Request Processing Pipeline

ASP .NET Core

Yash Lathiya

Table of Contents

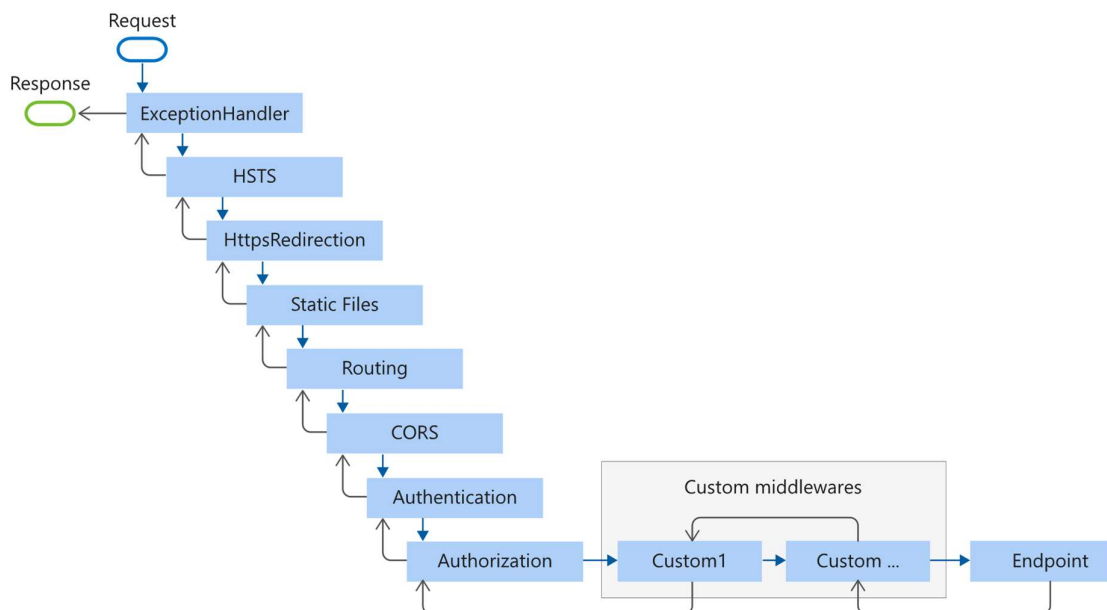| Sr No.	Topic	Date	Page No.
1	Middleware		
2	Routing		
3	Filters		
4	Controller Initialization		
5	Action Method		

Middleware
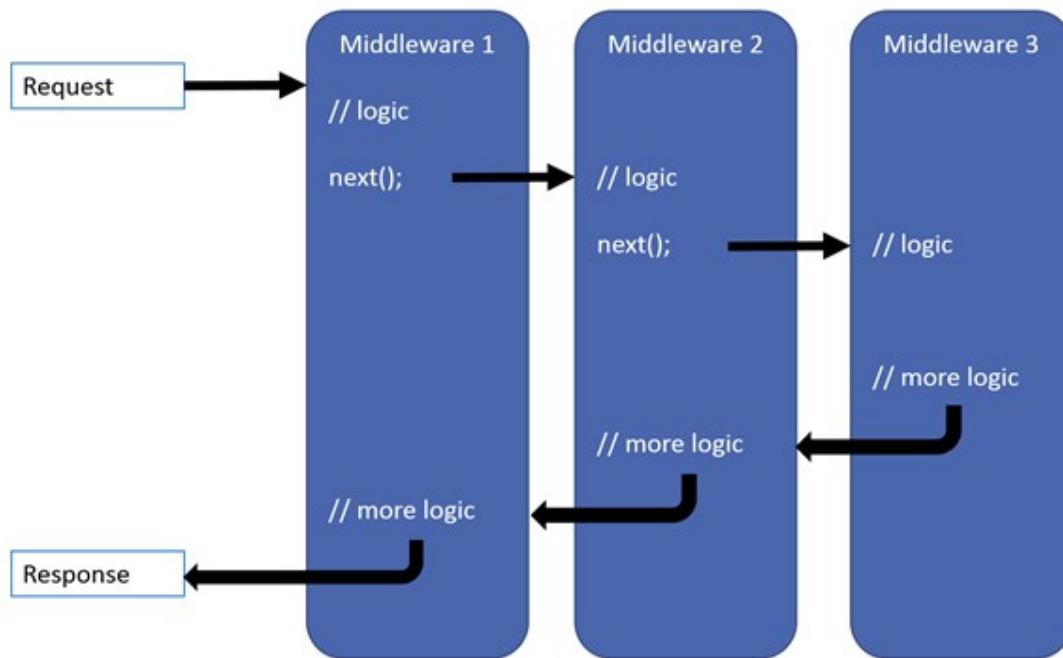
Middleware is software which handles requests through pipelines, it processes request & response one by one.

When middleware doesn't pass the response to next middleware, it is calling short-circuiting of the middleware.
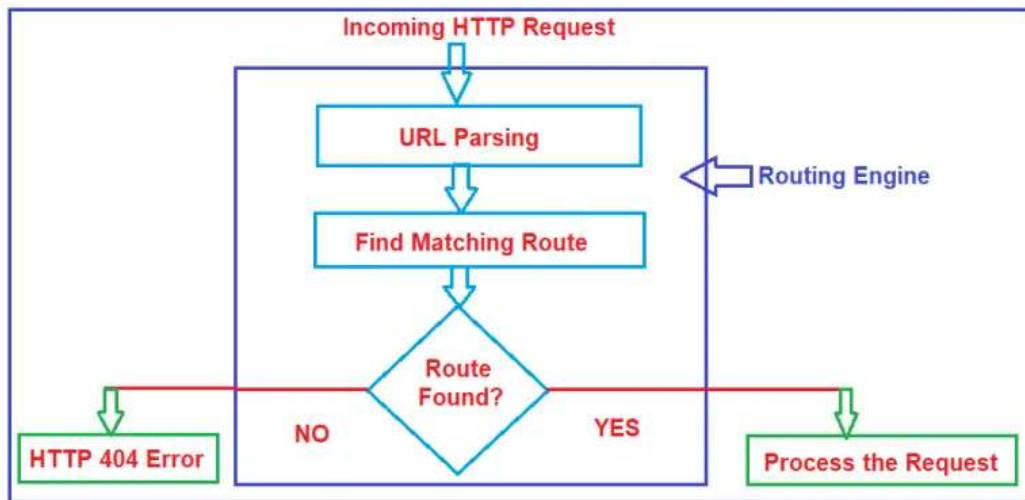
There are 3 types of middleware

Terminal Middleware — Middleware which deals with client

Non-terminal middleware — Which sends the request to next middleware

Custom Middleware — Developer specific middleware

Routing

Routing can be managed in `startup.cs` file,

Where can we configure specific routes

e.g. I have CLRoutingController and it contains greetings method.

```
// All methods in controller
endpoints.MapControllers();

// Map the Greeting action to the /api/CLRouting endpoint with GET method
//api/CLRouting/get
endpoints.MapGet("/api/CLRouting", async context =>
            {
                // You can handle the request logic directly here
                string name = context.Request.Query["name"];
                await context.Response.WriteAsync("Hello World + " + name);
            });

// Redirect requests from api/CLRouting/hello to api/CLRouting/greetings
endpoints.MapGet("/api/CLRouting/hello", context =>
            {
                string name = context.Request.Query["name"];

context.Response.Redirect($"/api/CLRouting/greetings?name={name}");
                return Task.CompletedTask;
            });

// Add fallback endpoint to catch all unmatched paths
endpoints.MapFallback(async context =>
            {
                context.Response.StatusCode = StatusCodes.Status404NotFound;
                await context.Response.WriteAsync("Bro , Entered path does
not exist !!!");
            });
```
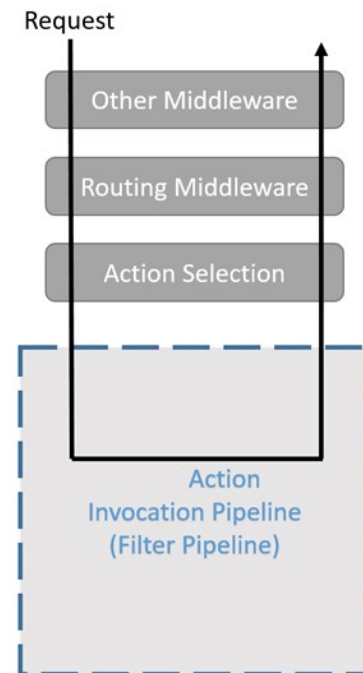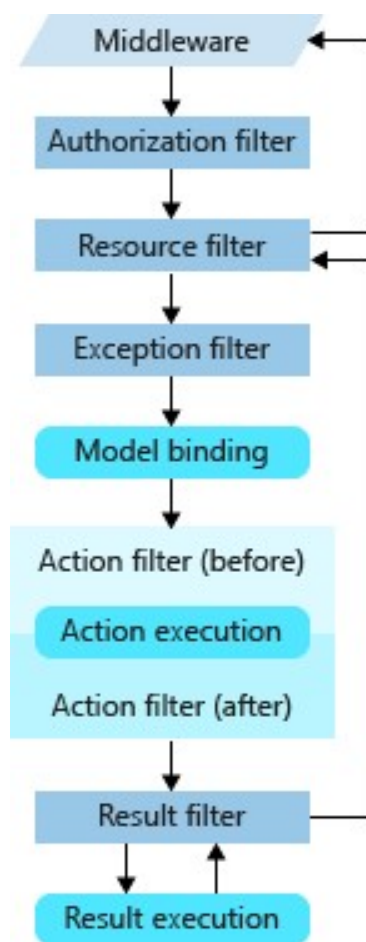
Filter

Filter is a implementation of request pipeline which can we add at any particular method or controller or we can declare that globally.

Built-in filter provides :

. Authorization

. Response Caching
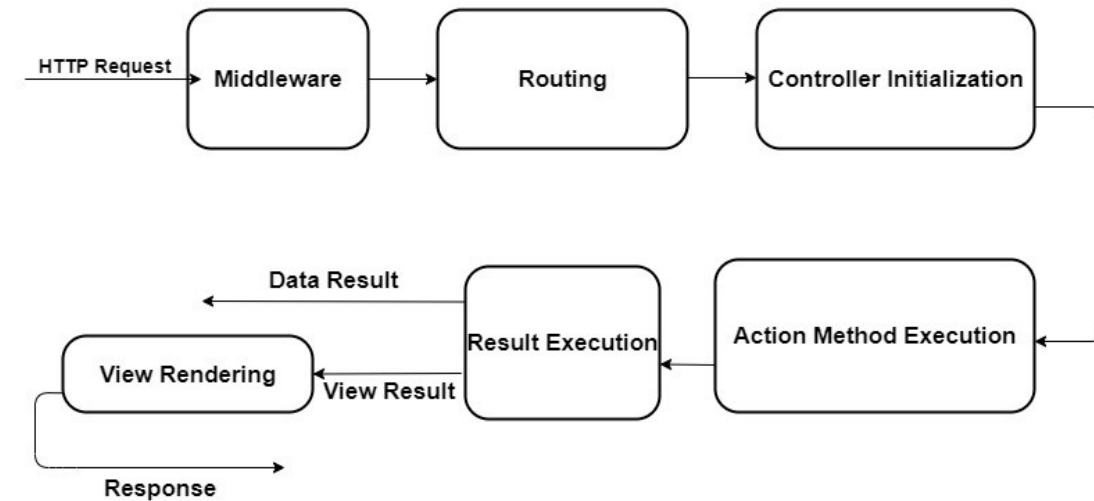
. Short – Circuiting the request

. Exception etc..

There are main two types of filter

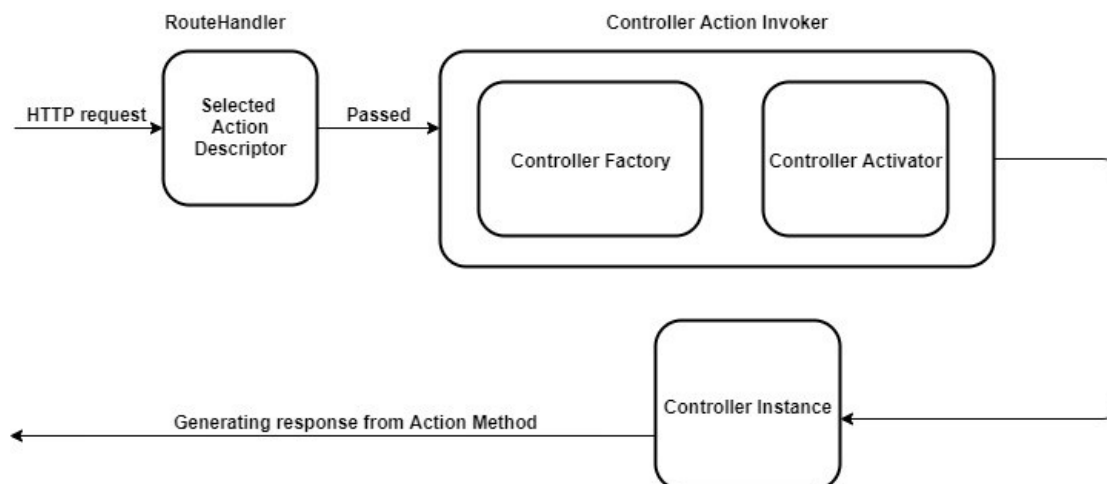. synchronous Filter

. Asynchronous Filter

Async filter can be used for logging, caching, authorization with additional services. Data processing & Resource cleanup

Controller Initialization

ASP.NET CORE MVC Request Life Cycle

HTTP Request → Middleware → Routing → Controller Initialization → Action Method Execution → Result Execution → Data Result

Result Execution → View Result → View Rendering → Response

Controller Initialization Process

HTTP request → **RouteHandler** (Selected Action Descriptor) → Passed → **Controller Action Invoker** (Controller Factory, Controller Activator) → Controller Instance → Generating response from Action Method

Action Method

As Json Data

```
return Ok(new
{
    key1 = "value1",
    key2 = "value2"
});
```

Redirect to another page / link

```
return Redirect("https://www.google.com/");
```

Return content

```
return Content("This is plain text", "text/plain");
```

Return file

```
return File(fileBytes, "application/pdf", "C_Sharp_Advance_Doc.pdf");
```

Return Status Code

```
return NotFound(); // status code – 404
```

```
return Unauthorized();
```