**注册**

**手动注册**

ABP使用Castle Windsor作为依赖注入框架。它是一个最成熟的DI框架。

用依赖注入框架时，你先要注册你的接口/类到依赖注入框架里，一般在你的[模块定义](http://www.cnblogs.com/kid1412/p/5973274.html" \t "http://www.cnblogs.com/kid1412/p/_blank)类的预初始化里注册

// 基本注册

IocManager.Register<IMyService, MyService>(DependencyLifeStyle.Transient);

// 泛型注册，IRepositoryEx<> 接口使用 RepositoryEx<> 实例

IocManager.IocContainer.Register(

    Component.For(typeof(IRepositoryEx<>)).ImplementedBy(typeof(RepositoryEx<>)).LifestyleTransient(),

    Component.For(typeof(IRepositoryEx<,>)).ImplementedBy(typeof(RepositoryEx<,>)).LifestyleTransient()

);

// 单例注册

IocManager.Instance.IocContainer.Register(

Component.For(typeof(DbContextOptions<IEManageSystemDbContext>)).Instance(dbOptionsBuilder.Options).LifestyleSingleton()

);

**自动注册**

ABP会按照约定自动注册所有仓储、领域服务、应用服务、Mvc控制器和Web Api控制器。

public interface IPersonAppService : IApplicationService

{

}

// 注意其命名

public class PersonAppService : IPersonAppService

{

}

ABP会自动注册它，因为它实现了IApplicationService接口（空的接口）。

**注册整个程序集**

IocManager.RegisterAssemblyByConvention(Assembly.GetExecutingAssembly());

**解析**

**构造函数或public属性注入**

最佳实践：你可以使用构造器和属性注入为你的类获取所需的依赖。

public CouponAppService couponAppService { get; set; }

如果一个类的属性时public，则abp会自动注入该类

**IocManager**

你可能需要用直接解析你的依赖来代替构造器和属性注入。这应该尽量避免，但有时却又无可避免。

CouponAppService couponAppService = IocManager.Instance.Resolve<CouponAppService>();