ABP的依赖注入系统是基于Microsoft的依赖注入扩展库开发的

**默认注册的类**

以下类会自动注册，无需我们手动注册

* 模块类注册为singleton.
* MVC控制器（继承Controller或AbpController）被注册为transient.
* MVC页面模型（继承PageModel或AbpPageModel）被注册为transient.
* MVC视图组件（继承ViewComponent或AbpViewComponent）被注册为transient.
* 应用程序服务（实现IApplicationService接口或继承ApplicationService类）注册为transient.
* 存储库（实现IRepository接口）注册为transient.
* 域服务（实现IDomainService接口）注册为transient.

还有继承自下面接口的类也会被自动注册

* ITransientDependency 注册为transient生命周期.
* ISingletonDependency 注册为singleton生命周期.
* IScopedDependency 注册为scoped生命周期.

还有通过Dependency特性标识的类也会被自动注册

[Dependency(ServiceLifetime.Transient, ReplaceServices = true)]

public class TaxCalculator

{

}

**手动注册**

我们可以在Module的ConfigureServices方法注册

public class BlogModule : AbpModule

{

    public override void ConfigureServices(ServiceConfigurationContext context)

    {

        //注册一个singleton实例

        context.Services.AddSingleton<TaxCalculator>(new TaxCalculator(taxRatio: 0.18));

        //注册一个从IServiceProvider解析得来的工厂方法

        context.Services.AddScoped<ITaxCalculator>(sp => sp.GetRequiredService<TaxCalculator>());

    }

}

**构造方法注入**

我们可以以在构造函数注入已注册的类

public class TaxAppService : ApplicationService

{

    private readonly ITaxCalculator \_taxCalculator;

    public TaxAppService(ITaxCalculator taxCalculator)

    {

        \_taxCalculator = taxCalculator;

    }

    public void DoSomething()

    {

        //...使用 \_taxCalculator...

    }

}

**属性注入**

public的属性会被自动注入服务类

public class MyService : ITransientDependency

{

    public ILogger<MyService> Logger { get; set; }

    public MyService()

    {

        Logger = NullLogger<MyService>.Instance;

    }

}

**从IServiceProvider解析服务**

我们还可以从IServiceProvider获取已注册的类

public class MyService : ITransientDependency

{

    private readonly IServiceProvider \_serviceProvider;

    public MyService(IServiceProvider serviceProvider)

    {

        \_serviceProvider = serviceProvider;

    }

    public void DoSomething()

    {

        var taxCalculator = \_serviceProvider.GetService<ITaxCalculator>();

        //...

    }

}

**关于手动释放注入的服务**

如你有一个后台服务，你想要使用依赖注入，则你应该按照如下方法去使用，

using (var scope = \_serviceProvider.CreateScope())

{

    var service1 = scope.ServiceProvider.GetService<IMyService1>();

    var service2 = scope.ServiceProvider.GetService<IMyService2>();

}