Instructor example

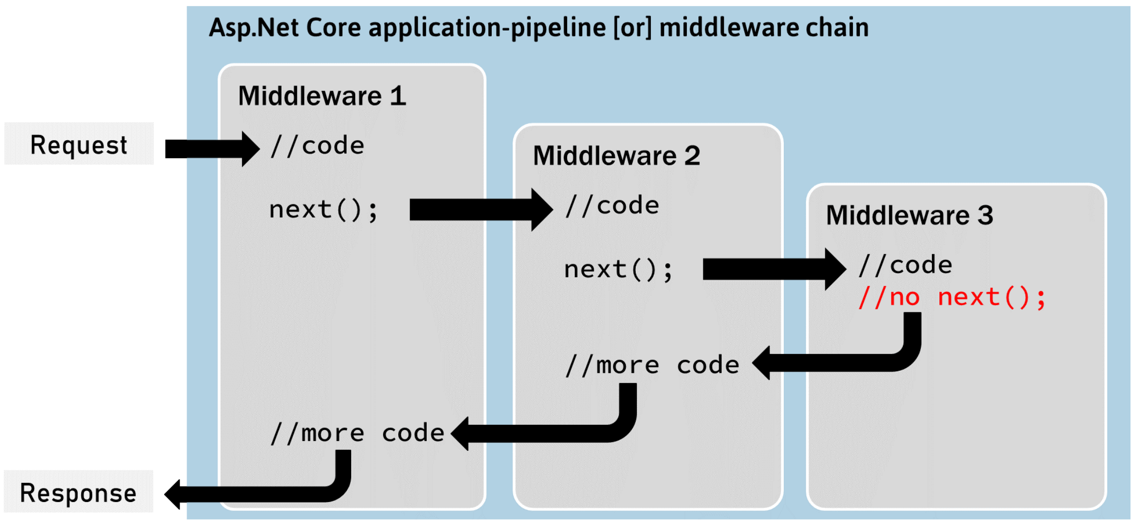
Web University by Harsha Vardhan

What is middleware?

It is the code that is injected into the application pipeline to handle requests and responses. They are just like chained to each other and form as a pipeline. The incoming requests are passed through this pipeline where all middleware is configured, and middleware can perform some action on the request before passing it to the next middleware. Same as for the responses, they are also passing through the middleware but in reverse order.

The main responsibilities of a middleware:

- Generate an HTTP response for an incoming HTTP request
- Intercept and make changes to an incoming HTTP request and pass it on to the next piece of middleware.
- Intercept and make changes to an outgoing HTTP response, and pass it on to the next piece of middleware.



What is the difference between IApplicationBuilder.Use() and IApplicationBuilder.Run()?

We can use both the methods while defining application request pipeline. Both are used to add middleware delegates to the application request pipeline. The middleware that is added using IApplicationBuilder.Use() may call the next middleware in the pipeline whereas the middleware that is added using the IApplicationBuilder.Run() method never calls the subsequent middleware. After IApplicationBuilder.Run method, system stop adding middleware in the request pipeline.

The IApplicationBuilder.Run() adds a terminating middleware; so no other middleware will run after the middleware added with Run().

What is the use of the "Map" extension while adding middleware to the ASP.NET Core pipeline?

It is used for branching the pipeline. It branches the ASP.NET Core pipeline based on request path matching. If the request path starts with the given path, middleware on to that branch will execute.

How do you create a custom middleware?

The custom middleware class can be used to separate the middleware code from the Program.cs (or Startup.cs in earlier versions).

You can create a custom middleware either by implementing IMiddleware interface or by using convention middleware

Implementing IMiddleware:

```
public class MyCustomMiddleware : IMiddleware
{
 public async Task InvokeAsync(HttpContext context, RequestDelegate next)
 {
  //TO DO: before logic
  await next(context);
  //TO DO: after logic
 }
}
```

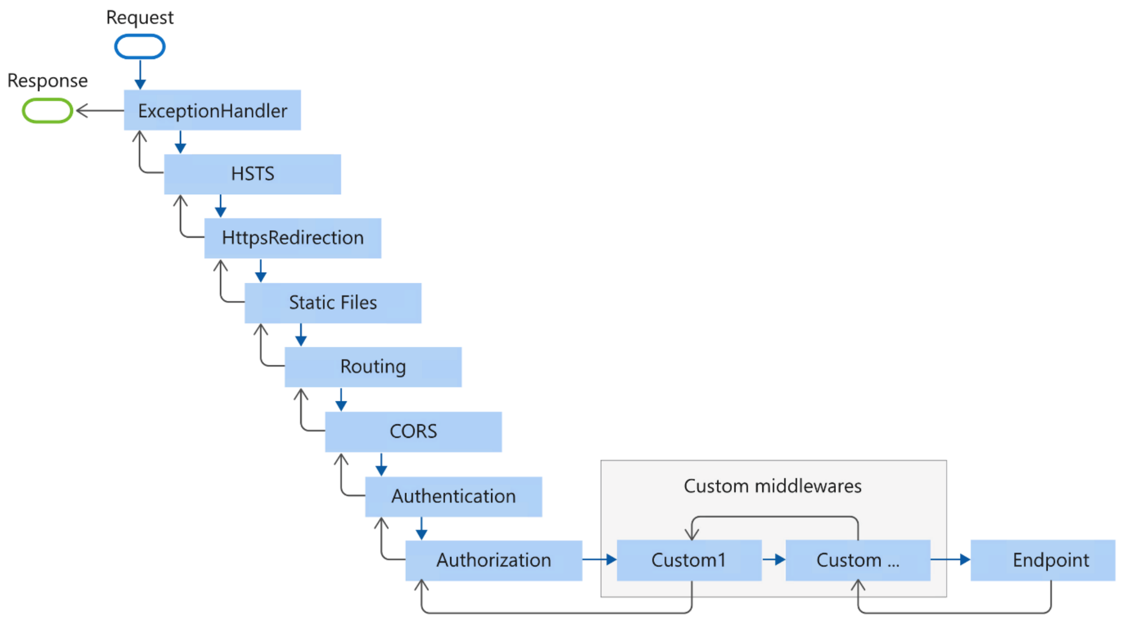
Convention Middleware:

```
public class MyCustomMiddleware
{
 private readonly RequestDelegate _next:
 public MiddlewareClassName(RequestDeletegate next)
 {
 }

 public async Task InvokeAsync(HttpContext context)
 {
  //TO DO: before logic
  await _next(context);
  //TO DO: after logic
 }
}
```

What is the right order of middleware used in production-level applications?

You can see how, in a typical app, existing middlewares are ordered and where custom middlewares are added. You have full control over how to reorder existing middlewares or inject new custom middlewares as necessary for your scenarios.



Your submission

Cesar David Guerrero Regino
Posted 5 months ago

What is middleware?

It is a component asembled into an application to handle requests and responses

What is the difference between IApplicationBuilder.Use() and IApplicationBuilder.Run()?

.Use methos allows to receive the next context which pass onto the next middleware to be called, whereas .Run does not receive one.

What is the use of the "Map" extension while adding middleware to the ASP.NET Core pipeline?

Map extension will allow to serch for keys in http requests

How do you create a custom middleware?

there are twoo ways, one inheriting from Imiddleware class and the second one creating a own class to implement it

What is the right order of middleware used in production-level applications?

I don´t remember all but I know the custom middlewares are place by the end