Cấu trúc ASP.NET Core

Nếu bạn mới bắt đầu với Asp.net Core và tìm kiếm thông tin trên Internet, bạn có thể sẽ cảm thấy rất hoang mang vì có quá nhiều nội dung khác nhau. Vấn đề là, Asp.net Core là một hệ thống phức tạp dành cho phát triển nhiều loại ứng dụng khác nhau hoạt động trên web. Do vậy, Asp.net Core bao gồm nhiều thành phần với vai trò khác nhau nhưng tương tác trong một khuôn khổ chung. Việc tìm hiểu cấu trúc Asp.net Core sẽ cho bạn cái nhìn tổng thể giúp định hình những gì sẽ học về sau, cũng như mối quan hệ giữa chúng.

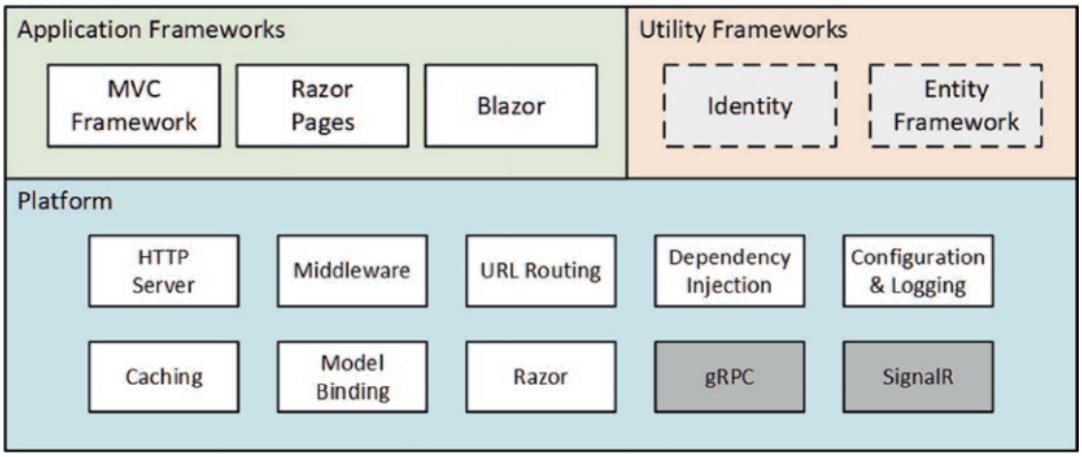
NỘI DUNG CỦA BÀI [Ẩn]

Cấu trúc Asp.net Core tổng quan

Khi bắt đầu tìm hiểu về Asp.net Core trên Internet, đa số các bạn sẽ cảm thấy ngợp. Cùng nói về Asp.net Core nhưng mỗi site có thể nói về những vấn đề hoàn toàn khác biệt.

Ví dụ, có site tập trung vào Asp.net Core MVC, có site lại tập trung vào Razor Page, hoặc Web API. Gần đây nhất bạn có thể gặp thêm Blazor. Hoặc khi đọc về MVC và Razor Pages bạn lại cùng gặp Razor.

Lý do là trong khuôn khổ của Asp.net Core có nhiều thành phần khác nhau. Hình dưới đây minh họa các thành phần này.



Cấu trúc Asp.net Core

Như bạn thấy, cấu trúc Asp.net Core bao gồm 3 phần: Application Frameworks, Utility Frameworks, và Platform.

Application Frameworks chứa những cái tên có lẽ tương đối quen thuộc như MVC Framework, Razor Pages hay Blazor. Đây là những framework giúp bạn xây dựng các dạng khác nhau của ứng dụng web.

Khối **Utility Frameworks** chứa những thứ cảm tưởng như không liên quan đến Asp.net Core: Identity và Entity Framework. Khối này chứa những framework hỗ trợ cho ứng dụng, bao gồm bảo mật và cơ sở dữ liệu.

Khối **Platform** là những gì tạo nên nền tảng chung nhất mà mọi loại ứng dụng Asp.net Core đều cần sử dụng đến.

Tiếp sau đây chúng ta sẽ trình bày chi tiết hơn về từng khối.

Application Frameworks

Đây là phần nổi của Asp.net Core khiến nhiều bạn cảm thấy lẫn lộn.

Application Framework chứa các framework: MVC, Razor Pages, và Blazor. Các framework này giúp (1) giải quyết các loại vấn đề khác nhau, và (2) giải quyết cùng một vấn đề theo những cách thức khác nhau.

Nhìn nhận theo một cách khác, đây là những mẫu kiến trúc khác nhau cho ứng dụng web mà Microsoft hỗ trợ. Bạn có thể lựa chọn bất kỳ framework nào phù hợp với yêu cầu của dự án và sở thích cá nhân. Do vậy, chúng không loại trừ nhau mà còn tương trợ lẫn nhau. Trong quá trình phát triển ứng dụng, bạn có thể chuyển đổi từ framework này sang framework khác, hoặc kết hợp chúng.

MVC là framework xây dựng theo nguyên lý phân chia nhiệm vụ (Separation of Concerns, SoC). MVC phân chia chức năng thành các phần của ứng dụng ra làm các nhóm độc lập, gọi là **M**odel, **V**iew và **C**ontroller.

> Nếu bạn từng làm việc với Asp.net (.NET Framework), bạn chắc chắn đã quen thuộc với framework có tên tương tự. MVC framework trong Asp.net Core thừa kế rất nhiều đặc điểm của Asp.net MVC giúp bạn dễ dàng chuyển đổi sang.

Trong Asp.net Core, cùng với sự phổ biến của ứng dụng đơn trang (Single-Page Application, SPA), MVC không còn đóng vai trò một framework chính nữa.

Razor Pages là một framework đơn giản hướng tới xây dựng các trang web tự thân độc lập. Mô hình của Razor Pages đơn giản hơn so với MVC.

> Razor Pages có thể xem là framework tương tự với Asp.net Web Pages.

Microsoft khuyến nghị quá trình học Asp.net Core cho người mới nên bắt đầu từ Razor Pages.

Blazor là framework dành cho phát triển ứng dụng đơn trang (SPA) nhưng sử dụng ngôn ngữ C# thay cho JavaScript chạy trên trình duyệt. Đây là framework mới nhất và đang

nhận được nhiều sự quan tâm.

Blazor bao gồm hai mô hình chính: Blazor Server và Blazor WebAssembly. Ngoài ra còn có các mô hình triển khai Blazor khác đang được phát triển.

Tập tài liệu này sẽ cố gắng hướng dẫn sử dụng cả ba framework này.

Utility Frameworks

Utility Frameworks chứa hai framework (không bắt buộc) nhưng lại được sử dụng gần như trong mọi ứng dụng Asp.net Core. Vì vậy, quá trình học Asp.net Core thực tế luôn gắn với hai framework này ở giai đoạn sau.

Entity Framework Core là một ORM (Object-Relational Mapping) giúp ứng dụng tương tác với cơ sở dữ liệu. Entity Framework Core giúp ánh xạ (hai chiều) giữa các bảng cơ sở dữ liệu (ví dụ, SQL Server) với các domain class của ứng dụng.

Entity Framework Core không gắn với Asp.net Core mà có thể sử dụng trong bất kỳ ứng dụng .NET nào.

Entity Framework Core có thể xem là phiên bản hiện đại đa nền tảng của Entity Framework – ORM chính của .NET Framework cổ điển. Nếu đã biết Entity Framework có thể dễ dàng chuyển đổi sang Entity Framework Core.

Asp.net Core Identity là framework dành cho xác thực (authentication) và xác minh quyền (authorization) người dùng trong ứng dụng.

Các ứng dụng đơn giản minh họa thường không sử dụng Identity nhưng đây là lại framework quan trọng hàng đầu với các ứng dụng thực tế.

Đây là hai framework phức tạp. Các tài liệu dạy Asp.net Core thường chỉ hướng dẫn sơ lược cách sử dụng các framework này. Thực tế, để giới thiệu đầy đủ mỗi framework cần nguyên một cuốn sách riêng.

Platform

Phần platform chứa nhiều thành phần cấp thấp cần thiết cho việc nhận và xử lý truy vấn HTTP, cũng như tạo ra các phản hồi phù hợp.

Mặc dù phần lớn thời gian trong quá trình phát triển ứng dụng bạn làm việc chủ yếu với applicaton framework, và bản thân các application framework cũng che đi rất nhiều phần của platform, bạn cần hiểu rõ các tính năng quan trọng của platform để có thể sử dụng hiệu quả application framework.

Các thành phần của platform khá đa dạng. Chúng ta sẽ điểm qua các thành phần này.

HTTP Server, còn gọi là built-in server với tên gọi Kestrel, có nhiệm vụ tiếp nhận truy vấn HTTP. Kestrel có thể hoạt động độc lập (tích hợp trong một ứng dụng khác) hoặc phối hợp với một web server thông thường (Apache, NGinx, IIS).

Middleware là nhóm thành phần có nhiệm vụ xử lý truy vấn HTTP. Các middleware được xếp theo chuỗi. Khi truy vấn chạy qua mỗi khâu của chuỗi middle sẽ được xem xét xử lý.

URL Routing là cơ chế ánh xạ chuỗi truy vấn HTTP sang thực thi một phương thức nào đó. Do vậy, mỗi URL (địa chỉ bạn gửi về server) sẽ tương ứng với thực thi một phương thức trên server.

Razor là cơ chế sinh ra HTML từ dữ liệu và logic của chương trình. Razor được gọi là view engine trong Asp.net Core. Razor sử dụng loại cú pháp đặc biệt kết hợp giữa C# và HTML.

Model Binding là cơ chế ánh xạ dữ liệu chứa trong truy vấn HTTP sang tham số của phương thức cần thực thi. Nhờ cơ chế Model Binding, việc xây dựng các phương thức trong Asp.net Core đơn giản như bất kỳ phương thức c# thông thường nào.

Dependency Injection là cơ chế cho phép tự động sinh và chèn object vào một object khác. Asp.net Core xây dựng sẵn cơ chế này cho bạn mà không cần đến một thư viện thứ ba (như Ninject, Unity).

Configuration & Logging là cơ chế hỗ trợ cấu hình và lưu vết quá trình thực thi ứng dụng.

Caching là cơ chế lưu tạm để tăng hiệu suất cho ứng dụng.

Nhìn chung, trong quá trình học, bạn sẽ lần lượt làm quen với tất cả các thành phần trên. Một số trong đó sẽ được xem xét chi tiết hơn.

SignalR và gRPC

Trong phần Platform có hai khối thể hiện hơi khác so với các thành phần còn lại là gRPC và SignalR.

SignalR là một framework riêng nhưng lại được sử dụng làm nền tảng cho ứng dụng Asp.net Core. SignalR cho phép tạo ra kênh truyền thông tốc độ cao (theo thời gian thực) và hai chiều giữa browser và web server.

SignalR là nền tảng cho Blazor Server – loại ứng dụng đơn trang mà toàn bộ phần xử lý được đẩy về server và nhận lại kết quả (DOM) theo thời gian thực.

Bạn cũng có thể tự dùng SignalR để tạo ra các loại ứng dụng web với tương tác hai chiều client – server theo thời gian thực.

gRPC là một chuẩn dành cho gọi hàm từ xa (Remote Procedure Call) đa nền tảng qua HTTP do Google (chữ g trong gRPC) phát triển. gRPC có thể hữu ích để các thành phần backend của ứng dụng (chạy trên server) trao đổi dữ liệu.

gRPC hiện nay chưa thực sự phổ biến và cũng không thế sử dụng trực tiếp trên trình duyệt.

Nhìn chung hai thành phần này bạn sẽ hiếm khi phải sử dụng đến, trừ trong những tình huống đặc biệt. Chúng được đưa vào đây chủ yếu vì đóng vai trò nền tảng cho một số

trường hợp đặc biệt.

> + 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!

Kết luận

Trong bài học này chúng ta đã trình bày sơ lược về cấu trúc của Asp.net Core.

Bạn chưa cần tìm hiểu cặn kẽ về từng thành phần một vì đây là nội dung của cả tập hướng dẫn này.

Tuy nhiên, bạn nên nắm được ý tưởng chung của chúng để định hướng được những gì cần học.