

ASP.NET Core 基础系列（6）（中间件）

原创 痕迹 CodeShare 4天前

收录于话题

#ASP.NET Core

8个

什么是ASP.NET Core Middleware?

ASP.NET Core中间件组件是被组装到应用程序管道中以处理HTTP请求和响应的软件组件（从技术上来说，组件只是C#类）。

ASP.NET Core应用程序中的每个中间件组件都执行以下任务。

- 选择是否将 HTTP 请求传递给管道中的下一个组件。这可以通过在中间件中调用下一个 `next()` 方法实现。
- 可以在管道中的下一个组件之前和之后执行工作。

在ASP.NET Core中，已经有很多内置的中间件组件可供使用，您可以直接使用它们。

如果需要，还可以在ASP.NET Core应用程序中创建自己的中间件组件。

您需要牢记的最重要的一点是，在ASP.NET Core中，给定的中间件组件应仅具有特定目的，即单一职责。

在ASP.NET Core应用程序中使用中间件组件的一些示例如下：

- 用于验证用户身份的中间件
- 中间件可用于记录请求和响应
- 用于处理错误的中间件
- 用于处理静态文件，例如图像，Javascript或CSS文件的中间件
- 用于在访问特定资源时授权用户的中间件

中间件组件是我们通常用于在ASP.NET Core应用程序中建立请求处理管道的组件。

如果您使用过.NET Framework的早期版本，那么您可能知道，我们使用HTTP处理程序和HTTP模块来设置请求处理管道。正是此管道将确定如何处理HTTP请求和响应。

如何在ASP.NET Core应用程序中配置中间件组件？

在ASP.NET Core应用程序中，我们需要在Startup.cs文件中存在的Startup类的 `Configure()` 方法内配置中间件组件。

这是在应用程序启动时将要运行的类。当我们使用空模板创建ASP.NET Core应用程序时，默认情况下，将使用 `Configure()` 方法创建Startup类，如下图所示。

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapGet("/", async context =>
        {
            await context.Response.WriteAsync("Hello World!");
        });
    });
}
```

因此，每当要在任何类型的.net核心应用程序中配置任何中间件组件时，都需要通过在IApplicationBuilder对象上调用Use方法在Startup类的Configure() 方法中对其进行配置。

如上图所示，configuration() 方法使用三个中间件组件设置了请求处理管道，如下所示。

- UseDeveloperExceptionPage()
- UseRouting()
- UseEndpoints()


在了解以上三个内置的中间件组件之前。首先让我们了解什么是中间件组件，以及这些中间件组件如何在ASP.NET Core应用程序中正常工作。

了解ASP.NET Core中的中间件组件

在ASP.NET Core应用程序中，中间件组件可以访问传入的HTTP请求和传出的HTTP响应。因此，ASP.NET Core中的中间件组件可以：

- 通过生成HTTP响应来处理传入的HTTP请求。
- 处理传入的HTTP请求，对其进行修改，然后将其传递给下一个中间件组件
- 处理传出的HTTP响应，进行修改，然后将其传递给下一个中间件组件或ASP.NET Core Web服务器。

为了更好地理解，请查看下图，该图显示了中间件组件如何在ASP.NET Core应用程序的请求处理管道中使用。



如上图所示，我们有一个日志记录中间件组件。该组件仅记录请求时间，然后将请求传递到下一个中间件组件，即请求管道中的静态文件中间件组件，以进行进一步处理。ASP.NET Core中的中间件组件也可以通过生成HTTP响应来处理HTTP请求。ASP.NET Core中间件组件也可能决定不调用请求管道中的下一个中间件组件。这个概念称为短路请求管道。

例如，我们有一个静态文件中间件组件。并且，如果传入的HTTP请求来自某些静态文件，例如图像，CSS文件，JavaScript等，则此Static Files Middleware组件可以处理请求，然后通过不调用管道中的下一个组件来缩短请求管道 即MVC中间件组件。正如上面讨论的，ASP.NET Core中间件组件可以访问管道中的HTTP请求和响应。因此，中间件组件也可以处理传出响应。例如，在我们的案例中，日志记录中间件组件可能会记录响应发送回客户端的时间。

ASP.NET Core应用程序中中间件组件的执行顺序是什么？

了解中间件组件的执行顺序非常重要。ASP.NET Core中间件组件的执行顺序与添加到管道中的顺序相同。因此，在将中间件组件添加到请求处理管道时，我们需要小心。根据应用程序的业务需求，您可以添加任意数量的中间件组件。

例如，如果您要开发具有某些静态HTML页面和图像的静态Web应用程序，则在请求处理管道中可能仅需要 “StaticFiles” 中间件组件。

但是，如果您正在开发安全的动态数据驱动的Web应用程序，则可能需要几个中间件组件，例如日志记录中间件，身份验证中间件，授权中间件，MVC中间件等。

什么是ASP.NET Core中的请求委托？

在ASP.NET Core中，请求委托用于构建请求管道，即请求委托用于处理每个传入的HTTP请求。在ASP.NET Core中，可以使用 “运行” ， “映射” 和 “使用” 扩展方法配置请求委托。

您可以使用嵌入式匿名方法（称为嵌入式中间件）指定请求委托，也可以使用可重用的类指定请求委托。

这些可重用的类和嵌入式匿名方法称为中间件或中间件组件。请求处理管道中的每个中间件组件负责调用管道中的下一个组件，或者通过不调用下一个中间件组件来使管道短路。

Use and Run方法的用途是什么？

在ASP.NET Core中，可以使用 “Use” 和 “Run” 扩展方法将内联中间件组件注册到请求处理管道中。

“Run” 扩展方法使我们可以添加终止中间件（不会在请求处理管道中调用下一个中间件组件的中间件）。

另一方面，“Use” 扩展方法使我们可以添加中间件组件，该中间件组件可以在请求处理管道中调用下一个中间件组件。

如果您观察 `Configure` 方法，那么您将看到它获得了 `ApplicationBuilder` 接口的一个实例，并将该实例与诸如 `Use` and `Run` 之类的扩展方法一起使用，它将配置中间件组件。

如您所见，在 `Configure` 方法中，使用 `ApplicationBuilder` 实例即 `app` 在请求处理管道中注册了三个中间件组件。他们如下：

- `UseDeveloperExceptionPage()`
- `UseRouting()`
- `UseEndpoints()`

UseDeveloperExceptionPage

如您所见，在 `Configure` 方法中，`UseDeveloperExceptionPage()` 中间件组件已注册到管道中，并且仅在将托管环境设置为 “Development” 时，该中间件组件才会出现。

当应用程序中发生未处理的异常时，该中间件组件将执行，并且由于它处于开发模式，因此它将向您显示代码的错误信息。您也可以考虑将其替换为其它内容。

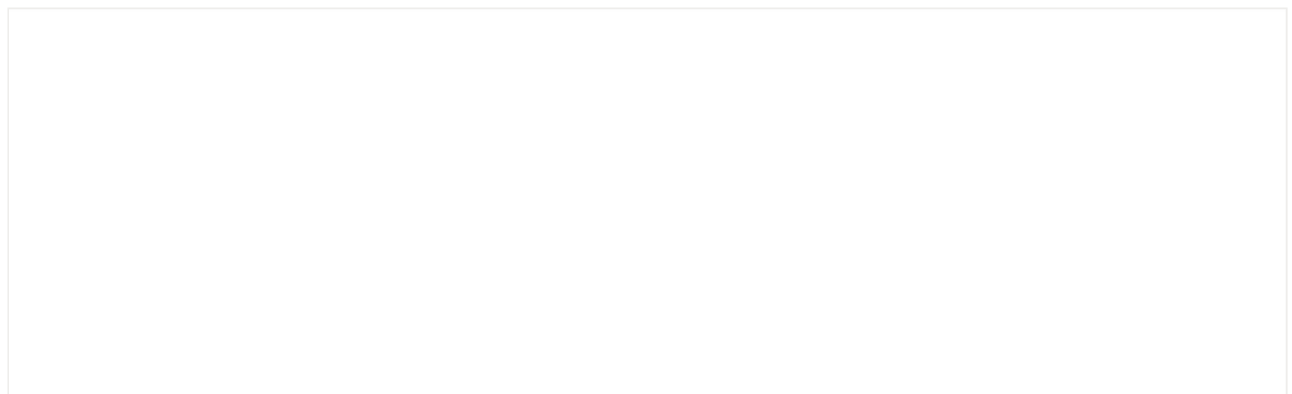
UseRouting

该中间件组件用于将端点路由中间件添加到请求处理管道，即它将URL（或传入的HTTP请求）映射到特定资源。

UseEndpoints

在此中间件中，将使用 `Map` 扩展方法来做出路由决策。以下是 `UseEndpoints` 中间件组件的默认实现。在 `MapGet` 扩展方法中，我们指定了URL模式，例如 “/”。这仅表示域名。

因此，只有域名的任何请求都将由该中间件处理。



除了 `MapGet`，您还可以使用 `Map` 方法，如下所示。

MapGet和Map方法有什么区别？

MapGet方法将处理GET HTTP请求，而Map方法将处理所有类型的HTTP请求，例如GET，POST，PUT和DELETE等。

如何使用Run() 扩展方法配置中间件组件？

首先我们需要了解如何使用“Run”扩展方法创建和配置自定义中间件组件。首先，注释一下Configure方法中存在的所有代码。

注释现有代码后，将以下代码复制并粘贴到Configure方法中。以下代码只是向应用程序的请求管道中添加了一个新的中间件组件，并仅打印了一条消息"My Name is Zhangsan"。

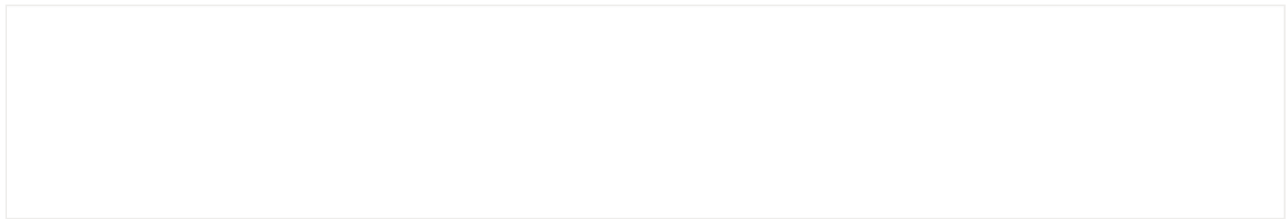
运行后,输出:

我们正在IApplicationBuilder实例（应用程序）上调用Run() 扩展方法，以将中间件组件注册到请求处理管道中。以下是Run方法的定义：

从Run() 方法的定义中可以看到，该方法被实现为IApplicationBuilder接口的扩展方法。这就是为什么我们能够使用IApplicationBuilder实例即app调用Run() 方法的原

因。

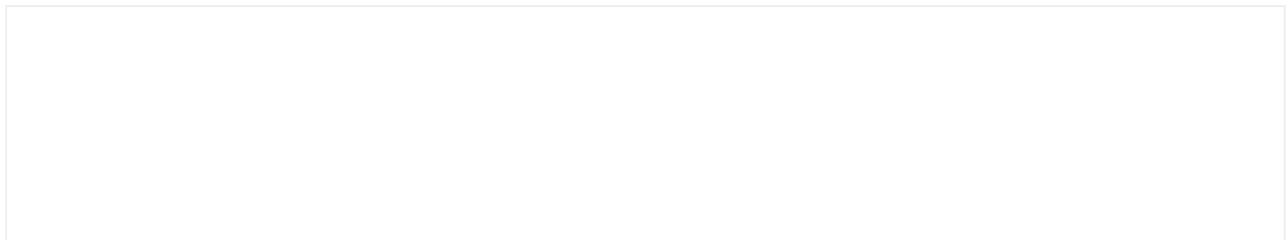
您还可以从上图看到Run() 方法采用RequestDelegate类型的输入参数。以下是RequestDelegate的定义。



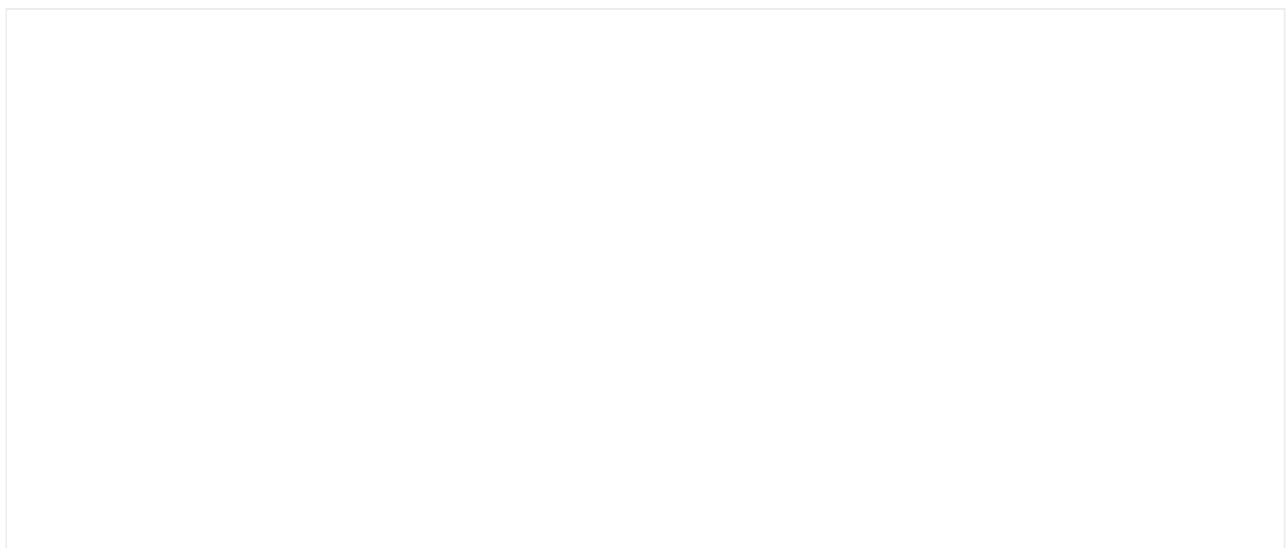
从上图可以看到，RequestDelegate是一个采用HttpContext对象类型的输入参数的委托。

正如我们上面讨论的那样，ASP.NET Core应用程序中的中间件组件可以访问HTTP请求和响应，这是因为上面的HttpContext对象。

在示例中，我们使用lambda表达式将请求委托内联作为匿名方法传递给内联，此外，我们还将HttpContext对象作为输入参数传递给请求委托。下图显示了以上内容：



向该应用程序再添加一个中间件。



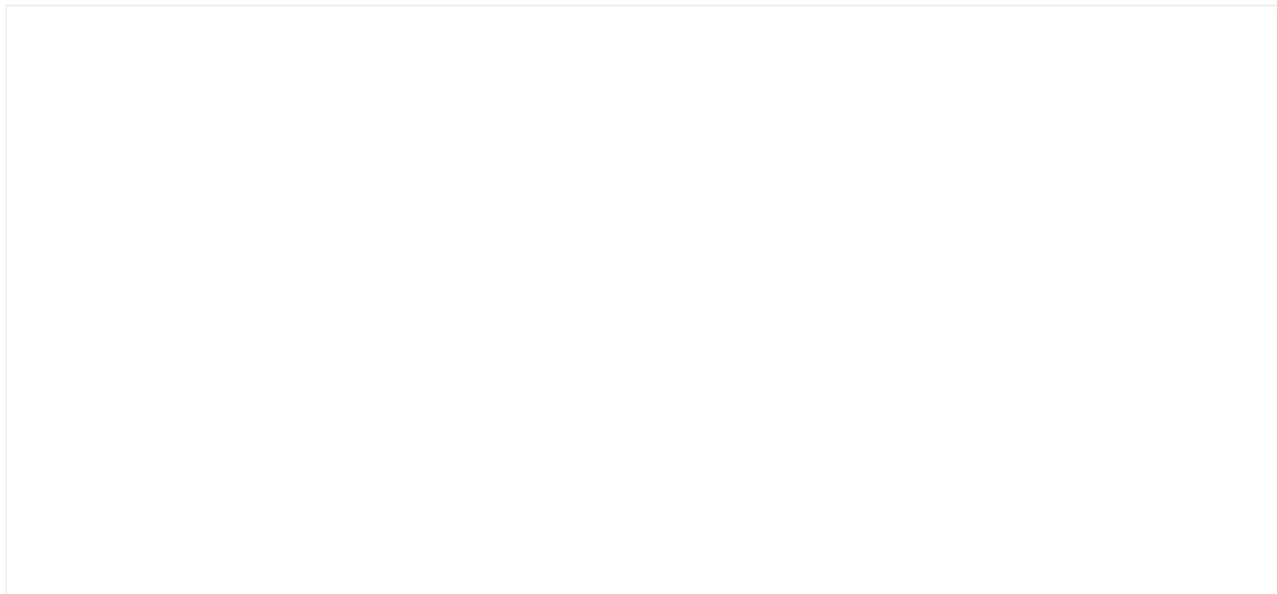
运行该应用程序，则将获得以下输出：

```
1 My Name is Zhangsan
```

输出来自第一个中间件组件。原因是，当我们使用Run() 扩展方法注册中间件组件时，该组件成为 终端组件，这意味着它不会在请求处理管道中调用下一个中间件组件。

使用Use扩展方法配置中间件组件

现在想到的问题是如何在请求处理管道中调用下一个组件，答案是使用Use扩展方法注册中间件组件，如下所示。



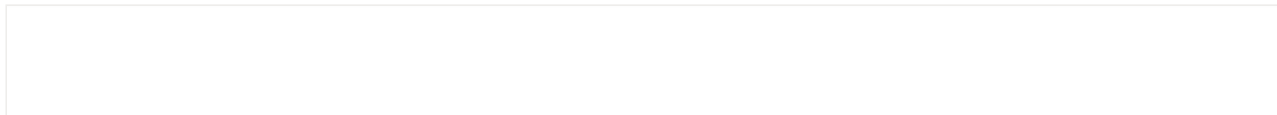
现在运行该应用程序，您将看到来自两个中间件组件的预期输出：

```
1 My Name is ZhangsanMy Name is LiSi
```

了解Use扩展方法

Use扩展方法将在行中定义的中间件委托添加到应用程序的请求管道中。

以下是Use扩展方法的定义：



此方法也实现为IApplicationBuilder接口上的扩展方法。这就是为什么我们能够使用IApplicationBuilder实例调用此方法的原因。从上面的定义可以看出，该方法采用两个输入参数。第一个参数是HttpContext上下文对象，通过它可以访问HTTP请求和响应。第二个参数是Func类型，即它是一个通用委托，可以处理请求或调用请求管道中的下一个中间件组件。

注意： 如果要请求从一个中间件发送到下一个中间件，则需要调用下一个方法。

[阅读原文](#) 阅读 81

分享

收藏

赞

在看 1