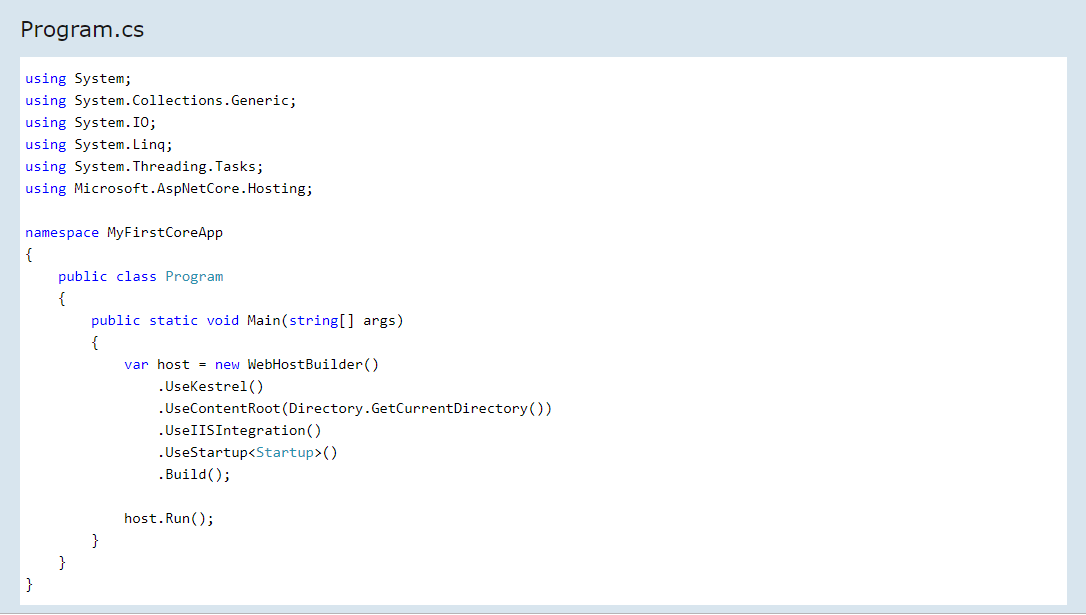
Program.cs

Program.cs là nơi thực hiện host web application trong asp.net core.

Config asp.net core 1.x



Các thành phần trong program.cs

- UseKestrel(): là một phương thức mở rộng chỉ định Kestrel là một server của web app. Kestrel là một máy chủ web đa nền tảng mã nguồn mở cho ASP.NET Core. Hiểu đơn giản nếu như sử dụng IIS để host thì ta chỉ có thể chạy trang web trên 1 nền tảng khi thực hiện host. Nhưng đối với Kestrel thì chỉ một lần host ta có thể sử dụng cho nhiều nền tảng khác nhau.

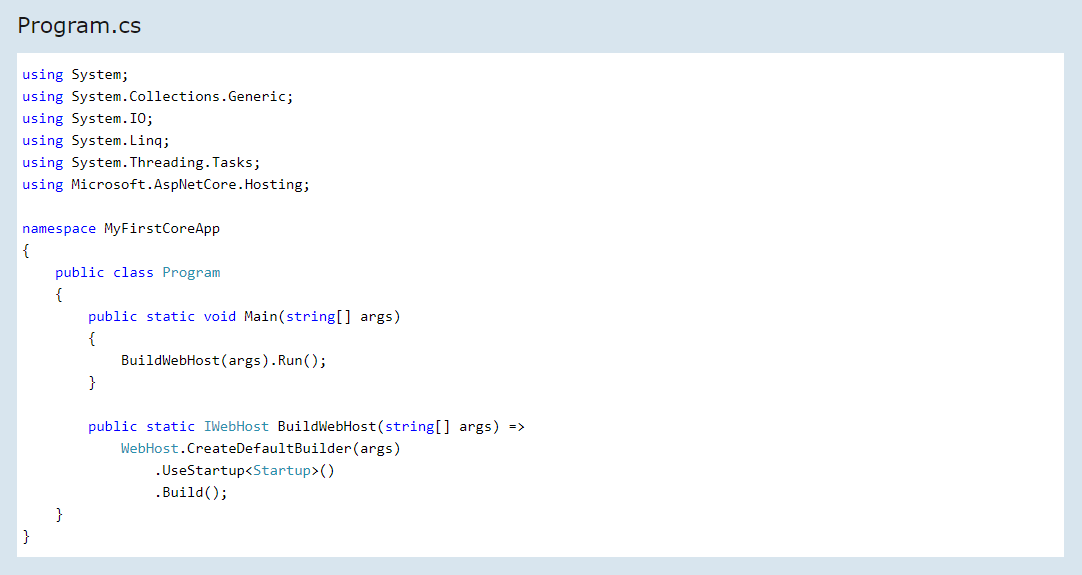
- UseContentRoot(Directory.GetCurrentDirectory()) : UseContentRoot xác định folder suoce của project

- UseIISIntegration() Chỉ định cho Server IIS là server bên ngoài hay là reverse proxy server .

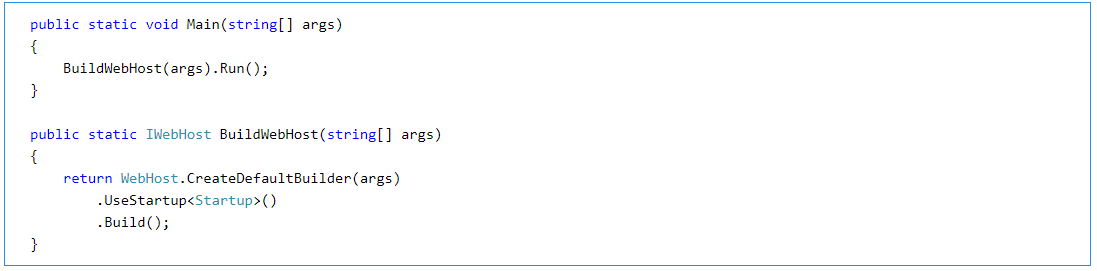
- UseStartup<Startup>() chỉ định class Startup được sử dụng trong web host. Visual studio sẽ tạo lớp Startup.cs là class quy định những khởi động mặc định của web app giống như Global.asax of .NET framework (tìm hiểu kỹ hơn ở phần Start up.cs)

- Build(); return về các config ở trên.

Config Progarm.cs với asp.net core 2.x.

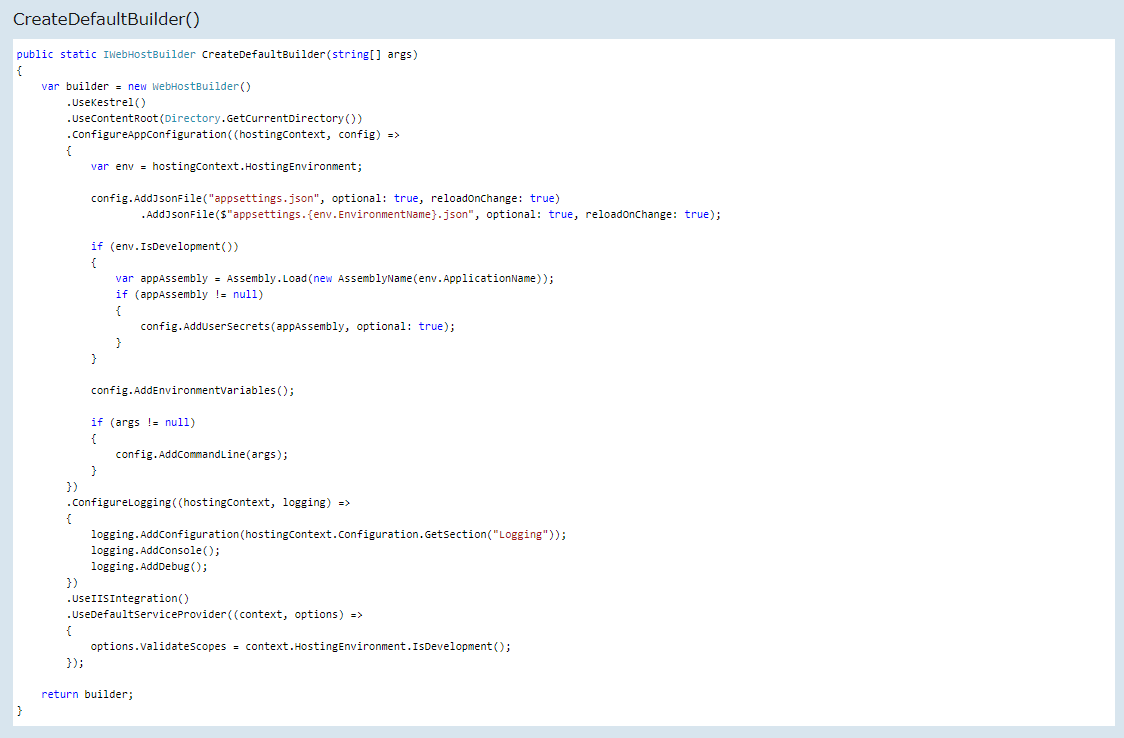


Thực hiện gọi phương thức BuildWebHost () được cấu hình sẵn ta cũng có thể tùy chỉnh trong phương thức. Trong phương thức BuildWebHost sẽ return về webhost.



CreateDefaultBuider() webhost là môt lớp tĩnh , được sử dụng để tạo IWebhost IWebHostBuilder

ta có thể tùy chỉnh CreateDefaultBuilder() để cấu hình server Kestrel.



Như chúng ta thấy phương thức CreateDefaultBuilder được tạo từ việc sử dụng webhostbuilder và settup Kestrel. nội đung bên trong createDefaultBuider

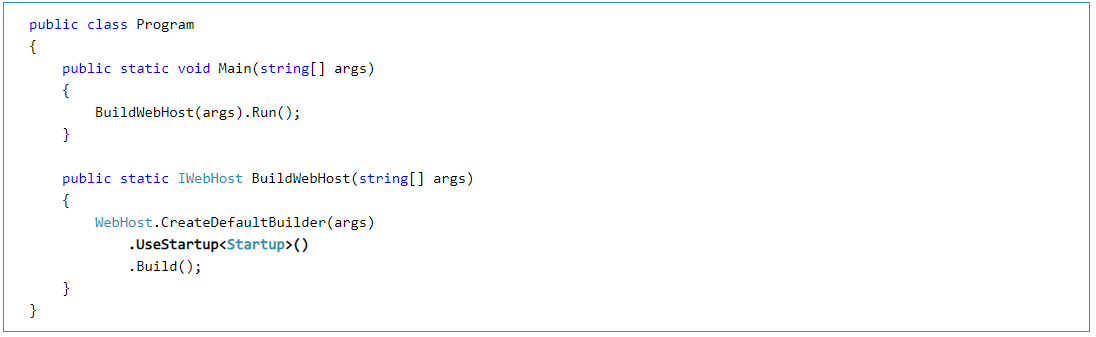
giống như asp.net core1.x trong hàm Main().

Startup Class

Là thành phần được chạy đầu tiên khi chạy ứng dụng.

Class Startup được sử dụng khi sử dung phương thức UseStartup<T>()

Tên class Startup được visual quy định tuy nhiên ta cũng có thể tạo một trang tương tự như vậy.



Visual cung cấp cho ta một class <Startup> được định nghĩa sẵn. Bao gồm 2 phương thứ là được khởi tạo sẵn là: ConfigureServices, Configure



Ta có thể thêm service từ bên ngoài vào bằng “ConfigureServices”. Để thực hiện config cho ứng dụng thì ta sẽ thực hiện trong phương thức Configure

ConfigureServices ()

ConfigureServices là phương thức:

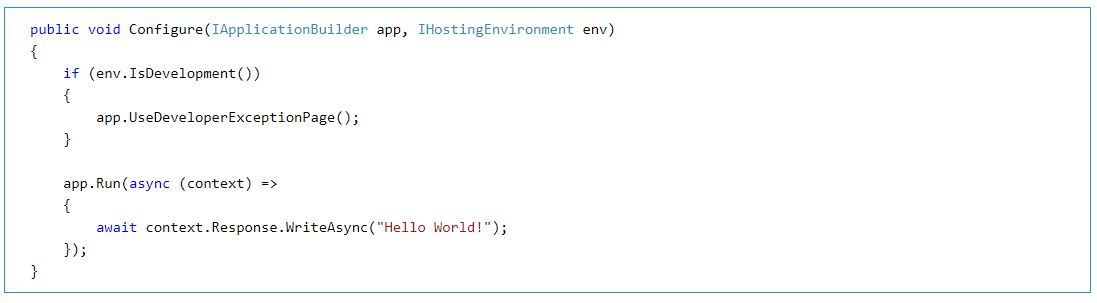
* Không bắt buộc (Optional).
* Được gọi trước phương thức Confiure để thực hiện config services
* Nơi quy định để thực hiện config option.
* Là phương thức đùng để cấu hình dịch vụ được thêm vào trong project.

Để gọi service vào projects ta sử dụng cú pháp sau:

* Add{Service}
* services.Configure{Service} gọi các phương thức config.

Configure ()

Đây là file sẻ giúp chúng cấu hình những config ban đầu của bạn như (router, register service bằng ioc, config authentication.....).



Configure thực hiện chỉ rõ cách thức hoạt động của việc nhận responds từ HTTP request.

Request pipeline sẽ được configured bởi 1 lớp trung gian là thành phần của [IApplicationBuilder](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.builder.iapplicationbuilder)

[IApplicationBuilder](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.builder.iapplicationbuilder) sẽ thược hiện config phương thức mà phương thức đó không đăng ký trong trong service. Hosting sẽ tạo một [IApplicationBuilder](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.builder.iapplicationbuilder) và thực hiện chuyển đến config.

[ASP.NET Core templates](https://docs.microsoft.com/en-us/dotnet/core/tools/dotnet-new)  thực hiện configure pipeline  với hổ trợ :

* Developer exception page
* [Exception handler](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/error-handling?view=aspnetcore-2.2#configure-a-custom-exception-handling-page)
* [HTTP Strict Transport Security (HSTS)](https://docs.microsoft.com/en-us/aspnet/core/security/enforcing-ssl?view=aspnetcore-2.2#http-strict-transport-security-protocol-hsts)
* [HTTPS redirection](https://docs.microsoft.com/en-us/aspnet/core/security/enforcing-ssl?view=aspnetcore-2.2)
* [Static files](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/static-files?view=aspnetcore-2.2)
* [General Data Protection Regulation (GDPR)](https://docs.microsoft.com/en-us/aspnet/core/security/gdpr?view=aspnetcore-2.2)
* ASP.NET Core [MVC](https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-2.2) and [Razor Pages](https://docs.microsoft.com/en-us/aspnet/core/razor-pages/index?view=aspnetcore-2.2)

Để them các thành phần bên ngoài vào để bổ trợ cho hệ thống ta sử dụng các phương thức Use

Ví dụ :



Convenience methods()

Việc gọi quá nhiều phương thức từ ConfigureService tha thực hiện gọi như sau cho đơn giản cú pháp:

