

Request Processing Pipeline

ASP .NET Core

Yash Lathiya

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

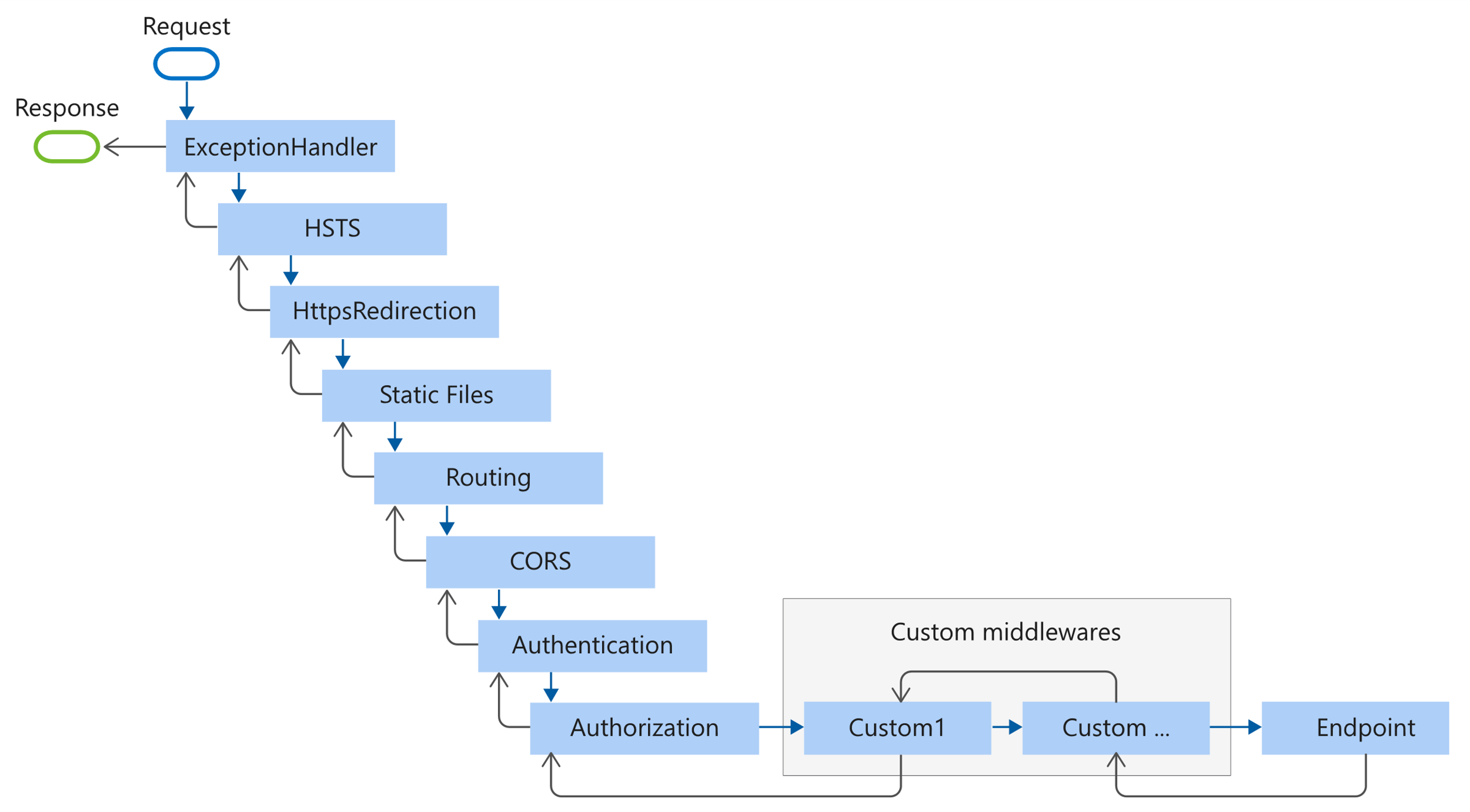
Table of Contents

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.



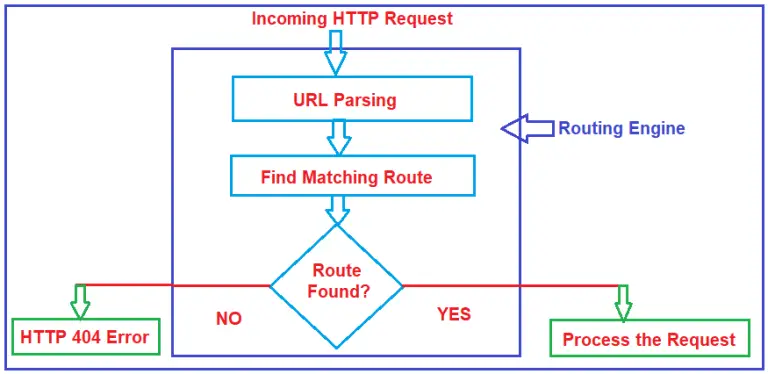
There are 3 types of middleware

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

Controller Initialization

Action Method