**Java的三种代理模式**

**1.代理模式**

代理(Proxy)是一种设计模式,提供了对目标对象另外的访问方式;即通过代理对象访问目标对象.这样做的好处是:可以在目标对象实现的基础上,增强额外的功能操作,即扩展目标对象的功能.

这里使用到编程中的一个思想:不要随意去修改别人已经写好的代码或者方法,如果需改修改,可以通过代理的方式来扩展该方法

举个例子来说明代理的作用:假设我们想邀请一位明星,那么并不是直接连接明星,而是联系明星的经纪人,来达到同样的目的.明星就是一个目标对象,他只要负责活动中的节目,而其他琐碎的事情就交给他的代理人(经纪人)来解决.这就是代理思想在现实中的一个例子

用图表示如下:



代理模式的关键点是:代理对象与目标对象.代理对象是对目标对象的扩展,并会调用目标对象

**1.1.静态代理（代码示例见demo01）**

静态代理在使用时,需要定义接口或者父类,被代理对象与代理对象一起实现相同的接口或者是继承相同父类.代理对象与目标对象要实现相同的接口,然后通过调用相同的方法来调用目标对象的方法。

静态代理总结:

1.可以做到在不修改目标对象的功能前提下,对目标功能扩展.

2.缺点:因为代理对象需要与目标对象实现一样的接口,所以会有很多代理类,类太多.同时,一旦接口增加方法,目标对象与代理对象都要维护.

解决静态代理中的缺点可以使用动态代理方式

**1.2.动态代理（代码示例见demo02）**

动态代理有以下特点:

1.代理对象,不需要实现接口

2.代理对象的生成,是利用JDK的API,动态的在内存中构建代理对象(需要我们指定创建代理对象/目标对象实现的接口的类型)

3.动态代理也叫做:JDK代理,接口代理

代理类所在包:java.lang.reflect.Proxy

JDK实现代理只需要使用newProxyInstance方法,但是该方法需要接收三个参数,完整的写法是:

static Object newProxyInstance(ClassLoader loader, Class<?>[] interfaces,InvocationHandler h )

注意该方法是在Proxy类中是静态方法,且接收的三个参数依次为:

ClassLoader loader:指定当前目标对象使用类加载器,获取加载器的方法是固定的

Class<?>[] interfaces:目标对象实现的接口的类型,使用泛型方式确认类型

InvocationHandler h:事件处理,执行目标对象的方法时,会触发事件处理器的方法,会把当前执行目标对象的方法作为参数传入

总结:

代理对象不需要实现接口,但是目标对象一定要实现接口,否则不能用动态代理

**1.3.Cglib代理（代码示例见demo03）**

上面的静态代理和动态代理模式都是要求目标对象是实现一个接口的目标对象,但是有时候目标对象只是一个单独的对象,并没有实现任何的接口,这个时候就可以使用以目标对象子类的方式类实现代理,这种方法就叫做:Cglib代理

Cglib代理,也叫作子类代理,它是在内存中构建一个子类对象从而实现对目标对象功能的扩展.

JDK的动态代理有一个限制,就是使用动态代理的对象必须实现一个或多个接口,如果想代理没有实现接口的类,就可以使用Cglib实现.

Cglib是一个强大的高性能的代码生成包,它可以在运行期扩展java类与实现java接口.它广泛的被许多AOP的框架使用,例如Spring AOP和synaop,为他们提供方法的interception(拦截)

Cglib包的底层是通过使用一个小而块的字节码处理框架ASM来转换字节码并生成新的类.不鼓励直接使用ASM,因为它要求你必须对JVM内部结构包括class文件的格式和指令集都很熟悉.

Cglib子类代理实现方法:

1.需要引入cglib的jar文件,但是Spring的核心包中已经包括了Cglib功能,所以直接引入pring-core-3.2.5.jar即可.

2.引入功能包后,就可以在内存中动态构建子类

3.代理的类不能为final,否则报错

4.目标对象的方法如果为final/static,那么就不会被拦截,即不会执行目标对象额外的业务方法.

在Spring的AOP编程中:

如果加入容器的目标对象有实现接口,用JDK代理

如果目标对象没有实现接口,用Cglib代理