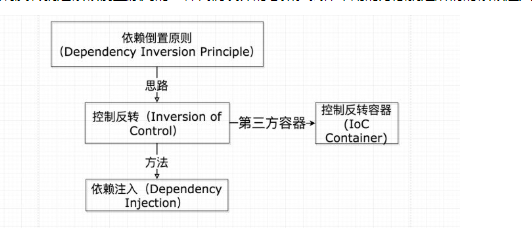
IOC 控制反转

依赖倒置原则



所谓依赖注入，就是把底层类作为参数传入上层类，实现上层类对下层类的“控制”。

AOP 面向切面编程：运行时织如—>代理对象

静态代理：由程序员创建或特定工具自动生成源代码，再对其编译。在程序运行前，代理类的.class文件就已经存在了。

动态代理：在程序运行时，运用反射机制动态创建而成。

　　静态代理的每一个代理类只能为一个接口服务，这样一来程序开发中必然会产生过多的代理，而且，所有的代理操作除了调用的方法不一样之外，其他的操作都一样，则此时肯定是重复代码。解决这一问题最好的做法是可以通过一个代理类完成全部的代理功能，那么此时就必须使用动态代理完成。

　　动态代理与静态代理对照的是动态代理类，动态代理类的字节码在程序运行时由Java反射机制动态生成，无需程序员手工编写它的源代码。动态代理不仅简化了编程工作，而且提高了软件系统的可扩展性，因为Java反射机制可以生成任意类型的动态代理类。java.lang.reflect包中的Proxy类和InvocationHandler接口提供了生成动态代理类的能力。

Spring AOP使用动态代理技术在运行期间织入增强的代码，主要有两种代理机制：基于JDK的动态代理；基于cglib的动态代理。JDK本身只提供接口的代理，而不支持类的代理。

CGLib采用非常底层的字节码技术，可以为一个类创建子类，并在子类中采用方法去技术拦截所有的父类方法的调用，并顺势织入横切逻辑。

Spring两种代理方式

　　若目标对象实现了接口，spring默认使用JDK的动态代理。

　　优点：因为有接口，所以使系统更加松耦合；

　　缺点：为每一个目标类创建接口；

　　若目标对象没有实现任何接口，spring使用CGLib进行动态代理。

　　优点：因为代理类与目标类是继承关系，所以不需要有接口的存在。

　　缺点：因为没有使用接口，所以系统的耦合性没有使用JDK的动态代理好。

　　若目标对象实现了接口，但是强制cglib代理，则使用cglib代理