Lập trình Ứng dụng Doanh nghiệp .NET

Giới thiệu học phần

- Số tín chỉ: 3 tín chỉ

- Mục tiêu:

+ ***Kiến thức***: Trang bị cho sinh viên cách xây dựng trang web bằng công nghệ ASP.NET, lập trình thao tác với CDSL, hiểu về điện toán đám mây.

+ ***Kỹ năng***: Sinh viên biết vận dụng các kiến thức về web, các ngôn ngữ HTML, ASP.NET, SQL Server, C# và các công cụ thiết kế web để xây dựng một trang web.

+ ***Năng lực tự chủ và trách nhiệm***: Thấy được ý nghĩa của các bài học từ đó có thái độ chủ động động, tích cực hăng say trong quá trình học tập, nghiên cứu.

- Hình thức thi: Thực hành phòng máy

Chương 1. Tổng quan về ASP.NET

Nội dung

- Định nghĩa và mô tả về các layer trong ứng dụng Web
- Giải thích về cấu trúc của một ứng dụng ASP.NET MVC
- Trình bày về các công nghệ hỗ trợ ASP.NET MVC
- Giải thích và mô tả cách tạo ứng dụng Web trên Visual Studio 2017

Tổng quát về phát triển ứng dụng Web

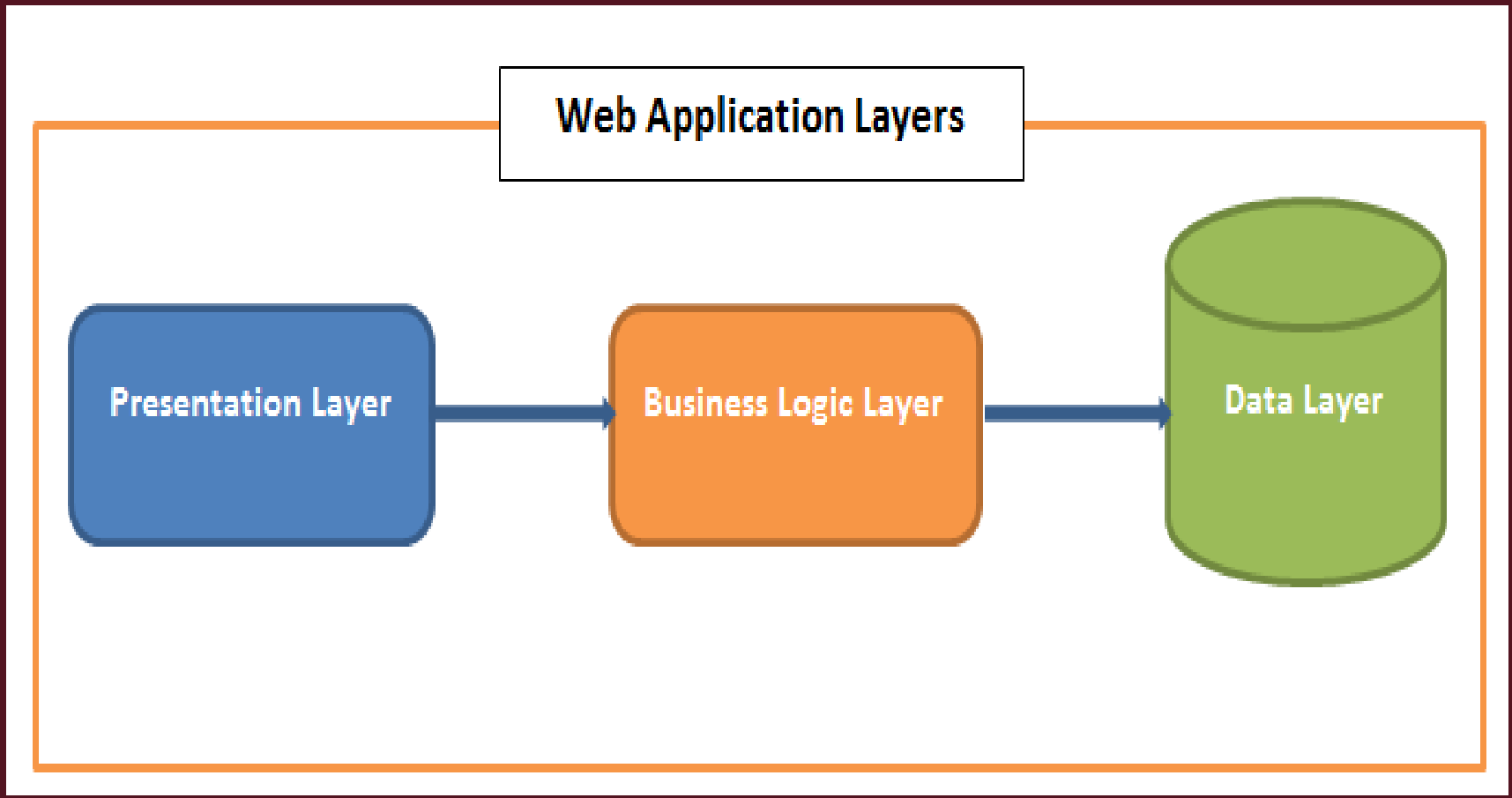
- **Ứng dụng Web**:
  - Là các ứng dụng chạy trên Web server và được truy cập trên các trình duyệt web.
  - Cho phép chia sẻ và truy cập thông tin Internet ở bất cứ đâu và tại bất cứ thời điểm nào.
  - Cho phép thực hiện các giao dịch thương mại, hay được gọi là ứng dụng E-commerce.

Các layer trong ứng dụng Web

- Các ứng dụng Web thường được chia thành 3 layer:
  - **Presentation layer**: Cho phép người dùng tương tác với ứng dụng.
  - **Business logic layer**: Điều khiển luồng thực thi và giao tiếp giữa Presentation layer và Data layer.
  - **Data layer**: Cung cấp dữ liệu được lưu trữ trong cơ sở dữ liệu cho Business logic layer.

Các layer trong ứng dụng Web

- Hình dưới đây biểu thị các layer trong ứng dụng Web:



Các kiến trúc của ứng dụng Web

- Kiến trúc của một ứng dụng phụ thuộc vào hệ thống mà các layer của ứng dụng được phân phối và giao tiếp với nhau.
- Một ứng dụng có thể dựa theo 1 trong các loại kiến trúc sau:
  - **Đơn tầng**: cả 3 layer được tích hợp với nhau và được cài đặt trên 1 máy tính đơn.
  - **Hai tầng**: 3 layer được phân bổ trên 2 tầng, 1 client và 1 server.
  - **Ba tầng**: 3 lớp của ứng dụng được phân bổ trên khắp các máy tính khác nhau.
  - **Đa tầng**: các thành phần của kiến trúc 3 tầng được phân tách rõ ràng hơn.

Các kiểu trang Web

- Một ứng dụng Web chứa nhiều trang Web.
- Một trang Web có thể được phân thành 2 loại:
  - **Trang Web tĩnh**: Chỉ chứa ngôn ngữ đánh dấu siêu văn bản (HTML) để trình bày nội dung tới người dùng.
  - **Trang Web động**: Chứa HTML kết hợp với các dòng lệnh phía server và phía client để phản hồi lại với các hành động của người dùng.

Sự phát triển của ASP.NET MVC

- ASP.NET MVC là một Framework phát triển các ứng dụng Web động, sử dụng .NET Framework.
- Trước ASP.NET MVC, các ứng dụng web động được phát triển trên .NET Framework bằng cách sử dụng:
  - ASP.NET Web Forms
  - ASP.NET Web pages

Ứng dụng ASP.NET

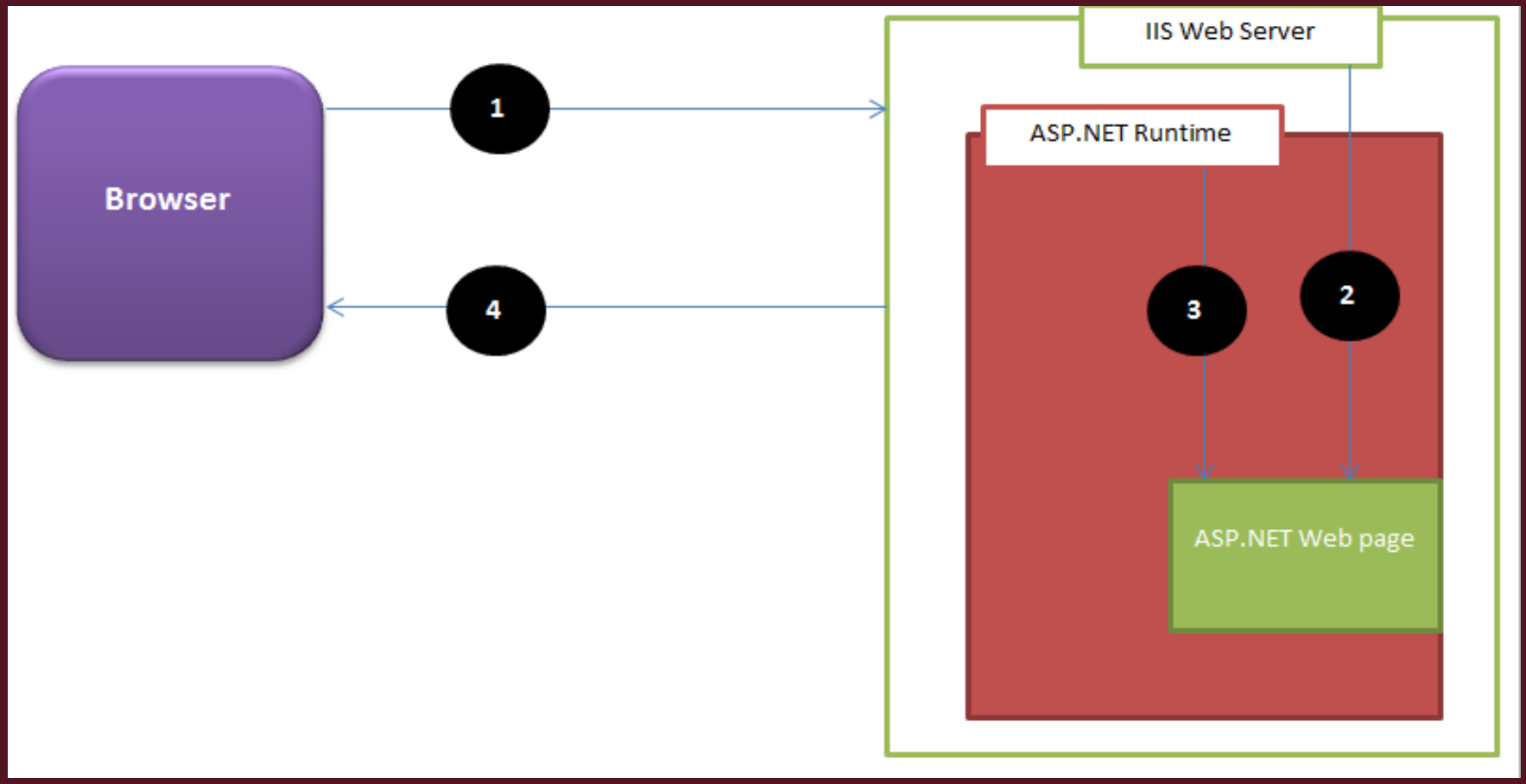
- ASP.NET:
  - Là một công nghệ phía server, sử dụng các tính năng nâng cao, cho phép bạn tạo ra các ứng dụng web động .
  - Bao gồm các trang web .aspx kết hợp giữa các dòng lệnh bên phía client và server để xây dựng các Website động.
  - Ứng dụng có thể được triển khai trên một Web server như Internet Information Services (IIS), là Web server của nền tảng Windows.

Ứng dụng ASP.NET

- Quy trình yêu cầu – phản hồi của trang Web ASP.NET bao gồm các bước sau:
  - Trình duyệt gửi 1 yêu cầu cho 1 trang Web ASP.NET.
  - Khi yêu cầu đến, Server IIS chặn yêu cầu này, tải xuống tập tin được yêu cầu, chuyển tiếp nó tới ASP.NET runtime để xử lý.
  - ASP.NET runtime có chứa công cụ dòng lệnh ASP.NET xử lý trang ASP.NET được yêu cầu và tạo ra phản hồi.
  - Server IIS gửi lại phản hồi tới Web server đã yêu cầu trang web.

Ứng dụng ASP.NET

- Hình dưới đây biểu thị luồng yêu cầu – phản hồi cho một trang Web ASP.NET :



ASP.NET MVC

- ASP.NET MVC dựa trên mô hình thiết kế MVC cho phép bạn xây dựng các giải pháp lập trình.
- Mô hình thiết kế MVC:
  - Cho phép bạn lập trình ứng dụng Web với các thành phần được kết nối linh hoạt.
  - Phân tách riêng biệt data access, business logic và presentation logic
- Khi sử dụng mô hình thiết kế MVC, một ứng dụng Web được chia thành 3 phần:
  - M odels: Các lớp đại diện cho dữ liệu của ứng dụng và sử dụng logic xác thực để thực thi các quy tắc nghiệp vụ cho dữ liệu đó.
  - V iews: Các tệp mẫu mà ứng dụng của bạn sử dụng để tạo động các phản hồi HTML.
  - C ontrollers: Các lớp xử lý các yêu cầu trình duyệt đến, truy xuất dữ liệu mô hình, sau đó chỉ định các mẫu xem trả về phản hồi cho trình duyệt.

ASP.NET MVC

- **Models:** Các đối tượng Models là một phần của ứng dụng, các đối tượng này thiết lập logic của phần dữ liệu của ứng dụng. Thông thường, các đối tượng model lấy và lưu trạng thái của model trong CSDL.

Ví dụ như, một đối tượng Employee (nhân viên) sẽ lấy dữ liệu từ CSDL, thao tác trên dữ liệu và sẽ cập nhật dữ liệu trở lại vào bảng Eployees ở SQL Server.

- **Views:** Views là các thành phần dùng để hiển thị giao diện người dùng (UI). Thông thường, view được tạo dựa vào thông tin dữ liệu model. Ví dụ như, view dùng để cập nhật bảng Employees sẽ hiển thị các hộp văn bản, drop-down list, và các check box dựa trên trạng thái hiện tại của một đối tượng Employee.

View là giao diện người dùng

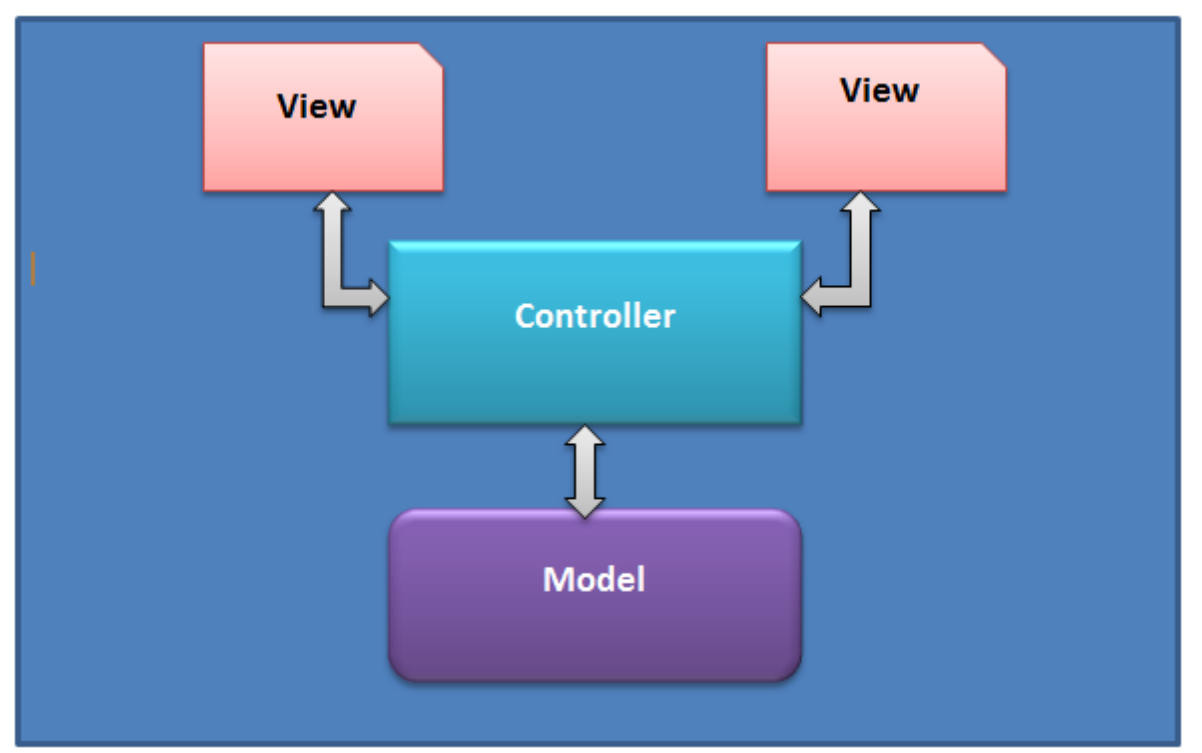
ASP.NET MVC

Controllers: Controller là các thành phần dùng để quản lý tương tác người dùng, làm việc với model và chọn view để hiển thị giao diện người dùng. Trong một ứng dụng MVC, view chỉ được dùng để hiển thị thông tin, controller chịu trách nhiệm quản lý và đáp trả nội dung người dùng nhập và tương tác với người dùng.

Controller điều khiểu các nhiều cầu từ Client.

Ví dụ: controller sẽ quản lý các dữ liệu người dùng gởi lên (query-string values) và gởi các giá trị đó đến model, model sẽ lấy dữ liệu từ CSDL nhờ vào các giá trị này

- Hình dưới đây biểu thị giao tiếp giữa các thành phần Model, View và Controller:

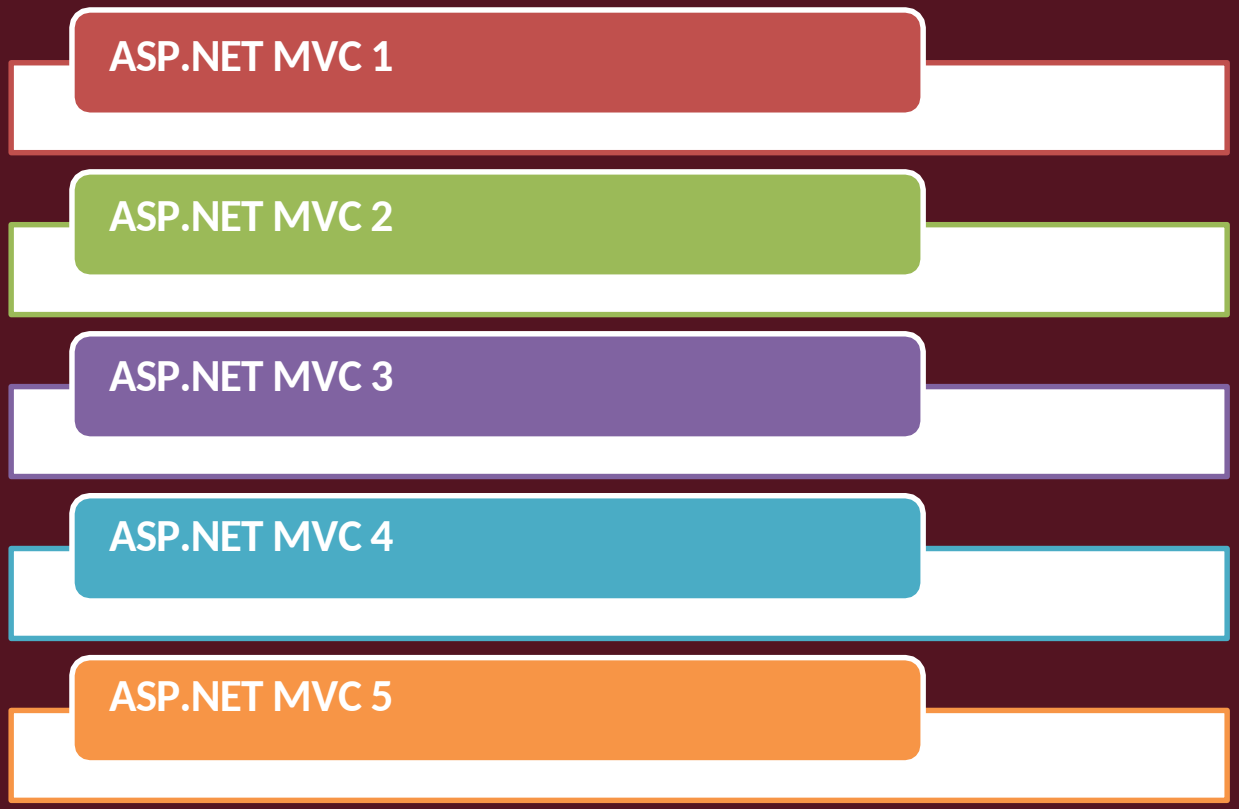


ASP.NET MVC

- Vì ASP.NET dựa trên mô hình MVC, nó cung cấp các lợi ích sau:
  - **Phân tách các quan hệ**: Cho phép bạn chắc chắn rằng các quan hệ ứng dụng ở các thành phần phần mềm khác nhau và độc lập.
  - **Đơn giản hóa kiểm thử và bảo trì**: Cho phép kiểm thử từng thành phần một cách độc lập. Điều này giúp bạn đảm bảo rằng nó hoạt động theo từng yêu cầu của ứng dụng và từ đó, đơn giản hóa quy trình kiểm thử, bảo trì và khắc phục sự cố
  - **Tính mở rộng**: Cho phép Model bao gồm 1 tập hợp các thành phần độc lập, bạn có thể dễ dàng chỉnh sửa hoặc thay thế dựa trên yêu cầu ứng dụng.

Lịch sử của MVC

- Các phiên bản của MVC:



ASP.NET MVC 1

- Bản đầu tiên của ASP.NET MVC, phát hành vào 13/3/2009.
- .NET Framework 3.5.
- Hỗ trợ Visual Studio 2008 và Visual Studio 2008 SP1 để phát triển các ứng dụng ASP.NET MVC 1.

Lịch sử của MVC

ASP.NET MVC 2

- Phát hành vào 10/3/2020 với .NET Framework 3.5 và 4.0.
- Hỗ trợ Visual Studio 2008 and 2010 để phát triển ứng dụng ASP.NET MVC 2.
- ASP.NET MVC 2 bao gồm các tính năng chính:
  - Trình hỗ trợ ý nghĩa của HTML
  - Chú thích dữ liệu
  - Xác thực phía Client
  - Tách ứng dụng thành các mô-đun
  - Bộ điều khiển bất đồng bộ

Lịch sử của MVC

ASP.NET MVC 3

- Phát hành vào 13/1/2011 với NET Framework 4.0.
- Hỗ trợ Visual Studio 2010 để phát triển các ứng dụng ASP.NET MVC 3.
- Phiên bản này bao gồm các tính năng chính như:
  - Công cụ Razor view
  - Cải thiện hỗ trợ chú thích dữ liệu
  - Hỗ trợ Entity Framework Code
  - Hỗ trợ ViewBag chuyển dữ liệu từ Controller sang view
  - Hỗ trợ Action Filters
  - Hỗ trợ JavaScript và jQuery

Lịch sử của MVC

ASP.NET MVC 4

- Được phát hành vào 15/8/2012 với .NET Framework 4.0 và 4.5.
- Hỗ trợ Visual Studio 2010 SP1 và Visual Studio 2012 để xây dựng ứng dụng ASP.NET MVC 4.
- Phiên bản này có các tính năng chính:
  - ASP.NET Web API
  - Trình điều khiển bất đồng bộ với Task Support
  - Hỗ trợ Windows Azure SDK

ASP.NET MVC 5

- Phiên bản này được phát hành vào 17/10/2013 với .NET Framework 4.5 và 4.5.1.
- Hỗ trợ Visual Studio 2013 để phát triển các ứng dụng ASP.NET MVC 5
- Phiên bản có cách tính năng chính:
  - Định danh ASP.NET
  - ASP.NET Web API 2

Kiến trúc của ứng dụng ASP.NET MVC

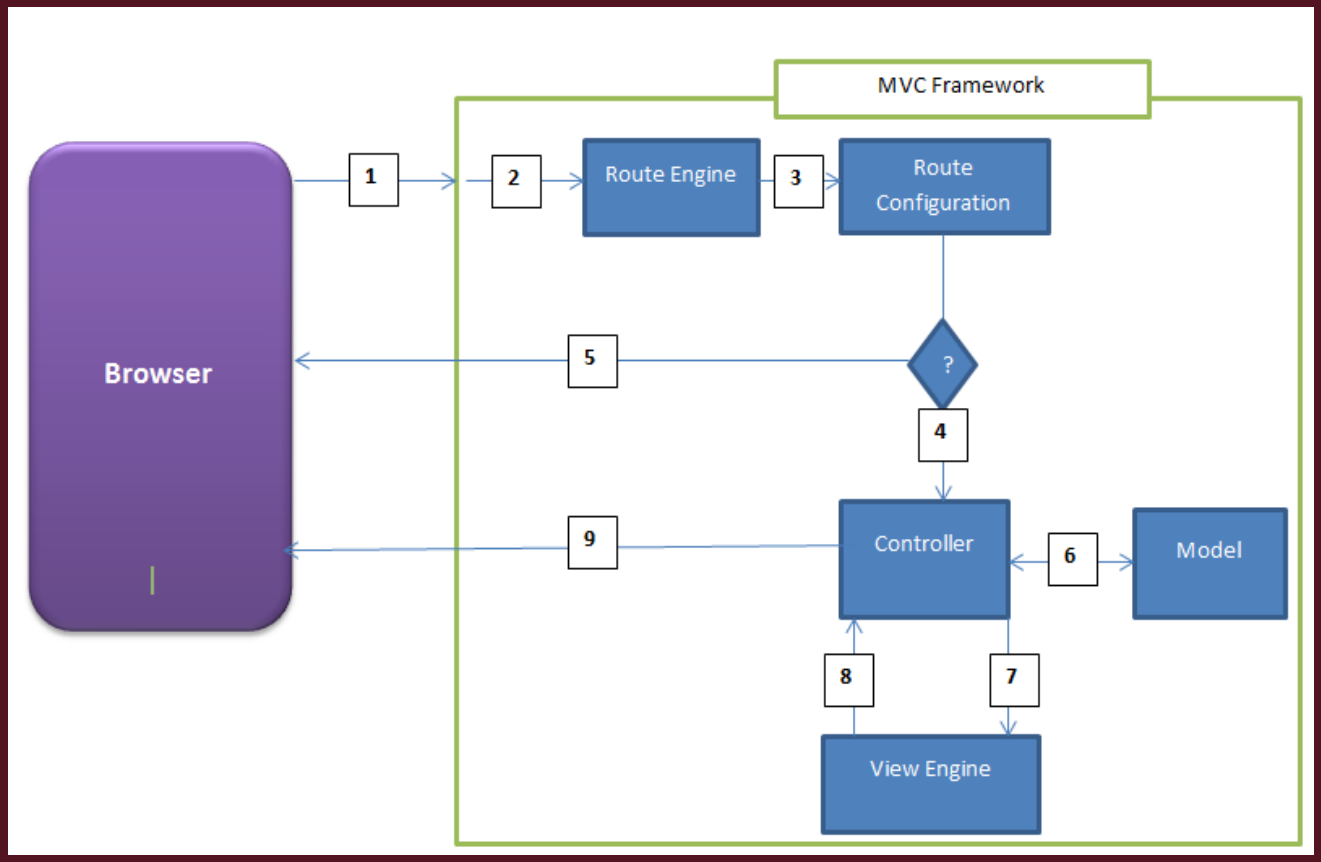
- Kiến trúc cơ bản của một ứng dụng ASP.NET MVC:
  - MVC Framework
  - Công cụ định tuyến
  - Cấu hình định tuyến
  - Controller
  - Model
  - Công cụ View
  - View
- Mỗi thành phần giao tiếp để xử lý các yêu cầu đến ứng dụng ASP.NET MVC.
- Quá trình xử lý một yêu cầu được gửi đến bao gồm một loạt các bước mà các thành phần này thực hiện.

Kiến trúc của ứng dụng ASP.NET MVC

- Các bước mà các thành phần của ASP.NET MVC Framework thực hiện khi xử lý một yêu cầu đến:
  - Trình duyệt gửi một yêu cầu đến ứng dụng ASP.NET MVC
  - MVC Framework chuyển tiếp yêu cầu tới công cụ định tuyến
  - Công cụ định tuyến kiểm tra cấu hình định tuyến của ứng dụng để tìm controller thích hợp xử lý yêu cầu.
  - Khi tìm thấy 1 controller, nó được gọi đến
  - Khi không tìm thấy controller phù hợp, công cụ định tuyến sẽ chỉ ra rằng không tìn thấy controller và MVC Framework sẽ thông báo lỗi với trình duyệt
  - Controller giao tiếp với Model.
  - Controller yêu cầu công cụ view cho một view dựa theo dữ liệu của Model.
  - Công cụ view trả về kết quả tới controller
  - Controller gửi lại kết quả dưới dạng phản hồi HTTTP cho trình duyệt

Kiến trúc của ứng dụng ASP.NET MVC

- Hình dưới đây biểu thị các bước mà các thành phần của ASP.NET MVC Framework thực hiện khi xử lý yêu cầu đến:



Các công nghệ hỗ trợ

- Ứng dụng ASP.NET MVC hỗ trợ nhiều công nghệ để tạo ra ứng dụng Web động và có tính phản hồi.
- Một số công nghệ hỗ trợ bạn có thể sử dụng khi tạo một ứng dụng ASP.NET MVC:

JavaScript

JQuery

Asynchronous JavaScript and XML (AJAX)

IIS

Windows Azure

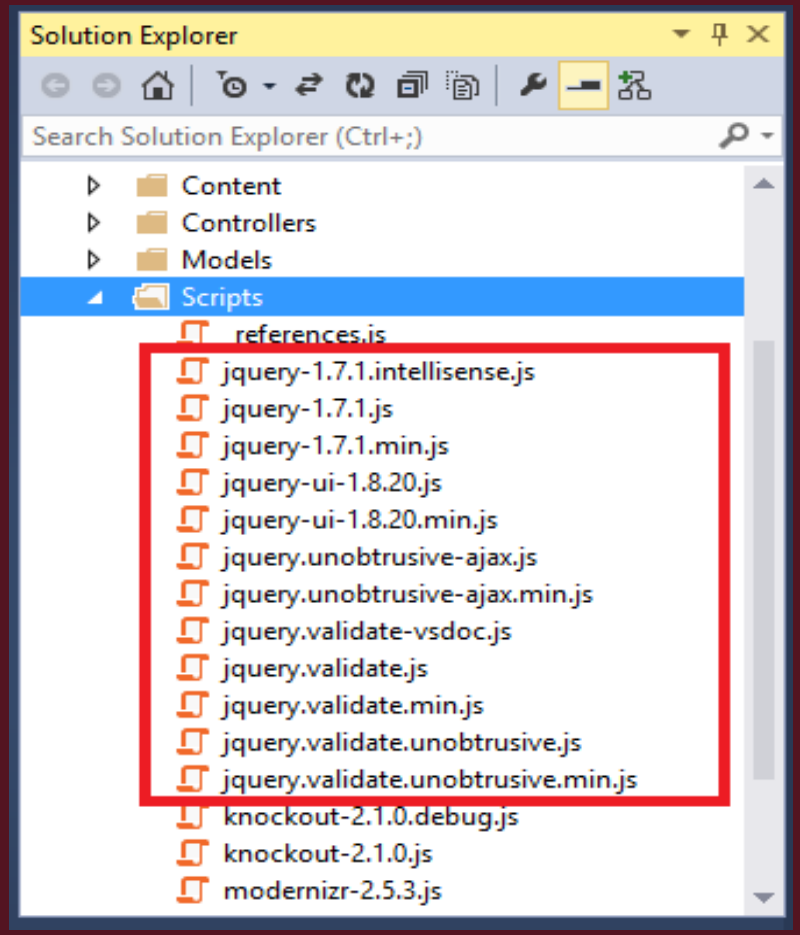
JavaScript

- Là một ngôn ngữ phía client cho phép một ứng dụng Web phản hồi lại các yêu cầu người dùng mà không cần tương tác với Web server.
- Cho phép tạo ra các ứng dụng Web có tính phản hồi, nâng cao trải nghiệm người dùng.
- Cho phép triển khai các tính năng như dễ dàng sử dụng UI, phản hồi nhanh với yêu cầu người dùng, chạy trên tất cả các trình duyệt khả dụng.

JQuery

- JQuery:
  - Là một thư viện JavaScript đơn giản hóa các dòng lệnh phía client của HTML
  - Được sử dụng ở các view để làm cho các view có tính động và tính phản hồi
  - Cho phép thực hiện xác nhận phía client của các form.
  - Cung cấp các plugin khác nhau, cho phép bạn nâng cao các thành phần UI của View.
- Khi tạo ra một project ASP.NET MVC trong Visual Studio 2013, project sẽ tự động bao gồm các thư viện jQuery với các tập tin **Scripts**.

- Hình dưới biểu thị các thư viện jQuery trong thư mục Scripts của project ASP.NET MVC:



- Để sử dụng 1 thư viện jQuery trong 1 view, bạn cần refer thư viện từ View.
- Code snippet dưới đây chỉ ra cách để refer một thư viện jQuery từ một View :

Code snippet:

```
<script src="/Scripts/jquery-1.7.1.min.js"
type="text/javascript"></script>
<!- -JQuery Code - ->
</script>
```

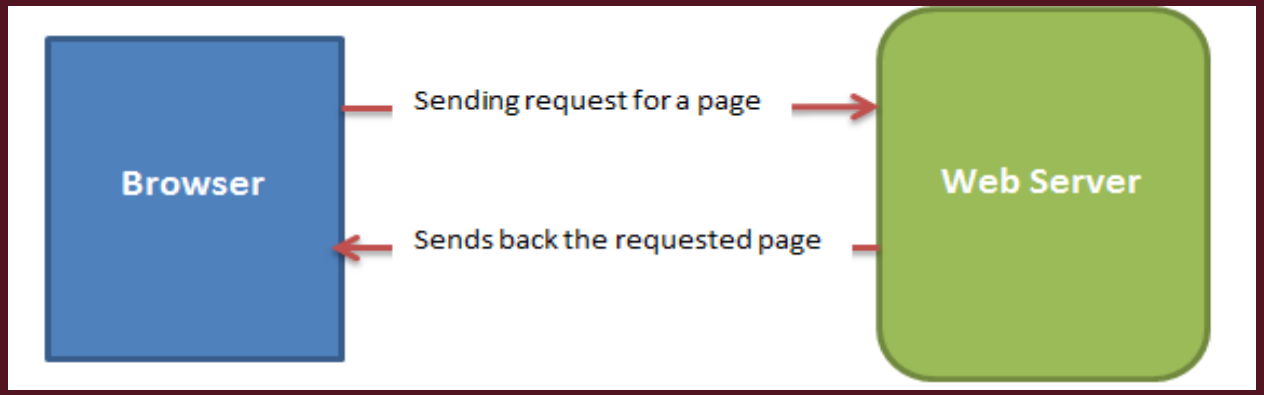
AJAX

- **AJAX:**
  - Là một kĩ thuật phát triển web được sử dụng để tạo ra các ứng dụng mang tính tương tác.
  - Cho phép truy xuất dữ liệu bất đồng bộ ở background của ứng dụng ASP.NET MVC mà không can thiệp vào cách hiển thị và hành vi của view hiện có.
  - Cho phép người dùng làm việc một cách liên tục trên ứng dụng mà không bị ảnh hưởng bởi các phản hồi nhận được từ server.
- JavaScript form là một phần không thể thiếu của các ứng dụng Web được phát triển bằng AJAX vì các ứng dụng này xử lý hầu hết các yêu cầu phía client, trừ khi cần kết nối tới Web server.
- Đối tượng XMLHttpRequest là một trong những đối tượng chính được JavaScript sử dụng vì nó phép giao tiếp bất đồng bộ giữa client và server.

IIS

- Một ứng dụng ASP.NET MVC yêu cầu một web server có thể xử lý các yêu cầu HTTP và tạo ra các phản hồi.
- Khi yêu cầu một trang web trên trình duyệt, bạn gõ 1 URL trên 1 trình duyệt, ví dụ như: `http://mvcexample.com/index.html`.
- URL này có các phần sau:
  - **http**: Một giao thức sử dụng để trao đổi yêu cầu và phản hồi
  - **mvctest.com**: Một tên miền ánh xạ tới một địa chỉ IP
  - **index.html**: Một tập tin mà bạn yêu cầu

- Hình dưới biểu thị giao tiếp giữa trình duyệt và server:

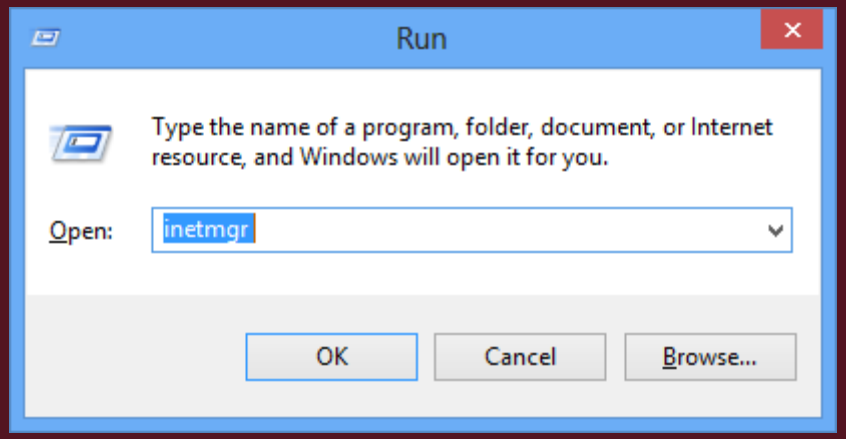


- Trong hình này:
  - Trình duyệt tạo ra kết nối với một Web server nơi địa chỉ IP được đăng kí.
  - Tiếp theo, trình duyệt sử dụng giao thức HTTP để gửi một yêu cầu tới server, yêu cầu tập tin.
  - Server sẽ tìm tập tin này, sau đó gửi lại nội dung dưới dạng một HTML markup tới trình duyệt.
  - Trình duyệt phân tích HTML markup vừa nhận và hiển thị kết quả tới bạn

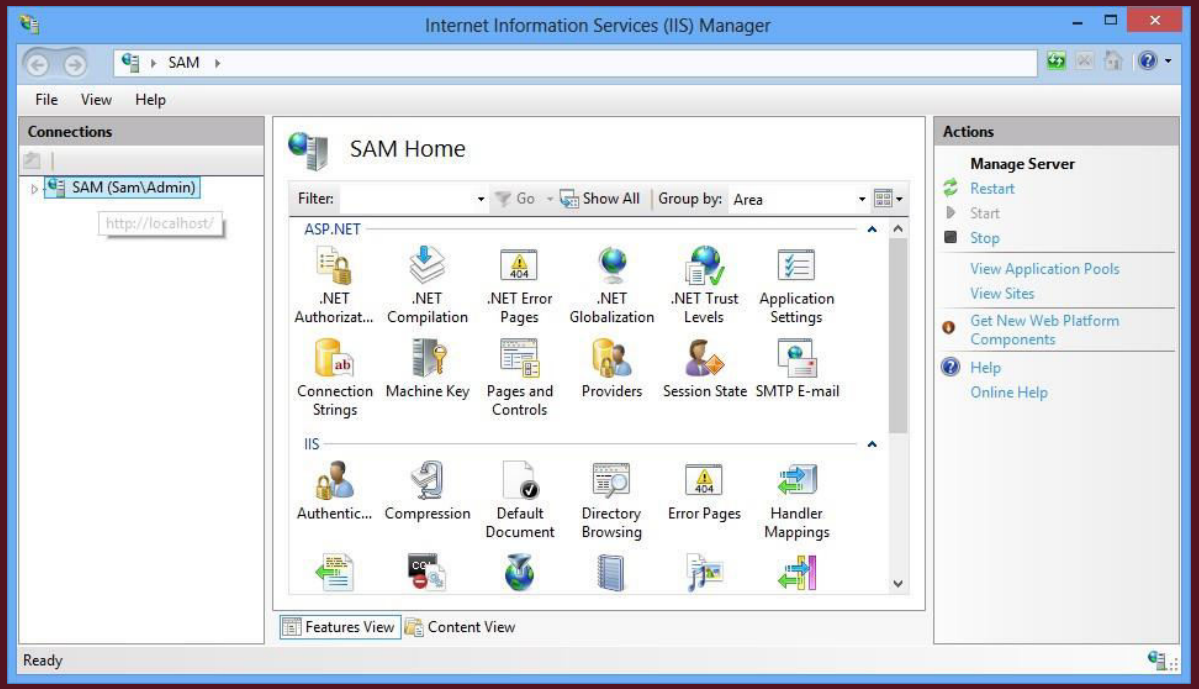
IIS

- IIS:
  - Là 1 sản phẩm của Microsoft, được thiết kế để truyền thông tin với độ bảo mật và tốc độ cao.
  - Hoạt động như 1 nền tảng mà bạn có thể mở rộng các khả năng tiêu chuẩn của Internet.
  - Được thiết kế mô-đun hóa, cho phép bạn chỉ cần chọn ra các mô-đun phù hợp với yêu cầu khi cài đặt IIS.
  - Một số các mô-đun mà bạn cần để tổ chức các ứng dụng ASP.NET MVC:
    - **Mô-đun nội dung**: Cho phép điều khiển nội dung được truyền đến client
    - **Mô-đun bảo mật**: Cho phép triển khai bảo mật thông qua cơ chế xác thực khác nhau, ủy quyền dựa trên URL, yêu cầu lọc.
    - **Mô-đun nén**: Cho phép nén các phản hồi gửi tới trình duyệt client bằng cách sử dụng tiêu chuẩn nén, ví dụ như Gzip
    - **Mô-đun caching**: Cho phép caching nội dung và cung cấp nội dung đó lưu trong bộ nhớ đệm cho các yêu cầu tiếp theo có cùng nguồn.
    - **Mô-đun ghi log và chẩn đoán**: Cho phép ghi log yêu cầu xử lý thông tin và trạng thái phản hồi, từ đó chẩn đoán được mục đích.

- Bạn có thể triển khai và quản lý các ứng dụng ASP.NET MVC trên IIS bằng cách sử dụng IIS Manager.
- Để truy cập IIS Manager, làm theo các bước sau:
  1. Chắc chắn rằng IIS được cài đặt trên máy tính của bạn
  2. Nhấn tổ hợp phím **Windows+R** . Hộp thoại **Run** xuất hiện.
  3. Gõ `inetmgr`, như hình dưới dây:



4. Nhấn **OK**. Cửa sổ **IIS Manager** mở ra, như hình dưới:



- Sử dụng cửa sổ IIS Manager để cấu hình các tính năng của IIS, ví dụ như gán quyền cho các ứng dụng, quản lý bảo mật server, quản lý định danh và định quyền của ứng dụng.

Windows Azure

- Trước nền tảng đám mây, các ứng dụng được triển khai trên các server có khả năng truy cập internet, được hỗ trợ bởi các datacenter.
- Tuy vậy, rất khó cho các cá nhân và tổ chức có thể thiết lập được một cơ sở hạ tầng như vậy.
- Thêm vào đó, việc duy trì được hạ tầng như vậy lại càng khó khan hơn.
- Như một giải pháp, nền tảng đám mây được giới thiệu là nơi mà các cá nhân hay tổ chức có lựa chọn để tổ chức cũng như duy trì các ứng dụng trong hạ tầng mà họ không cần phải quản lý.

Windows Azure

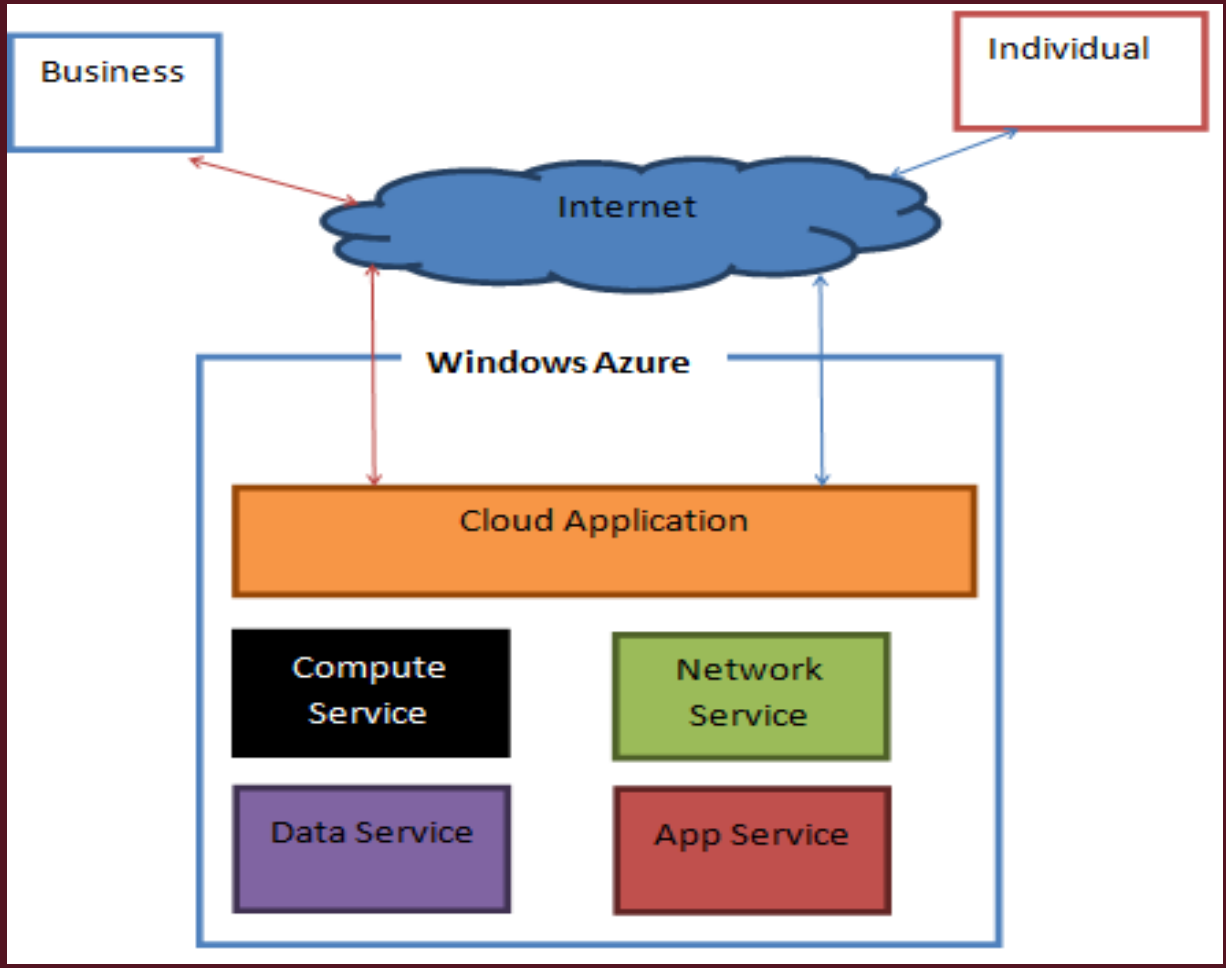
- **Windows Azure:**
  - Là một giải pháp đám mây được cung cấp bởi Microsoft.
  - Cho phép bạn tạo ra các Word documents, chia sẻ chúng, lưu chúng mà không cần yêu cầu bất cứ phần mềm nào được cài đặt trên máy tính hoặc nâng cấp các phần cứng trên máy tính.
  - Cung cấp nền tảng để xây dựng nên các ứng dụng có thể tận dụng đám mây để đáp ứng nhu cầu của người dùng, từ đơn giản như gửi email đến quản lý một ứng dụng ASP.NET MVC phức tạp triển khai một cửa hàng đấu giá trực tuyến với cơ sở người dùng trên toàn cầu.
- Thay vì phải tải, cài đặt, sử dụng một sản phẩm trên chính máy tính của họ, bạn hoàn toàn có thể sử dụng Windows Azure như một dịch vụ có các chức năng y hệt các chức năng trên

Windows Azure

- Window Azure cung cấp những dịch vụ chính sau:
  - **Dịch vụ điện toán**: Cung cấp cơ sở hạ tầng để cung cấp năng lượng cần thiết để chạy các ứng dụng đám mây qua các dịch vụ, ví dụ như máy ảo, website, dịch vụ Mobile
  - **Dịch vụ mạng**: Dịch vụ cung cấp các ứng dụng đám mây cho người dùng và các datacenter.
  - **Dịch vụ dữ liệu**: Cung cấp dịch vụ cho phép các ứng dụng đám mây có thể lưu trữ, quản lý, bảo mật dữ liệu ứng dụng một cách an toàn
  - **Dịch vụ ứng dụng**: Cung cấp dịch vụ cho phép các ứng dụng đám mây nâng cao hiệu suất và tính bảo mật. Dịch vụ này cũng cho phép các ứng dụng đám mây được tổ chức trên Window Azure có thể tích hợp được với các ứng dụng đám mây khác

Windows Azure

- Hình dưới đây biểu diễn các dịch vụ đám mây chính của Windows Azure:

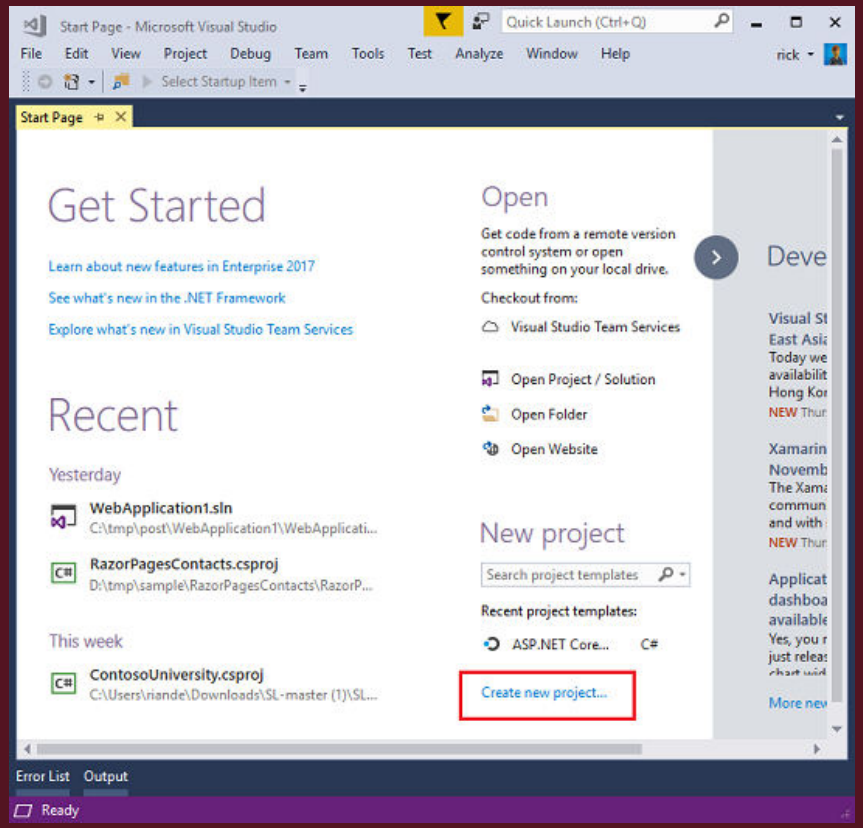


Hỗ trợ MVC trong Visual Studio

- Visual Studio 2017:
  - Đơn giản hóa quá trình tạo các ứng dụng ASP.NET MVC bằng cách cung cấp các template có sẵn.
  - Cung cấp một template MVC tự động tạo một cấu trúc ứng dụng MVC với các tập tin cơ bản để chạy ứng dụng

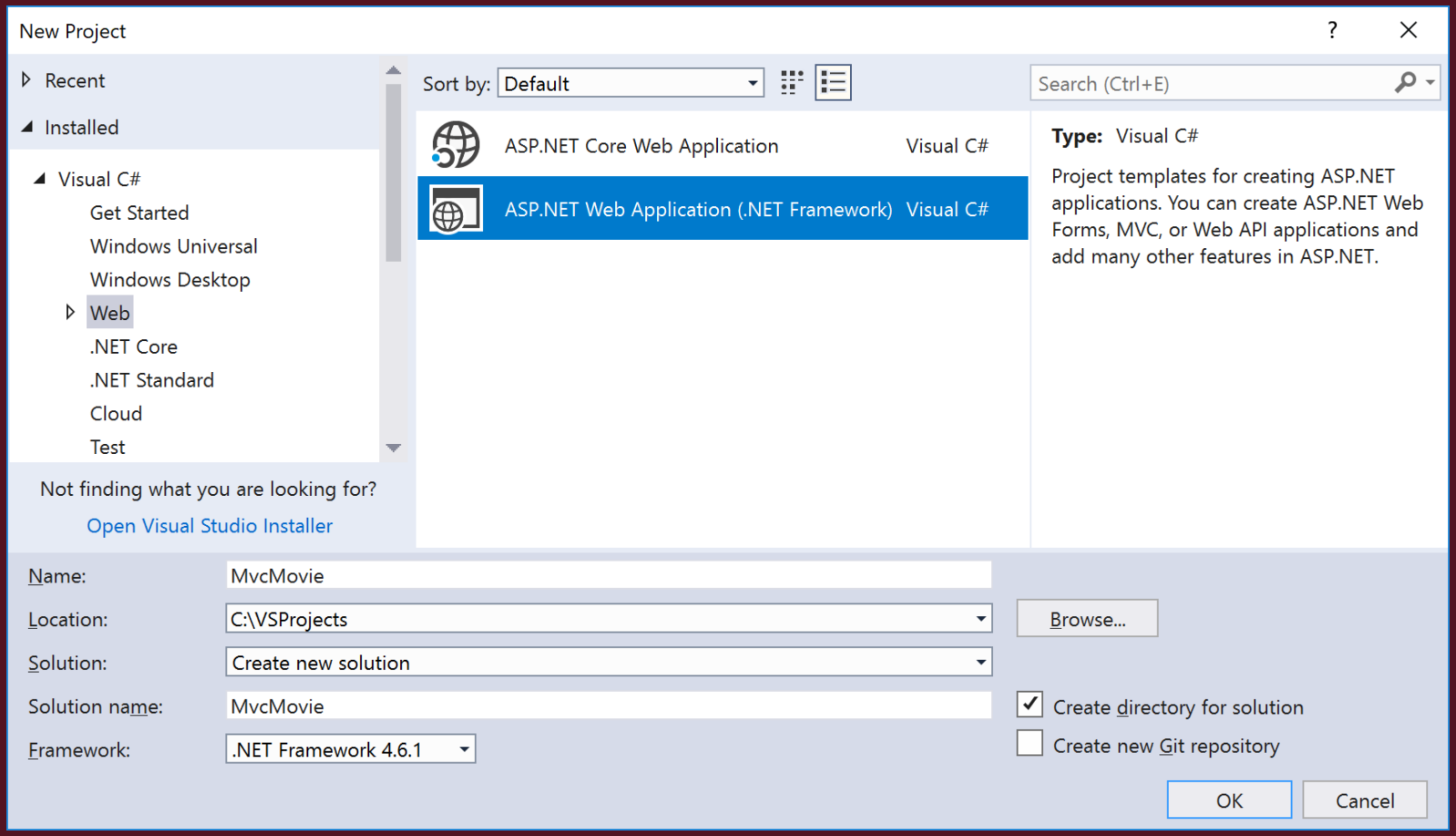
Tạo một project ASP.NET

- Để tạo một ứng dụng ASP.NET MVC trên Visual Studio 2017, làm theo các bước sau:
  1. Nhấn tổ hợp phím **Windows+Q**.
  2. Hộp thoại **Search** xuất hiện, gõ Visual Studio 2017. Biểu tượng **Visual Studio 2017** xuất hiện dưới mục Apps
  3. Nháy đúp vào biểu tượng **Visual Studio 2017**. **Start Page của** Visual Studio 2017 xuất hiện như hình dưới:



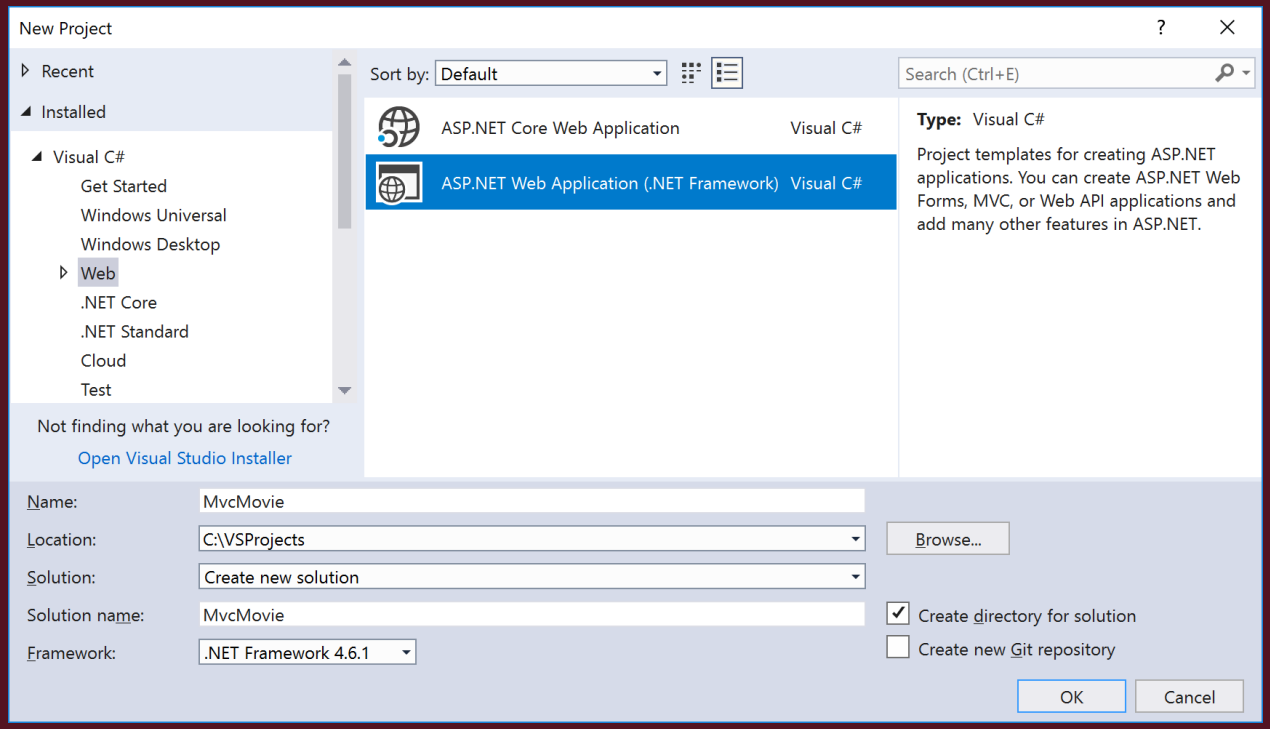
Tạo một project ASP.NET MVC

4. Nhấn **File→New→Project** ở thanh menu của Visual Studio 2017.
5. Ở hộp thoại **New Project** chọn **Web** ở dưới dòng **Installed,** sau đó chọn **ASP.NET Web Application** template



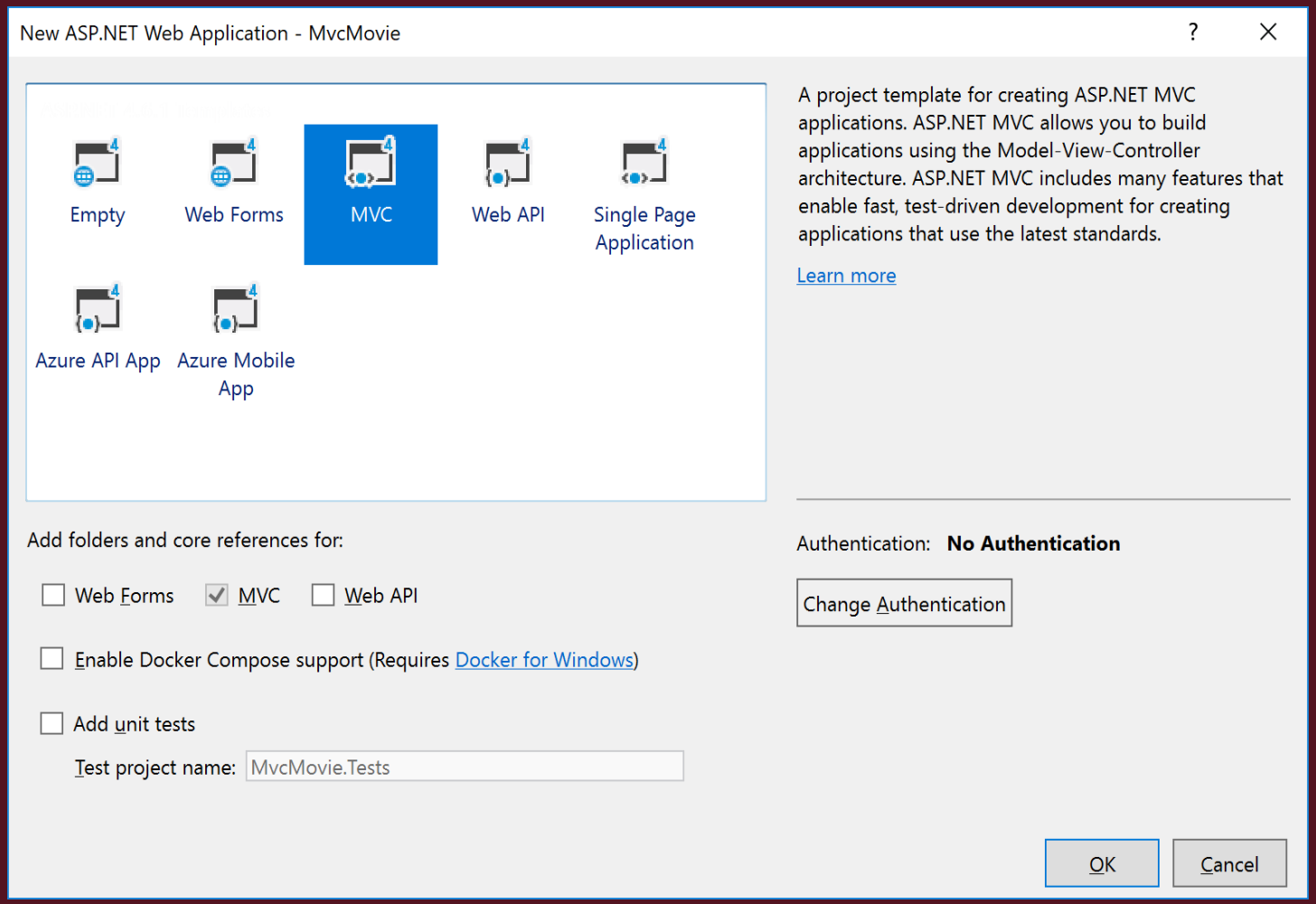
Tạo một project ASP.NET MVC

6. Gõ **MvcMovie** ở dòng **Name**.
7. Nhấn **Browse**. Hộp thoại **Project Location** xuất hiện cho phép bạn thấy được địa chỉ mà ứng dụng được tạo
8. Nhấn **Select Folder**. Hộp thoại **New Project** xuất hiện với địa chỉ đã định



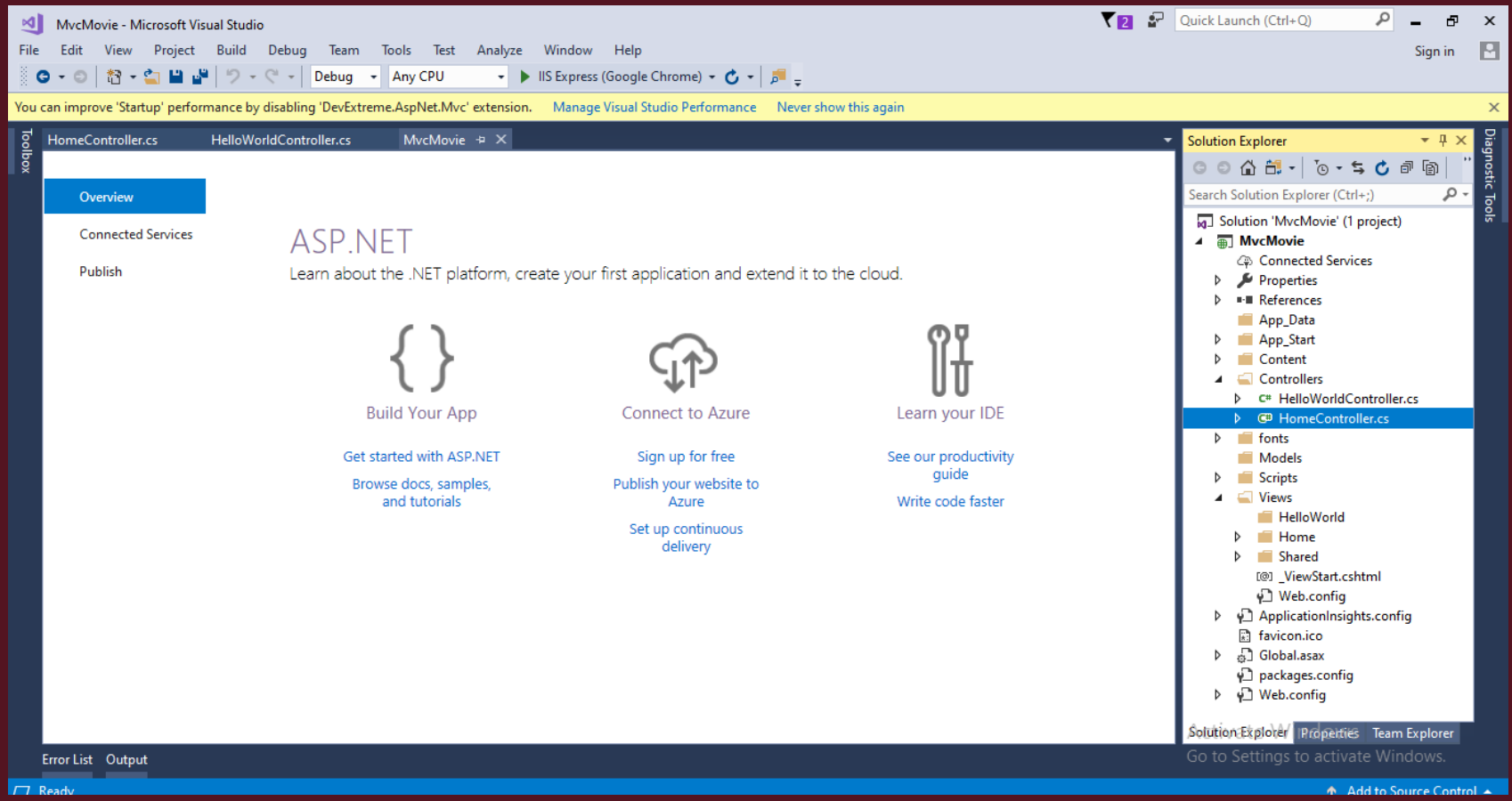
Tạo một project ASP.NET MVC

9. Nhấn **OK**. Hộp thoại **New ASP.NET Project – MvcMovie** xuất hiện
10. Chọn **MVC**



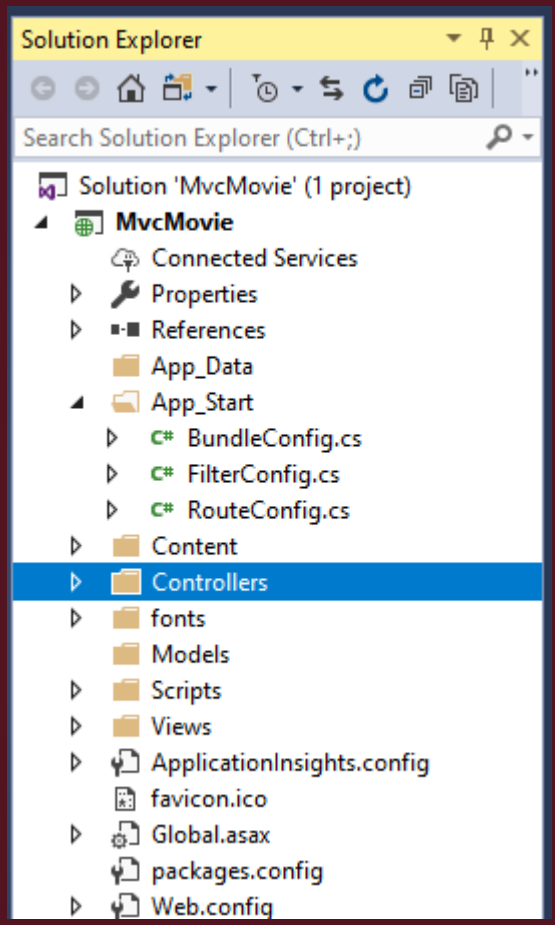
Tạo một project ASP.NET MVC

11. Nhấn **OK**. Visual Studio 2017 hiển thị ứng dụng vừa được tạo:



Cấu trúc của một project ASP.NET MVC

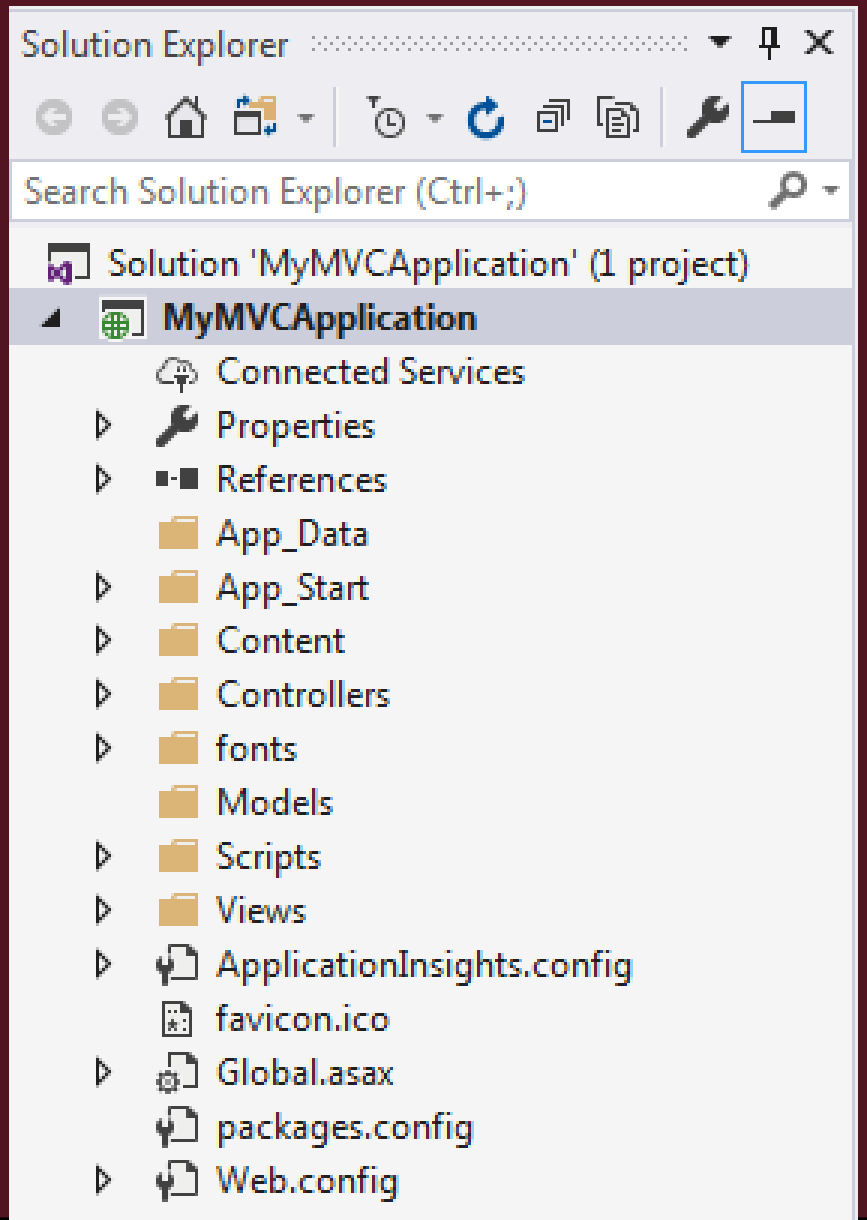
- Khi sử dụng Visual Studio 2017 để tạo một ứng dụng ASP.NET MVC, nó sẽ tự động thêm một số tập tin và thư mục vào Project
- Hình dưới đây biểu thị một số tập tin và thư mục được Visual Studio 2017 tạo ra khi bạn tạo một ứng dụng ASP.NET MVC:



Cấu trúc của một project ASP.NET MVC

- Cấu trúc của một ứng dụng ASP.NET MVC bao gồm các thư mục sau:
  - **Controllers**: Bao gồm các class Controller xử lý các yêu cầu URL.
  - **Models**: Bao gồm các class đại diện cũng như thao tác dữ liệu và các đối tượng
  - **Views**: Bao gồm các tập tin template UI chịu trách nhiệm hiển thị đầu ra, ví dụ như HTML
  - **Scripts**: Chứa các tập tin thư viện JavaScript
  - **Images**: Chứa các ảnh mà bạn cần sử dụng trong ứng dụng
  - **Content**: Chứa các nội dung CSS và các nội dung khác, trừ scripts và hình ảnh
  - **Filters**: Chứa filter code.
  - **App_Data**: Chứa các tập tin dữ liệu đọc/ghi
  - **App_Start**: Chứa các tập tin chứa code cấu hình mà bạn có thể sử dụng các tính năng như Routing, Bundling, and Web API.

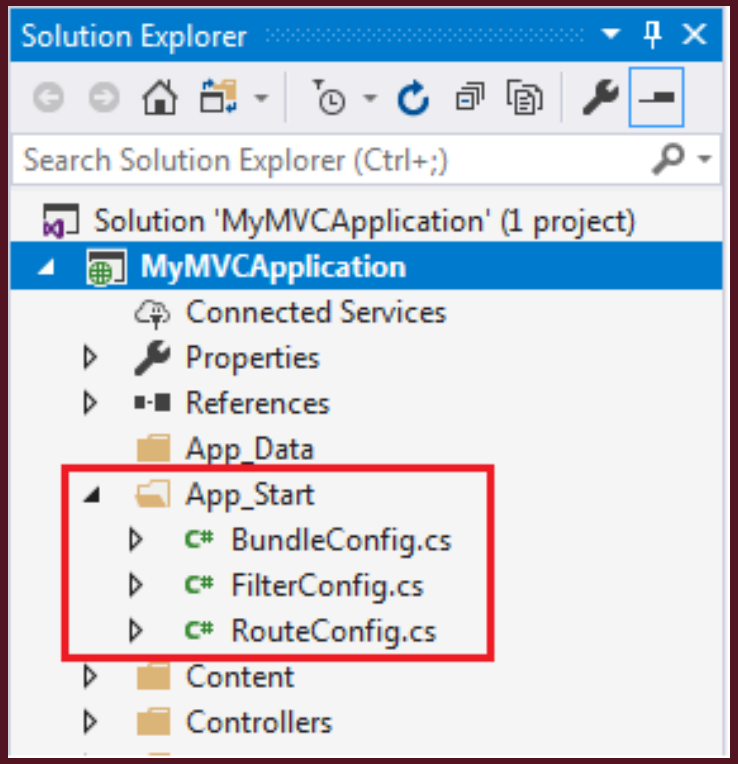
Cấu trúc của một project ASP.NET MVC



App_Data

App_Data: thư mục này chứa dữ liệu như: LocalDB, .mdf files, xml files. IIS sẽ không bao giờ nhận các tệp từ thư mục App_Data.

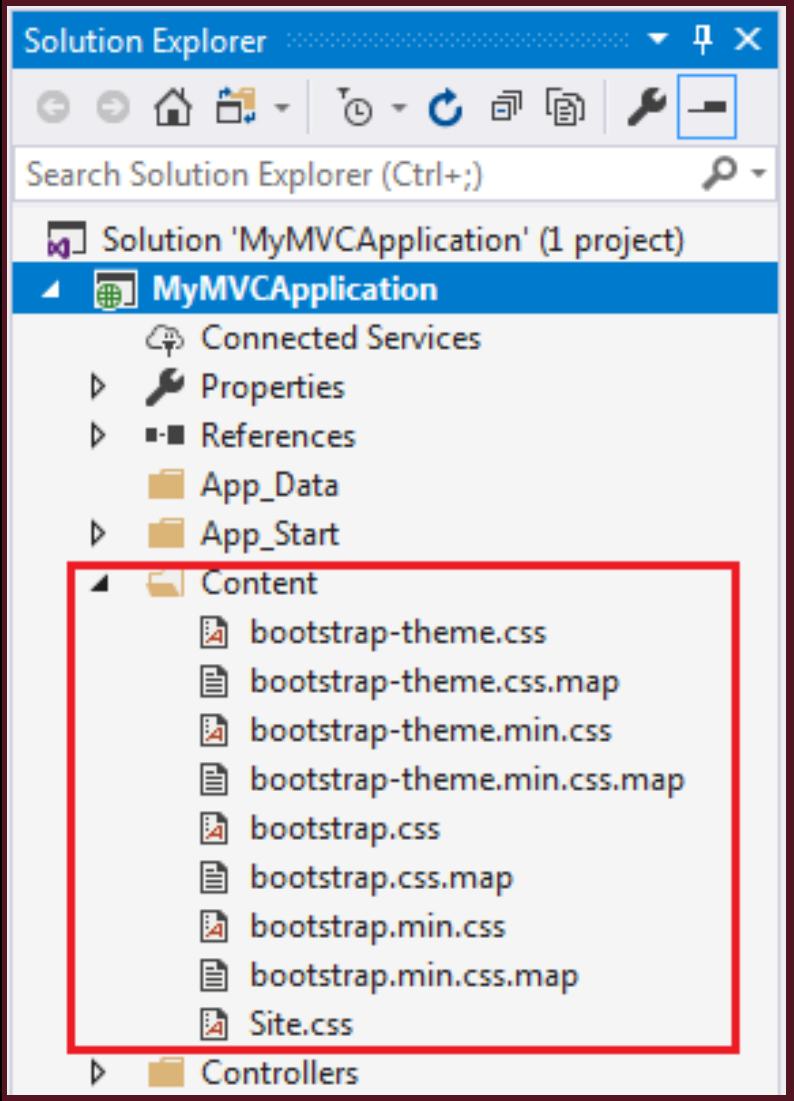
Cấu trúc của một project ASP.NET MVC



App_Start

thư mục chưa các file cấu hình khởi động và biên dịch của project. Chúng ta chú ý 2 file đó là **FilterConfig.cs**, dùng để khai báo các filter sử dụng trước khi thực hiện 1 hành động nào đó, tương tự như Middleware của PHP. Và file **RouteConfig.cs**, cái tên đã nói lên tất cả, chúng ta sẽ định nghĩa các routes của web ở trong file này.

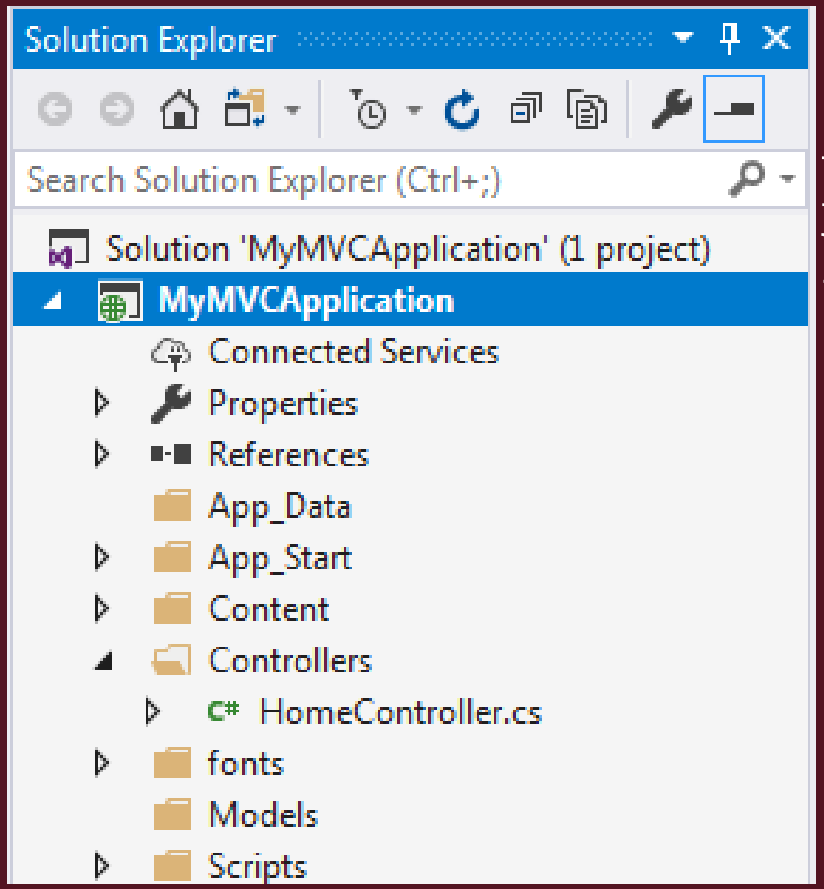
Cấu trúc của một project ASP.NET MVC



Content

Content: thư mục này chứa các file :css, images và icons .Ứng dụng MVC 5 chứa các file mặc định: bootstrap.css, bootstrap.min.css và Site.css .

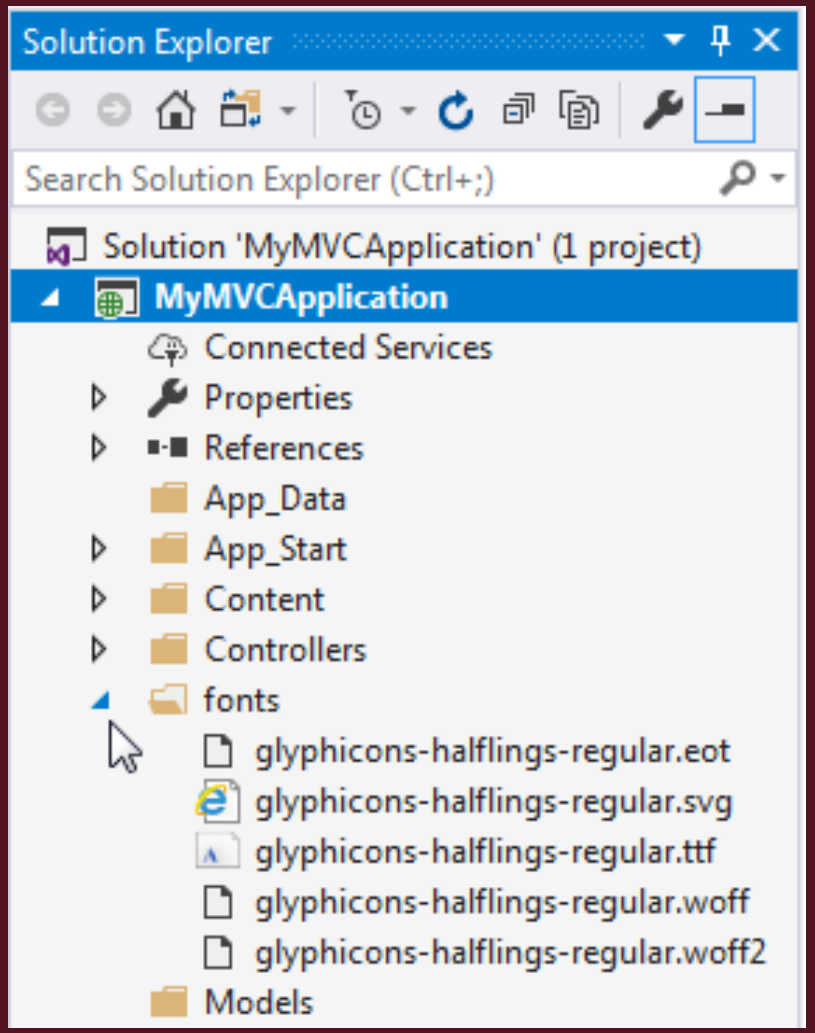
Cấu trúc của một project ASP.NET MVC



Controllers

Controller xử lý yêu cầu và phản hồi cho người dùng. MVC yêu cầu tên của tất cả các tệp điều khiển kết thúc bằng "Controller".

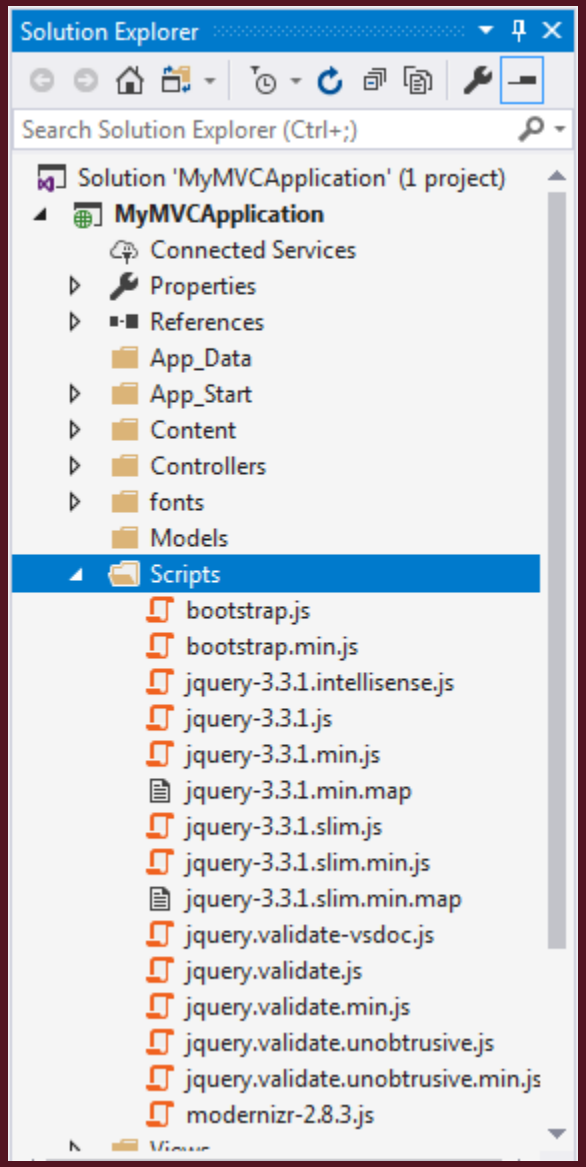
Cấu trúc của một project ASP.NET MVC



fonts

Fonts: thư mục này chứa các tập tin font files của ứng dụng.

Cấu trúc của một project ASP.NET MVC



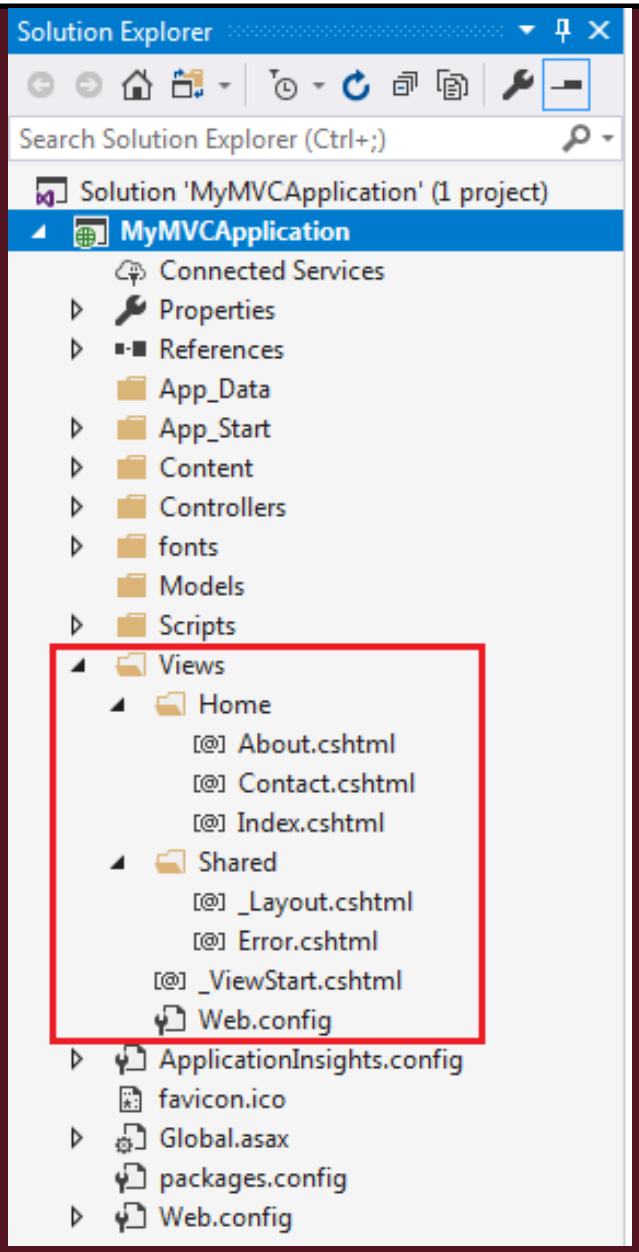
Models

Models chứa các file tương tác với CSDL..

Scripts

thư mục này chứa các tập tin JavaScript hay VBScript. MVC 5 chứa các tập tin mặc định :javascript bootstrap, jquery 1.10.

Cấu trúc của một project ASP.NET MVC



Views

Views: Thư mục chứa các file HTML với đuôi là .cshtml.

Cấu trúc của một project ASP.NET MVC

Ngoài ra , MVC chứa các tập tin cầu hình:

Global.asax

Global.asax: cho phép bạn viết code chạy để đáp ứng với các sự kiện ở mức ứng dụng: Application_BeginRequest, application_start, application_error, session_start, session_end, v.v..

Packages.config

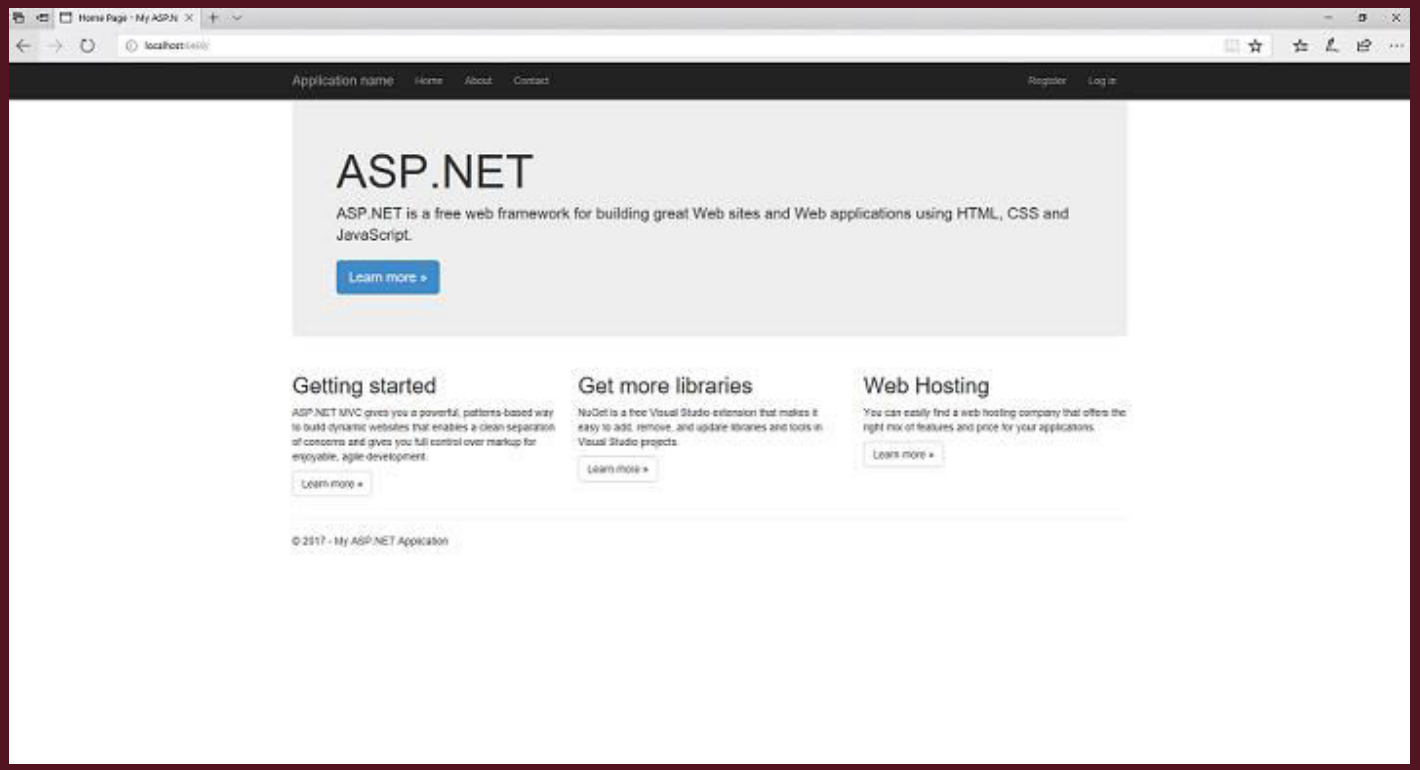
Packages.config: tập tin được NuGet quản lý để theo dõi những gói và phiên bản bạn đã cài đặt trong ứng dụng.

Web.config

Web.config : File khá quan trọng, định nghĩa các cài đặt cho project.

Executing ASP.NET MVC Project

- Nhấn **Debug→Start (**Hoặc nhấn **F5)** để thi hành ứng dụng mà không debug. Trình duyệt sẽ hiển thị đầu ra của ứng dụng mặc định:



Tổng kết

- MVC Framework thực hiện quy trình xác thực để xác thực dữ liệu người dùng
- Ứng dụng web là các ứng dụng được thực thi trên một Web server và được truy cập từ một trình duyệt Web
- Ứng dụng web được chia làm 3 lớp chính, mỗi lớp thực hiện một chức năng riêng biệt
- Kiến trúc của một ứng dụng phụ thuộc vào hệ thống trong đó các layers của ứng dụng được phân phối và giao tiếp với nhau
- ASP.NET là một công nghệ phía Server cho phép bạn tạo ra ứng dụng Web có tính động sử dụng các tính năng nâng cao, như tính đơn giản, tính bảo mật, khả năng mở rộng, dựa trên .NET Framework
- ASP.NET MVC dựa trên mô hình thiết kế MVC cho phép bạn phát triển các giải pháp phần mềm
- Một ứng dụng ASP.NET MVC yêu cầu một Web server cho phép xử lý các yêu cầu HTTP và tạo phản hồi
- Windows Azure là một giải pháp đám mây được cung cấp bởi Microsoft.

Chương 2: Mô hình MVC trong ASP.NET

Chương 2: Mô hình MVC trong ASP.NET

2.1 Controller trong ASP.NET MVC

2.2 View trong ASP.NET MVC

2.3 Model trong ASP.NET MVC

2.1 Controller trong ASP.NET MVC

Nội dung

- Định nghĩa và mô tả về controller
- Mô tả cách hoạt động của phương thức
- Giải thích cách gọi tới phương thức hành động
- Giải thích yêu cầu định tuyến
- Mô tả các mẫu URL

Làm việc với controller

- Một controller trong một ứng dụng ASP.NET MVC
  - Quản lý luồng của ứng dụng.
  - Có nhiệm vụ chặn các yêu cầu tới và thực thi các mã code ứng dụng phù hợp
  - Giao tiếp với các Model của ứng dụng và chọn các view được yêu cầu để kết xuất với yêu cầu
  - Là một class C# kế thừa class `Controller` của namespace `System.Web.Mvc` .
  - Cho phép phân tách business logic của ứng dụng với presentation logic

Làm việc với Controller

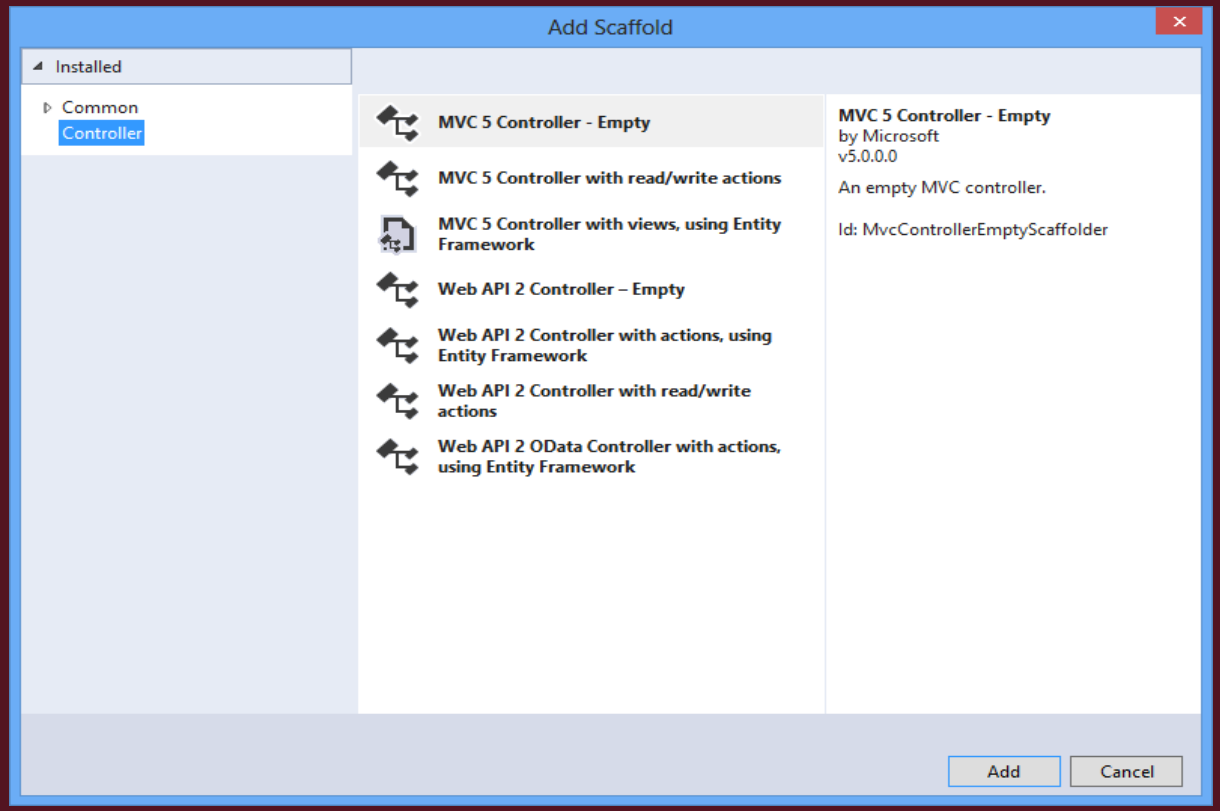
- Trong một ứng dụng ASP.NET MVC, 1 controller có nhiệm vụ:
  - Xác định phương thức phù hợp để gọi cho yêu cầu được gửi đến.
  - Xác thực dữ liệu của yêu cầu đến trước khi gọi đến phương thức.
  - Lấy dữ liệu yêu cầu và chuyển tới phương thức được yêu cầu dưới dạng đối số.
  - Xử lý mọi ngoại lệ mà phương thức ném ra
  - Hỗ trợ kết xuất View dựa trên kết quả của phương thức

Tạo một controller

- Trong ASP.NET MVC, class `ControllerBase` của namespace `System.Web.Mvc` là class cơ sở cho mọi controller
- Class `Controller` kế thừa từ class `ControllerBase` để cung cấp triển khai mặc định của một controller.
- Để tạo một controller trong ứng dụng ASP.NET MVC, bạn cần phải tạo ra một class kế thừa từ class `Controller` .
- Thay vì tạo ra controller một cách thủ công, bạn có thể dùng Visual Studio 2017 để tạo ra các cấu trúc thư mục cho ứng dụng một cách tự động

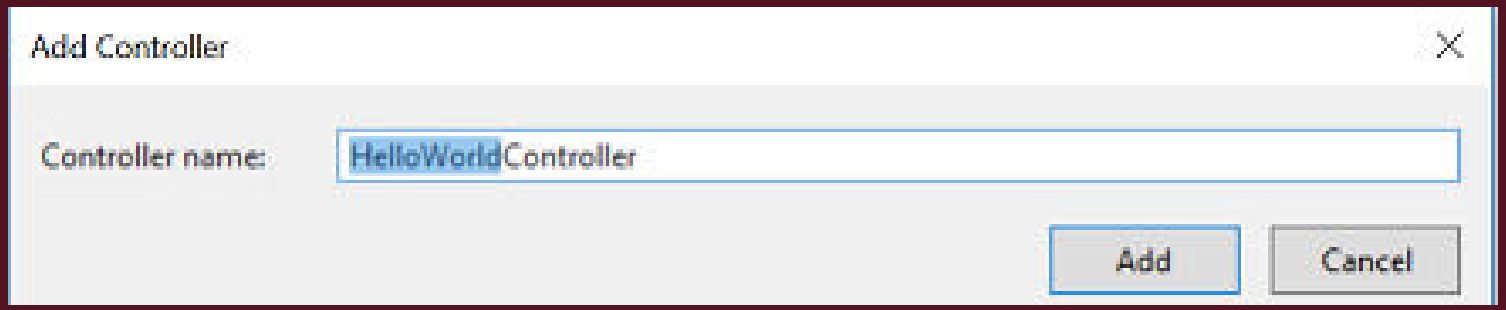
Tạo một controller

- Trong Visual Studio 2017 IDE, tạo 1 controller theo các bước sau:
  1. Bấm chuột phải vào thư mục **Controllers** ở cửa sổ **Solution Explore**
  2. Chọn **Add→Controller** từ thanh menu mới xuất hiện. Hộp thoại **Add Scaffold xuất hiện**:



Tạo một controller

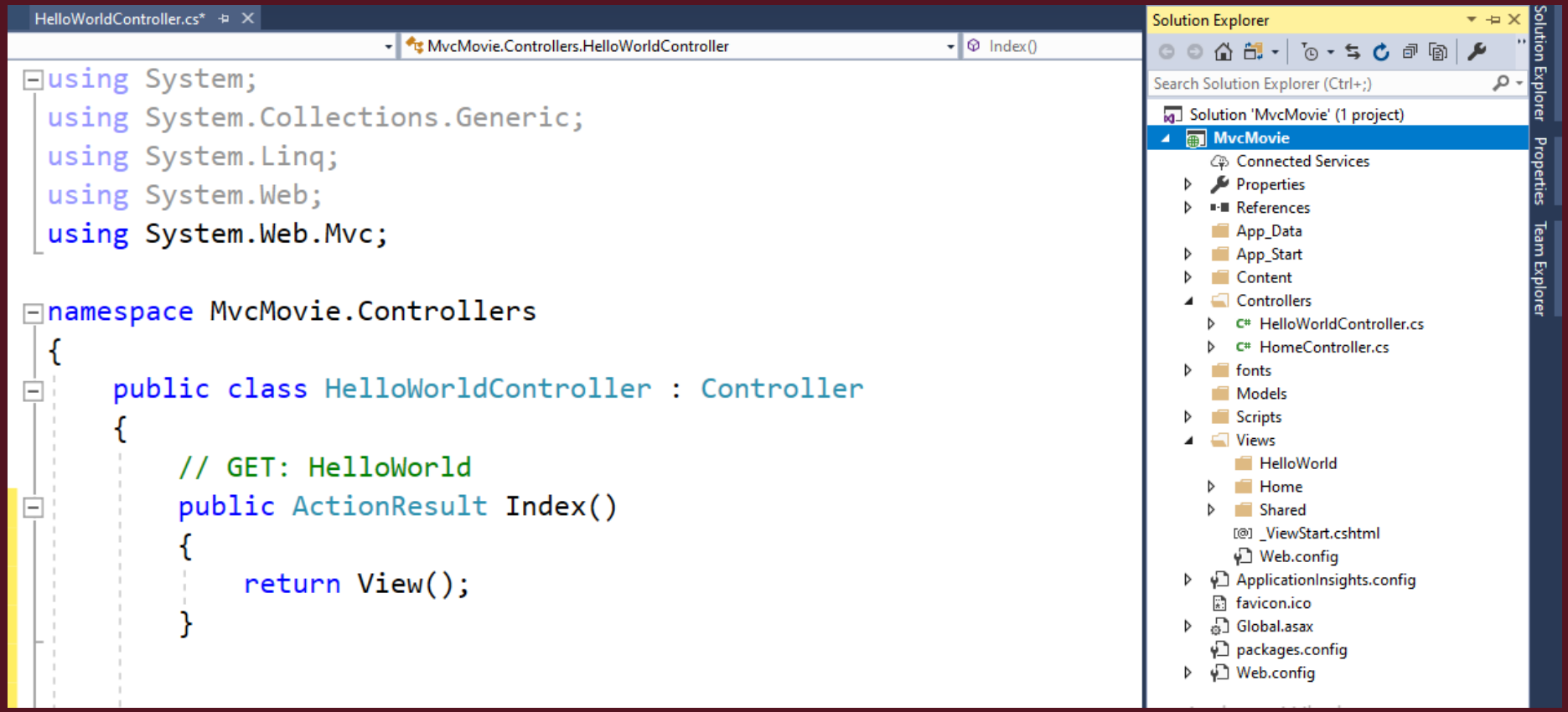
3. Chọn **MVC 5 Controller - Empty** trong hộp thoại **Add Scaffold**
4. Chọn **Add**. Cửa sổ **Add Controller** xuất hiện.
5. Gõ *HelloWorldController* trong phần **Controller name** như hình dưới:



6. Chọn **Add**. Cửa sổ **Solution Explorer** hiển thị controller *HelloWorldController* vừa mới tạo ở bên dưới thư mục **Controllers và** một thư mục mới **Views \ HelloWorld** được tạo .

Tạo một controller

- Ảnh dưới hiển thị nội dung file *HelloWorldController.cs và* cửa sổ Solution Explorer với controller vừa mới được tạo:

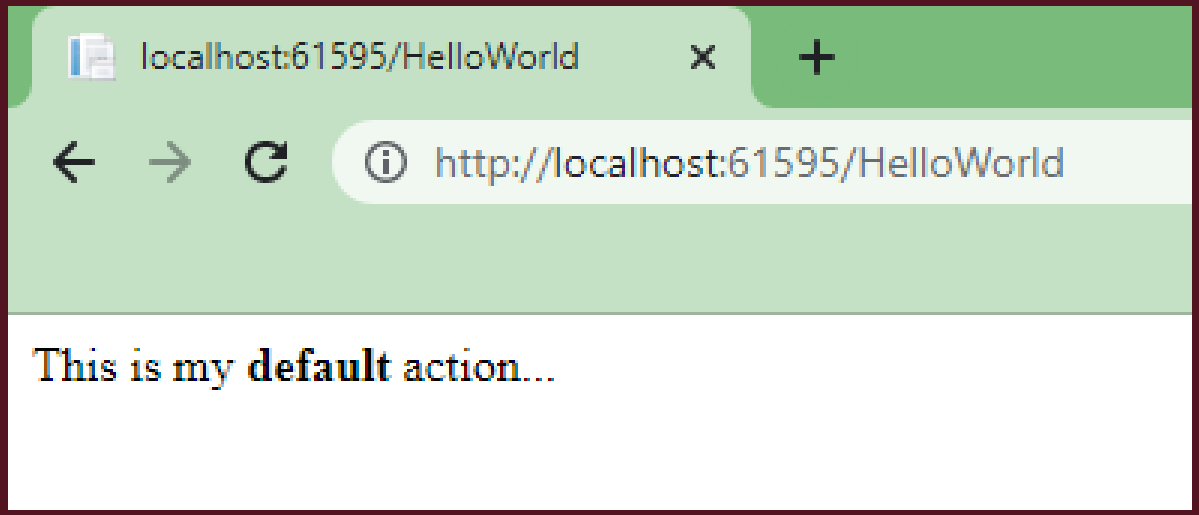


Thay thế nội dung của tệp bằng mã sau:

```
namespace MvcMovie.Controllers
{
    public class HelloWorldController : Controller
    {
        //
        // GET: /HelloWorld/

        public string Index()
        {
            return "This is my <b>default</b> action...";
        }

        //
        // GET: /HelloWorld/Welcome/

        public string Welcome()
        {
            return "This is the Welcome action method...";
        }
    }
}
```

Chạy ứng dụng

Nhấn F5 để chạy ứng dụng. Trong trình duyệt, thêm "HelloWorld" vào đường dẫn trên thanh địa chỉ để được theo mẫu: http://localhost:61595/HelloWorld



Tạo một controller

- Bạn có thể dùng cú pháp bên dưới để tạo một class Controller:

Code Snippet:

```
using System.Web.Mvc;
public class <Controller_Name>Controller:Controller
{
//Some code
}
```

Trong đó,

- `Controller_Name`: là tên của controller

Tạo một controller

- Dưới đây là một bộ khung của một class Controller:

Code Snippet :

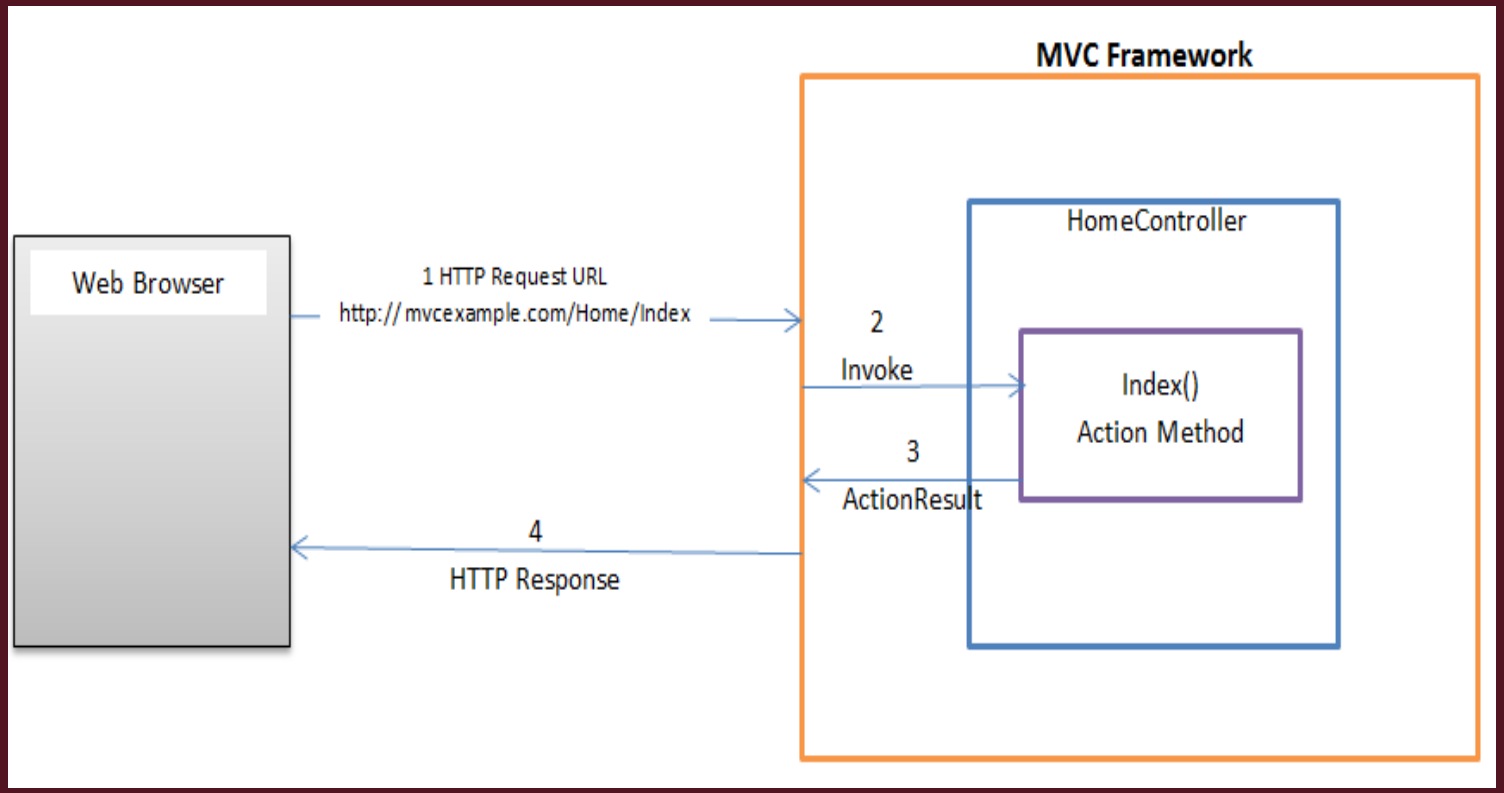
```
using System.Web.Mvc
public class TestController : Controller
{
   //Some code
}
```

  - Trong đoạn code này, một controller được tạo với tên `TestController`.

Làm việc với phương thức hành động

- Một class controller chứa một hoặc nhiều phương thức hành động, còn được gọi là các hành động của controller
- Các phương thức hành động:
  - Có nhiệm vụ xử lý các yêu cầu được gửi tới controller.
  - Thường sẽ trả về kiểu đối tượng `ActionResult` đóng gói kết quả của việc thực thi phương thức
- Hình dưới biểu thị cách phương thức hành động hoạt động:

Làm việc với phương thức hành động



- Các bước trong hình trên như sau:
  1. Trình duyệt gửi một yêu cầu HTTP.
  2. MVC Framework gọi đến phương thức hành động controller dựa trên URL yêu cầu
  3. Phương thức hành động thực thi và trả về đối tượng `ActionResult`. Đối tượng này đóng gói kết quả mà phương thức hành động thực thi.
  4. MVC Framework chuyển đổi `ActionResult` thành phản hồi HTTP và gửi lại trình duyệt

Làm việc với phương thức hành động

- Các quy tắc khi tạo ra phương thức hành động:
  - Phải được khai báo là **public**.
  - Không thể khai báo là **static**.
  - Không thể có nạp chồng phương thức
- Cú pháp tạo ra một phương thức hành động trong class `Controller`:

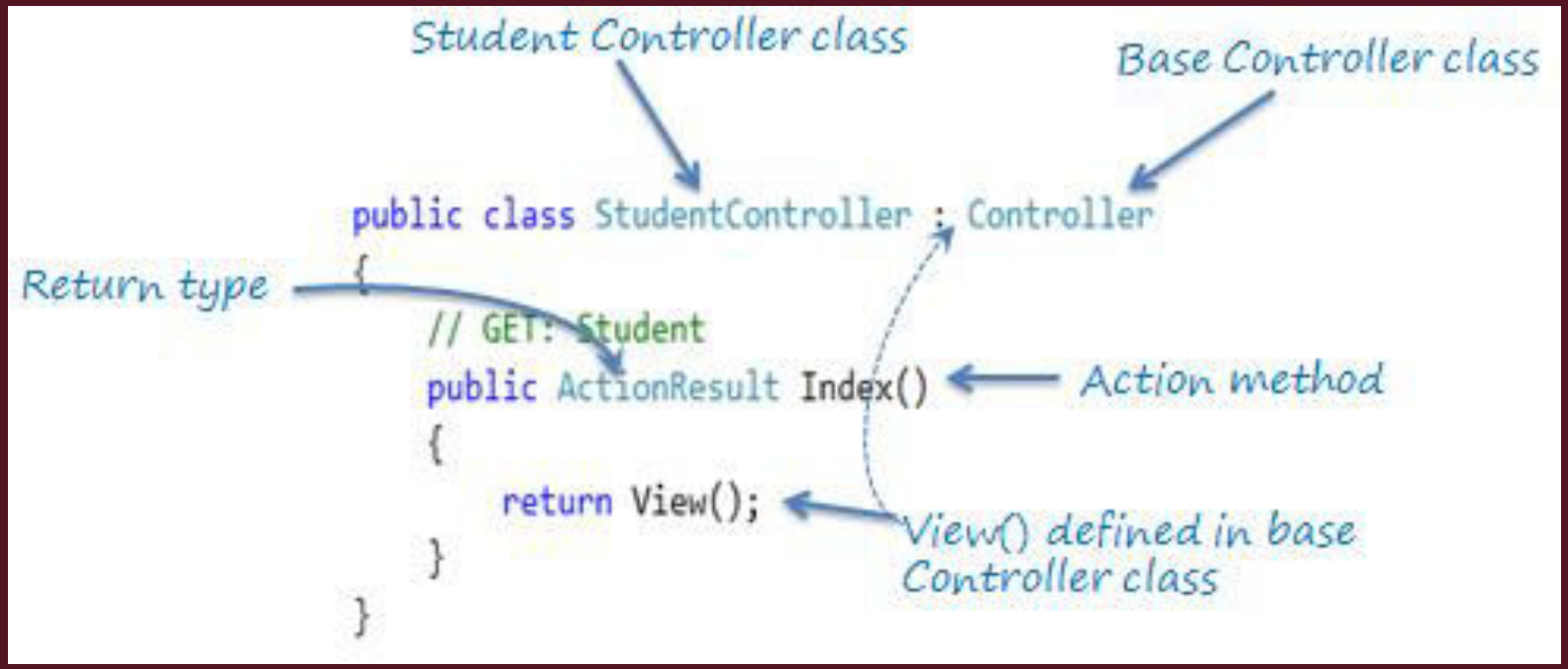
Code snippet

```
public ActionResult <ActionMethod_Name>()
 {
  /*Code to execute logic and return the result as ActionResult*/
 }
```

Trong đó,

- `<ActionMethod_Name>`: tên của phương thức hành động.

Ví dụ về phương thức action Index của lớp **StudentController**



phương thức Index là một phương thức public và nó trả về ActionResult bằng phương thức View (). Phương thức View () được định nghĩa trong lớp cơ sở Controller, trả về ActionResult

Làm việc với phương thức hành động

- Đoạn code sau biểu thị 2 phương thức hành động, với tên lần lượt là ***Index*** và ***About*** trong class `HomeController`:

Code Snippet:

```
using System.Web.Mvc;
public class HomeController : Controller
{
 public ActionResult Index()
   {
    /*Code to execute logic and return the result as ActionResult*/
   }
public ActionResult About()
   {
   /*Code to execute logic and return the result as ActionResult*/
   }
}
```

- Đoạn code trên tạo ra 2 phương thức với tên `Index` và `About` trong class `HomeController`. Cả 2 phương thức này đều được khai báo là `public` và đều trả về đối tượng `ActionResult`.

Làm việc với phương thức hành động

- Mặc dù hầu hết các phương thức hành động đều trả về đối tượng `ActionResult`, một phương thức hành động có thể trả về các dạng khác, ví dụ như `String`, `int`, or `bool`, như đoạn code dưới đây:

Code Snippet 3:

```
using System.Web.Mvc;
public class HomeController : Controller
 {
public bool IsValid()
   {
return true;
   }
public String Contact()
   {
return "Contact us at www.zynla.com";
   }
}
```

- Đoạn code tạo ra 2 phương thức: phương thức `IsValid` trả về giá trị `bool` và `Contact` trả về giá trị `String`.

ActionResult

- ActionResult:
  - Là một abstract class, là lớp cơ sở cho các lớp triển khai và cho ra các loại kết quả khác nhau.
  - Bao gồm HTML kết hợp với các dòng lệnh phía server và client để phản hồi các hành động người dùng.

ActionResult

Bảng dưới biểu thị các class được dùng phổ biến được kế thừa từ class `ActionResult` để cung cấp các kết quả khác nhau của một phương thức hành động:

Result Class	Description	Base Controller Method
ViewResult	Trả về HTML.	View()
EmptyResult	Represents No response.	
ContentResult	Không trả về cái gì cả.	Content()
FileContentResult, FilePathResult, FileStreamResult	Trả về nội dung của tập tin	File()

ActionResult

Result Class	Description	Base Controller Method
JavaScriptResult	Trả về JavaScript script.	JavaScript()
JsonResult	Trả về một JSON có thể sử dụng trong AJAX	Json()
RedirectResult	Chuyển hướng người dùng	Redirect()
RedirectToRouteResult	Chuyển hướng hành động khác.	RedirectToRoute()
PartialViewResult	Trả về HTML từ Partial view	PartialView()
HttpUnauthorizedResult	Trả về lỗi và HTTP Code	

Gọi phương thức hành động

- Trong một ứng dụng ASP.NET MVC, bạn có thể tạo ra nhiều phương thức hành động trong 1 controller
- Bạn có thể gọi đến 1 phương thức hành động bằng cách xác định 1 URL trong trình duyệt web có chứa tên Controller và phương thức hành động được gọi tới

Code Snippet:

```
http:// <domain_name> /<controller_name>/<actionmethod_name>
```

Trong đó,

- <domain_name>: tên miền của ứng dụng.
- <controller_name>: tên controller không có hậu tố Controller
- <actionmethod_name>: tên phương thức hành động được gọi tới.

Gọi phương thức hành động

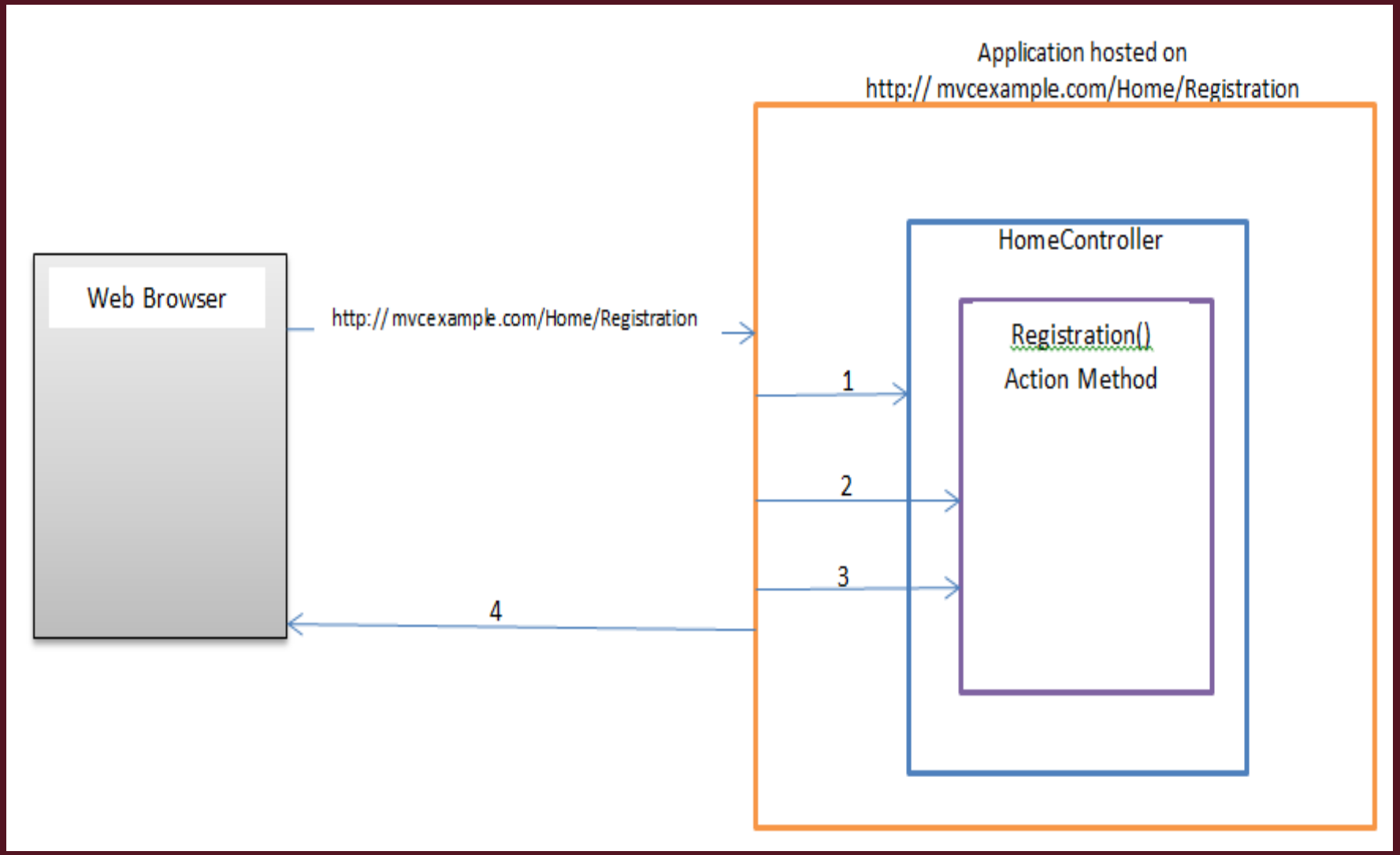
- Ví dụ, xét URL:

```
http://mvcexample.com/Home/Registration
```

- Khi URL được gửi tới ứng dụng thông qua trình duyệt web, MVC Framework sẽ thực hiện theo các bước sau:
  - Tìm class HomeController.
  - Tìm phương thức hành động `Registration()` trong class `HomeController`.
  - Thực thi phương thức `Registration()`
  - Gửi lại phản hồi cho trình duyệt.

Gọi phương thức hành động

- Các bước trên được mô tả dưới hình sau:



Truyền tham số

- Đôi lúc, bạn phải cung cấp input khác ngoài tên trang web khi yêu cầu một trang web.
- Xét URL sau:
  - `http://www.mvcexample.com/student/details?Id= 006`
  - URL trên sẽ gọi tới phương thức Details() của class `StudentController`
  - URL cũng chứa một tham số `Id` có giá trị `006`.
- Phương thức hành động `Details()` phải tiếp nhận tham số `Id` có kiểu `string` để có thể trả về các bản ghi về `Student` dựa theo tham số `Id`.

- Đoạn code sau biểu thị hành động Details tiếp nhận tham số Id:

Code Snippet:

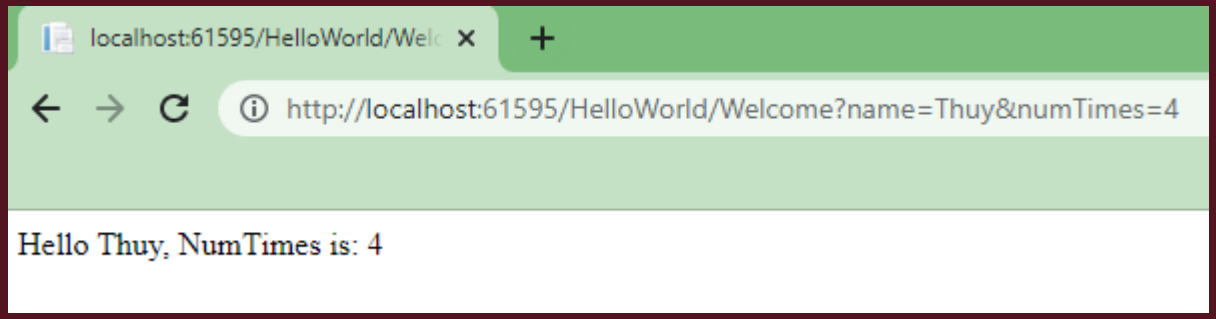
```
public ActionResult Details(string Id)
{
 /*Return student records
  based on the Id parameter as
  an ActionResultobject*/


}
```

Ví dụ truyền tham số

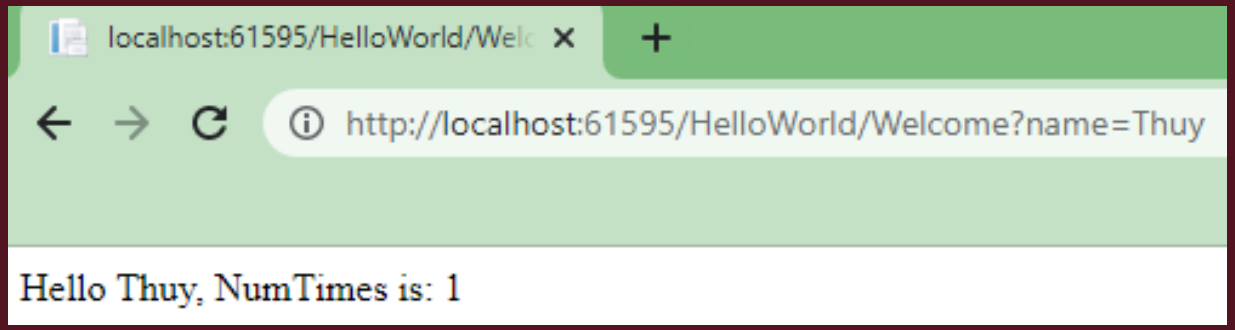
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MvcMovie.Controllers
{
    public class HelloWorldController : Controller
    {
        // GET: /HelloWorld/
        public string Index()
        {
            return "This is my <b>default</b> action...";
        }
        // GET: /HelloWorld/Welcome/
        public string Welcome(string name, int numTimes = 1)
        {
            return HttpUtility.HtmlEncode("Hello " + name + ", NumTimes is: " + numTimes);
        }
    }
}
```

Ví dụ truyền tham số

http://localhost:61595/HelloWorld/Welcome?name=Thuy&numTimes=4



http://localhost:61595/HelloWorld/Welcome?name=Thuy

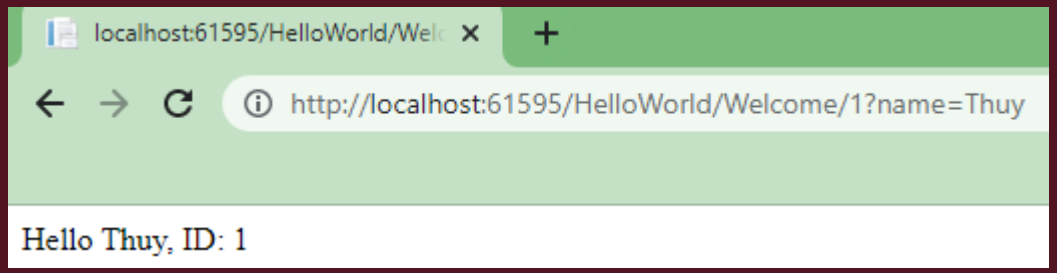


Ví dụ truyền tham số

Thay phương thức Welcome trong HelloWorldControler bằng:

```
public string Welcome(string name, int ID = 1) {
 return HttpUtility.HtmlEncode("Hello " + name + ", ID: " + ID);
}
```

Chạy ứng dụng và nhập URL sau:
http://localhost:61595/HelloWorld/Welcome/1?name=Thuy



Lần này, phân đoạn URL khớp với tham số tuyến ID. Phương thức hành động Welcome chứa một tham số (ID) khớp với đặc tả URL trong phương thức RegisterRoutes.

Ví dụ truyền tham số

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

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

Trong các ứng dụng ASP.NET MVC, việc truyền các tham số dưới dạng dữ liệu tuyến đường (như chúng ta đã làm với ID ở trên) thông thường hơn là chuyển chúng dưới dạng chuỗi truy vấn. Bạn cũng có thể thêm một tuyến đường để chuyển cả tham số Name và ID trong làm dữ liệu tuyến đường trong URL. Trong tệp *App_Start* \ *RouteConfig.cs* , hãy thêm tuyến đường “Hello":

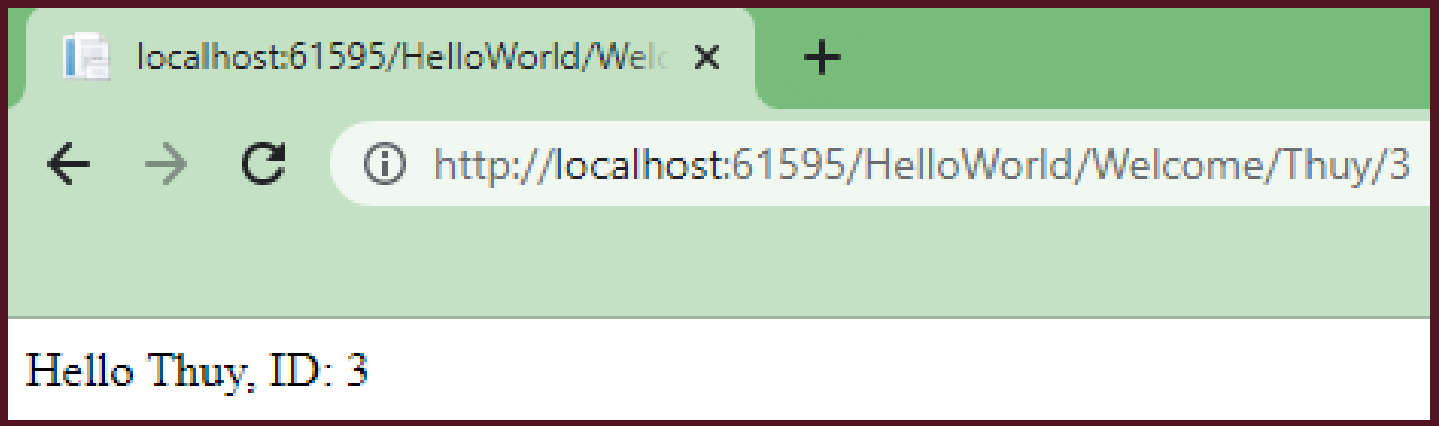
Ví dụ truyền tham số

```
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 }
      );

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

Ví dụ truyền tham số

Chạy ứng dụng và nhập URL sau:

http://localhost:61595/HelloWorld/Welcome/Thuy/3



Yêu cầu định tuyến

- MVC Framework giới thiệu định tuyến cho phép bạn xác định các mẫu URL với placeholder ánh xạ tới URL yêu cầu.
- Trong ứng dụng ASP.NET MVC, định tuyến:
  - Xác định cách ứng dụng sẽ xử lý và phản hồi với yêu cầu HTTP.
  - Mô tả đúng hành động của Controller mà yêu cầu cần định tuyến

Sử dụng định tuyến

- Định tuyến là quy trình ánh xạ các yêu cầu đến tới các hành động controller xác định.
- 2 chức năng chính của định tuyến:
  - Ánh xạ yêu cầu đến tới hành động của controller.
  - Xây dựng các URL gửi đi tương ứng với các hành động của controller
- Định tuyến được thực hiện bằng cách cấu hình các mẫu định tuyến trong ứng dụng. Quy trình cấu hình bao gồm các bước:
  - Tạo ra mẫu định tuyến
  - Đăng kí mẫu định tuyến với bảng định tuyến của MVC Framework
- Tại thời điểm tạo ứng dụng ASP.NET MVC , ứng dụng có thể đăng kí nhiều mẫu định tuyến với bảng định tuyến của MVC Framework.
- Bảng định tuyến cung cấp thông tin về cách công cụ định tuyến xử lý các yêu cầu được nối tới các mẫu đó.

Định tuyến mặc định

- Một ứng dụng MVC yêu cầu một định tuyến có thể xử lý các yêu cầu người dùng.
- Khi tạo ra một ứng dụng ASP.NET MVC trong Visual Studio .NET, một định tuyến được cấu hình tự động trong tập tin `RouteConfig.cs`
- Hình dưới biểu thị phương thức `MapRoute()` của class `RouteConfig`:

Code Snippet:

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

Định tuyến mặc định

- Trong đoạn code:
  - Các định tuyến thuộc loại `System.Web.Routing.RouteCollection` biểu diễn một tập hợp các định tuyến của ứng dụng
  - Phương thức `MapRoute()` định nghĩa một định tuyến đã được đặt tên là Default, một mẫu URL và một định tuyến mặc định.
  - Định tuyến mặc định được sử dụng nếu yêu cầu URL không phù hợp với mẫu URL được xác định trong phương thức `MapRoute()`. Ví dụ: nếu yêu cầu URL không chứa tên controller và một action, yêu cầu đó sẽ được định tuyến với action `Index` của controller `Home`.

Đăng kí định tuyến mặc định

- Một ứng dụng MVC chứa 1 tập tin Global.asax:
  - Khởi tạo ứng dụng với các chức năng của MVC Framework khi ứng dụng khởi động.
  - Chứa 1 class `MVCApplication` với phương thức `Application_Start()` .Trong phương thức này, bạn cần đăng kí định tuyến mặc định để Framework sử dụng định tuyến khi một yêu cầu được gửi tới ứng dụng.

Đăng kí định tuyến mặc định

- Đoạn code sau biểu thị class `MVCApplication` của file `Global.asax`:

Code Snippet:

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;

namespaceUrlsAndRoutes {
public class MvcApplication : System.Web.HttpApplication {
protected void Application_Start() {
RouteConfig.RegisterRoutes(RouteTable.Routes);

/*Code for registering other MVC components*/
}
}
}
```

Đăng kí định tuyến mặc định

- Trong đoạn code trên:
  - Class `MVCApplication` kế thừa từ class `System.Web.HttpApplication` định nghĩa các tính năng cơ bản được yêu cầu bởi các đối tượng trong ứng dụng ASP.NET
  - Phương thức `Application_Start()` gọi tới phương thức `RouteConfig.RegisterRoutes()` để đăng kí định tuyến mặc định để sử dụng trong ứng dụng.

Các mẫu URL

- Mẫu URL :
  - Phải được định nghĩa khi tạo một định tuyến.
  - Được ánh xạ với URL yêu cầu bởi công cụ định tuyến của MVC Framework.
  - Chứa các giá trị chữ và các placeholder được phân tách bởi kí tự '/'. Các placeholder trong 1 mẫu URL nằm trong 2 dấu ngoặc nhọn '{' và '}'. Ví dụ:

    `"{controller}/{action}/{id}"`
- URL phù hợp với mẫu URL trên:

  `http://www.mvcexample.com/student/records/36`
- Khi công cụ định tuyến nối URL trên với mẫu URL:
  - Chỉ định `student` là giá trị của `{controller}` placeholder
  - Chỉ định `records` là giá trị của `{action}` placeholder
  - Chỉ định `36` là giá trị của `{id}` placeholder

Các mẫu URL

- Một tham số URL có thể có một tổ hợp các giá trị chữ và các placeholder. Ví dụ:.

```
"student/{action}/{id}"
```

- Một số URL phù hợp để nối tới mẫu URL trên:
  - http://www.mvcexample.com/student/records/36
  - http://www.mvcexample.com/student/delete/21
  - http://www.mvcexample.com/student/view/23

Thứ tự định tuyến

- Đôi khi, bạn cần phải đăng kí nhiều định tuyến trong 1 ứng dụng ASP.NET MVC.
- Bạn cần cấu hình trình tự tùy theo việc định tuyến nào sẽ được thực thi.
- Công cụ định tuyến nối 1 URL yêu cầu với 1 mẫu URL bắt đầu từ định tuyến được đăng kí đầu tiên
- Khi bắt gặp 1 định tuyến vừa nối, công cụ định tuyến sẽ dừng quá trình nối. Điều này có thể dẫn tới 1 số kết quả không mong muốn

Thứ tự định tuyến

- Đoạn code dưới đây biểu diễn 2 định tuyến:

Code Snippet:

```
routes.MapRoute(
 name: "general",
  url: "{controller}/{action}",
  defaults: new { controller = "Home", action = "Index"});
    routes.MapRoute(
    name: "manager",
    url: "Manager/{action}",
    defaults: new { controller = "Manager", action = "Browse"}
   );
```

- Trong đoạn code:
  - Chứa 2 placeholder và đặt giá trị mặc định của tham số controller là `Home` và của tham số action là `Index`.
  - Định tuyến thứ 2 chứa 1 giá trị chữ là `Manager`, 1 placeholder, chỉ định giá trị của tham số controller là `Manager` và của tham số action là `Browse`.

Ràng buộc định tuyến

- Trong ứng dụng ASP.NET MVC, công cụ định tuyến cho phép bạn có thể áp dụng các ràng buộc xung quanh các giá trị placeholder.
- Bạn có thể sử dụng các ràng buộc trong các tình huống mà một ứng dụng có các URL định tuyến y hệt nhau nhưng dựa theo yêu cầu của ứng dụng, công cụ định tuyến sẽ giải quyết các URL cho các controller hoặc hành động khác nhau.

Code Snippet:

```
routes.MapRoute(
"Product",
"{controller}/{action}/{id}",
  new { controller = "Product", action = "Browse", id =
UrlParameter.Optional },
  new { id = "(|Jewellery|Jeans|Mobile)" }
  );
```

- Trong đoạn code, 1 ràng buộc được áp dụng cho định tuyến.Placeholder id chỉ có thể có 1 trong các giá trị `Jewellery`, `Jeans`, hoặc `Mobile`

Hạn chế định tuyến

- Bảng sau biểu thị các kết quả khi công cụ định tuyến ánh xạ các URL khác nhau dựa theo các ràng buộc định tuyến :

URL	Matching Results
http://www.mvcexample.com	Yes. The default values of the controller and action are specified as Product and Browse respectively.
http://www.mvcexample.com/Product	Yes. The default values of the action is specified as Browse.
http://www.mvcexample.com/Product/Browse	Yes. The id specified as an optional parameter.
http://www.mvcexample.com/Product/Browse/Jewellery	Yes. Jewellery specified in the URL is present in the list containing valid values for id parameter.
http://www.mvcexample.com/Product/Browse/Jeans	Yes. Jeans specified in the URL is present in the list containing valid values for id parameter.

Hạn chế định tuyến

URL	Matching Results
http://www.mvcexample.com/Product/Browse/Mobile	Yes. Mobile specified in the URL is present in the list containing valid values for id parameter.
http://www.mvcexample.com/Product/Browse/Laptop	No. Laptop specified in the URL is not present in the list containing valid values for id parameter.
http://www.mvcexample.com/Product/Browse/Glasses	No. Glasses specified in the URL is not present in the list containing valid values for id parameter.

Hạn chế định tuyến

- Đoạn code sau biểu thị một định tuyến với 1 ràng buộc định tuyến mà placeholder `id` placeholder chỉ có thể nối với giá trị `integer` :

Code Snippet:

```
routes.MapRoute(
name: "Product",
url: "{controller}/{action}/{id}",
defaults: new { controller = "Product", action = "Browse",
id=UrlParameter.
Optional},
constraints: new { id = @"\d*" }
);
```

- Trong đoạn code, định tuyến có 3 placeholder là `controller`, `action`, và `id`. Trong định tuyến này, giá trị mặc định của controller được đặt là `Product`, của action được đặt là `Browse`.
- Thêm vào đó, một ràng buộc được áp dụng với tham số `id`. Ràng buộc này sử dụng một biểu thức chính quy để quy định rằng tham số id chỉ có thể nối tới các giá trị kiểu `integer`.

Bỏ qua định tuyến

- MVC Framework cung cấp cho bạn khả năng bỏ qua định tuyến
- Sử dụng phương thức `IgnoreRoute()` của class `RoutesTable` để xác định định tuyến mà bộ định tuyến MVC sẽ bỏ qua.
- Đoạn code sau biểu diễn cách bỏ qua định tuyến:

Code Snippet:

```
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
```

- Trong đoạn code, phương thức `routes.IgnoreRoute()` chỉ ra rằng các nguồn với phần mở rộng `.axd` sẽ bị bỏ qua bởi công cụ định tuyến và được xử lý trực tiếp như phản hồi

Tổng kết

- Một controller có nhiệm vụ chặn các yêu cầu đến và thi hành các mã code thích hợp của ứng dụng
- Để tạo ra 1 controller trong ứng dụng ASP.NET MVC, bạn cần tạo ra một class C# kế thừa từ lớp Controller
- Một lớp controller có thể chứa 1 hoặc nhiều phương thức hành động, hay hành động của controller
- Mặc dù hầu hết các phương thức hành động đều trả về dạng đối tượng *ActionResult*, một phương thức hành động cũng có thể trả về các kiểu khác như String, int, bool
- Định tuyến là quy trình ánh xạ các yêu cầu đến với các hành động xác định của controller
- Khi tạo ra một định tuyến, bạn cần định nghĩa mẫu URL có thể chứa giá trị chữ và các placeholder được ngăn cách bởi kí tự '/'
- Công cụ định tuyến cho phép bạn áp dụng các ràng buộc xung quanh các giá trị placeholder và cũng có thể bỏ qua các định tuyến