# Khái niệm.

* Mẫu kiến trúc Model – View – Controller được sử dụng nhằm chi ứng dụng thành ba thành phần chính: model, view và controller. Nền tảng ASP.NET MVC giúp cho chúng ta có thể tạo được các ứng dụng web áp dụng mô hình MVC thay vì tạo ứng dụng theo mẫu ASP.NET Web Forsm. Nền tảng ASP.NET MVC có đặc điểm nổi bật là nhẹ (lighweigt), dễ kiểm thử phần giao diện (so với ứng dụng Web Forms), tích hợp các tính năng có sẵn của ASP.NET. Nền tảng ASP.NET MVC được định nghĩa trong namespace System.Web.Mvc và là một phần của name space System.Web.
* **Vài nét về mô hình 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 Product (sản phẩm) sẽ lấy dữ liệu từ CSDL, thao tác trên dữ liệu và sẽ cập nhật dữ liệu trở lại vào bảng Products ở SQL Server.

Trong các ứng dụng nhỏ, model thường là chỉ là một khái niệm nhằm phân biệt hơn là được cài đặt thực thụ, ví dụ, nếu ứng dụng chỉ đọc dữ liệu từ CSDL và gởi chúng đến view, ứng dụng khong cần phải có tầng model và các lớp lien quan. Trong trường hợp này, dữ liệu được lấy như là một đối tượng model (hơn là tầng model).

**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 Products sẽ hiển thị các hộp văn bản, drop-down list, và các check box dựa trên trạng thái hiện tại của một đối tượng Product.

**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. 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.

* Ngày 15-08-2012, Microsoft đã cho ra phiên bản ASP.NET MVC 4 với khá nhiều tính năng mới, giao diện cũng được cải thiện khá nhiều so với phiên bản trước đó.

# Lịch sử.

|  |  |
| --- | --- |
| **Ngày** | **Phiên bản** |
| 10-12-2007 | ASP.NET MVC CTP |
| 13-03-2009 | ASP.NET MVC 1.0 |
| 10-03-2010 | ASP.NET MVC 2.0 |
| 13-01-2011 | ASP.NET MVC 3.0 |
| 20-09-2011 | ASP.NET MVC 4.0 Developer Preview |
| 15-02-2012 | ASP.NET MVC 4.0 Beta |
| 15-08-2012 | ASP.NET MVC 4 |

# Tổng quan về những tính năng mới

## ASP.NET Web API

- ASP.NET MVC 4 bao gồm cả ASP.NET Web API, 1 framework mới cho phép tạo ra các dịch vụ HTTP mà có thể được truy cập bởi các trình duyệt của client và cả các thiết bị di động hoặc máy tính bảng. Bạn có thể sử dụng XML hoặc JSON cho API của bạn. JSON đặc biệt tiện dụng cho các ứng dụng mobile kết nối chậm. Bạn có thể dùng Jquery để gọi 1 API.

-Nó cũng là 1 platform lý tưởng để xây dựng các dịch vụ RESTful.

-Các đặc tính nổi bật của Web API:

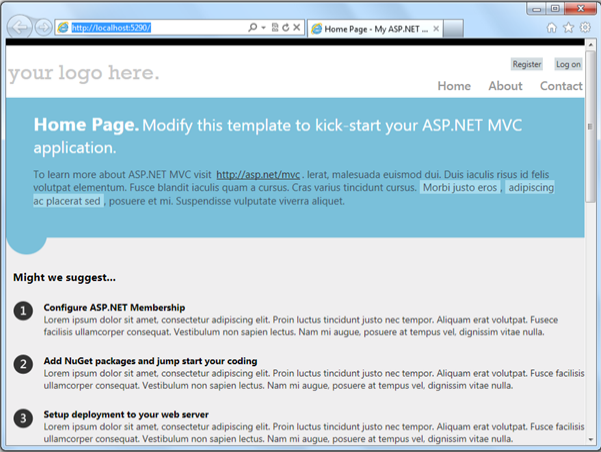
* Mô hình lập trình HTTP hiện đại (Modern HTTP programming model) :
* Truy cập trực tiếp và thao tác với các HTTP request và response trong Web APIs sử dụng strongly typed HTTP object model. Có thể lập trình theo mô hình này ở phần client, bằng cách sử dụng kiểu HttpClient.
* Hỗ trợ đầy đủ cho việc định tuyến (Full support for routes):
* Web APIs hỗ trợ đầy đủ các khả năng định tuyến, bao gồm cả các tham số định tuyến (route parameters) và các ràng buộc (constraints).
* Ngoài ra, Web APIs còn hỡ trợ đấy đủ cho các HTTP Verbs (POST, GET, PUT, DELETE..). Chúng ta không cần phải dùng những attribute như [HttpPost] cho các lớp cho các phương thức nữa.
* Thương thảo nội dung (Content negotiation):
* Client và server có thể làm việc cùng nhau để đưa ra 1 định dạng đúng cho dữ liệu được trả về từ 1 API.
* Nó cung cấp hỗ trợ mặc định cho các định dạng: XML, JSON và Form URL-encoded, và bạn hoàn toàn có thể mở rộng hỗ trợ này ra bằng cách thêm vào những định dạng mới của riêng bạn.
* Model binding and validation:
* Cung cấp cách thức dễ dàng nhất để rút trích dữ liệu từ các phần khác nhau của 1 HTTP request và chuyển các phần đó thành các đối tượng .NET. Các đối tượng này có thể được dùng bởi các Web API action.
* Các bộ lọc (Filters):
* Web API hỗ trợ các bộ lọc, bao gồm các bộ lọc đã biết như [Authorize].
* Bạn hoàn toàn có thể tạo ra 1 bộ lọc mới cho các action, xác thực người dùng (authorization), xử lý ngoại lệ (exception handling).
* Query composition:
* Bằng cách trả về Iqueryable<T>, Web API hỗ trợ việc truy vấn dữ liệu thông qua việc sử dụng OData URL conventions.
* Tăng cường khả năng kiểm thử các chi tiết HTTP (Improved testability of HTTP details):
* Các Web API action giờ đây có thể làm việc với những thực thể (instances) *HttpRequestMessage* và *HttpResponceMessage* để quy định các chi tiết của HTTP.
* Improved Inversion of Control (IoC) via DependencyResolver:
* Web API giờ đây có thể sử dụng mẫu thiết kế *Service locator* bằng cách sử dụng MVC dependency resolver để triệu gọi các đối tượng từ các lớp khác nhau.
* Code-based configuration:
* Cấu hình Web API chỉ được thực hiện bằng các mã lệnh.

→Các tập tin cấu hình khá ngắn gọn hơn.

* Self-host:
* Web APIs có thể được host trên tiến trình của bạn thay vì chỉ IIS mà vẫn tận dụng được tính năng định tuyến.

## Enhancements to Default Project Templates:

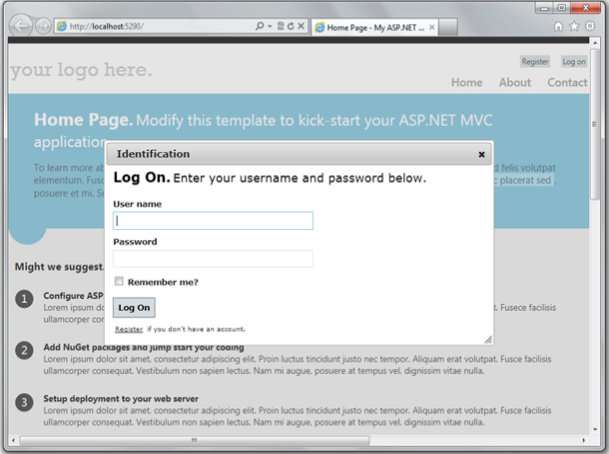
* Template được dùng để tạo mới 1 project MVC 4 được cải thiện về giao diện:



* Ngoài ra, để cải thiện giao diện, Một số chức năng mới đã được sử dụng trong Template này. Template sử dụng 1 kỹ thuật gọi là *adaptive rendering* để cải thiện giao diện cả trình duyệt trên máy tính lẫn điện thoại:

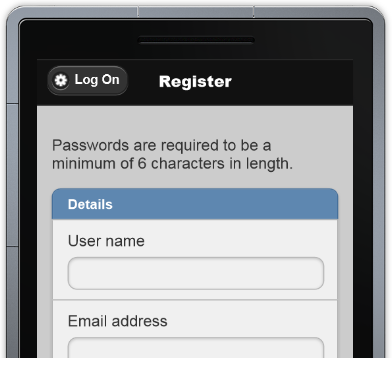


* Để hiểu rõ hơn về kỹ thuật *adaptive rendering*,bạn có thể sử dụng chương trình giả lập điện thoại trên máy tính hoặc đơn giản chỉ cần resize lại kích thước cửa sổ trình duyệt trên máy tính. Khi cửa sổ trình duyệt nhỏ tới 1 mức nào đó, layout của trang sẽ thay đổi.
* Một điểm mới khác đối với template mặc định của project là việc sử dụng Javascript để cung cấp 1 giao diện người dùng (UI) phong phú hơn. Những link Login và Register mà được dùng trong teamplate là những ví dụ cụ thể cho thấy việc sử dụng các Jquery UI Dialog để hiển thị 1 màn hình Login khá bắt mắt:



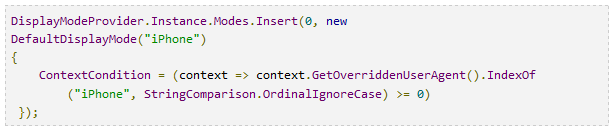
## Mobile Project Template:

* Nếu bạn bắt đầu 1 project mới và muốn tạo mới 1 trang cho trình duyệt của mobile hoặc máy tính bảng, bạn có thể sử dụng Mobile Application template mới. Template này dựa trên Jquery Mobile, 1 thư viện mã nguồn mở cho phép tối ưu hóa giao diện cảm ứng, Ứng dụng cảm ứng của bạn sẽ dễ dàng sử dụng hơn.



## Display Modes:

* Cho phép 1 ứng dụng có thể lựa chọn các *view* chế độ nhìn dựa trên trình duyệt đang yêu cầu. Ví dụ, nếu 1 trình duyệt desktop yêu cầu trang Home Page, ứng dụng có thể dùng template *Views\Home\Index.cshtml*. Nếu một trình duyệt mobile yêu cầu trang Home Page, ứng dụng có thể trả về template *Views\Home\Index.mobile.cshtml.*
* Các Layout và partials có thể được override cho các loại trình duyệt khác nhau. Ví dụ:
* Nếu thư mục *View\Shared* của bạn có chứa cả 2 template: *\_Layout.cshtml* và *\_Layout.mobile.cshtml,* Mặc định ứng dụng sẽ sử dụng *\_Layout.mobile.cshtml* cho các yêu cầu từ các trình duyệt mobile và *\_Layout.cshtml* cho các yêu cầu khác.
* Nếu thư mục có chứa cả 2 template: *\_MyPartial.cshtml* và \_*MyPartial.mobile.cshtml*, lời chỉ thị *@Html.Partial("\_MyPartial")* sẽ render \_*MyPartial.mobile.cshtml* cho các yêu cầu đến từ trình duyệt mobile và \_*MyPartial.cshtml* cho các yêu cầu khác.
* Nếu bạn muốn tạo mới view, layout hoặc partial view cho các thiết bị khác, bạn hoàn toàn có thể đăng ký 1 thực thể (instance) DefaultDisplayMode để chỉ rỏ tên gọi sẽ được dùng khi có 1 yêu cầu thỏa mãn những điều kiện cụ thể. Ví dụ, bạn có thể thêm đoạn code sau vào phương thức Application\_Start trong file Global.asax để đăng ký 1 chuỗi “iPhone” như 1 chế độ nhìn (Display mode) mà sẽ được chấp nhận khi có trình duyệt iPhone thực hiện yêu cầu:



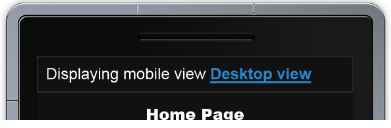
* Sau khi chạy đoạn code này, khi một trình duyệt iPhone thực hiện 1 yêu cầu, ứng dụng của bạn sẽ sử dụng layout *Views\Shared\\_Layout.iPhone.cshtml* (Nếu layout này tồn tại).

## jQuery Mobile, the View Switcher, and Browser Overriding:

* jQuery Mobile là một thư viện mã nguồn mở cho phép tối ưu hóa giao diện cảm ứng (touch-optimized UI). Nếu bạn muốn sử dụng jQuery Mobile kết hợp với ứng dụng của mình, bạn có thể download và cài đặt 1 NuGet package. Để cài đặt gói này từ Visual Studio Package Console, gõ mã lệnh dưới đây:



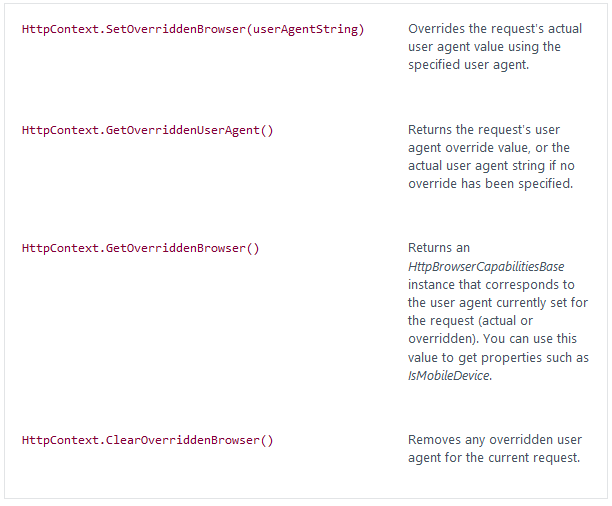
* Đoạn lệnh này giúp cài đặt jQuery Mobile và một vài files tiện ích (helper), bao gồm:
* *Views/Shared/\_Layout.Mobile.cshtml*, đây là một layout jQuerry Mobile cơ bản.
* Một component view-switcher bao gồm partial *Views/Shared/\_ViewSwitcher.cshtml* và controller *ViewSwitcherController.cs*
* Sau khi cài đặt gói này, chạy ứng dụng của bạn dùng trình duyệt mobile, bạn sẽ thấy 1 vài điểm đổi mới trên trang của mình, bởi vì jQuery Mobile tác động tới việc xử lý layout và styling.
* Một quy ước cho những trang web mobile-optimized là thêm 1 link “Desktop view” hoặc “Full site mode”…để khi người dùng click vào có thể chuyển trang sang phiên bản desktop. jQuery.Mobile.MVC package có chứa 1 thành phần (component) đơn giản để phục vụ mục đích này. Nó được dùng mặc định trong view *Views\Shared\\_Layout.Mobile.cshtml* và khi trang được hiển thị nó sẽ trông giống như hình bên dưới:



* Nếu người dùng click vào link này, trang sẽ được chuyển sang chế độ desktop.
* Do mặc định layout của desktop không chứa view-switcher, người dùng không thể chuyển trang về chế độ mobile. Để làm được việc này, thêm đoạn code dưới đây vào thẻ <body> :



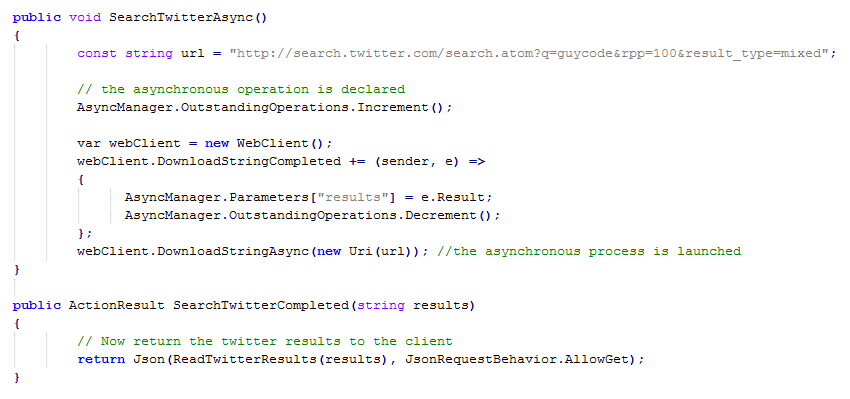
* View switcher dùng một tính năng mới gọi là Browser Overriding. Tính năng này cho phép ứng dụng của bạn xử lý những yêu cầu như thể những yêu cầu đó đến từ 1 trình duyệt khác (user agent) chứ không phải là trình duyệt thực tế. Danh sách các phương thức mà Browser Overriding cung cấp:



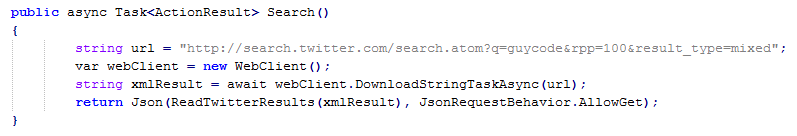
* Browser Overriding là một tính năng mới cốt lõi của ASP.NET MVC 4 và được tích hợp sẵn mà không cần cài đặt jQuery.Mobile.MVC package. Tuy nhiên, nó chỉ tác động tới sự lựa chọn view, layout và partial-view chứ không tác động tới bất cứ tính năng ASP.NET mới nào khác mà phụ thuộc vào đối tượng *Request.Browser.*
* Mặc định, user-agent override được lưu trữ bằng cách sử dụng cookie. Nếu bạn muốn lưu trữ ở nơi khác (ví dụ như lưu trong database), bạn có thể thay thế default provider (BrowserOverrideStores.Current).

## Task Support for Asynchronous Controllers:

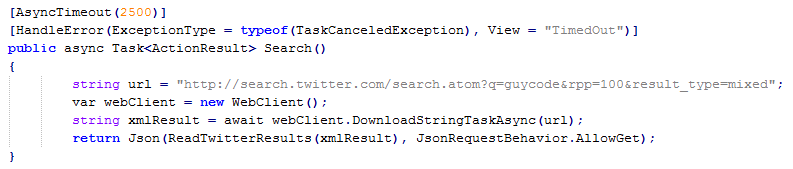
* Đối với những action đòi hỏi thời gian xử lý lâu (Upload file…) -> sử dụng Async Controller.
* Asynchronous controllers đã có mặt ở phiên bản ASP.NET MVC 3, tuy nhiên để sử dụng nó bạn phải viết những đoạn code khá dài dòng:
  + Viết 2 phương thức tên giống nhau nhưng phương thức đầu tiên kết thúc với cụm từ Async. Phương thức thứ 2 kết thúc với cụm từ Completed



* Trong MVC 4, bạn có thể viết các phương thức async action như những phương thức đơn lẻ -> rút ngắn code. Phương thức này trả về đối tượng kiểu Task hoặc Task<ActionResult>:



* Phương thức này còn hỗ trợ timeout. Để thiết lập thời gian giới hạn bạn dùng dòng lệnh AsyncTimeout(value).
* Ví dụ dưới đây thiết lập timeout là 2500 mili giây. Nếu quá thời gian giới hạn này, view “Timeout” sẽ được hiển thị cho người dùng.



## Azure SDK

* ASP.NET MVC 4 hỗ trợ phiên bản Window Azure SDK 1.6 và những phiên bản mới hơn.

## Database Migrations

* Các projects trong ASP.NET MVC 4 hiện tại bao gồm luôn cả Entity Framework 5. Một trong những đặc điểm tuyệt vời của Entity Framework 5 là hỗ trợ database migrations. Tính năng này cho phép chúng ta dễ dàng phát triển lược đồ cơ sở dữ liệu bằng cách sử dụng code-focused migration (sự chuyển đổi tập trung vào mã) nhằm bảo vệ dữ liệu trong cơ sở dữ liệu.

## Empty Project Template

* MVC Empty project bây giờ hoàn toàn trống để chúng ta có thể bắt đầu dự án mới hoàn toàn sạch sẽ. Các phiên bản trước của mẫu Empty project được đổi tên thành Basic.

## Thêm Controller vào bất kỳ thư mục nào trong dự án

* Chúng ta có thể click chuột phải và chọn Add Controller từ bất kỳ thư mục trong dự án MVC. Điều này cho phép chúng ta tổ chức linh hoạt các Controller như mong muốn bao gồm cả việc giữ MVC Controller và Web API Controller trong các thư mục riêng biệt.

## Bundling và Minification

* Bundling và minification framework cho phép chúng ta làm giảm số lượng yêu cầu HTTP là một trang Web cần được thực hiện bằng cách kết hợp các tập tin cá nhân thành một tập tin đóng gói duy nhất, cho cả script và CSS. Nó có thể làm giảm kích thước tổng thể của những yêu cầu đó bằng cách làm giảm nội dung của các gói này. Làm giảm có thể bao gồm các thao tác như loại bỏ các khoảng trắng để rút ngắn tên biến hay thậm chí bỏ đi những CSS selectors dựa trên ngữ nghĩa của chúng. Các gói được khai báo và cấu hình trong code và có thể dễ dàng tham chiếu trong views thông qua các phương thức trợ giúp mà có thể tạo ra một liên kết duy nhất tới gói đó hoặc là khi debug thì có thể tạo nhiều liên kết tới những nội dung con bên trong gói.

## Cho phép Logins từ Facebook và những tài khoản khác sử dụng OAuth và OpenID

* Mẫu mặc định trong ASP.NET MVC Internet Project bao gồm hỗ trợ OAuth và OpenID login bằng cách sử dụng thư viện DotNetOpenAuth.