本篇文章需要对依赖注入概念有了解

**.NET官方提供的依赖注入**

微软老大有提供一个依赖注入的实现，需要安装如下包

Microsoft.Extensions.DependencyInjection

1. 依赖服务注册

如下代码示例如何注册依赖服务，看注释

// Microsoft.Extensions.DependencyInjection 提供的依赖注入

IServiceCollection services = new ServiceCollection()

    // 添加依赖的4种方式

    .AddSingleton<TestClass>()

    .AddSingleton<ITestClass, TestClass>()

    .AddSingleton<ITestClass>(new TestClass())

    .AddSingleton<ITestClass>((IServiceProvider \_) => new TestClass())

    // 添加单例服务

    .AddSingleton<TestClass>()

    // 添加域服务

    .AddScoped<TestClass2>()

    // 添加临时服务

    .AddTransient<TestClass3>();

1. 解析服务

如下代码示例获取服务，看注释

// 生成服务提供器（根IServiceProvider）

IServiceProvider serviceProvider = services.BuildServiceProvider();

// 获取注册的服务

TestClass testClass = serviceProvider.GetService<TestClass>();

// 生成域，并获服务（scope.ServiceProvider为域IServiceProvider）

IServiceScope scope = serviceProvider.CreateScope();

TestClass2 ts2 = scope.ServiceProvider.GetService<TestClass2>();

**服务生命周期**

依赖注入提供了3种生命周期

1. 单例服务

生成的服务放在根IServiceProvider中，根IServiceProvider释放后服务才会释放

1. 域服务

生成的服务放在调用GetService的IServiceProvider中，IServiceProvider释放后服务才会释放

1. 临时服务

生成的服务放在调用GetService的IServiceProvider中，每次调用GetService都会生成新的服务

1. 代码示例生命周期

// True，单例服务保存在根 IServiceProvider 中，所有域 IServiceProvider 获取的服务从根获取

scope1Provider.GetService<TestClass>() == scope2Provider.GetService<TestClass>();

// True，域服务保存在域 IServiceProvider 中

scope1Provider.GetService<TestClass2>() == scope1Provider.GetService<TestClass2>();

// False，不同域的域服务是不同的

scope1Provider.GetService<TestClass2>() == scope2Provider.GetService<TestClass2>();

// False，临时服务每次调用 GetService 时都会生成新的

scope1Provider.GetService<TestClass3>() == scope2Provider.GetService<TestClass3>();