|  |
| --- |
| 63734a0bd9a5160c7013baab3ecaed1e  **TỔNG HỢP KIẾN THỨC**  **TP.HCM, ngày 9 tháng 3 năm 2025** |

**MỤC LỤC**

[I. .NET Core 3](#_Toc17487)

[ .NET Core vs .NET Framework 3](#_Toc20678)

[1. Networking 3](#_Toc29699)

[1.1 Migration 3](#_Toc6420)

[2. ASP.NET MVC 4](#_Toc887)

[2.1 Model Binding (Ánh xạ dữ liệu) & Model Validation 4](#_Toc26045)

[ Binding 4](#_Toc4111)

[ Validation 4](#_Toc15706)

[ Dựng Validation 5](#_Toc4511)

[ Dựng Binding 6](#_Toc7567)

[ Syntax 7](#_Toc18417)

[3. ASP.NET CORE 7](#_Toc25831)

[3.1 Middleware 7](#_Toc26022)

[ Pipeline 7](#_Toc6324)

[ Thứ tự middleware 9](#_Toc27413)

[ MVC Endpoint (Endpoint Middleware) 9](#_Toc20223)

[ Custom middleware 9](#_Toc30987)

[ Thêm phương thức mở rộng 11](#_Toc1213)

[ Truyền dữ liệu giữa các Middleware 12](#_Toc25238)

**I. .NET Core**

* **.NET Core vs .NET Framework**

|  |  |  |
| --- | --- | --- |
| **BASED ON** | **.NET core** | **.NET Framework** |
| **Open Source** | √ | × |
| **Cross-Platform** | Windows,Linux, Mac OS | Only windows |
| **Application Models** | Focus on Web,windows mobile, windows store | Desktop,web applications,windows forms, WPF |
| **Installation** | Packaged and installed independently,Cross-platform | A single package |
| **Support** | Microserives, Rest APIs | Rest APIs |
| **WCF Services** | × | √ |
| **Performance and Scalability** | high | less |
| **Android Development** | iOS, Android, Windows phones | × |
| **Packaging and Shipping** | .Net core is shipped as a collection of Nugget packages | All the libraries of .NET Framework are packaged and shipped together |
| **Deployment Model** | Updated instantly on one machine at at a time => flexible | Deployed on the Internet Information Server only |
| **Compatibility** | Windows,Linux, Mac OS | Only windows |
| **CLI Tools** |  |  |
| **Security** | × | √ |

**1. Networking**

**1.1 Migration**

Migration là kỹ thuật trong việc tương tác với cơ sở dữ liệu, khi thay đổi CSDL ở code sẽ cập nhật lên CSDL, đảm bảo toàn vẹn dữ liệu và lịch sử thay đổi.

Khi sử dụng EF làm việc với DB, có các cách tạo dựng database:

- Database first (Có database trước sau đó viết code)

- Code first (Viết code tạo database)

- Model first (Tạo model, sau đó code, tạo database sau)

 <https://www.dammio.com/2017/04/20/entity-framework-phan-5-tao-mo-hinh-truoc-model-first>

|  |  |
| --- | --- |
| **Lệnh** | **Ý nghĩa** |
| dotnet tool install --global dotnet-ef | Cài đặt công cụ .net ef |
| dotnet tool update --global dotnet-ef | Cập nhật công cụ .net ef |
| dotnet ef migrations add Init-1 | Tạo migration với tên Init-1 |
| dotnet ef migrations list | Danh sách các migration |
| dotnet ef database update | Cập nhật database với migration cuối |
| dotnet ef database update Init-1 | Cập nhật database với migration init-1 |
| dotnet ef migrations remove | Xóa migration cuối |
| dotnet ef migrations script --output  migrations.sql | Xuất lệnh sql khi thực hiện migration |
| dotnet ef database drop -f | Xóa database |

|  |  |
| --- | --- |
| **Package Manager Console** | |
| **Thêm migration** | Add-migration[name-migration] |
| **Cập nhật database từ migration** | Update-database  => Update-database -migration [migration-name] |
| **Xóa migration** | Remove-migration |

1. **ASP.NET MVC**

**2.1 Model Binding (Ánh xạ dữ liệu) & Model Validation**

* Binding

Controller và Razor pages làm việc với dữ liệu bằng HTTP requests, ví dụ, route data gửi record key, posts form fileds gửi giá trị **properties** model

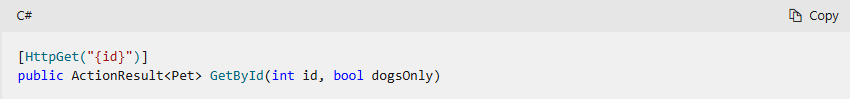
- Truy xuất dữ liệu (**Retrives data**) từ nhiều nguồn: route data, form flelds, and query strings.

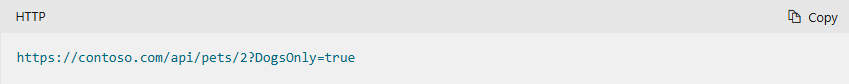
- Chuyển dữ liệu từ controllers và Razor pages vào các thông số (**parameters**) và **public properies**

- Chuyển đổi chuỗi dữ liệu sang kiểu .NET

- Cập nhật thuộc tính (**properties**) của các kiểu phức tạp (**complex types**)

Mục đích



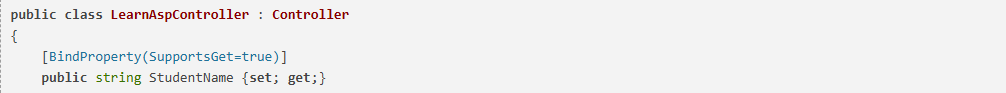


→ {“DogsOnly”= true; “id”=2 }

Khi hệ thống tìm từ nguồn đến, nếu không có sẽ thiết lập giá trị mặc định cho tham số của phương thức

Lỗi khi binding hoặc validation sẽ lưu trong **ControllerBase.ModelState** (dùng **ModelState.IsValid** để kiểm tra ánh xạ dữ liệu.

**[BindProperty] :** Để nguồn dữ liệu qua **HTTP requests** (Form, route data, url request,…)tự động convert dữ liệu và gán vào thuộc tính của Controller



**- [FormQuery] :** dữ liệu trích xuất từ Url query (/abc/?key=value)

**- [FormRoute] :** dữ liệu lấy từ giá trị trong tham số của Route

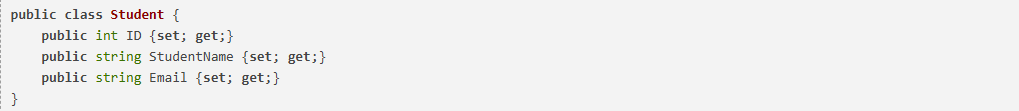
**- [FormForm] :** dữ liệu lấy từ Form

**- [FormBody]:** dữ liệu lấy từ Body của Http Request

**- [FormHeader] :** dữ liệu lấy từ Header của Http Request

**Name trong [BindingProperty], [FromQuery], [FromRoute] … :** khi binding dữ liệu mà tên ở nguồn đến và tên thuộc tính Controller khác nhau dùng thuộc tính Name







→ prefix.property\_name (tên key được tổ hợp từ tiền tố thuộc tính và tên thuộc tính).

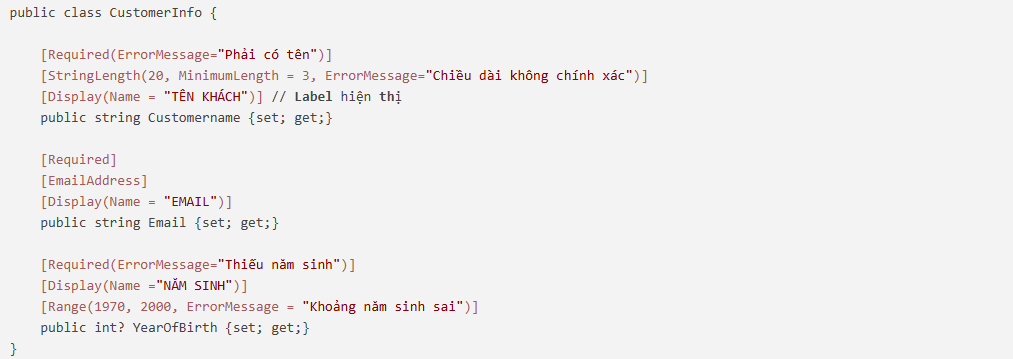
Khi các cơ chế binding,validation mặc định không đáp ứng nhu cầu nào đó có thể xây dựng 1 binding,validation riêng.

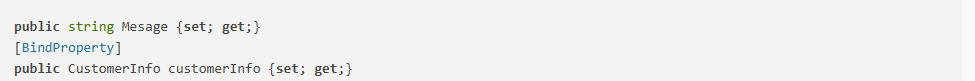
* Validation

Để đảm bảo dữ liệu chính xác, cần kiểm tra trước khi xử lý, .Net core hỗ trợ DataAnnotation Attribute với mô hình binding.

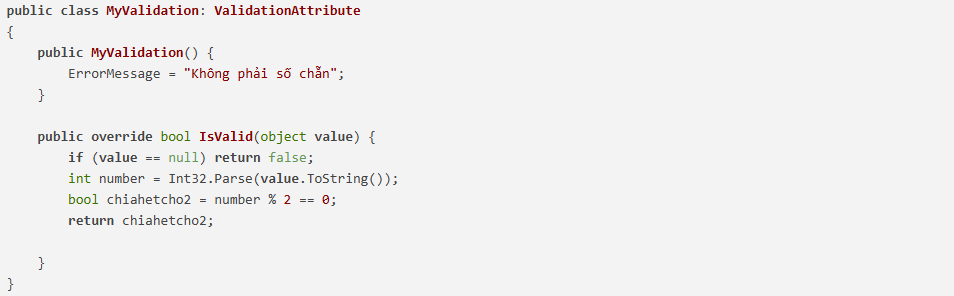
|  |  |
| --- | --- |
| Attribute | Sử dụng |
| Required | Thuộc tính cần phải có giá trị |
| StringLength | Chiều dài chuỗi (tối thiểu,tối đa) |
| Range | Số nằm trong khoảng |
| RegularExpression | Gía trị phù hợp với biểu thức chính quy |
| CustomValidation | Thuộc tính sử validation tự tạo |
| EmailAddress | Là địa chỉ Email |
| FileExtention | Phần mở rộng file |
| MaxLength | Dài tối đa của mảng, chuỗi |
| MinLength | Dài tối thiểu |
| Phone | Là số điện thoại |

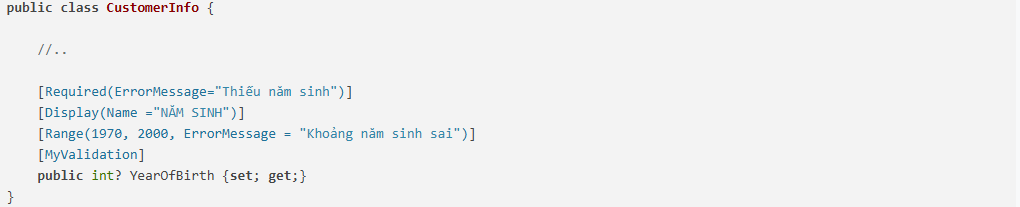
Ví dụ:



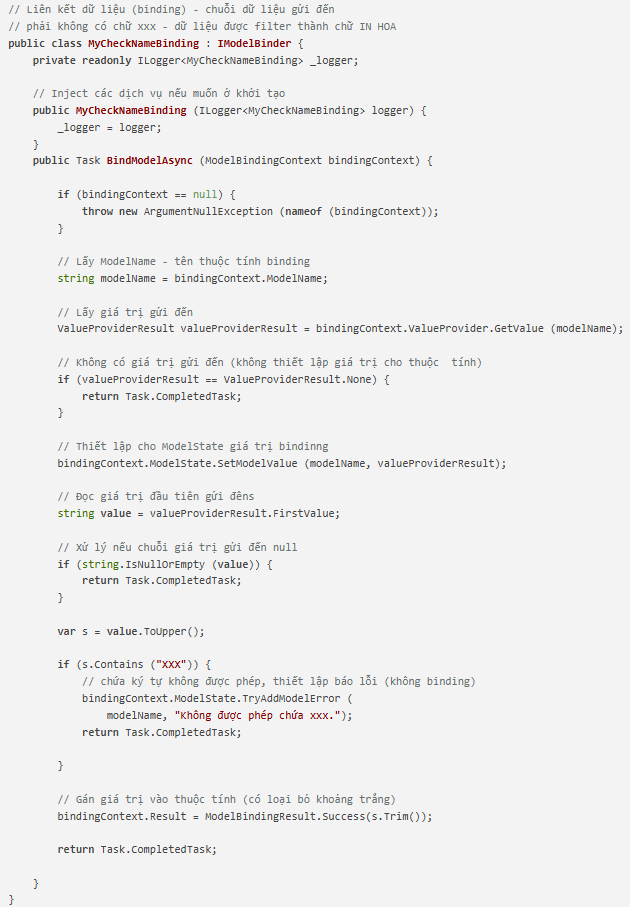


* Dựng Validation





* Dựng Binding



* Syntax

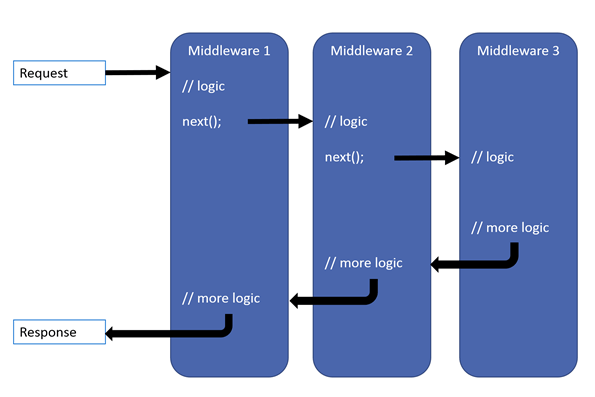




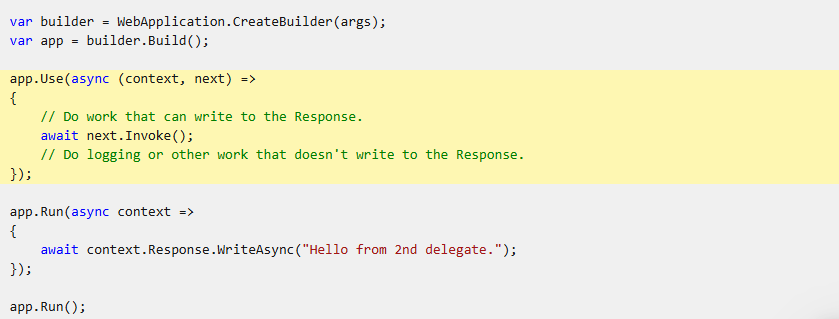
1. **ASP.NET CORE**
   1. **Middleware**

* Pipeline

Trong ứng dụng .net core , các middleware kết nối với nhau thành xích, chuỗi các middleware theo thứ tự như hình gọi là pipeline.



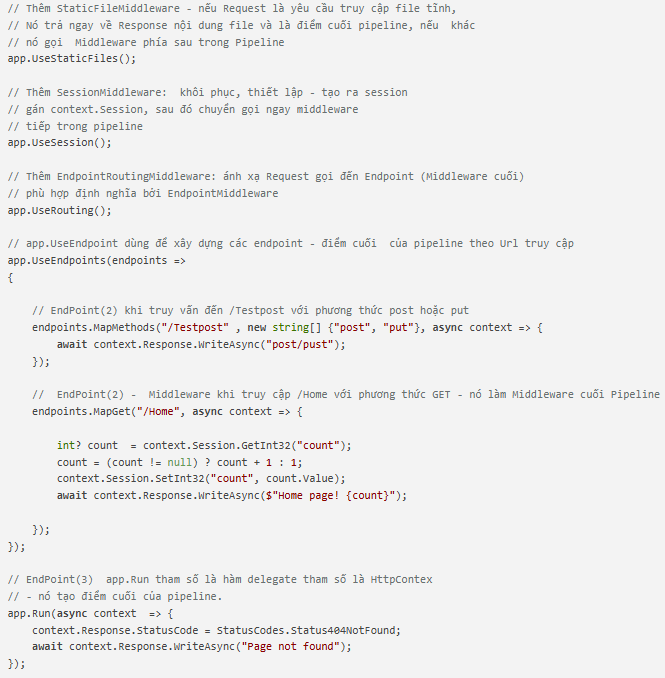
Các middleware là các dịch vụ nhỏ, đăng ký vào ứng dụng bằng cách sử dụng đối tượng IapplicationBuilder, sau đó ứng dụng sẽ tạo luồng xử lý (pipeline) cho các truy vấn gửi lên.



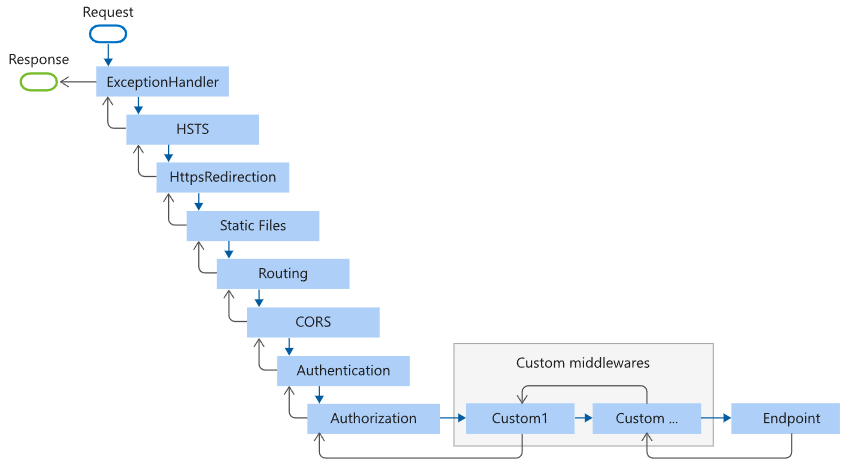
Các middleware trong pipeline xử lý các HttpMessage có dạng:

- Đầu vào là tham số kiểu HttpContext (chứa HttpRequest và HttpResponse), thi hành code và sau đó chuyển đến middleware tiếp theo.

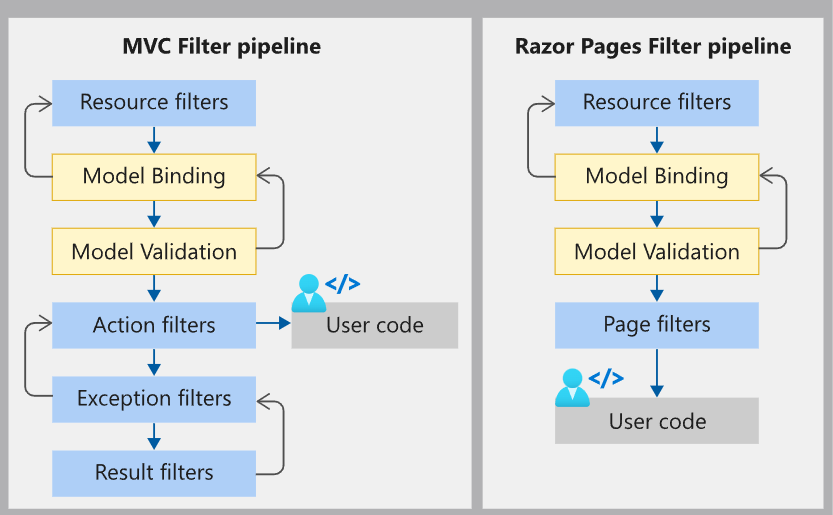
- app.Run() tham số (sử dụng phương thức overload) là hàm delegate tham số là HttpContext - **là điểm cuối của pipeline**



* Thứ tự middleware



* MVC Endpoint (Endpoint Middleware)



* Custom middleware

Cấu trúc middleware trong ASP.NET:

- Có một phương thức khởi tạo public với tham số thứ nhất kiểu **RequestDelegate,** các tham số tiếp theo phải inject được từ DI của hệ thống.

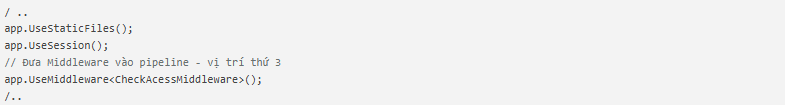
- Tối thiểu 1 phương thức Invoke, InvokeAsync với tham số là HttpContext, trả về Task (**InvokeAsync**).

- Trong invoke, InvokeAsync viết code xử lý middleware,nếu muốn chuyển đến middleware khác bằng cách gọi RequestDelegate đã truyền trong hàm tạo, nếu không cần chuyển middleware tiếp theo thì cần đảm bảo HttpResponse trong HttpContext phù hợp

Ví dụ 1:



→ Đưa middleware vào pipeline



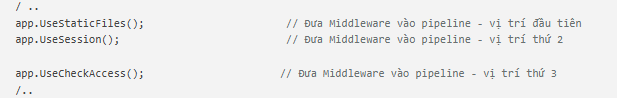
Ví dụ 2:



* Thêm phương thức mở rộng

Ví dụ 1:





Ví dụ 2:



Program.cs



#Xây dựng middleware triển khai từ giao diện IMiddleware



→ Để sử dụng middleware này, phải đăng ký vào DI container (Dependency Injection)



→ Sau đó thêm vào pipeline



* Truyền dữ liệu giữa các Middleware

Truyền dữ liệu giữa các middleware sử dụng **HttpContext.Items**