# Middleware in ASP.NET Core Web API

## 📌 What is Middleware?

Middleware in ASP.NET Core Web API is a software component that processes HTTP requests and responses in a pipeline. Each middleware component in the pipeline:  
✅ Handles the request before passing it to the next middleware.  
✅ Processes the response before sending it back to the client.  
  
Middleware executes \*\*in the order it is added\*\* in `Program.cs`. If a middleware does not call the next middleware, it can \*\*short-circuit\*\* request processing.

## 📌 Built-in Middleware in ASP.NET Core

ASP.NET Core provides several built-in middleware components to handle security, authentication, logging, static files, routing, and more.

|  |  |  |
| --- | --- | --- |
| Middleware | Purpose | Order in Program.cs |
| Exception Handling Middleware | Catches unhandled exceptions and returns error responses. | Should be first. |
| HTTPS Redirection Middleware | Redirects all HTTP requests to HTTPS. | Before routing. |
| HSTS Middleware | Enforces HTTP Strict Transport Security (HSTS). | Registered early in production. |
| Static Files Middleware | Serves static files (HTML, CSS, JS). | Before routing. |
| Routing Middleware | Determines which controller/action should handle the request. | Before authentication & authorization. |
| CORS Middleware | Enables Cross-Origin Resource Sharing (CORS). | Before routing. |
| Authentication Middleware | Validates authentication tokens (JWT, cookies, etc.). | Before authorization. |
| Authorization Middleware | Ensures authenticated users have the required permissions. | After authentication. |
| Endpoint Middleware | Executes the selected controller action. | Last in the pipeline. |

## 📌 Middleware Execution Order in `Program.cs`

Middleware must be registered \*\*in the correct order\*\* in `Program.cs` for proper functionality.

var builder = WebApplication.CreateBuilder(args);  
  
var app = builder.Build();  
  
// 1️⃣ Exception Handling Middleware (Handles errors globally)  
app.UseMiddleware<ExceptionHandlingMiddleware>();   
  
// 2️⃣ HTTPS Redirection Middleware  
app.UseHttpsRedirection();  
  
// 3️⃣ HSTS Middleware (Only in production)  
if (!app.Environment.IsDevelopment())  
{  
 app.UseHsts();  
}  
  
// 4️⃣ Static Files Middleware (For serving static assets)  
app.UseStaticFiles();  
  
// 5️⃣ Routing Middleware (Determines request handling)  
app.UseRouting();  
  
// 6️⃣ CORS Middleware (For cross-origin requests)  
app.UseCors();  
  
// 7️⃣ Authentication Middleware (Validates authentication tokens)  
app.UseAuthentication();  
  
// 8️⃣ Authorization Middleware (Ensures user permissions)  
app.UseAuthorization();  
  
// 9️⃣ Endpoint Middleware (Executes the selected endpoint)  
app.MapControllers();  
  
app.Run();

## 📌 Why Does Middleware Order Matter?

Middleware order is crucial because ASP.NET Core processes requests in the sequence middleware are added in `Program.cs`. Incorrect ordering can lead to security risks, routing issues, or request failures.

✅ \*\*Exception Handling should be first\*\* – Ensures all unhandled exceptions are caught.  
✅ \*\*Routing should be before authentication\*\* – Requests must be routed before checking authentication.  
✅ \*\*Authentication should be before authorization\*\* – Users must be authenticated before checking permissions.  
✅ \*\*Static files should be before routing\*\* – Ensures requests for CSS/JS/images are served before trying API routes.

## 📌 Summary

✅ Middleware processes requests and responses in a pipeline.  
✅ ASP.NET Core provides several built-in middleware components.  
✅ Middleware must be registered in the correct order for proper functionality.  
✅ `Program.cs` defines the order in which middleware is executed.  
✅ Endpoint Middleware (`MapControllers()`) should be the last in the pipeline.

## 🚀 Next Steps

Would you like to:

1. Implement Custom Middleware for Logging Requests?  
2. Add Rate Limiting Middleware to Prevent Abuse?  
3. Explore Middleware for API Performance Optimization?