ASP.NET Core 基础系列（7）(请求管道)

原创 痕迹 CodeShare 3天前

收录于话题
#ASP.NET Core

8个

了解ASP.NET处理管道

为了理解ASP.NET Core中的请求处理管道概念，让我们修改Startup类的Configure()方法，如下所示。
在这里，我们将三个中间件组件注册到请求处理管道中。如您所见，前两个组件是使用Use() 扩展方法注册的，因此它们有机会在请求处理管道中调用下一个中间件组件。
最后一个使用Run() 扩展方法注册，因为它将成为我们的终止组件，即它将不会调用下一个组件。

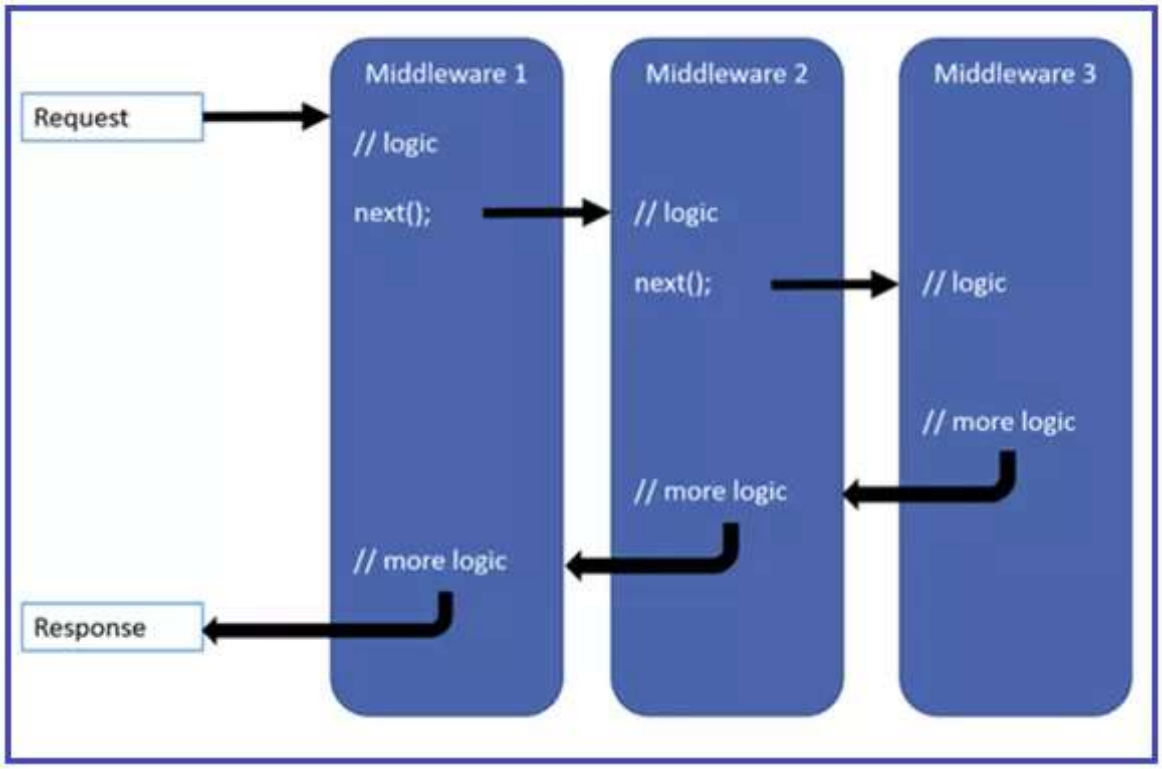
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.Use(async (context, next) =>
    {
        await context.Response.WriteAsync("Middleware1: Incoming Request\n");
        await next();
        await context.Response.WriteAsync("Middleware1: Outgoing Response\n");
    });

    app.Use(async (context, next) =>
    {
        await context.Response.WriteAsync("Middleware2: Incoming Request\n");
        await next();
        await context.Response.WriteAsync("Middleware2: Outgoing Response\n");
    });

    app.Run(async (context) =>
    {
        await context.Response.WriteAsync("Middleware3: Incoming Request handled and response generated\n");
    });
}
```

了解ASP.NET Core请求处理管道执行顺序

为了理解这一点，让我们将上面的输出与下图进行比较，以更简单的方式理解ASP.NET Core请求处理管道。



当传入的HTTP请求到达时，它首先由第一个中间件组件（即Middleware1）接收，该组件在响应流中记录“ Middleware1：传入请求”。因此，首先，我们首先在浏览器上看到此消息。

第一个中间件 记录了信息，然后它将调用next()方法，该方法将在请求处理管道中调用第二个中间件，即Middleware2。
第二个中间件 记录了“中间件2：传入请求”信息，因此我们在第一个日志之后看到了该日志信息。然后第二个中间件调用next()，它将在请求管道中调用第三个中间件Middleware3。
第三个中间件 处理请求，然后产生响应。因此，我们在浏览器中看到的第三个信息是“Middleware3：传入请求已处理并生成响应”。

该中间件组件是使用Run()扩展方法注册的，因此它是终端组件。因此，从这一点开始，请求管道开始反向。这意味着从该中间件将控制权交还给第二个中间件，第二个中间件将信息记录为“中间件2：外发响应”，然后将控制权交还给第一个中间件组件，第一个中间件组件记录信息就像我们在浏览器中看到的一样，是"Middleware1：外发响应"。

要记住的要点：
ASP.NET Core请求处理管道由一系列中间件组件组成，这些中间件组件将一个接一个地调用。
每个中间件组件都可以在使用next方法调用下一个组件之前和之后执行一些操作。中间件组件还可以决定不调用下一个中间件组件，这称为短路请求管道。
asp.net核心中的中间件组件可以访问传入请求和传出响应。
您需要牢记的最重要的一点是，在Startup类的Configure方法中添加中间件组件的顺序定义了将在请求时调用这些中间件组件的顺序以及对它们的相反顺序。响应。因此，顺序对于定义应用程序的安全性，性能和功能至关重要。