## Startup类

* 可选择性地包括ConfigureServices方法以配置应用服务。服务是一个提供应用功能的可重用组件。在ConfigureServices中注册服务，并通过[依赖关系注入 (DI)](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-3.1) 或 [ApplicationServices](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.builder.iapplicationbuilder.applicationservices) 在整个应用中使用服务。
* 包括 [Configure](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.hosting.startupbase.configure) 方法以创建应用的请求处理管道。

### 使用Startup筛选器扩展Startup

## 选项

接口分隔原则（ISP）或封装：依赖于配置设置的方案（类）仅依赖于其使用的配置设置。

关注点分离：应用不同部件的设置不彼此依赖或相互耦合

## Netcore终结点 endpoind

可执行：具有 [RequestDelegate](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.http.endpoint.requestdelegate" \l "Microsoft_AspNetCore_Http_Endpoint_RequestDelegate)。（执行http请求委托）

可扩展：具有[元数据](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.http.endpoint.metadata)集合。

Selectable:可选择性包含[路由信息](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.routing.routeendpoint.routepattern)。

可枚举：可通过从 [DI](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-3.1) 中检索 [EndpointDataSource](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.routing.endpointdatasource) 来列出终结点集合

## 中间件

使用自定义的中间件

### 定义类

public class RequestCultureMiddleware

{

private readonly RequestDelegate \_next;

public RequestCultureMiddleware(RequestDelegate next)

{

\_next = next;

}

public async Task InvokeAsync(HttpContext context)

{

var cultrueQuery = context.Request.Query["cultrue"];

if (!string.IsNullOrWhiteSpace(cultrueQuery))

{

var culture = new CultureInfo(cultrueQuery);

CultureInfo.CurrentCulture = culture;

CultureInfo.CurrentUICulture = culture;

}

await \_next(context);

}

}

类必须实现 InvokeAsync或 Invoke 必须包含RequestDelegate next

### 2.定义扩展方法

public static class RequestCultureMiddlewareExtensions

{

public static IApplicationBuilder UseRequestCulture(this IApplicationBuilder builder)

{

return builder.UseMiddleware<RequestCultureMiddleware>();

}

}

### 调用

app.UseRequestCulture();

## 管道

请求正文和响应正文有两个抽象元素：[Stream](https://docs.microsoft.com/zh-cn/dotnet/api/system.io.stream) 和 [Pipe](https://docs.microsoft.com/zh-cn/dotnet/api/system.io.pipelines.pipe)。 对于请求读取，[HttpRequest.Body](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.http.httprequest.body#Microsoft_AspNetCore_Http_HttpRequest_Body) 为 [Stream](https://docs.microsoft.com/zh-cn/dotnet/api/system.io.stream)，HttpRequest.BodyReader 为 [PipeReader](https://docs.microsoft.com/zh-cn/dotnet/api/system.io.pipelines.pipereader)。 对于响应写入，[HttpResponse.Body](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.http.httpresponse.body#Microsoft_AspNetCore_Http_HttpResponse_Body) 为 [Stream](https://docs.microsoft.com/zh-cn/dotnet/api/system.io.stream)，HttpResponse.BodyWriter 为 [PipeWriter](https://docs.microsoft.com/zh-cn/dotnet/api/system.io.pipelines.pipewriter)。

建议使用[管道](https://docs.microsoft.com/zh-cn/dotnet/standard/io/pipelines)替代流。 在一些简单操作中，使用流会比较简单，但管道具有性能优势，并且在大多数场景中更易于使用。