参考文章：https://www.cnblogs.com/artech/p/inside-asp-net-core-1.html

面试有些公司总是爱问”你是如何理解 .net core 的”，”你说下.net core吧”,这让我不知从何说起，索性就整理一份.net core的启动原理，虽然再开发中很少用到，但面试过程还是很有用的

**WebHostBuilder生成并启动站点**

**站点启动示例**

public static void Main (string[] args) {

    // WebHost生成器

    WebHostBuilder webHostBuilder = new WebHostBuilder ();

    // 配置Kestrel服务器

    webHostBuilder.UseKestrel ();

    // 配置中间件

    webHostBuilder.Configure(

        (IApplicationBuilder app) => app.Run(async context => await context.Response.WriteAsync("Hello World")));

    // 生成WebHost

    IWebHost webHost = webHostBuilder.Build ();

    // 运行WebHost

    webHost.Run ();

}

**.net core的站点由 WebHostBuilder 生成**

WebHostBuilder 具有ConfigureServices 方法，ConfigureServices 的参数是一个Action，在站点生成时才会调用，用于向依赖注入容器IServiceCollection注册服务，

**UseKestrel 指明内置服务器**

WebHostBuilder的UseKestrel 指明使用内置的Kestrel服务器作为站点服务器，UseKestrel实际上调用ConfigureServices 向我们的依赖注入容器注册KestrelServer服务器类

services.AddSingleton<IServer, KestrelServer> ();

**UseUrls 方法指明站点地址**

WebHostBuilder的UseUrls 方法指明站点地址，如果步指明，默认地址为localhost:5000，UseUrls 实际上将我们传入的地址保存到WebHostBuilder类的键值对中，KestrelServer服务器启动时，会从它那里获取地址

**配置中间件**

接下来我们可以配置中间件，我们可以使用WebHostBuilder的Configure配置中间件，或者我们使用UseStartup指明Startup类，然后再Startup类里进行配置

UseStartup实际上调用ConfigureServices是向依赖注入容器注册Startup

services.AddSingleton (typeof (IStartup), startupType);

**Build 生成WebHost**

接着通过WebHostBuilder的Build 生成WebHost

Build生成我们的IServiceCollection

然后调用我们前面通过ConfigureServices 注册的Action，IServiceCollection作为Action的参数

然后通过IServiceCollection获取IStartup的实例，然后调用IStartup的ConfigureServices，然后利用ConfigureServices返回的IServiceProvider作为站点的依赖注入容器（注：如果我们编写的Startup的ConfigureServices返回是void，Build会利用IServiceCollection生成IServiceProvider）

**run方法启动站点**

运行WebHost的run方法启动站点

run方法新建了一个IApplicationBuilder传给你IStartup实例的Configure方法注册中间件

调用IApplicationBuilder的build生成RequestDelegate（封装了所有中间件的处理流程）

新建一个IHttpApplication（代表一个应用）并将RequestDelegate传给它

从依赖注入容器获取IServer，将IHttpApplication作为参数调用其StartAsync方法启动服务器

**KestrelServer接收和处理Http请求**

KestrelServer监听我们对应的地址和端口，当有一个请求到来时，KestrelServer执行如下的处理流程

**将http请求封装成IFeatureCollection**

KestrelServer会将http请求封装成IFeatureCollection，IFeatureCollection为特征列表

**创建上下文**

KestrelServer将IFeatureCollection作为参数调用IHttpApplication的CreateContext创建上下文，CreateContext通过IHttpContextFactory生成Http上下文

**调用处理中间件**

KestrelServer将Http上下文作为参数调用IHttpApplication的ProcessRequestAsync，ProcessRequestAsync会调用我们封装的中间件

**销毁上下文**

中间件返回后，KestrelServer调用IHttpApplication的DisposeContext销毁上下文