mở giải pháp Begin nằm ở thư mục Source/Ex4-Async/Begin. Nếu không, chúng ta có thể tiếp tục sử dụng giải pháp End thu được bằng cách hoàn thành bài tập trước.

a) Nếu chúng ta đã mở được giải pháp Begin được cung cấp, chúng ta sẽ cần phải tải về một số gói NuGet mất liên lạc trước khi tiếp tục. Để làm điều này, hãy nhấp vào menu Project và chọn Manage NuGet Packages.

b) Trong hộp thoại Manage NuGet Packages, nhấp vào Restore để tải về các gói còn thiếu.

c) Cuối cùng, dịch giải pháp bằng cách nhấp vào Build | Build Solution.

[2]. Mở lớp HomeController.cs từ thư mục Controllers.

[3]. Thêm lệnh using vào đầu file.

```
using System.Threading.Tasks;
```

[4]. Cập nhật lớp HomeController để kế thừa từ AsyncController . Các bộ điều khiển xuất phát từ AsyncController cho phép ASP.NET để xử lý các yêu cầu không đồng bộ và chúng vẫn có thể phục vụ các phương thực hành động đồng bộ.

```
public class HomeController: AsyncController
{
```

[5]. Thêm từ khóa async vào trước phương thức Index làm cho nó trả về kiểu <ActionResult>

```
public async Task<ActionResult> Index ()
{
    ...
```

***Lưu ý:*** Từ khóa async là một trong những từ khoá mới của NET Framework 4.5 cung cấp, nó nói với trình biên dịch rằng phương thức này có chứa mã không đồng bộ. Task đối tượng đại diện cho một hoạt động không đồng bộ có thể hoàn thành tại một thời điểm nào đó trong tương lai.

[6]. Thay thế lời gọi client.GetAsync () với phiên bản async đầy đủ bằng cách sử dụng từ khóa await như hình dưới đây.

```
public async Task<ActionResult> Index()
{
    var client = new HttpClient();
    var response = await client.GetAsync(Url.Action("gallery", "photo", null, Request.Url.Scheme));
    ...
```

***Lưu ý:*** Trong các phiên bản trước, chúng ta đã được sử dụng thuộc tính Result của đối tượng Task để khóa thread cho đến khi kết quả được trả về (đồng bộ).

Thêm từ khóa await để báo cho trình biên dịch về việc chờ đợi cho nhiệm vụ trả về từ lời gọi phương thức không đồng bộ. Điều này có nghĩa rằng phần còn lại của mã này sẽ được thực hiện như là một lời gọi lại chỉ sau khi phương thức hoàn tất chờ đợi.

Một điều cần ghi nhận là chúng ta không cần phải thay đổi khối try-catch để làm công việc này: các trường hợp ngoại lệ xảy ra ở hậu trường nền hoặc ở phía trước đều bị bắt mà không cần thêm bất kỳ công việc nào bằng cách sử dụng một bộ xử lý được cung cấp bởi framewwork.

[7]. Thay đổi mã để tiếp tục với việc thực hiện không đồng bộ bằng cách thay thế các dòng với mã mới như hình dưới đây

```
public async Task<ActionResult> Index()
{
    var client = new HttpClient();
    var response = await client.GetAsync(Url.Action("gallery", "photo", null, Request.Url.Scheme));
    var value = await response.Content.ReadAsStringAsync();
    var result = await JsonConvert.DeserializeObjectAsync<List<Photo>>(value);

    return View(result);
}
```

[8]. Chạy ứng dụng. Chúng ta sẽ thấy không có thay đổi lớn, nhưng mã của chúng ta sẽ không bị khóa trong khi chờ đợi các nguồn tài nguyên máy chủ và cải thiện hiệu suất.

***Lưu ý:*** Chúng ta có thể tìm hiểu thêm về các tính năng mới lập trình không đồng bộ trong bài lab "Lập trình không đồng bộ trong NET 4.5 với C# và Visual Basic. bao gồm trong Visual Studio Training Kit ".

### 2.5.2 Nhiệm vụ 2: Xử lý thời hạngi với Cancelation Tokens

Phương thức hành động không đồng bộ trả về Task có hỗ trợ thời hạng.

Trong nhiệm vụ này, chúng ta sẽ cập nhật mã phương thức Index để xử lý một kịch bản thời gian bằng cách sử dụng mã thông báo hủy bỏ.

[1]. Quay trở lại Visual Studio và nhấn SHIFT + F5 để ngăn chặn gỡ lỗi.

[2]. Thêm dòng sau bằng cách sử dụng báo cáo để các HomeController.cs tập tin.

```
using System.Threading;
```

[3]. Cập nhật hành động Index nhận một đối số CancellationToken .

```
Public async Task<ActionResult> Index(CancellationToken cancellationToken)
{
    ...
```

[4]. Cập nhật lời gọi GetAsync () để truyền cancellationToken.

```
public async Task<ActionResult> Index(CancellationToken cancellationToken)
{
    var client = new HttpClient();
    var response = await client.GetAsync(Url.Action("gallery", "photo", null, Request.Url.Scheme),
cancellationToken);
    var value = await response.Content.ReadAsStringAsync();
    var result = await JsonConvert.DeserializeObjectAsync<List<Photo>>(value);

    return View(result);
}
```

[5]. Đánh dấu phương thức Index() với thuộc tính AsyncTimeout với thiết lập thời hạng là 500 mili giây và đồng thời đánh dấu thuộc tính HandleError để cấu hình để xử lý TaskCanceledException bằng cách chuyển hướng đến một View TimedOut.

```
[AsyncTimeout (500)]
[HandleError(ExceptionType=typeof(TimeoutException), View="TimedOut" )]
Public async Task<ActionResult> Index(CancellationToken cancellationToken)
{
```

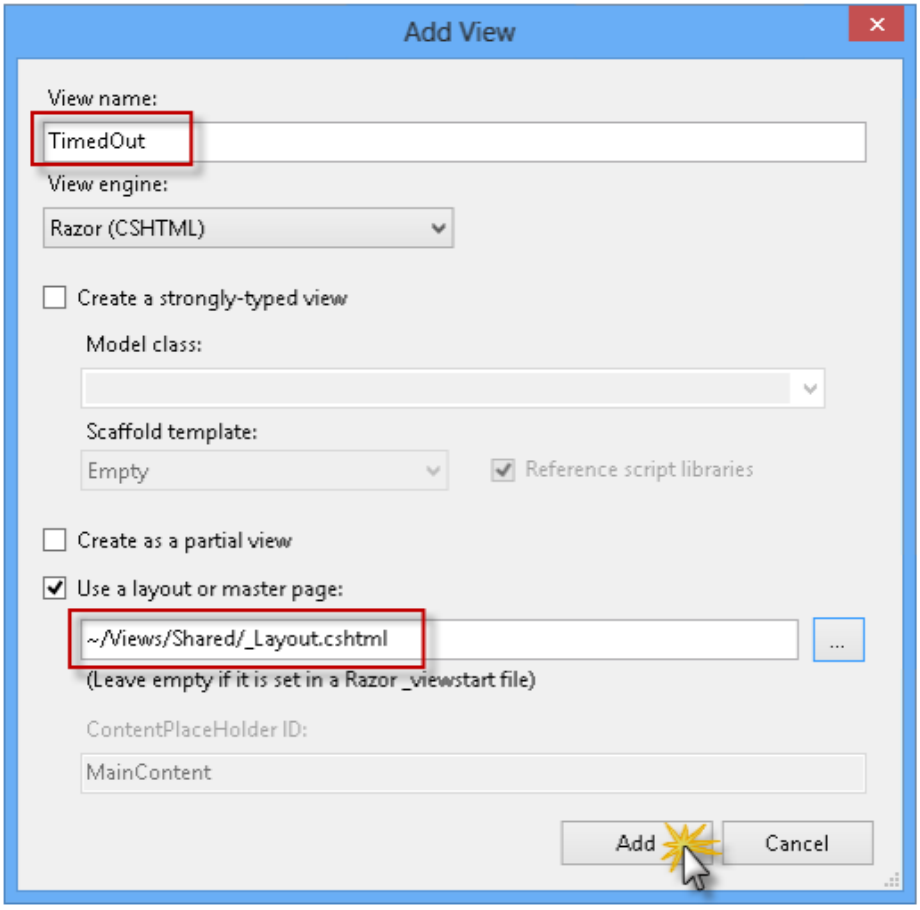
[6]. Mở lớp PhotoController và cập nhật phương thức Gallery để kéo dài thi hành 1000 miliseconds (1 giây) để mô phỏng một nhiệm vụ chạy lâu.

```
public ActionResult Gallery ()
{
    System.Threading.Thread.Sleep (1000);

    Return File( "~/App_Data/Photos.json" , "application/json" );
}
```

[7]. Mở tập tin Web.config và cho phép tùy chỉnh lỗi bằng cách thêm vào các thẻ sau đây.

```
< system.web >
  < customErrors  mode = "On" > </ customErrors >
  ...
```

[8]. Tạo ra một View mới trong Views/Shared có tên là TimedOut và sử dụng layout mặc định. Trong Solution Explorer, nhấn phải chuột vào Views/Shared và chọn Add | View .

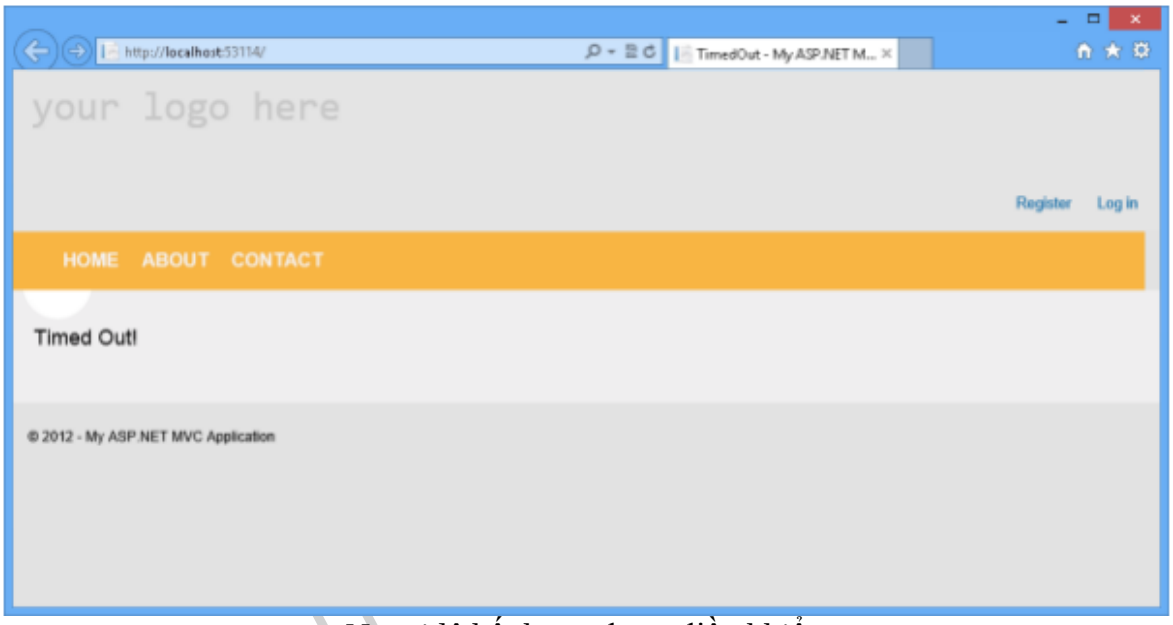



Sử dụng View khác nhau cho mỗi thiết bị di động

[9]. Cập nhật nội dung View TimedOut như hình dưới đây.

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

<h2> Time Out! </h2>
```

[1]0. Chạy ứng dụng và điều hướng đến URL gốc. Do chúng ta đã bổ sung thêm lệnh Thread.Sleep 1000 mili giây nên chúng ta sẽ nhận được một thông báo lỗi lỗi thời hạng (Time Out) được tạo ra bởi thuộc tính AsyncTimeout và bắt lỗi bởi thuộc tính HandleError .




Ngoại lệ hết hạng được điều khiển

## 2.6 Tóm tắt

Trong bài lab này, chúng ta đã quan sát thấy một số các tính năng mới cung cấp trong ASP.NET MVC 4. Các khái niệm sau đây đã được thảo luận:

- ✓ Tận dụng lợi thế của những cải tiến trong dự án ASP.NET MVC mẫu, bao gồm cả các mẫu dự án ứng dụng trên điện thoại di động mới
- ✓ Sử dụng thuộc tính viewport của HTML5 và truy vấn media của CSS để cải thiện hiển thị trên các thiết bị di động
- ✓ Điện thoại di động sử dụng jQuery để cải tiến tiến bộ và xây dựng tối ưu hóa cảm ứng giao diện web
- ✓ Tạo View cho các thiết bị di động cụ thể.

- ✓ Sử dụng view-switcher để chuyển đổi giữa các View điện thoại di động và máy tính để bàn trong ứng dụng
- ✓ Tạo bộ điều khiển không đồng bộ sử dụng hỗ trợ của Task