**Mục Lục**

[**User API** 2](#_Toc49691931)

[**1.** **Cấu hình trong ASP.NET Core** 2](#_Toc49691932)

[**1.1.** **Load cấu hình** 2](#_Toc49691933)

[**1.1.1.** **Load cấu hình từ file Json** 4](#_Toc49691934)

[**1.1.2.** **Load từ User Secrets** 4](#_Toc49691935)

[**1.1.3.** **Load cấu hình từ biến môi trường** 4](#_Toc49691936)

[**1.1.4.** **Đọc từ command line arguments** 4](#_Toc49691937)

[**1.2.** **Đọc cấu hình** 5](#_Toc49691938)

[**2.** **Middleware/Pipeline trong C#** 5](#_Toc49691939)

[**2.1.** **Custom middleware** 7](#_Toc49691940)

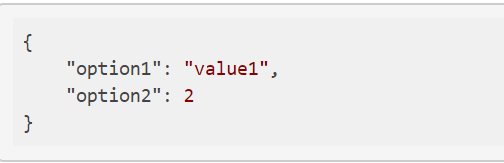
[**2.2.** **Map** 8](#_Toc49691941)

# **User API**

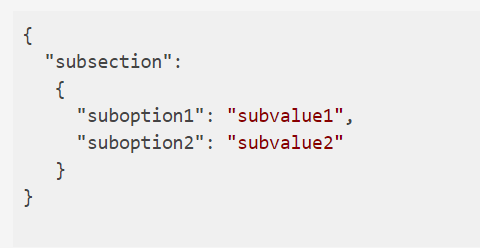
1. **Cấu hình trong ASP.NET Core**

Cấu hình trong ASP.NET Core được lưu dưới dạng key-value, người dùng có thể lưu cấu hình trong các file định dạng

* JSON
* XML
* INI



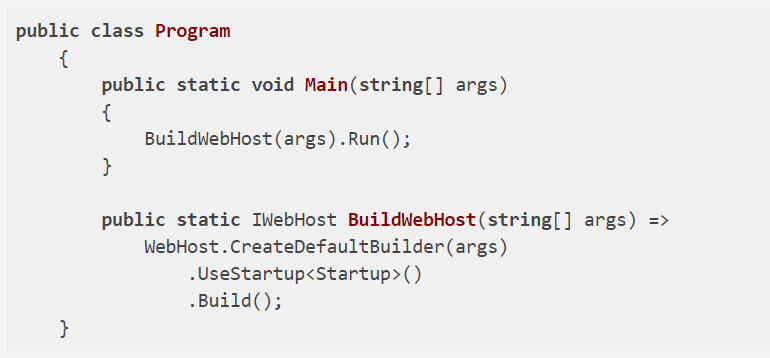
hình 1: cấu hình đơn giản



hình 2: cấu hình đơn giản

ASP.NET Core hỗ trợ đọc cấu hình từ các nguồn khác nhau và các định dạng khác nhau. Một vài nguồn được sử dụng phổ biến:

* Định dạng file: JSON, INI, XML
* Command line Arguments (tham số dòng lệnh)
* Environment variables (biến môi trường)
* Custom Provider (cái này là tự tạo ra provider riêng theo ý muốn)
  1. **Load cấu hình**



hình 3: phương thức main

phương thức Main gọi đến CreateDefaultBuilder là một Helper Class, một trong những công việc là load cấu hình từ các nguồn



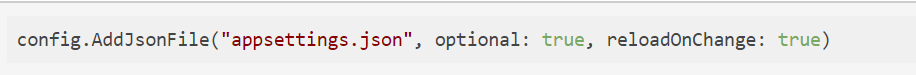
hình 4: phương thức ConfigureAppConfiguration

Phương thức ConfigureAppConfigurationnhận 2 tham số.

* Tham số thứ nhất: là thể hiện của WebHostBuilderContext
* Tham số thứ hai:là thể hiện của IConfiguretionBuilder.WebHostBuilderContext đưa ra thuộc tính HostingEnvironment giúp chúng ta biết đang chạy ở môi trường nào(Development, Production, Staging)

**IConfigurationBuilder**đưa ra một số phương thức để load file cấu hình

* + 1. **Load cấu hình từ file Json**



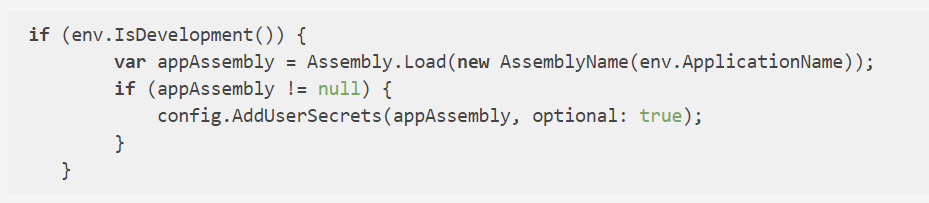
hình 5: phương thức AddJsonFile

Phương thức AddJsonFile giúp load cấu hình từ file json

Có ba tham số truyền vào:

* Tham số thứ nhất: đường dẫn tương đối đến file cấu hình
* Tham số thứ hai: nếu để true nó sẽ không sinh lỗi nếu không tìm thấy đường dẫn cung cấp bởi tham số đầu tiên
* Tham số thứ ba: nếu để true, chương trình sẽ tự động load lại cấu hình nếu file cấu hình bị thay đổi
  + 1. **Load từ User Secrets**

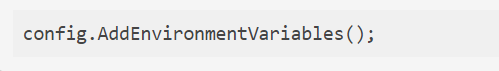
Đây là tính năng mới trong ASP.NET Core khi mà cấu hình được lưu trữ bên ngoài mã nguồn. Nó được gọi là user secrets và chỉ có thể áp dụng cho môi trường Development. Điều này hữu ích trong trường hợp bạn có những cấu hình chỉ muốn sử dụng trên máy dev thôi, và không muốn nó nằm trong mã nguồn khi đưa code lên source control chung.



hình 6: user secrets

* + 1. **Load cấu hình từ biến môi trường**

Phương thức AddEnvironmentVariables() đọc tất cả các user và system environment variable trong hệ điều hành ra.



hình 7: phương thức AddEnvironmentVariables

* + 1. **Đọc từ command line arguments**

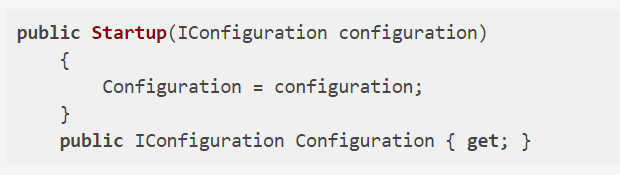
Phương thức AddCommandLine đọc tất cả các tham số của dòng lệnh khi các bạn gọi ứng dụng bằng command line mà truyền các tham số.



hình 8: phương thức AddCommandLine

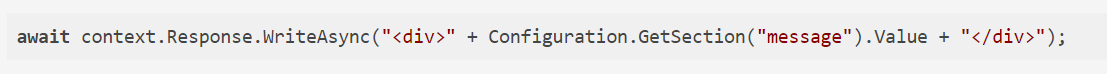
* 1. **Đọc cấu hình**

Để đọc cấu hình, chúng ta cần thể hiện của **IConfiguration**. Chúng ta có thể dùng dependency injection để lấy thể hiện của nó trong constructor của **Startup**class. Bạn có thể dùng kỹ thuật này tương tự với Controller.



hình 9: IConfiguration

Sau đó nhận giá trị với phương thức **Configuration.GetSection**:



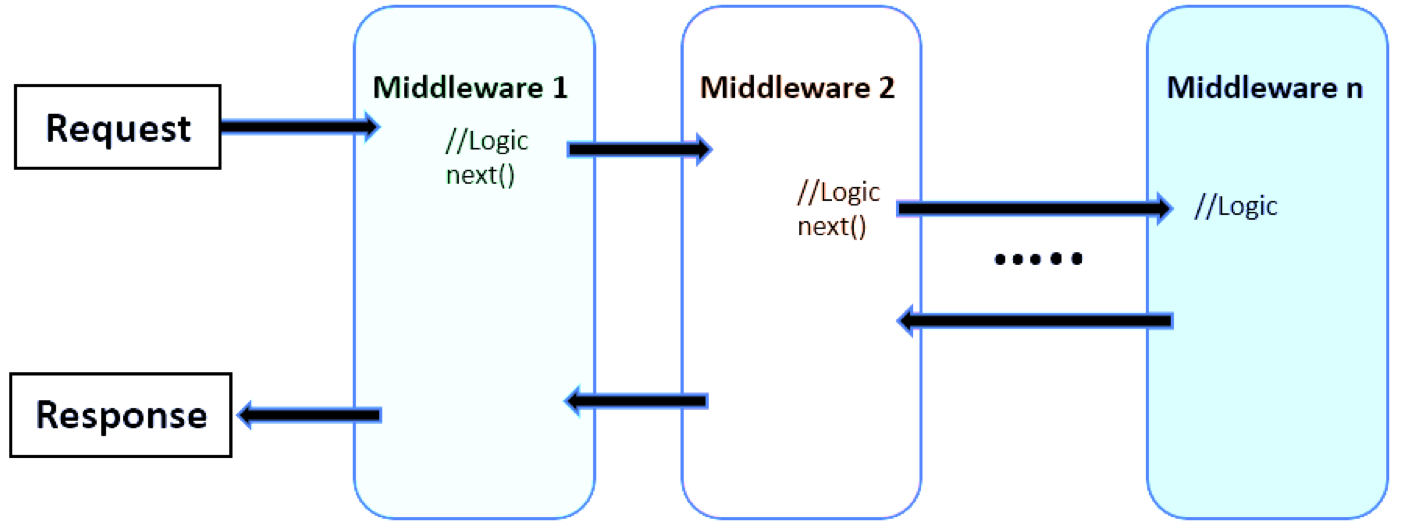
hình 10: phương thức GetSection

1. **Middleware/Pipeline trong C#**

Một Middleware là một module code nó nhận yêu cầu gửi đến Request và trả về Response. Cụ thể trong ASP.NET Core, middlewarre có thể:

* Nhận một HTTP Request gửi đến và phát sinh ra HTTP Response để trả về
* Nhận một HTTP Request gửi đến, thi hành một số tác vụ (có thể là sửa đổi HTTP Request), sau đó chuyển đến một middleware khác.
* Nhận HTTP Response, sửa nó và chuyển đến một Middleware khác

**Pipeline**: Trong ứng dụng ASP.NET Core, các **middlware** kết nối lại với nhau thành một xích, middleware đầu tiên nhận HTTP Request, xử lý nó và có thể chuyển cho middleware tiếp theo hoặc trả về ngay HTTP Response. Chuỗi các **middleware** theo thứ tự như vậy gọi là **pipeline.**



hình 11: Middleware

Các middleware như là các dịch vụ nhỏ, đăng ký vào ứng dụng bằng cách sử dụng đối tượng IApplicationBuilder, sau đó ứng dụng căn sẽ xây dựng lên các pipeline (luồng xử lý) cho các truy vấn gửi đến

Trong ASP.NET Core, middleware được định nghĩa trong **Configure lớp** Startup, đồng thời ASP.NET Core cung cấp nhiều middleware có sẵn



hình 12: Configure middleware

hình trên thể hiện một số middleware có sẵn được gọi: UseHttpRedirection(), UseStaticFile(), UseCookiePolicy(), UseRouting(), UseAuthentication(), UseAuthorization(), UseSession()

* 1. **Custom middleware**

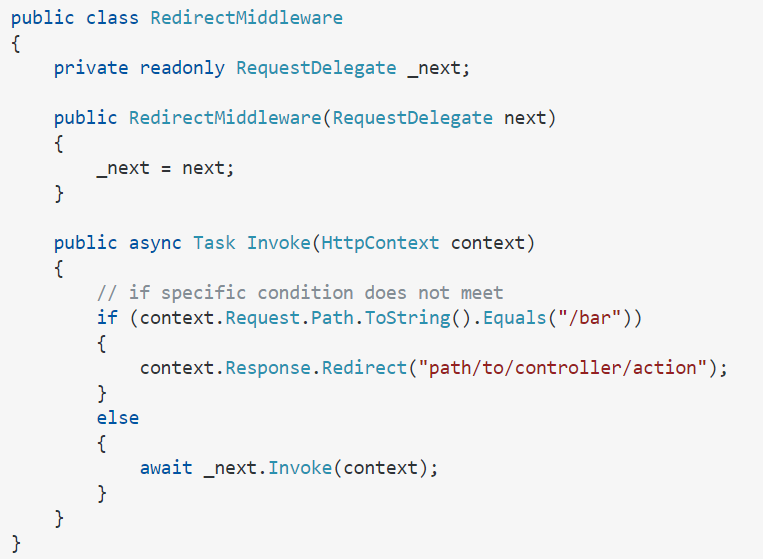
Có hai cách chính để tạo và thực thi một middleware

* Tạo inline middleware ngay trong Startup.Configure()
* Tạo class middleware



hình 13: inline middleware

hình trên có dòng code app.UseMiddleware<RedirectMiddleware>() chính là cách gọi đến một class middleware



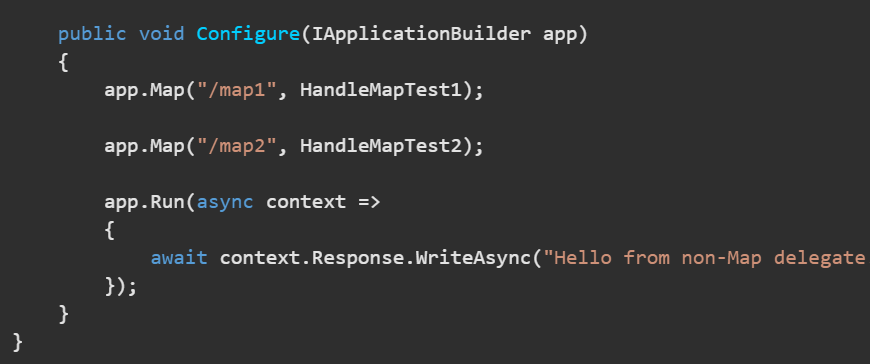
hình 14: class middleware

Như đã nói từ đầu phần 2, middleware có thể gọi đến một middleware khác, để làm việc đấy chỉ cần dòng code: await \_next(context)

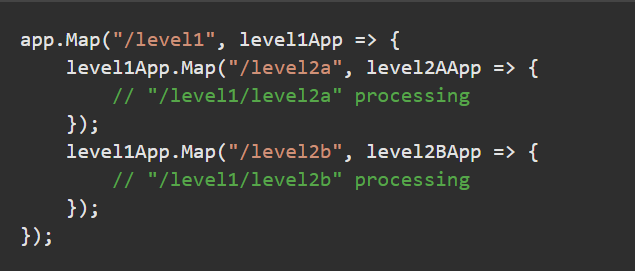
Nếu muốn gọi controller trong middleware chỉ cần dòng code: await \_next.Invoke(context)

* 1. **Map**

Map được sử dụng để có thể rẽ nhánh trong pipeline, ví dụ bạn chỉ muốn middleware tác động đến một phần chứ không phải tất cả thì bạn cần dùng đến middle



hình 15: Map



hình 16: Map