# 引言

## 目的

指导开发人员正确地使用框架。

## 背景

Core.Framework作为项目的基础支撑，为上层具体应用提供实现技术的支持，并定义好了基本的框架使用规范，使上层应用能关注于业务，同时避免多项目之间类似业务逻辑的重复劳动。

# 框架使用规定

## 正确的引用方式

**对于采用Core.Infrastructure.Impl.XXX等方式命名的程序集**，Application Layer与 Domain Layer是不能引用的，因为此类程序集，都表明了自身是使用特定技术完成了对接口的实现，一旦直接引用将造成程序集引用的感染，后期维护成本较高。

**Core.Infrastructure.Crosscutting这个程序集**，是可以在任何层次上直接引用的，他定义了框架基础结构的一系列接口，要获取接口的实例对象，采用Container来进行获取。

**框架以及上层应用都采用上引用下的引用方式**，禁止逆向引用。即DistributedServices可以引用Application、Domain、Infrastructure，Application可以引用Domain、Infrastructure，Domain可以引用Infrastructure，禁止Infrastructure引用Domain这样的操作。

## 采用正确的方式加载实例对象

为了降低各层之间的耦合，框架限定了Infrastructure层的具体实现类型，必须采用IOC进行注入，上层只需要依赖接口就能完成需求， 所以IOC容器的配置也需要采用一定规范。

详细的系统加载代码，请查阅Core.Framework.Test内的TestsInitialize.cs，里面有完整的示范代码与注释。

IOC的配置采用代码映射的方案，也是为了降低维护工作量，因为底层经常更换的可能性较小，而采用配置文件进行配置，可读性较差，还不能在编译时排除一些明显错误，所以采用代码映射的方案。

## 底层配置，唯一的例外

要让系统运行起来，还需要在底层进行数据库映射配置，以及DTO的映射配置，这两个配置程序集，是唯一违反了框架上引用下的引用规范，其实这两个配置程序集，放置到分布式服务层也是可以的，但是由于其自身依赖了具体实现技术，而且职责单一，引用范围也比较单一，所以放在Infrastructure也是可以接受的，但是一定要注意保证程序集内的职责单一，只完成映射配置工作，不做任何多余工作。

详细代码请参考：Core.Infrastructure.Impl.Test.DbConfig 与 Core.Infrastructure.Impl.Test.DtoAdapter。