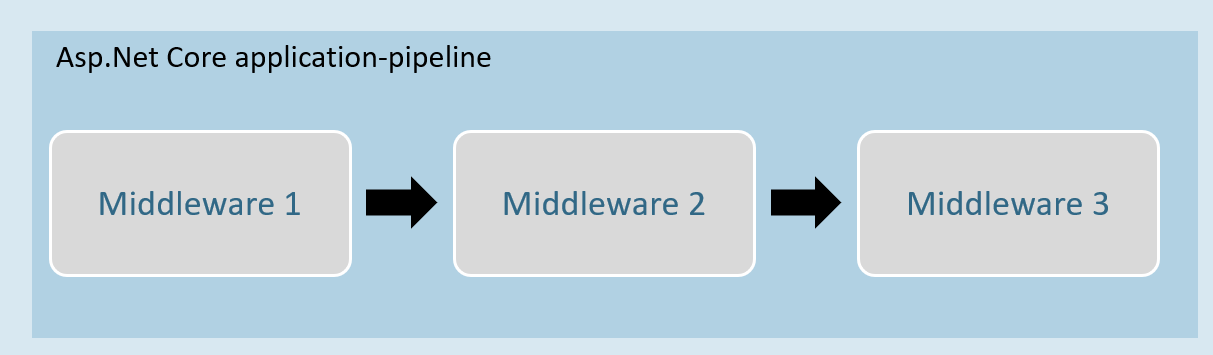
**Section Cheat Sheet (PPT)**

Introduction to Middleware

Middleware is a component that is assembled into the application pipeline to handle requests and responses.

Middlewares are chained one-after-other and execute in the same sequence how they're added.





Middleware can be a request delegate (anonymous method or lambda expression) [or] a class.

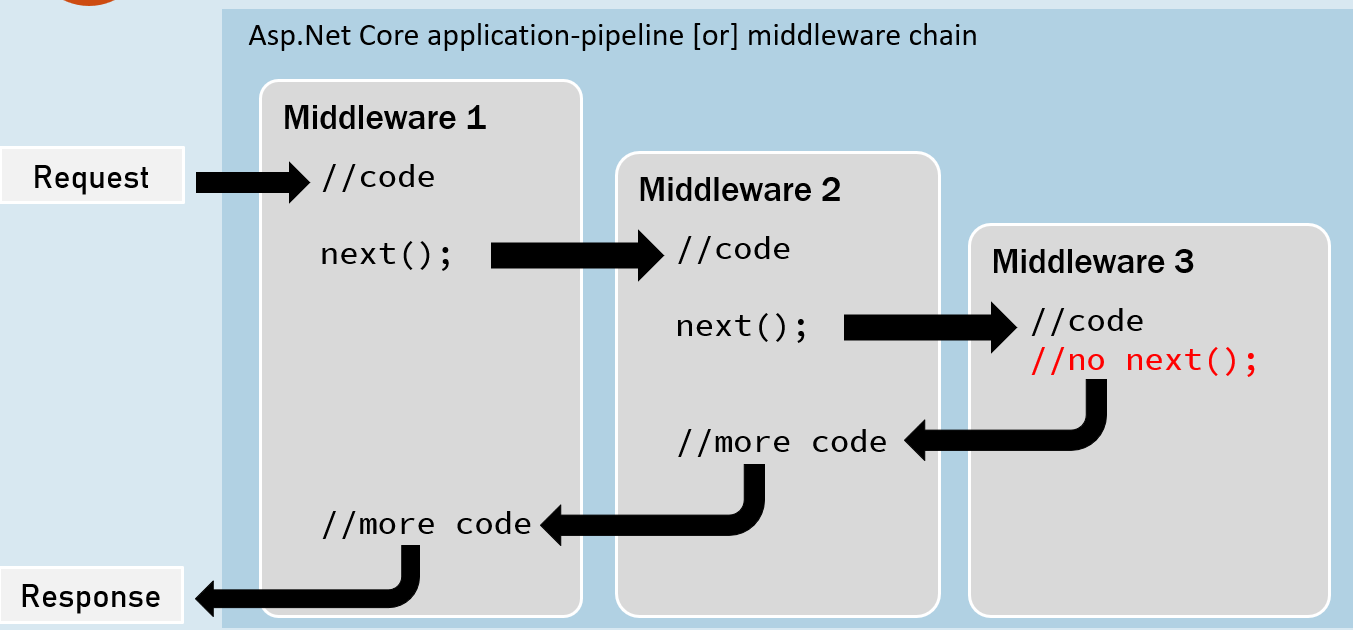
Middleware - Run

**app.Run( )**

1. app.Run(async (HttpContext context) =>
2. {
3. //code
4. });

The extension method called “Run” is used to execute a terminating / short-circuiting middleware that doesn’t forward the request to the next middleware.

Middleware Chain



**app.Use( )**

1. app.Use(async (HttpContext context, RequestDelegate next) =>
2. {
3. //before logic
4. await next(context);
5. //after logic
6. });

The extension method called “Use” is used to execute a non-terminating / short-circuiting middleware that may / may not forward the request to the next middleware.

Middleware Class

Middleware class is used to separate the middleware logic from a lambda expression to a separate / reusable class.

1. class MiddlewareClassName : IMiddleware
2. {
3. public async Task InvokeAsync(HttpContext context, RequestDelegate next)
4. {
5. //before logic
6. await next(context);
7. //after logic
8. }
9. }

app.UseMiddleware<MiddlewareClassName>();

Middleware Extensions

1. class MiddlewareClassName : IMiddleware
2. {
3. public async Task InvokeAsync(HttpContext context,RequestDelegate next)
4. {
5. //before logic
6. await next(context);
7. //after logic
8. }
9. });

Middleware extension method is used to invoke the middleware with a single method call.

1. static class ClassName
2. {
3. public static IApplicationBuilder ExtensionMethodName(this IApplicationBuilder app)
4. {
5. return app.UseMiddleware<MiddlewareClassName>();
6. }
7. }

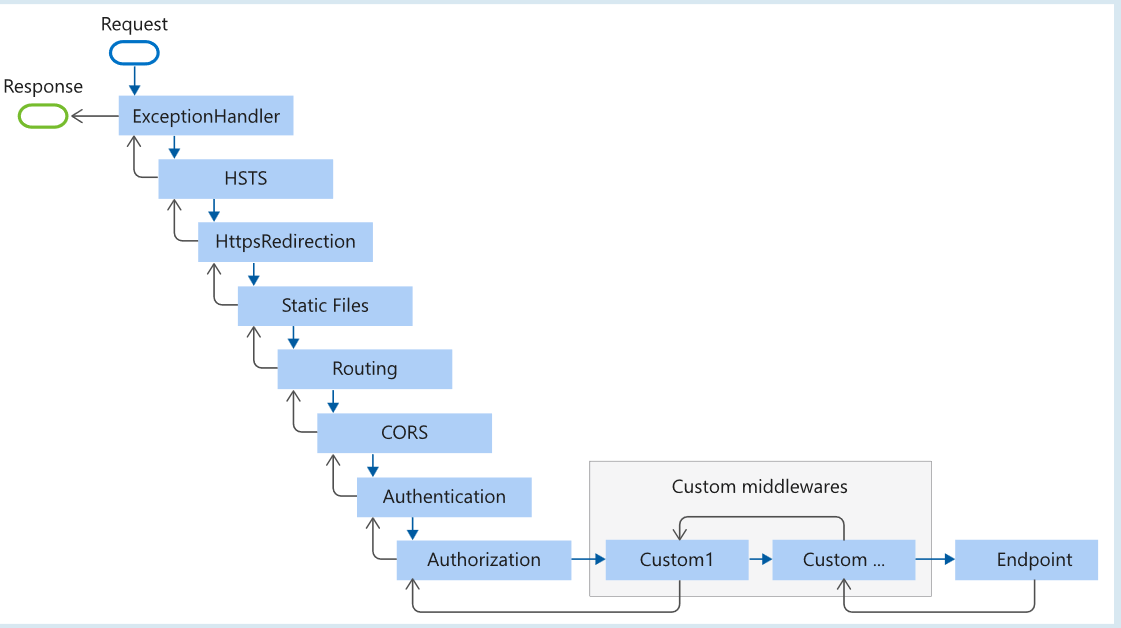
app.ExtensionMethodName();

Conventional Middleware

1. class MiddlewareClassName
2. {
3. private readonly RequestDelegate \_next;
5. public MiddlewareClassName(RequestDelegate next)
6. {
7. \_next = next;
8. }
10. public async Task InvokeAsync(HttpContext context)
11. {
12. //before logic
13. await \_next(context);
14. //after logic
15. }
16. });
17. static class ClassName
18. {
19. public static IApplicationBuilder ExtensionMethodName(this IApplicationBuilder app)
20. {
21. return app.UseMiddleware<MiddlewareClassName>();
22. }
23. }

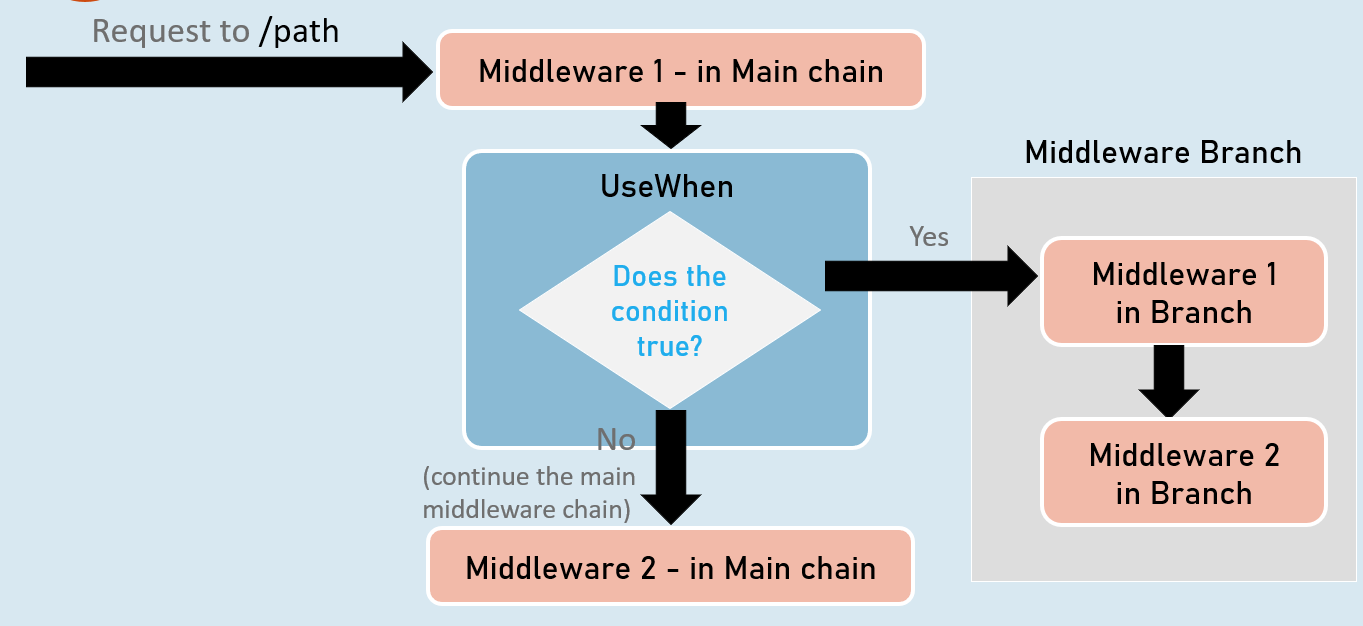
app.ExtensionMethodName();

The Right Order of Middleware



1. app.UseExceptionHandler("/Error");
2. app.UseHsts();
3. app.UseHttpsRedirection();
4. app.UseStaticFiles();
5. app.UseRouting();
6. app.UseCors();
7. app.UseAuthentication();
8. app.UseAuthorization();
9. app.UseSession();
10. app.MapControllers();
11. //add your custom middlewares
12. app.Run();

Middleware - UseWhen



**app.UseWhen( )**

1. app.UseWhen(
2. context => { return boolean; },
3. app =>
4. {
5. //add your middlewares
6. }
7. );

The extension method called “UseWhen” is used to execute a branch of middleware only when the specified condition is true.