ASP.NET Core中创建中间件的几种方式

原创 大姚 追逐时光者 2024年07月12日 07:00 广东

点击蓝字 设为星标 / 常备干货不走丢~

前言

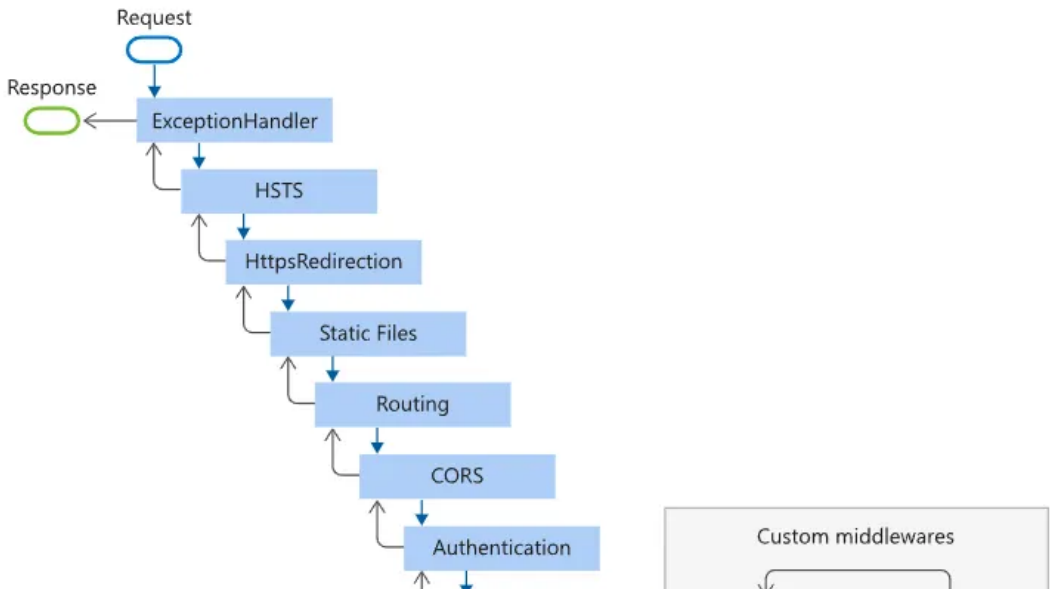
今天我们一起来盘点一下在ASP.NET Core应用程序中添加和创建中间件常见的四种方式。

中间件介绍

ASP.NET Core中间件（Middleware）是用于处理HTTP请求和响应的组件，它们被安排在请求处理管道中，并按顺序执行。中间件的设计是为了使其在请求处理管道中能够以灵活和可扩展的方式处理 HTTP 请求和响应。

下图显示了 ASP.NET Core MVC 和 Razor Pages 应用的完整请求处理管道：

> 了解现有中间件的顺序，以及在哪里添加自定义中间件。你可以完全控制如何重新排列现有中间件，或根据场景需要注入新的自定义中间件。



公众号 · 追逐时光者

中间件用途

开发者通过在请求处理管道中添加不同的中间件（Middleware）组件，可以实现应用程序的认证和授权、日志记录、异常处理、静态文件处理、路由和端点映射、CORS（跨域资源共享）、会话管理、请求压缩、国际化和本地化、缓存等各种功能。

通过请求委托添加中间件

我们可以通过在 WebApplication 实例上调用 Use 方法，并提供一个带有两个参数的 lambda 方法来实现。第一个参数是 HttpContext，第二个参数是管道中的实际下一个请求委托。

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

app.Use(async (context, next) =>
{
    // 在这里处理请求
    // ...

    await next.Invoke();

    // 在这里处理响应
    // ...
});

app.Run();
```

按约定添加中间件

ASP.NET Core中提供了许多内置中间件，例如静态文件中间件、路由、认证、授权中间件等。这些中间件通常已经预先定义好了，开发者只需按照约定调用相应的方法即可。

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

// 使用静态文件中间件
app.UseStaticFiles();

// 使用路由中间件
app.UseRouting();

// 使用认证中间件
app.UseAuthentication();

// 使用授权中间件
app.UseAuthorization();

app.Run();
```

创建自定义中间件类

创建自定义中间件类

首先我们创建一个自定义中间件类 `RequestLoggingMiddleware` ，它将记录每个请求的详细信息。

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

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

    public async Task InvokeAsync(HttpContext context)
    {
        // 记录请求信息
        Console.WriteLine($"Request Method: {context.Request.Method}");
        Console.WriteLine($"Request Path: {context.Request.Path}");
```

```
        // 调用下一个中间件
        await _next(context);

        // 记录响应状态码
        Console.WriteLine($"Response Status Code: {context.Response.StatusCode}");
    }
}
```

创建扩展方法

为了方便在应用程序中注册中间件，我们可以创建一个扩展方法。

```
public static class RequestLoggingMiddlewareExtensions
{
    public static IApplicationBuilder UseRequestLogging(this IApplicationBuilder builder)
    {
        return builder.UseMiddleware<RequestLoggingMiddleware>();
    }
}
```

在应用程序中使用自定义中间件

在 `Program.cs` 文件中，使用自定义中间件。

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

        // 使用自定义中间件
        app.UseRequestLogging();

        app.Run();
```

添加基于工厂的中间件

IMiddlewareFactory 是 ASP.NET Core 中用于创建和管理中间件实例的接口。它提供了一种灵活的方式来控制中间件的创建和生命周期管理，特别是在需要复杂依赖注入或条件实例化的场景中。

创建自定义中间件工厂

实现 IMiddlewareFactory 接口的自定义工厂类。

```
public class CustomMiddlewareFactory(IServiceProvider serviceProvider) : IMiddlewareFactory
{
    private readonly IServiceProvider _serviceProvider = serviceProvider;

    public IMiddleware? Create(Type middlewareType)
    {
        // 使用服务提供者创建中间件实例
        return _serviceProvider.GetService(middlewareType) as IMiddleware;
    }

    public void Release(IMiddleware middleware)
    {
        // 如果需要，可以在这里释放中间件实例（容器负责释放资源）
        (middleware as IDisposable)?.Dispose();
    }
}
```

创建自定义中间件

实现 IMiddleware 接口的自定义中间件类。

```
public class CustomMiddleware : IMiddleware
{
    public async Task InvokeAsync(HttpContext context, RequestDelegate next)
    {
        Console.WriteLine("在这里处理请求.......");

        await next(context);
```

```
            Console.WriteLine("在这里处理响应.......");
        }
    }
```

注册中间件和工厂

在 `Program.cs` 文件中注册自定义中间件和工厂。

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

// 注册中间件和工厂到依赖注入容器
builder.Services.AddTransient<CustomMiddleware>();
builder.Services.AddTransient<IMiddlewareFactory, CustomMiddlewareFactory>();

var app = builder.Build();

// 使用基于工厂的中间件
app.UseMiddleware<CustomMiddleware>();
app.Run();
```

最后总结

在ASP.NET Core中添加和创建中间件的方式有很多种，本文列举了四种常见的方式，具体取决于你的需求和偏好。每种方式都有其适用的场景，选择合适的方法可以使你的代码更加简洁和易于维护。

参考文章

- https://learn.microsoft.com/zh-cn/aspnet/core/fundamentals/middleware/?view=aspnetcore-8.0
- https://learn.microsoft.com/zh-cn/aspnet/core/fundamentals/middleware/extensibility?view=aspnetcore-8.0

- 免费开源的程序员简历模板
- 了解作者&获取更多学习资料
- 程序员常用的开发工具软件推荐
- 加入DotNetGuide技术社区交流群
- C#/.NET/.NET Core推荐学习书籍
- C#/.NET/.NET Core学习视频汇总
- .NET/.NET Core ORM框架资源汇总
- ASP.NET Core开发者学习指南路线图
- C#/.NET/.NET Core面试宝典（基础版）
- C#/.NET/.NET Core优秀项目和框架推荐
- C#/.NET/.NET Core学习、工作、面试指南

追逐时光者

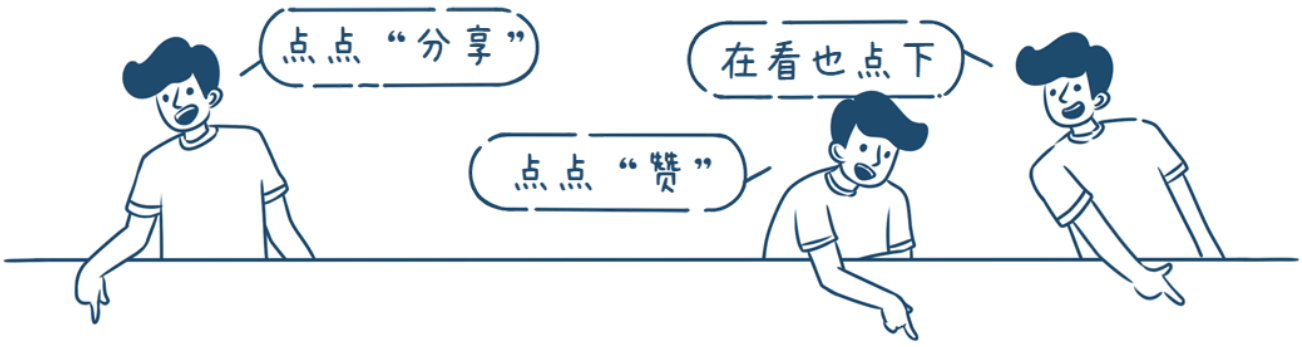
DotNetGuide官方公众号，微软MVP，专注于C#/.NET/.NET Core学习、工作、面试干...

506篇原创内容

公众号

学习是一个永无止境的过程，你知道的越多，你不知道的也会越多，在有限的时间内坚持每天多学一点，你一定能成为你想要成为的那个人。不积跬步无以至千里，不积小流无以成江河！！！

See you next good day



C# 267　.NET 335　.NET Core 205　拾遗补漏 37　中间件 1

C# · 目录

上一篇
2024年最新最全Visual Studio实用插件推荐！

下一篇
再也不用为找.NET相关的项目和框架发愁了

阅读原文