Day 59/60 - Creating Custom Middleware in ASP.NET Core

Custom middleware is a core part of ASP.NET Core that allows you to intercept, modify, or handle HTTP requests and responses. By creating your own middleware, you can add functionality like logging, authentication, or custom error handling without modifying your controllers.

What is Middleware?

Middleware components are assembled into a pipeline where each component can:
- Process an incoming HTTP request
- Decide whether to pass the request further down the pipeline
- Modify the response on the way back

Creating Custom Middleware

1. Create a Middleware Class

Define a class with a constructor accepting a RequestDelegate and an Invoke method that takes an HttpContext.

```
public class CustomLoggingMiddleware
{
 private readonly RequestDelegate _next;

 public CustomLoggingMiddleware(RequestDelegate next)
 {
 _next = next;
 }

 public async Task InvokeAsync(HttpContext context)
 {
 // Pre-processing logic: Log request details
 Console.WriteLine($"Handling request: {context.Request.Method} {context.Request.Path}");

 // Call the next middleware in the pipeline
 await _next(context);

 // Post-processing logic: Log response status code
 Console.WriteLine($"Finished handling request. Response status: {context.Response.StatusCode}");
 }
}
```

2. Extension Method for Easy Registration

Create an extension method on IApplicationBuilder to add the middleware to the pipeline.

```
public static class CustomLoggingMiddlewareExtensions
{
 public static IApplicationBuilder UseCustomLogging(this IApplicationBuilder builder)
 {
 return builder.UseMiddleware<CustomLoggingMiddleware>();
 }
}
```

3. Register the Middleware

In the Configure method of your Startup class (or in your minimal API setup), add your middleware to the pipeline.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
 // Register custom middleware
 app.UseCustomLogging();

 // Other middleware registrations like routing, authentication, etc.
 app.UseRouting();
 app.UseEndpoints(endpoints => endpoints.MapControllers());
}
```

Benefits of Custom Middleware

- It allows you to encapsulate cross-cutting concerns in one place.
- It provides flexibility to modify the HTTP request and response.
- It makes your application more modular and easier to maintain.

Custom middleware is a powerful tool in ASP.NET Core, offering a clean way to handle common tasks without cluttering your business logic. Try creating your own middleware to add a custom logging, error handling, or authentication mechanism to your application.