Asp.net core的依赖注入：我们向IServiceCollection服务集合注册我们的类，然后在控制器的构造方法中直接以该类为参数，而不用手动实例

**依赖注入基本使用**

服务注册：

public void ConfigureServices(IServiceCollection services)

{

    services.AddScoped<IMyDependency, MyDependency>();

    ...

}

使用（称为依赖注入）

public  class  MyController :  Controller  {

    // 通过构造函数注入依赖

    public  MyController (IMyDependency  myDependency)  {

        var my =  myDependency;

    }

    public IActionResult MyAction () {

        // 通过 HttpContext 获取依赖

        var myDependency = (IMyDependency ) HttpContext.RequestServices.GetService(typeof(IMyDependency ));

        return View ();

    }

}

## **框架提供的服务**

在ConfigureServices中，那些以Add开头的方法，都是在向IServiceCollection注入服务

public void ConfigureServices(IServiceCollection services)

{

    // 注册数据库上下文

    services.AddDbContext<ApplicationDbContext>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

    // 注册 MVC 所需的服务

    services.AddMvc();

}

## **依赖注入容器替换**

如果内置的依赖注入容器无法满足需求，可以在 Startup.ConfigureServices 方法中返回新的依赖注入容器。

public IServiceProvider ConfigureServices(IServiceCollection services){

    services.AddMvc();

    ...

    // 返回 Autofac 依赖注入容器

    return new AutofacServiceProvider(container);

}

要使用第三方容器，Startup.ConfigureServices 必须返回 IServiceProvider。

**Startup中使用依赖注入**

在Startup构造函数中我们可以使用依赖注入

public class Startup {

    public Startup (IConfiguration configuration) {

        Configuration = configuration;

    }

    ...

}

**Asp.net core以注册的服务**

在运行我们的Startup之前，Asp.net core已经注入了如下几个服务

IApplicationBuilder：中间件生成器，利用其注册中间件

ILoggerFatory：日志工厂

IHostingEnvironment：宿主环境信息

IConfiguration：根配置信息，包含有appsettings.json文件生成的配置