## **什么是中间件（**请求委托**）？**

中间件是一种装配到应用程序管道以处理请求和响应的软件。 每个组件：

* 选择是否将请求传递到管道中的下一个组件。
* 可在调用管道中的下一个组件前后执行工作。

## **使用 IApplicationBuilder 创建中间件管道**

ASP.NET Core 请求管道包含一系列相继调用的请求委托，如下图所示（执行过程遵循黑色箭头）：



每个委托均可在下一个委托前后执行操作。

此外，委托还可以决定不将请求传递给下一个委托， next 参数表示管道中的下一个委托。

public class Startup {

    public void Configure (IApplicationBuilder app) {

        // 添加一个中间件

        app.Use (async (context, next) => {

            // Do work that doesn't write to the Response.

            await next.Invoke ();

            // Do logging or other work that doesn't write to the Response.

        });

        // Run 是终点中间件

        app.Run (async context => {

            // 写入响应

            await context.Response.WriteAsync ("Hello from 2nd delegate.");

        });

    }

}

## **中间件排序**

中间件的添加顺序就是中间件的执行顺序

public void Configure(IApplicationBuilder app)

{

    app.UseExceptionHandler("/Home/Error");

app.UseStaticFiles();

app.UseAuthentication();

    app.UseMvcWithDefaultRoute();

}

在以上代码中，UseExceptionHandler 是添加到管道的第一个中间件组件，因此，该组件可捕获在后面的调用中发生的任何异常。

### **Use、Run 和 Map**

使用 Use、Run 和 Map 配置 HTTP 管道

Use 方法可使管道短路（即不调用 next 请求委托）

Run 添加一个中间件

Map 将请求路径映射到新的请求管道上面

[MapWhen](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.builder.mapwhenextensions) 基于给定谓词的结果将请求分发到新的请求管道上。

public class Startup

{

    private static void HandleMapTest1(IApplicationBuilder app)

    {

        app.Run(async context =>

        {

            await context.Response.WriteAsync("Map Test 1");

        });

    }

    private static void HandleBranch(IApplicationBuilder app)

    {

        app.Run(async context =>

        {

            var branchVer = context.Request.Query["branch"];

            await context.Response.WriteAsync($"Branch used = {branchVer}");

        });

}

    public void Configure(IApplicationBuilder app)

    {

        // 如果请求是 /map1，则执行 HandleMapTest1

        app.Map("/map1", HandleMapTest1);

        // 如果请求参数包含 "branch"，则执行 HandleBranch

        app.MapWhen(context => context.Request.Query.ContainsKey("branch"),

                               HandleBranch);

        app.Run(async context =>

        {

            await context.Response.WriteAsync("Hello from non-Map delegate. <p>");

        });

    }

}

Map 支持嵌套，例如：

app.Map ("/level1", level1App => {

    level1App.Map ("/level2a", level2AApp => {

        // "/level1/level2a"

        //...

    });

    level1App.Map ("/level2b", level2BApp => {

        // "/level1/level2b"

        //...

    });

});

## **像使用UseMvc一样添加中间件**

定义中间件类（注：中间件按照约定定义，并没有继承任何接口）：

public class RequestCultureMiddleware {

    private readonly RequestDelegate \_next;

    public RequestCultureMiddleware (RequestDelegate next) {

        \_next = next;

}

    public async Task InvokeAsync (HttpContext context) {

        ...

        await \_next (context);

    }

}

扩展 [IApplicationBuilder](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.builder.iapplicationbuilder) 的方法：

public static class RequestCultureMiddlewareExtensions

{

    public static IApplicationBuilder UseRequestCulture (

        this IApplicationBuilder builder)

    {

        return builder.UseMiddleware<RequestCultureMiddleware> ();

    }

}

调用扩展方法：

public void Configure (IApplicationBuilder app)

{

    app.UseRequestCulture ();

...

}

## **内置中间件**

ASP.NET Core 附带以下中间件组件，以及用于添加这些组件的顺序的说明：

| **中间件** | **描述** | **顺序** |
| --- | --- | --- |
| [身份验证](https://docs.microsoft.com/zh-cn/aspnet/core/security/authentication/identity?view=aspnetcore-2.1) | 提供身份验证支持。 | 在需要 HttpContext.User 之前。OAuth 回叫的终端。 |
| [CORS](https://docs.microsoft.com/zh-cn/aspnet/core/security/cors?view=aspnetcore-2.1) | 配置跨域资源共享。 | 在使用 CORS 的组件之前。 |
| [诊断](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/error-handling?view=aspnetcore-2.1) | 配置诊断。 | 在生成错误的组件之前。 |
| [转接头](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.builder.forwardedheadersextensions) | 将代理标头转发到当前请求。 | 在使用更新的字段（示例：架构、主机、客户端 IP、方法）的组件之前。 |
| [HTTP 方法重写](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.builder.httpmethodoverrideextensions) | 允许传入 POST 请求重写方法。 | 在使用已更新方法的组件之前。 |
| [HTTPS 重定向](https://docs.microsoft.com/zh-cn/aspnet/core/security/enforcing-ssl?view=aspnetcore-2.1" \l "require-https) | 将所有 HTTP 请求重定向到 HTTPS（ASP.NET Core 2.1 或更高版本）。 | 在使用 URL 的组件之前。 |
| [HTTP 严格传输安全性 (HSTS)](https://docs.microsoft.com/zh-cn/aspnet/core/security/enforcing-ssl?view=aspnetcore-2.1" \l "http-strict-transport-security-protocol-hsts) | 添加特殊响应标头的安全增强中间件（ASP.NET Core 2.1 或更高版本）。 | 在发送响应之前，修改请求的组件（例如转接头、URL 重写）之后。 |
| [响应缓存](https://docs.microsoft.com/zh-cn/aspnet/core/performance/caching/middleware?view=aspnetcore-2.1) | 提供对缓存响应的支持。 | 在需要缓存的组件之前。 |
| [响应压缩](https://docs.microsoft.com/zh-cn/aspnet/core/performance/response-compression?view=aspnetcore-2.1) | 提供对压缩响应的支持。 | 在需要压缩的组件之前。 |
| [请求本地化](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/localization?view=aspnetcore-2.1) | 提供本地化支持。 | 在对本地化敏感的组件之前。 |
| [路由](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/routing?view=aspnetcore-2.1) | 定义和约束请求路由。 | 用于匹配路由的终端。 |
| [会话](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/app-state?view=aspnetcore-2.1) | 提供对管理用户会话的支持。 | 在需要会话的组件之前。 |
| [静态文件](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/static-files?view=aspnetcore-2.1) | 为提供静态文件和目录浏览提供支持。 | 如果请求与文件匹配，则为终端。 |
| [URL 重写](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/url-rewriting?view=aspnetcore-2.1) | 提供对重写 URL 和重定向请求的支持。 | 在使用 URL 的组件之前。 |
| [WebSockets](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/websockets?view=aspnetcore-2.1) | 启用 WebSockets 协议。 | 在接受 WebSocket 请求所需的组件之前。 |