Mẫu kiến trúc MVC (Model – View – Controller) trong ASP.NET Core

ASP.NET Core MVC là tên gọi của framework trong ASP.NET Core thực thi mô hình kiến trúc MVC. Framework này giúp phát triển nhiều loại ứng dụng khác nhau, từ ứng dụng web truyền thống đến ứng dụng đơn trang hoặc Web API.

Từ bài học này chúng ta chuyển sang nội dung về MVC framework trong ASP.NET Core. Với bài học đầu tiên, chúng ta sẽ xem xét tổng quan về kiến trúc MVC cũng như đặc điểm triển khai kiến trúc này trên ASP.NET Core.

> Nếu bạn chưa từng học phát triển ứng dụng web hoặc không quen thuộc với phát triển ứng dụng web với các công nghệ Microsoft, bạn nên xuất phát với Razor Pages. Phần nội dung về Razor Pages trình bày chi tiết những vấn đề cơ bản cho người mới bắt đầu.

NỘI DUNG CỦA BÀI [Ẩn]

Mẫu kiến trúc MVC

MVC là tên gọi tắt của Model – View – Controller – một *mẫu kiến trúc* (architectural pattern) lâu đời và rất phổ biến trong phát triển phần mềm.

Mẫu kiến trúc MVC được áp dụng rộng rãi trong ứng dụng web, desktop và mobile. Trên thực tế, mẫu MVC nguyên bản vốn được xây dựng dành cho ứng dụng với giao diện đồ họa tương tự ứng dụng desktop chứ không phải cho ứng dụng web. Tuy nhiên, hiện nay mẫu kiến trúc này được sử dụng rộng rãi nhất trong các web framework.

> Lưu ý phân biệt mẫu kiến trúc (architechtural pattern) với mẫu thiết kế (design pattern). Nhiều tài liệu sử dụng lẫn lộn các thuật ngữ này.

ASP.NET Core MVC chỉ là một trong số các web framework vận dụng mẫu kiến trúc MVC. Bạn có thể cũng đã nghe về Django (Python), Rails (Ruby) hay Spring (Java). Đây là các web framework nổi tiếng khác vận dụng kiến trúc MVC.

Mỗi ứng dụng hoặc framework diễn giải MVC theo cách riêng của mình và thường tập trung hơn vào một số khía cạnh của mẫu kiến trúc này. Ví dụ, mẫu MVC ứng dụng trên Rails hay Spring không hoàn toàn giống với ASP.NET Core, mặc dù đều có 3 thành phần cơ bản tương tự nhau.

Tuy nhiên, dù vận dụng ở đâu và theo cách nào, mẫu này cũng hướng tới phân tách việc biểu diễn dữ liệu khỏi quản lý và xử lý dữ liệu. Để tạo ra sự phân tách này, mẫu MVC phân chia ứng dụng ra 3 thành phần với nhiệm vụ cơ bản như sau:

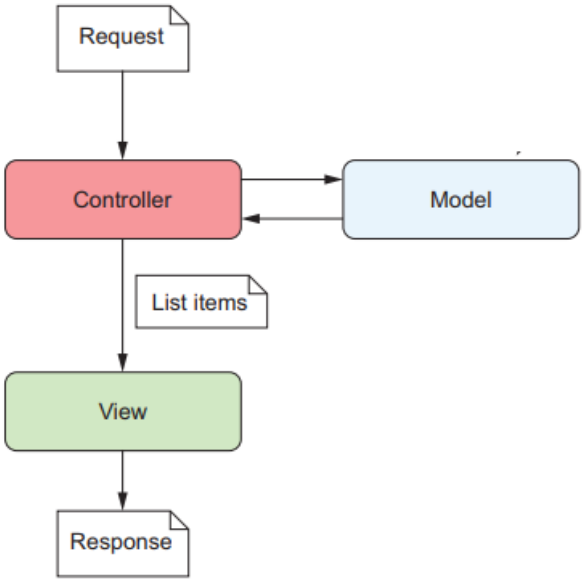
- Model – dữ liệu và trạng thái của ứng dụng.
- View – khuôn mẫu dành cho hiển thị dữ liệu.
- Controller – cập nhật model và lựa chọn view.

Mỗi thành phần của ứng dụng MVC chỉ chịu trách nhiệm cho một mảng duy nhất. Khi các thành phần kết hợp lại và tương tác với nhau sẽ tạo ra giao diện người dùng.

Tương tác giữa các thành phần trong kiến trúc MVC

Nhìn chung, thứ tự các sự kiện xảy ra khi ứng dụng phản ứng với tương tác/yêu cầu của người dùng như sau:

- Controller nhận yêu cầu.
- Controller lấy dữ liệu từ model, hoặc cập nhật dữ liệu của model.
- Controller lựa chọn view phù hợp cho việc hiển thị dữ liệu và chuyển dữ liệu của model sang cho view.
- View sử dụng dữ liệu của model để sinh ra giao diện.



Với cách mô tả MVC như trên, controller đảm nhiệm vai trò điểm tiếp nhận của tương tác. Người dùng trao đổi với controller để bắt đầu một tương tác.

Trong ứng dụng web, tương tác này chính là truy vấn HTTP. Như vậy khi một truy vấn HTTP tới ứng dụng, controller sẽ xử lý truy vấn. Tùy thuộc vào truy vấn, controller có thể thực hiện nhiều hoạt động khác nhau. Tuy nhiên các hoạt động này hầu như đều thực hiện trong sự kết hợp với model.

Model hoàn toàn độc lập với cách thức hiển thị của dữ liệu trên giao diện do view đảm nhiệm.

> Model có thể hiểu, theo cách đơn giản nhất, là dữ liệu. Tuy nhiên, trong các MVC framework, model có thể hiểu theo nhiều cách khác nhau. Trong phần sau của bài học này chúng ta sẽ trao đổi kỹ hơn về "model" trong ASP.NET Core MVC.

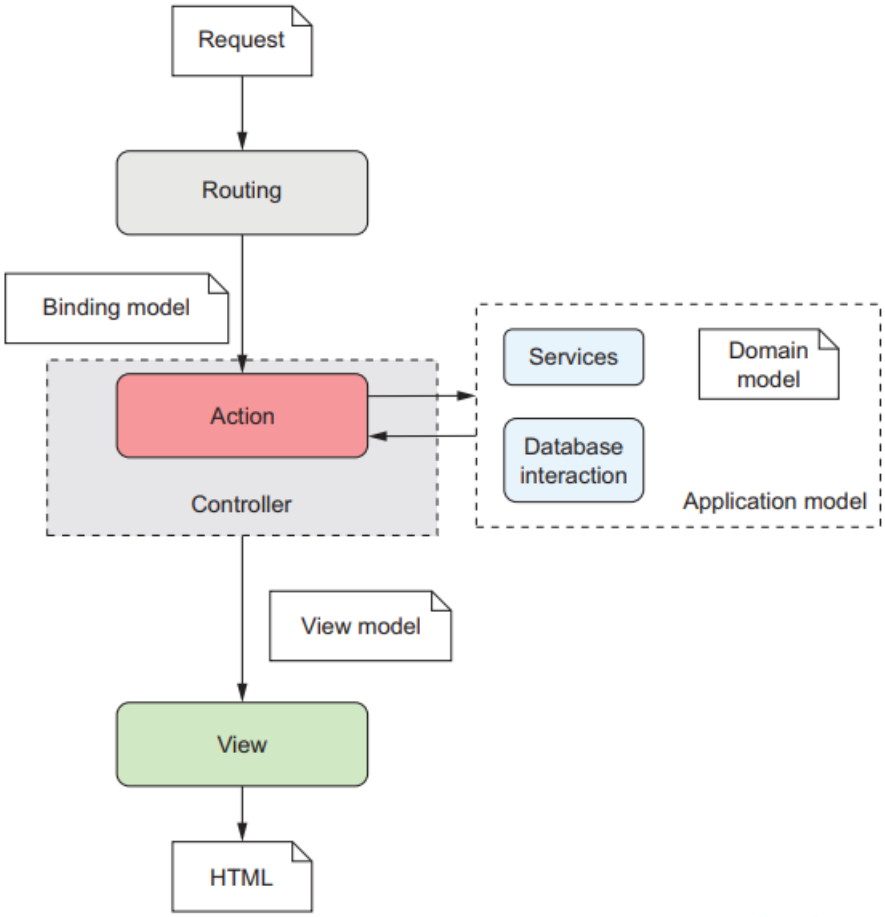
Sự độc lập giữa model và view giúp đơn giản hóa việc test. Thông thường rất khó để test giao diện. Khi không liên quan đến giao diện, thành phần model còn lại sẽ dễ dàng hơn để test.

Sự độc lập này cũng cho phép controller lựa chọn view cho phù hợp với yêu cầu. Tức là, với cùng một dữ liệu (model), controller có thể lựa chọn những cách hiển thị (view) khác nhau. Ví dụ, trong một ứng dụng web thông thường, controller sẽ lựa chọn HTML view. Nếu truy vấn đến từ một ứng dụng khác, controller có thể lựa chọn loại view đặc biệt ở dạng JSON hoặc XML.

MVC trong ASP.NET Core

Ở trên chúng ta đã nhắc tới kiến trúc MVC ở mức độ sơ lược. Như đã nói, mỗi framework vận dụng MVC với những đặc thù riêng. ASP.NET Core cũng vậy. Chúng ta sẽ xem xét một số khái niệm gắn với cách thực hiện kiến trúc MVC trong ASP.NET Core.

Dưới đây là sơ đồ chi tiết minh họa hoạt động của một ứng dụng xây dựng theo kiến trúc MVC trong ASP.NET Core.



Đây là một chu kỳ xử lý từ lúc nhận được truy vấn HTTP cho đến khi sinh ra HTML.

Truy vấn HTTP sẽ được cơ chế **routing** ánh xạ sang một phương thức xác định gọi là action.

Trong ASP.NET Core MVC, **action** là phương thức được thực thi để đáp ứng lại một truy vấn.

Để thực thi, action cần đến dữ liệu đầu vào chứa trong truy vấn HTTP. Dữ liệu được trích ra từ truy vấn thông qua cơ chế **model binding**.

Binding model là một object đóng vai trò "thùng chứa" dữ liệu trích xuất ra từ truy vấn để cung cấp cho action. Binding model là kết quả hoạt động của cơ chế model binding và là tham số đầu vào cho action.

Controller trong ASP.NET Core là class chứa các action có quan hệ nhất định.

Action khi thực thi sẽ tương tác với các thành phần còn lại của ứng dụng như các dịch vụ, cơ sở dữ liệu.

Với cách tiếp cận DDD (Domain-driven Design), phần dữ liệu nghiệp vụ được thể hiện qua các **domain model**.

Tất cả các thành phần dịch vụ, domain model, v.v., được gọi chung là **application model**.

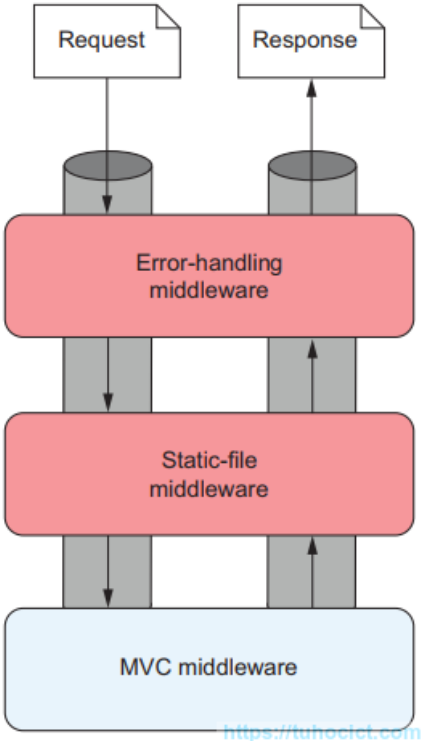
Quá trình tương tác này sẽ sinh ra dữ liệu phục vụ cho hiển thị, gọi là view model.

View model là object đơn giản chứa dữ liệu cần thiết để sinh ra giao diện. Thông thường view model là một biến thể của dữ liệu lấy được từ application model cùng với dữ liệu phụ trợ cho hiển thị (như tiêu đề, phân trang, v.v.).

View trong ASP.NET Core MVC là các trang Razor chứa loại mã hỗn hợp C# + HTML theo cú pháp Razor. Vì vậy người ta cũng thường gọi view trong ASP.NET Core MVC là **Razor view**. Kết quả xử lý của Razor view là HTML.

MVC middleware trong ASP.NET Core

Mô hình kiến trúc MVC trong ASP.NET Core được thực thi trong một middleware có tên là **MVC middleware**.



MVC Middleware đặt ở cuối chuỗi xử lý của ASP.NET Core. Toàn bộ code bạn viết chính là để mở rộng MVC Middleware theo nhu cầu xử lý bài toán của bạn, như xây dựng các lớp controller riêng hoặc các Razor view.

> Razor Pages mà bạn đã học được xây dựng bên trên MVC Middleware.

Tất cả các thành phần cần thiết cho một ứng dụng MVC như routing, base controller class, model binding, Razor view engine, v.v., đều được thực thi sẵn. Bạn có thể dễ dàng sử dụng, mở rộng, hoặc thậm chí thay thế các thành phần này.

Để tiện lợi cho người lập trình, MVC Middleware thực hiện *cấu hình dựa trên quy ước* (configuaration over convention), như tên gọi của controller, vị trí lưu file của các Razor

view, v.v..

Do vậy, trong quá trình học làm việc với ASP.NET Core MVC bạn cũng phải học rất nhiều quy ước khác nhau. Tuy nhiên, giống như các thành phần khác của ASP.NET Core, bạn có thể thay đổi các quy ước theo nhu cầu.

Bonus: Các loại model trong ASP.NET Core MVC

Đây là một nội dung đặc biệt mà có lẽ ít tài liệu giải thích cặn kẽ cho bạn.

Trong mô tả của kiến trúc MVC khái niệm model tương đối chung chung: thành phần chứa tất cả các loại dữ liệu và hoạt động không liên quan đến giao diện.

Cách mô tả model như vậy gây ra khó hiểu. Ít nhất chúng ta thực sự khó hình dung ra đâu là model trong một ứng dụng xây dựng theo kiến trúc MVC.

Lấy ví dụ trong ứng dụng quản lý sách điện tử, liệu model là một object Book, hay là tập hợp các object Book, hay là một khái niệm trừu tượng?

Trong ASP.NET Core, "model" được sử dụng để chỉ MỘT SỐ thành phần khác nhau, bao gồm: binding model, application model, domain model, view model.

Các thành phần này có vị trí và vai trò rất khác biệt nhau trong ứng dụng ASP.NET Core MVC.

Binding model là kết quả của việc trích xuất dữ liệu từ truy vấn HTTP (chứa trong route data, query string, form) và biến đổi thành object của .NET. Binding model có vai trò là tham số đầu vào cho action. Quá trình trích xuất và biến đổi dữ liệu này trong ASP.NET Core được gọi là model binding.

Domain model là kết quả của phân tích dữ liệu theo mô hình DDD. Dù bạn xây dựng ứng dụng theo kiến trúc nào đi chăng nữa thì cũng vẫn cần có domain model. Domain model không phải là đặc trưng của kiến trúc MVC.

Tất cả những gì mà action có thể sử dụng để thực hiện nhiệm vụ của mình đều được xem là thuộc **application model**. Đây là khái niệm rất chung và cũng không đặc trưng của MVC. Trong các loại ứng dụng khác bạn cũng có thể gọi chung các thành phần dịch vụ, cơ sở dữ liệu, domain model là application model.

View model là dữ liệu được action tạo ra và truyền sang cho view. View model có lẽ là thành phần gần gũi nhất với định nghĩa M-model trong MVC.

View model không "thuần nhất". Nghĩa là nó có thể là các loại model khác hoặc là tập hợp của nhiều loại dữ liệu khác nhau.

Ví dụ, view model có thể là: một danh sách các object của domain model; một object cụ thể của domain model; một bộ phận của một object domain model; kết hợp giữa domain model

và thông tin phụ trợ như danh sách phân trang.

View model thường tạo ra từ các hoạt động liên quan đến cơ sở dữ liệu (thuộc về application model).

Kết luận

Trong bài học này chúng ta đã giới thiệu sơ lược về kiến trúc MVC cũng như đặc điểm của MVC trong ASP.NET Core. Qua đây có thể thấy, MVC trong một ứng dụng web thực tế không quá phức tạp. Hoặc cũng có thể nói theo cách khác, các web framework vận dụng kiến trúc MVC như ASP.NET Core đã giúp chúng ta rất nhiều trong việc tạo ra các ứng dụng theo kiến trúc này.

> + Nếu bạn thấy site hữu ích, trước khi rời đi hãy **giúp đỡ** site bằng một hành động nhỏ để site có thể phát triển và phục vụ bạn tốt hơn.
> + Nếu bạn thấy bài viết hữu ích, hãy giúp **chia sẻ** tới mọi người.
> + Nếu có thắc mắc hoặc cần trao đổi thêm, mời bạn viết trong phần **thảo luận** cuối trang.
> Cảm ơn bạn!