Middlewares

لو بتشتغل على تطوير الويب بـ ASP.NET Core، أكيد سمعت كلمة **Middleware** وشوية حسيت إنها معقدة. بس الحقيقة انها زي "الحراس" اللي بيمرروا الطلبات (Requests) من الـ Client للـ Server ويرجعوا الردود (Responses)، وبتقدر تتحكم في كل حاجة زي الأمان، الـ Logging، أو حتى الـ Routing.

يعني إيه Middleware؟

الـ **Middleware** هي مكونات صغيرة بتشتغل مع بعض في خط أنابيب (Pipeline) متسلسل. كل واحد فيهم بياخد الـ Request، يعمل عليه حاجة (زي التحقق من الصلاحيات أو تسجيل اللوج)، وبعدين يمرره للـ Middleware اللي بعده. في النهاية، الـ Response بيرجع عكسيًا من الخط نفسه.

في ASP.NET Core، الـ Middleware بتتميز بالمرونة وبتشتغل على مستوى الـ HTTP Request/Response كله. بتسمى "Middleware" لأنها في الوسط بين الـ Client والتطبيق الفعلي.

ليه بنستخدم Middleware؟

علشان الـ ASP.NET Core مصممة علشان تكون modular، يعني مش لازم تحط كل اللوجيك في Controller واحد. الـ Middleware بتسمحلك:

- فصل الاهتمامات (Separation of Concerns): كل Middleware مسؤول عن مهمة واحدة.
- إعادة استخدام الكود: تقدر تضيف Middleware جاهزة زي Authentication أو CORS.
- تحسين الأداء: بتتحكم في التدفق قبل ما يوصل للكود الرئيسي.

أنواع الـ Middleware الأساسية:

- **Built-in Middleware** (الجاهزة من Microsoft)

  - **UseAuthentication()**: بيتحقق من هوية المستخدم.
  - **UseAuthorization()**: بيحدد الصلاحيات.
  - **UseStaticFiles()**: بيخدم الملفات الثابتة زي CSS و JS.
  - **UseRouting()**: بيحدد الـ Routes.

- **Custom Middleware** (اللي بتكتبه بنفسك)

بتقدر تعمل واحدة خاصة بك، زي واحدة للـ Logging أو Rate Limiting.

- **Terminal Middleware**

زي UseEndpoints()، اللي بتوقف الـ Pipeline وترجع الـ Response (مثل الـ MVC Controllers).

إزاي تنفذ Middleware؟ أنواع التنفيذ المختلفة

في ASP.NET Core، في طرق مختلفة لتنفيذ الـ Middleware، حسب احتياجك. هقسمها لك على ثلاث طرق رئيسية: المضمن (Inline)، المخصص (Custom)، واللي بطريقة التمديد (Extension Method). كل واحدة ليها مميزاتها، وهشوف معاك أمثلة عملية.

- **Inline Middleware**

ده الطريقة الأبسط والمباشرة لتحديد الـ Middleware. مثالي للمنطق السريع، النماذج الأولية، أو معالجة الطلبات الخفيفة.

مثال: تسجيل الطلب والرد

```
public void Configure(IApplicationBuilder app)
{
    app.Use(async (context, next) =>
    {
        Console.WriteLine("Request: " + context.Request.Path);
        await next.Invoke();
        Console.WriteLine("Response: " +
context.Response.StatusCode);
    });

    app.Run(async context =>
    {
        await context.Response.WriteAsync("Hello from inline
middleware!");
    });
}
```

استخداماته:

- تسجيل سريع (Quick Logging).
- تشخيص الأخطاء (Diagnostics).
- تصفية طلبات بسيطة (Simple Request Filtering).

2. Custom Middleware

بتحدد فئة منفصلة تحتوي على المنطق، وتقدر تستخدمها عبر مشاريع متعددة.

مثال: Middleware للتسجيل

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

    public LoggingMiddleware(RequestDelegate next)
    {
        _next = next;
    }
```

```
    public async Task InvokeAsync(HttpContext context)
    {
        Console.WriteLine("Logging request: " +
context.Request.Path);
        await _next(context);
        Console.WriteLine("Logging response: " +
context.Response.StatusCode);
    }
}
```

تسجيلها في Startup.cs (أو Program.cs):

```
public void Configure(IApplicationBuilder app)
{
    app.UseMiddleware<LoggingMiddleware>();
}
```

استخداماته:

- منطق معقد (Complex Logic).
- مكونات قابلة لإعادة الاستخدام (Reusable Components).
- فصل أفضل للاهتمامات (Better Separation of Concerns).

3. Extension Method Middleware

الطريقة دي بتغلف منطق تسجيل الـ Middleware و بتخليها قابلة للإعادة استخدام، وده بيحافظ على نظافة Startup أو Program ويسهل قراءتها.

مثال: تمديد لـ Middleware التسجيل

```
public static class LoggingMiddlewareExtensions
{
    public static IApplicationBuilder UseLogging(this
IApplicationBuilder builder)
    {
```

```
        return builder.UseMiddleware<LoggingMiddleware>();
    }
}
```

استخدامها في Startup.cs:

```
public void Configure(IApplicationBuilder app)
{
    app.UseLogging();
}
```

استخداماته:

- تسجيل Middleware نظيف وسلس (Clean and Fluent Middleware Registration).
- استخدام متسق عبر المشاريع (Consistent Usage Across Projects).
- يبسط تكوين الخط الأنابيب (Simplifies Pipeline Configuration).

مثال توضيحي:

تخيل عندك تطبيق ويب، والـ Request بيجي من المتصفح فهيعدي بالترتيب علي:

1. **الأول**: UseHttpsRedirection() - بيتأكد إن الـ Request HTTPS، لو لا بيعيد توجيه.
2. **التاني**: UseStaticFiles() - لو الطلب ملف ثابت، بيخدمه ويخلص.
3. **التالت**: UseAuthentication() - بيتحقق لو المستخدم مسجل دخول.
4. **الرابع**: MapControllers() - بيوصل الطلب للـ Controller ويرجع الـ Response.

في الكود، بتضيفها في Program.cs (أو Startup.cs في الإصدارات القديمة):

```
var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();
```

```
app.UseHttpsRedirection();  // Middleware أولى
app.UseStaticFiles();       // Middleware تانية
app.UseAuthentication();    // Middleware تالتة
app.UseAuthorization();     // Middleware رابعة
app.MapControllers();       // Terminal Middleware

app.Run();
```

لو الـ Request مش HTTPS، هيوقف هنا ويعيد توجيه، ومش هيوصل للباقي. بس لو كله تمام، هيكمل الخط كله ويرجع الـ Response عكسيًا (مثل إضافة Headers في الـ Response).

طيب الناس بتتلخبط في إيه؟

- **ترتيب الـ Middleware**: الترتيب مهم جدًا! لو حطيت UseAuthentication بعد MapControllers مثلا، مش هيشتغل صح لأن الطلب هيتنفذ قبل التحقق.
- **Short-circuiting**: بعض الـ Middleware بتقف الـ Pipeline لو الشرط مش متحقق، فبتحتاج تفهم إزاي كل واحد بيتصرف.
- **الـ Exception Handling**: لو حصل خطأ في Middleware، استخدم UseExceptionHandler() في البداية عشان تلتقطها.

⚙️ Tips مهمة للتعامل مع Middleware:

✔️ **دائمًا رتبها صح**: Authentication قبل Authorization، و Static Files قبل Routing.

✔️ **اكتب Custom Middleware بسيط**: استخدم IApplicationBuilder و RequestDelegate:

```
public class LoggingMiddleware
{
```

```
    private readonly RequestDelegate _next;
    public LoggingMiddleware(RequestDelegate next) { _next = next; }
    public async Task InvokeAsync(HttpContext context)
    {
        Console.WriteLine("Request Started");
        await _next(context);  // مرر للـ Next
        Console.WriteLine("Request Ended");
    }
}
// في Program.cs: app.UseMiddleware<LoggingMiddleware>();
```

✔️ **استخدم Extensions**: اعمل extension method عشان تخلي الكود أنظف، زي app.UseMyCustomMiddleware().

✔️ **اختبر الـ Pipeline**: استخدم tools زي Postman عشان تشوف إزاي الطلب بيمرر.