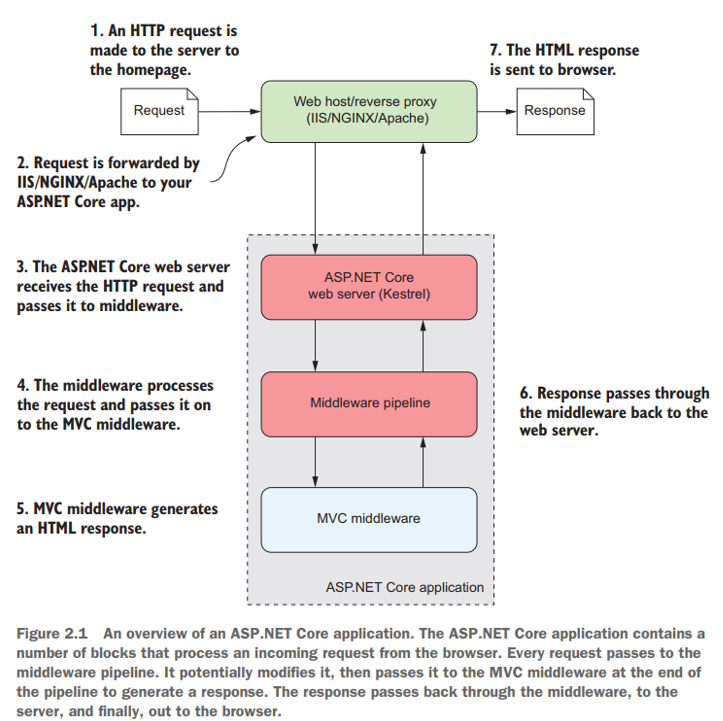


1. Kestrel làm host
2. Kestrel tạo một instance HttpContext đại diện cho http request và http response
3. Nhúng Startup Config vào trong server để xử lý.



Một middleware thực chất là 1 class đứng ra để chỉnh sửa khi n được invoke.

Middleware có thể làm gì?

1. Chỉnh sửa HttpContext và forward tiếp tới middleware đằng sau
2. Đoản mạch trả về response

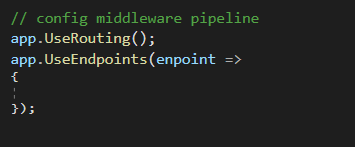
Mỗi một middleware đều có cơ hội để xử lý http request và http resonse.

Đi kèm với middleware thì có các services liên quan.

**Defintion** Các services cũng chỉ là các class đứng ra xử lý một unit task đó.

Những middleware quan trọng.

1. Routing



Có 2 dạng routing là : Global và attribute

Routing template.

ví dụ cú pháp :

"{controller=home}/{action=index}/id?"

Template cần phải có cái gì.

1. controller và action chỉ có thể có 1 cái trong chuối string

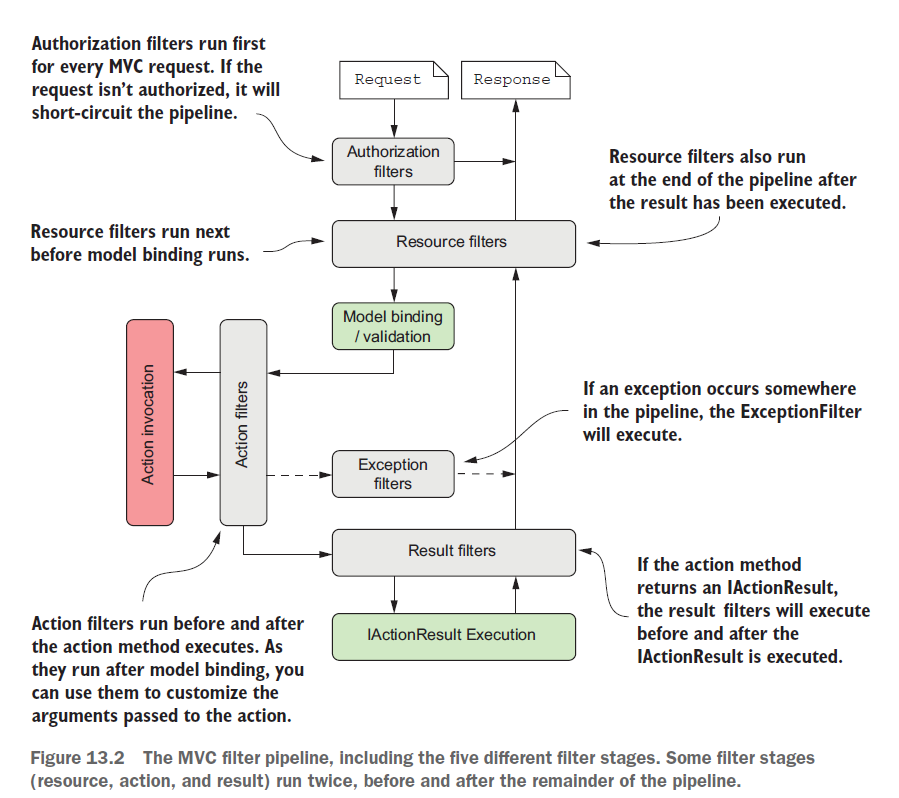
Routing gồm 2 giai đoạn.

Giao đoạn config: thì routing sử dụng app.UseEndpoints để config các templae.

Sau đó app.UseRouting sẽ lấy các template này để routing.

Quá trình route nó tạo một dictionary với cặp key/value lấy được từ http request

### MVC middleware



### Authentication-Authorizatication

Authentication là định danh user.

1. Cookie
2. Jwt bearer

Giải mã thông tin từ cookie và Jwt và đẩy thông tin đấy vào HttpContext.User

Authorization là cấp quyền,

và yêu cầu đầu tiên là ông phải authentication thành công.

Identity lam nhung viec gì

1. Lưu trữ thông tin của user
2. tích hợp luôn cookie authentication
3. Các library hỗ trợ việc tương tác với user database

IdentityServer4

1. Tạo token, tạo authorization code
2. điều hướng, tương tác với dữ liệu nhận được từ Identity, để mà có tạo token hay k
3. Đánh chặn các enpoint để thực thi cái giao thức
4. quản lý client (clientid, secret)
5. quản lý scope

Một Policy chứa nhiều Requirements

Một Requirement chứa nhiều handler

Để mà một Requirement Ok, thì ít nhất có một handler trả về là thành công ||

hadler1||hadler2||hadler3

Để mà 1 policy OK, thì tất cả các requirement phải thành công &&

requirement1&&requirement2&&....

### DI

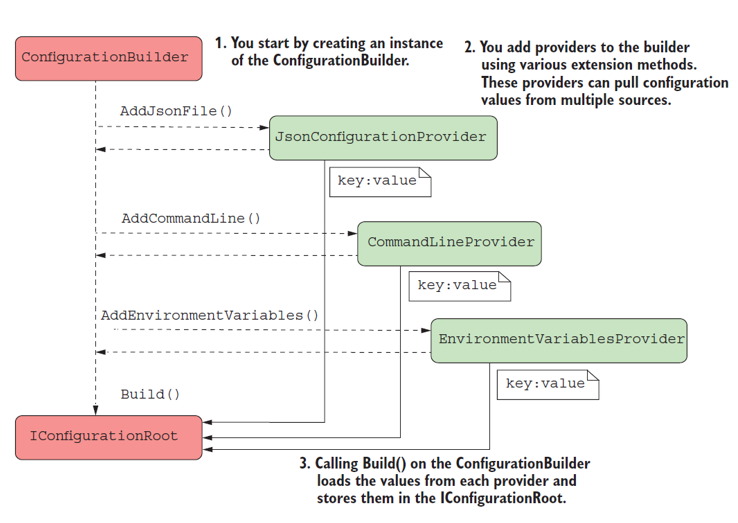
Life time của object được lấy từ container

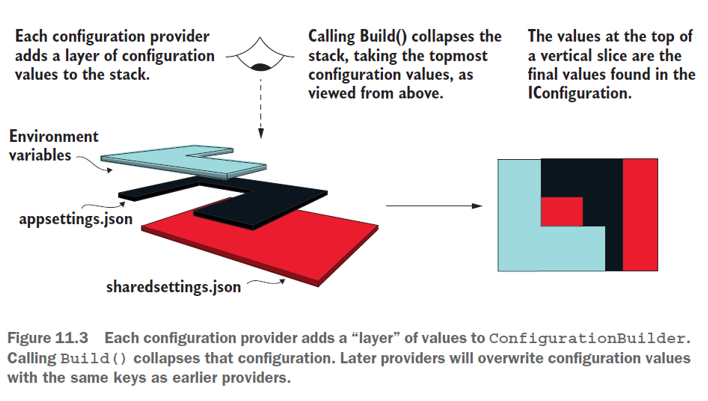
1. Singleton – tồn tại cùng với application từ khi nó được tạo đến khi application chết, và mọi cuộc gọi đến nó đều là 1 instance giống nhau
2. Scope – trong cùng một http request and response.
3. trasist – tạo mỗi khi được gọi

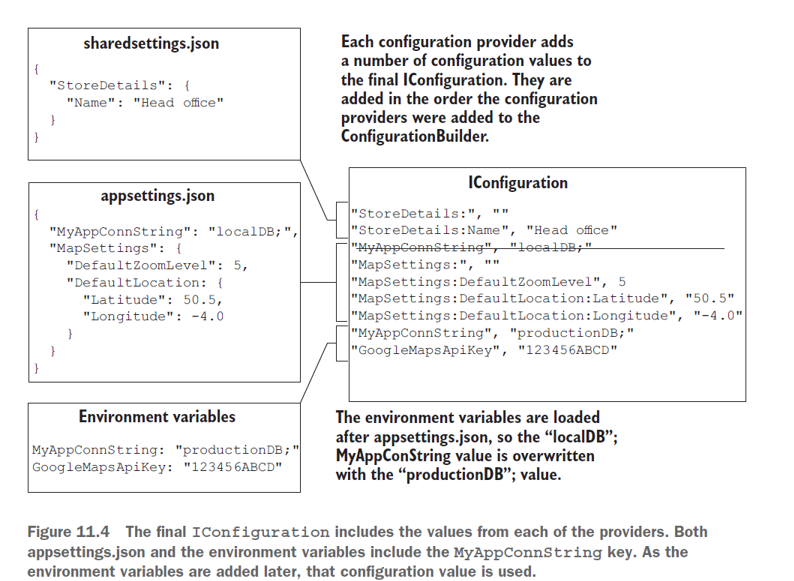
Chi phí để tạo thằng singleton quá lớn, hoặc thằng singleton không giữ trạng thái

Cảnh báo, giả dụ scope dùng đến service là singleton.

Chỉ dùng khi mà life time của dependency nhỏ hơn thằng sử dụng.







### Unit Test

Hàm test gồm 3 phần

1. Arrange: khởi tạo các biến cần thiết
2. Execute: chạy hàm/class cần test
3. Assert: đánh giá kết quả

Các khía cạnh test.

* The happy path – các arguments được cung cấp các giá trị mà hàm mong muốn
* The error path – các arguments được cung cấp giá trị mà không hợp lệ
* Edge cases – các trường hợp mà arguments nằm ngay cạnh giá trị mong muốn