Cấu trúc dự án, cấu hình ứng dụng, middleware trong ASP.NET Core

Cấu trúc dự án là cách thức bố trí và ý nghĩa của từng thành phần của dự án (project) ASP.NET Core. Cấu trúc dự án có liên quan mật thiết đến cách thức cấu hình cho ứng dụng. So với ASP.NET, cấu trúc và cấu hình dự án ASP.NET Core có nhiều điểm khác biệt. Bạn phải nắm rõ các vấn đề này trước khi đi vào viết code.

NỘI DUNG CỦA BÀI [Ẩn]

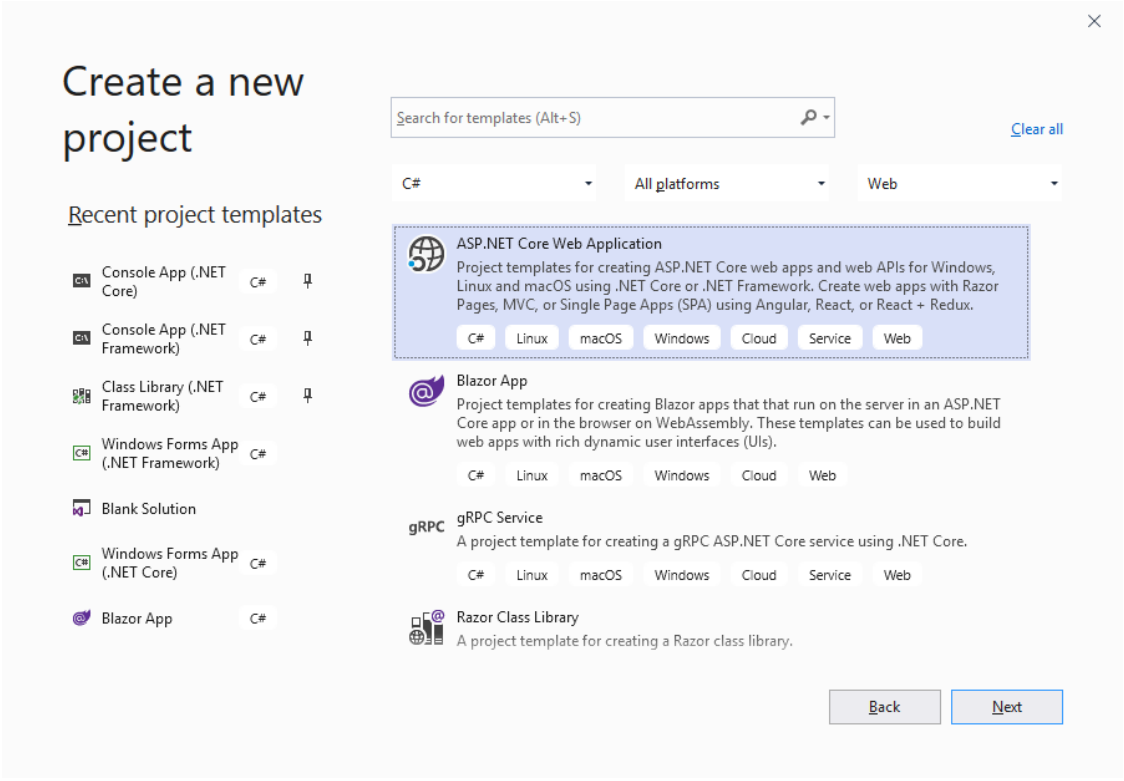
Thực hành 1: Tạo dự án ASP.NET Core trống

Trong bài học đầu tiên bạn đã biết cách tạo dự án (project) ASP.NET Core sử dụng template MVC. Bạn để ý thấy rằng trong project này xuất hiện khá nhiều file và folder. Loại template này phù hợp để bạn bắt đầu làm việc.

Tuy nhiên, khi bạn mới học lập trình ASP.NET Core, template MVC có thể sẽ làm bạn bị rối. Vì vậy, chúng ta sẽ tạo một dự án trống (empty project) với các file/thư mục cơ bản nhất cho mục đích học tập.

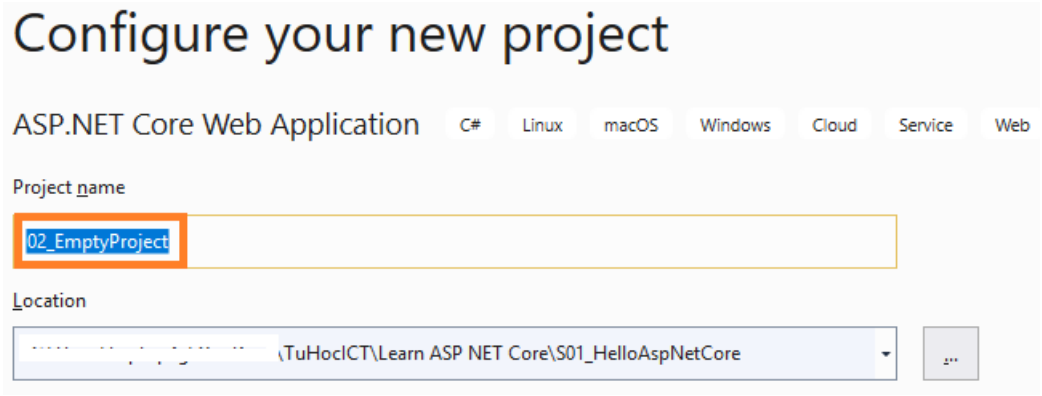
Hãy thực hiện theo các bước sau đây để tạo ra một project ASP.NET Core trống.

Bước 1.



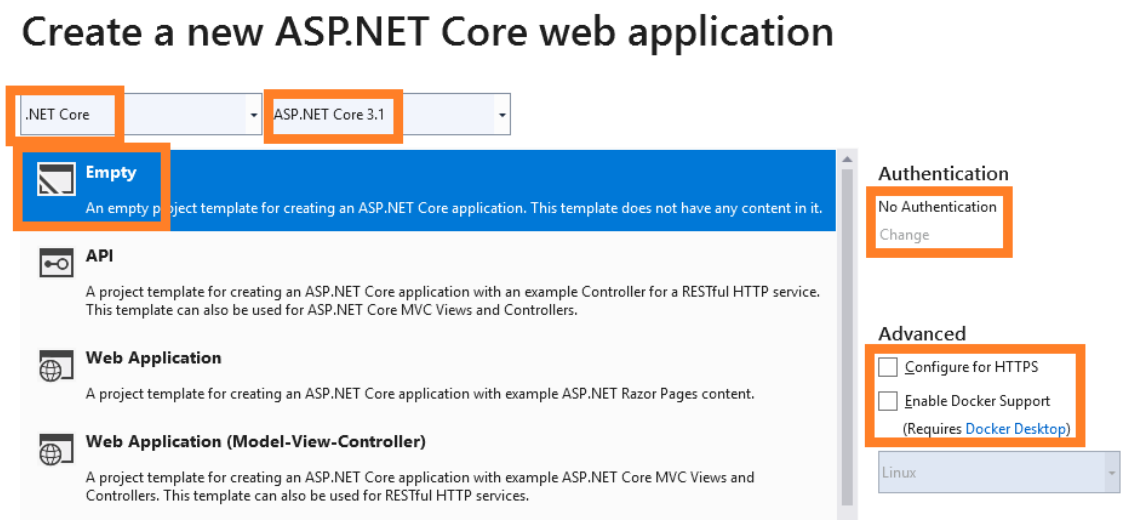
Chọn project template cho ASP.NET Core Web Application

Bước 2.



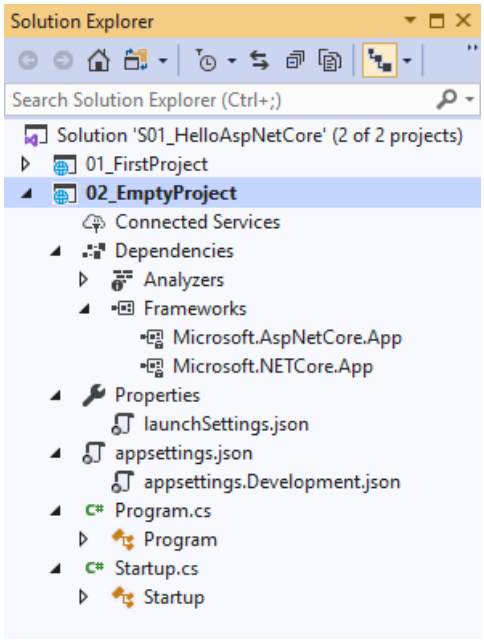
Đặt tên project và chọn thư mục lưu project

Bước 3.



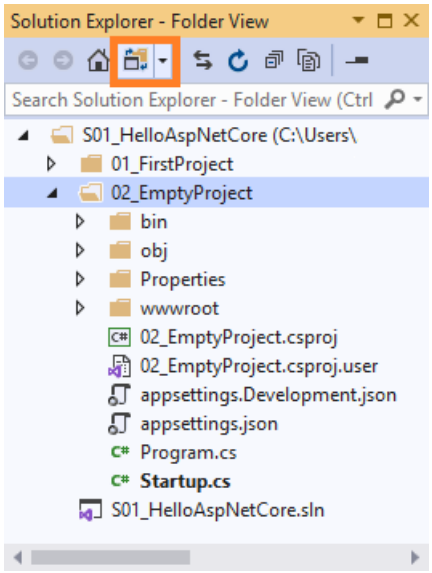
Chọn platform, version và template

Bạn sẽ thu được một project có cấu trúc như dưới đây:



Cấu trúc một empty project ASP.NET Core

Nếu bấm nút *Switch Views*, Solution Explorer sẽ chuyển sang chế độ hiển thị file/folder:



Chế độ hiển thị file/folder

> Khác với ASP.NET, trong ASP.NET Core cấu trúc dự án tự động đồng bộ với cấu trúc file/folder. Có nghĩa là, nếu bạn copy/tạo mới file/thư mục, nó sẽ tự động xuất hiện trong cấu trúc project và ngược lại.

Cấu trúc dự án ASP.NET Core

Như bạn đã thấy trong phần thực hành trên, một project ASP.NET Core đơn giản nhất chứa các thành phần:

Connected Services: danh sách các dịch vụ online mà project này sử dụng, ví dụ WCF Web Service, các dịch vụ trên đám mây Azure, v.v..

Dependencies: danh sách các gói thư viện NuGet được cài đặt và sử dụng trong project. Dù là một project trống đơn giản nhất cũng phải sử dụng một số gói thư viện nhất định.

Properties: chứa thông tin cấu hình của project. Nếu mở bằng Visual Studio, bạn sẽ gặp giao diện đồ họa quen thuộc vốn có trong tất cả các project C#. Trong ASP.NET Core, các thông tin này thực chất được lưu trong một file json mà bạn có thể trực tiếp điều chỉnh. Cách thức lưu mới này rất hữu ích nếu bạn sử dụng một trình viết code khác (như Visual Studio Code).

appsettings.json: chứa các thông tin cấu hình cho hoạt động của ứng dụng, như connection string, các biến môi trường, tham số dòng lệnh, v.v.. Bạn sẽ lần lượt gặp và làm việc với file này trong quá trình học.

Program.cs: File này chứa class Program chịu trách nhiệm cấu hình nền tảng (infrastructure) của ứng dụng. Class này cũng chứa entry point của ứng dụng. Nhìn chung, các cấu hình bên trong Program hầu như không thay đổi trong mỗi project. Nếu không có yêu cầu gì đặc biệt, bạn sẽ không cần điều chỉnh gì trong class Program và file Program.cs.

> Nếu bạn quen thuộc với ứng dụng Console/WPF/Winforms, bạn hẳn biết khái niệm *entry point* của ứng dụng C# – phương thức `void Main` – nơi bắt đầu mọi hoạt động của ứng dụng. Ứng dụng ASP.NET Core mặc định được host bên trên một ứng dụng Console, do đó nó cũng có void Main. Bạn thậm chí có thể điều chỉnh để host ứng dụng ASP.NET Core bên trên một ứng dụng winforms hoặc WPF nếu muốn.

Startup.cs: File này chứa class với các phương thức cấu hình cho hoạt động của ứng dụng, ví dụ bạn sẽ sử dụng những middleware nào, thứ tự sắp xếp các middleware trong pipeline ra sao. Bạn cũng có thể cấu hình sử dụng các loại dịch vụ (service) nào, như Dependency Injection, Logging. Bạn sẽ gặp một số cấu hình đơn giản trong phần tiếp theo của bài học này. Chi tiết cách tạo class cấu hình bạn sẽ gặp lại trong một bài học khác.

wwwroot: Thư mục chứa các file tĩnh như html, css, javascript, hình ảnh. Các file tĩnh phải nằm trong thư mục này thì mới có thể cung cấp cho trình duyệt. Trong ASP.NET Core, thư mục này có tên gọi là riêng là **web root**.

Để dễ dàng quản lý các file, web root thường được chia thành các thư mục con: ***css*** (cho các file css), ***js*** (cho các file javascript), ***lib*** (cho các thư viện javascript như jQuery). Ngoài ra, file biểu tượng của site (favicon.ico) cũng được đặt trực tiếp trong web root.

Trong phần thực hành 2 dưới đây bạn sẽ học cách cấu hình sử dụng file tĩnh trong ASP.NET Core.

> Nếu bạn tạo project với ASP.NET Core 2.x, thư mục wwwroot được tạo tự động. Từ ASP.NET Core 3, trong template trống sẽ không tạo ra thư mục wwwroot. Bạn có thể tự thêm thư mục này vào project khi cần thiết. Để thêm thư mục, bạn click phải chuột vào tên project => Add => New Folder.

Startup class

Trong phần này chúng ta sẽ trình bày sơ lược về Startup class. Như đã nói, Startup là class dùng cho cấu hình hoạt động của ứng dụng. Cụ thể hơn, class này chịu trách nhiệm (1) đăng ký các *dịch vụ* (service) được sử dụng trong ứng dụng và (2) đăng ký và ghép nối các *middleware* dùng để xử lý truy vấn HTTP.

Tên class không quan trọng. Mặc định class này được đặt tên là `Startup`. Bạn có thể đổi tên nó thành bất kỳ tên gì khác (miễn là đúng quy tắc của C#). Tuy nhiên, dù đặt tên class là gì, bạn bắt buộc phải có hai phương thức sau:

```
1.  public void ConfigureServices(IServiceCollection services) {...}
2.  public void Configure(IApplicationBuilder app, IWebHostEnvironment env) {...}
```

Trong đó, `ConfigureServices` là phương thức dùng để đăng ký các dịch vụ, `Configure` là phương thức dùng để đăng ký các middleware.

> Sở dĩ có yêu cầu trên là vì ASP.NET Core sử dụng cơ chế reflection để tìm hai phương thức có tên tương ứng để thực thi.

Trong ASP.NET Core, **dịch vụ** (service) dùng để chỉ một class có khả năng cung cấp chức năng cho ứng dụng. Ví dụ, trong một ứng dụng về thương mại điện tử bạn có thể sử dụng một class riêng chuyên môn để tính thuế, tạm gọi là TaxCalculator. Các class dịch vụ như vậy thường do bạn tự xây dựng.

Việc đăng ký class dịch vụ với ứng dụng đảm bảo rằng một object của class dịch vụ sẽ được ***tự động*** tạo ra khi cần thiết mà bạn không cần tự mình khởi tạo.

Thông thường object sẽ được tạo ra và truyền cho nơi cần dùng thông qua cơ chế Dependency Injection tích hợp sẵn trong ASP.NET Core. Bạn không cần sử dụng các DI container thứ ba nữa (như Ninject, Unity).

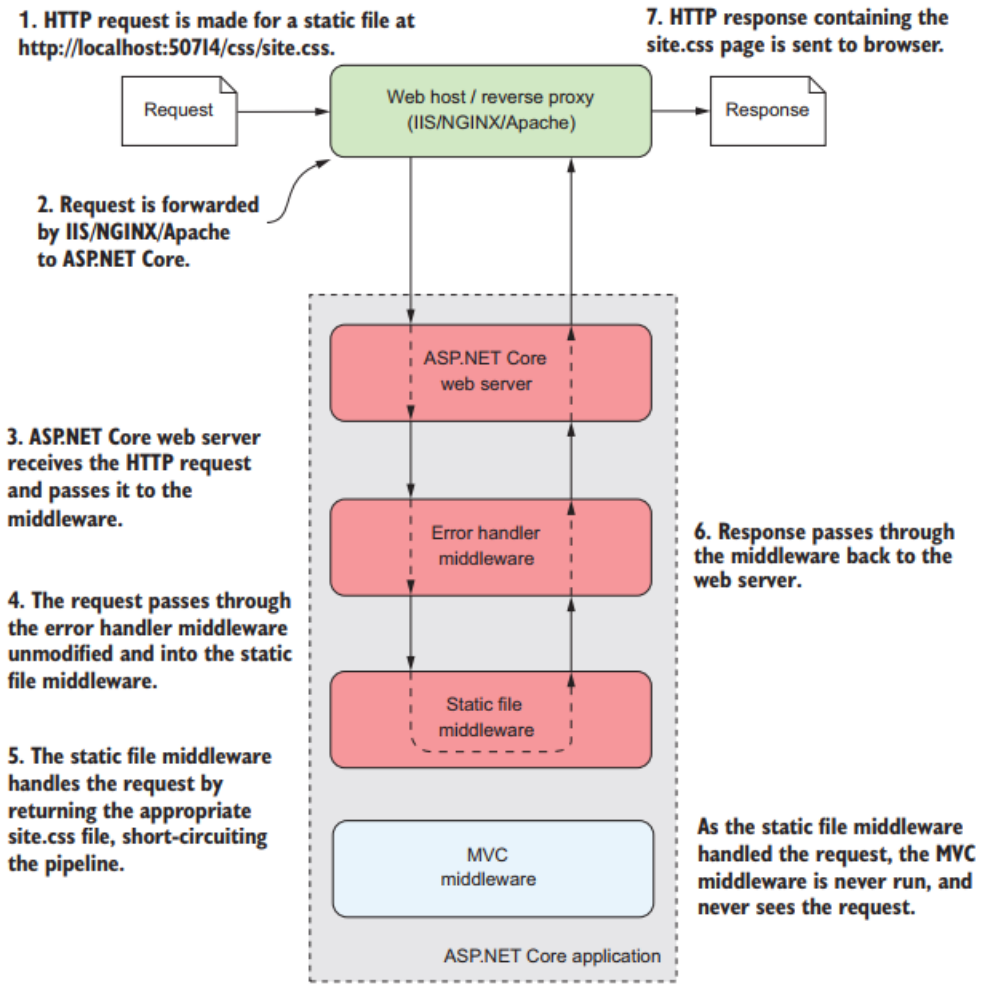
Ngoài ra, cơ chế dịch vụ này khuyến khích ứng dụng của bạn tuân thủ nguyên lý Single Responsibility (SRP) – nguyên lý đầu tiên trong bộ nguyên lý SOLID.

> Bạn sẽ học chi tiết hơn về các đăng ký dịch vụ và Dependency Injection trong một bài học khác.

Khái niệm middleware trong Asp.net Core

Middleware là một khái niệm mới trong ASP.NET Core dùng để chỉ một module/class có khả năng xử lý truy vấn HTTP và trả lại kết quả ở dạng HTTP Response.

Nếu nhớ lại cách thức hoạt động của ứng dụng ASP.NET trong bài học trước, bạn hẳn chú ý thấy rằng, sau khi Kestrel nhận truy vấn từ reverse proxy (IIS, Apache, Nginx), truy vấn này sẽ di chuyển dọc theo chuỗi (pipeline) class middleware cho đến khi một nó gặp một middleware phù hợp để xử lý. Khi middleware xử lý xong, kết quả sẽ trả ngược lại đường đi của truy vấn.



Ví dụ chuỗi middleware trong Asp.net Core

> Bạn cũng có thể tưởng tượng quy trình xử lý với middleware giống như một phân xưởng có hai băng chuyền ngược chiều. Truy vấn HTTP tới từ trình duyệt sẽ chạy trên một băng chuyền (pipeline). Mỗi middleware là một công nhân đứng dọc băng chuyền đó và chỉ chịu trách nhiệm xử lý một phần xác định. Nếu hoàn tất quá trình xử lý, sản phẩm sẽ chuyển sang băng chuyền thứ hai trả ngược lại cho trình duyệt.

Do một truy vấn có thể chứa những yêu cầu rất khác biệt, đồng thời mỗi ứng dụng có những mục đích khác nhau, tổ hợp middleware cho mỗi ứng dụng không hoàn toàn giống nhau. "Tổ hợp middleware" thể hiện qua loại middlewere được sử dụng và thứ tự bố trí chúng trên pipeline.

Lấy ví dụ, nếu ứng dụng ASP.NET Core chỉ đơn giản là cung cấp các file HTML đã thiết kế sẵn, nó không cần những class xử lý phức tạp. Thay vào đó, nó chỉ cần khả năng tìm và trả lại nguyên vẹn các file HTML trên ổ đĩa. Loại middleware này trong ASP.NET Core được gọi là StaticFiles. Bạn sẽ sử dụng middleware này trong phần thực hành tiếp theo.

Cơ chế đăng ký và ghép nối middleware này giúp ứng dụng ASP.NET Core chỉ cần sử dụng những gì mình cần, không dùng những chức năng dư thừa, qua đó làm ứng dụng nhẹ và nhanh hơn.

Trong các phần thực hành sau đây bạn sẽ làm quen với một số middleware cơ bản thường sử dụng nhất.

Thực hành 2: Cấu hình sử dụng static file

File tĩnh (static file) là những file đã được tạo sẵn và được cung cấp nguyên vẹn cho trình duyệt khi được yêu cầu. Các loại file sau đây được coi là static file: html (file dữ liệu chính của các trang web tĩnh), css (cascading style sheet – chịu trách nhiệm định dạng trang), js (file mã nguồn Javascript), các loại file ảnh. Các file tĩnh kết hợp lại với nhau tạo ra các trang web tĩnh (static page) – dạng thức cơ bản nhất của web.

Phục vụ các file static là nhiệm vụ cơ bản nhất của các web server. Tuy nhiên trong ASP.NET Core tính năng này yêu cầu bạn phải thực hiện một số thao tác cấu hình. Nói cách khác, mặc định ASP.NET Core không bật tính năng này.

> Nếu bạn đã từng làm việc với các framework/công nghệ phát triển ứng dụng khác, bạn hẳn đã để ý, khi tạo một project trống trong ASP.NET Core, bạn không hề nhìn thấy những gì quen thuộc như default.html, index.html.

Trong phần thực hành này chúng ta sẽ thực hiện một cấu hình đơn giản giúp ứng dụng ASP.NET Core có thể phục vụ các file tĩnh cho trình duyệt.

> Bạn tiếp tục sử dụng project vừa tạo ở phần ***thực hành 1***.

Bước 1. Tạo thư mục wwwroot

Mặc định ASP.NET Core yêu cầu tất cả các file tĩnh phải nằm trong thư mục wwwroot trực thuộc thư mục dự án. Vì vậy, nếu trong project bạn chưa có thư mục này, hãy tạo mới bằng cách click phải vào tên project => chọn Add => New Folder => đặt tên cho thư mục vừa tạo là wwwroot.

Bước 2. Tạo file default.html

Trong thư mục wwwroot tạo file default.html bằng cách: click phải vào thư mục wwwroot => Add => New Item => chọn template HTML Page => đặt tên file là default.html => OK.

Mở file vừa tạo và điều chỉnh nội dung như sau:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Hello ASP.NET Core</title>
</head>
<body>
    <h1>This file is served from ASP.NET  CORE!</h1>
</body>
</html>
```

Bước 3. Điều chỉnh class Startup

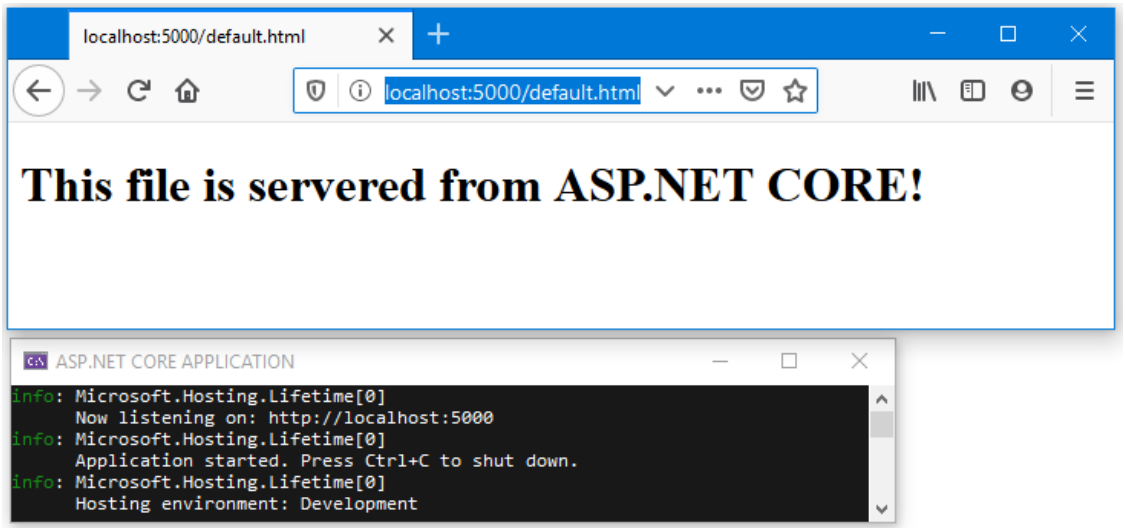
Mở file Startup.cs và điều chỉnh phương thức Configure như sau (bổ sung dòng `app.UseStaticFiles();`):

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseStaticFiles();

    app.UseRouting();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapGet("/", async context =>
        {
            await context.Response.WriteAsync("Hello World!");
        });
    });
}
```

Dòng lệnh `app.UseStaticFiles();` yêu cầu gắn thêm middleware xử lý static file vào đầu pipeline của ứng dụng. Nghĩa là nếu trình duyệt yêu cầu một file tĩnh, ứng dụng ASP.NET Core sẽ tìm kiếm file tương ứng trong wwwroot. Nếu tìm thấy sẽ trả lại ngay file tương ứng và kết thúc pipeline xử lý.

Bước 4. Chạy thử ứng dụng

Giả sử bạn chạy ứng dụng ở dạng Console. Kestrel khi này sẽ lắng nghe ở cổng 5000 (mặc định). Bạn gõ URL yêu cầu lấy file default.html vào thanh địa chỉ của trình duyệt. Kết quả như dưới đây.



Thực hành 3. JavaScript và CSS

Trong phần thực hành này chúng ta sẽ học cách sử dụng file JavaScript và CSS cùng với file HTML tĩnh.

Bạn thực hiện ***nối tiếp Thực hành 2***.

Bước 1. JavaScript

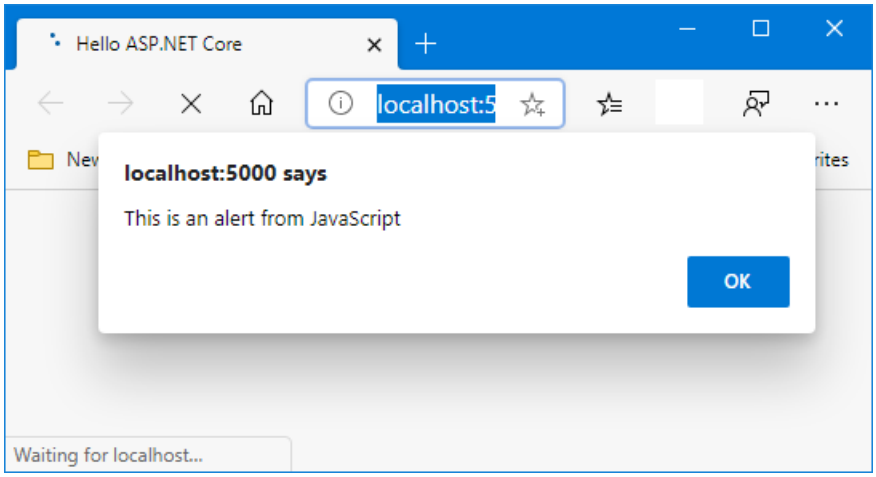
Thêm file script.js vào thư mục wwwroot và viết code như sau:

```
alert("This is an alert from JavaScript");
```

Điều chỉnh file default.html như sau:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Hello ASP.NET Core</title>
    <script src="script.js"></script>
</head>
<body>
    <h1>This file is served from ASP.NET  CORE!</h1>
</body>
</html>
```

Khi chạy thử bạn sẽ thu được kết quả như sau:



Bước 2. CSS

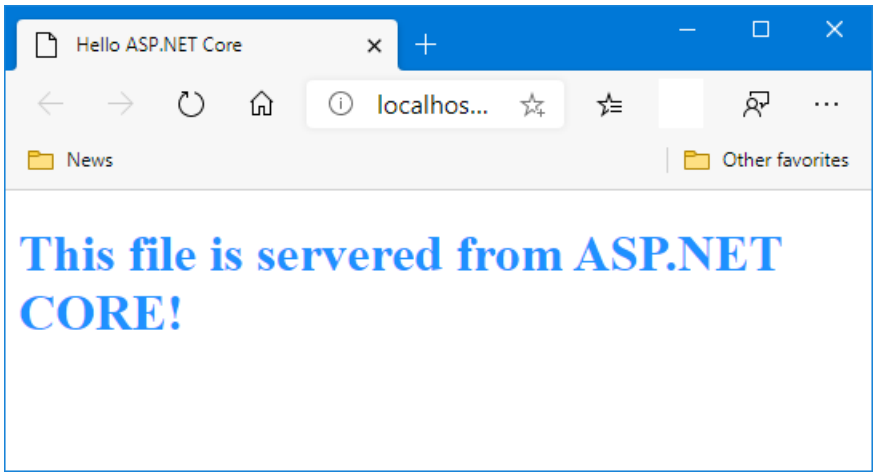
Thêm file styles.css vào thư mục wwwroot và viết code như sau:

```
h1 {color: dodgerblue;}
```

Điều chỉnh file default.html như sau:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Hello ASP.NET Core</title>
    <link rel="stylesheet" type="text/css" href="styles.css" />
    <script src="script.js"></script>
</head>
<body>
    <h1>This file is served from ASP.NET  CORE!</h1>
</body>
</html>
```

Khi chạy bạn sẽ thu được kết quả như sau:



Như vậy, qua hai phần thực hành này bạn đã nắm được cách cung cấp các nội dung tĩnh cho trình duyệt từ ứng dụng ASP.NET Core. Đây là một yêu cầu rất cơ bản mà bạn phải thành thạo.

Thực hành 4: Cấu hình file mặc định

Khi bạn đã nắm được cách thức cấu hình cơ bản cho middleware pipeline của ứng dụng ASP.NET, giờ là lúc chúng ta thực hiện thêm một cấu hình đơn giản nữa.

Các web server bình thường sẽ tự lựa chọn một file để cung cấp cho trình duyệt nếu trong URL không chỉ định rõ file nào cần lấy. Ví dụ, nếu trình duyệt gửi request với URL chỉ chứa "/" (root), thông thường các web server sẽ chọn một trong số các file index.html, index.htm (Apache), default.html, default.htm (IIS) để cung cấp cho trình duyệt. Những file này được gọi là file mặc định.

Trong ASP.NET Core, nếu muốn sử dụng file mặc định bạn phải thực hiện vài thao tác cấu hình đơn giản.

> Bạn thực hiện nối tiếp Thực hành 3.

Bước 1. Bổ sung middleware DefaultFiles

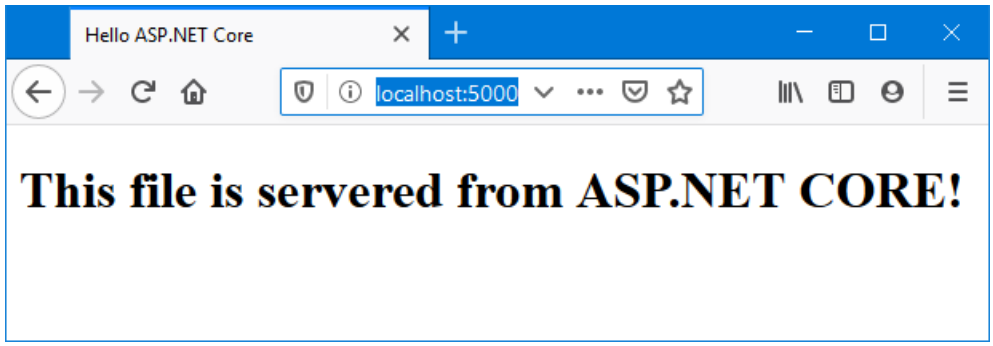
Mở file Startup.cs và điều chỉnh phương thức Configure như sau:

```
22.  public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
23.  {
24.      if (env.IsDevelopment())
25.      {
26.          app.UseDeveloperExceptionPage();
27.      }
28.
29.      var options = new DefaultFilesOptions();
30.      options.DefaultFileNames.Add("home.html");
31.      options.DefaultFileNames.Add("home.htm");
32.      app.UseDefaultFiles(options);
33.
34.      app.UseStaticFiles();
```

```
35.
36.          app.UseRouting();
37.
38.          app.UseEndpoints(endpoints =>
39.          {
40.              endpoints.MapGet("/", async context =>
41.              {
42.                  await context.Response.WriteAsync("Hello World!");
43.              });
44.          });
45.      }
```

Bước 2. Chạy thử chương trình

Chạy chương trình và nhập URL là localhost:5000. Bạn thu được cùng kết quả như trong phần thực hành 2. Sự khác biệt là giờ bạn không cần chỉ định rõ file cần lấy nữa.



Trong điều chỉnh trên bạn yêu cầu sử dụng middleware `DefaultFiles` ở đầu pipeline (trước `StaticFiles`). Middleware này có khả năng phục vụ file tĩnh mặc định nếu URL không chỉ rõ file cần lấy. Tên file mặc định sẽ được middleware này tìm kiếm là *default.html*, *default.htm*, *index.html*, *index.htm*.

Bạn cũng có thể bổ sung thêm tên file mặc định theo ý mình bằng cách thêm tham số (kiểu `DefaultFilesOptions`). Trong ví dụ trên, chúng ta bổ sung thêm hai tên file mặc định là *home.html* và *home.htm*.

Kết luận

Trong bài học này bạn đã học về cấu trúc dự án ASP.NET Core, cách cấu hình và sử dụng một số middleware đơn giản để phục vụ static file cho trình duyệt (StaticFiles) và file mặc định (DefaultFileNames).

Tải mã nguồn solution
S01_HelloAspNetCore
1 file(s) 15Mb

TẢI MÃ NGUỒN SOLUTION

+ 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!