关于动态代理

a、什么是代理（程序中）

就是对某些类的方法不够强大时，对其进行增强用的。

b、代理的分类

静态代理

装饰者模式

代理类从一开始就是存在的。并且会生成字节码。在程序运行时，字节码就进入了虚拟机

动态代理

代理类是随用随创建，随用随加载。

作用：在不改变源码的情况下，在程序运行时对方法进行增强。

c、动态代理的分类

基于接口的动态代理

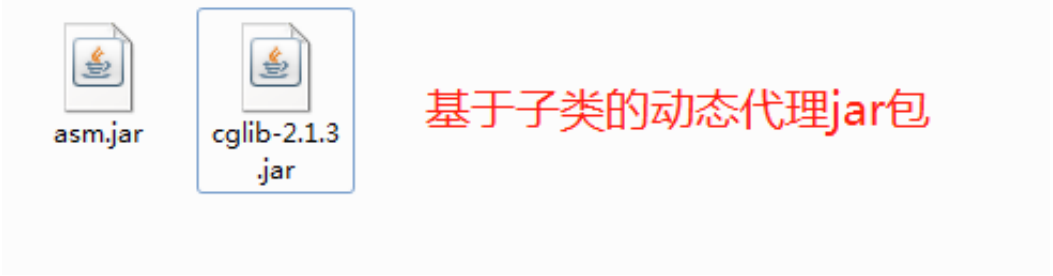
要求：被代理类最少实现一个接口。

它是jdk官方提供的

基于子类的动态代理

要求：被代理类不能是final修饰的。

它是第三方提供的。cglib



明确：

动态代理就是用来对被代理对象的方法进行增强用的。

能用jdk官方的不用第三方的。

1、什么是AOP

面向切面编程。

是一种对OOP思想再增强的编程思想。

它是使用动态代理的技术对我们的一些公共的代码进行统一管理。

在需要增强时，使用动态代理对我们的业务方法进行增强。

2、它给我们的开发带来的好处

维护方便。

能提高执行效率

减少冗余代码

AOP中的术语：

连接点：

业务层接口中的业务核心方法就是连接点。

它就是公共代码(增强的方法)和业务主线(需要被增强的方法)之间的纽带。

切入点：

指实际被增强过的连接点。

切入点一定是连接点。连接点不一定是切入点。

通知：

指提供了增强代码的类。(提供了公共代码的类)

通知中的方法分不同的类型：

前置通知：以业务核心方法（切入点方法）为定位点。在切入点方法之前执行的就是前置通知。

后置通知：以业务核心方法（切入点方法）为定位点。在切入点方法之后执行的就是后置通知。

例外通知/异常通知：以业务核心方法（切入点方法）为定位点。写在catch里面的就是异常通知。

最终通知：以业务核心方法（切入点方法）为定位点。写在finally里面的就是最终通知。

环绕通知：整个invoke方法叫做环绕通知。

切面：也叫方面。指通知所关心的某个方面。比如：事务是一个方面。日志又是另外一个方面。

织入：业务核心方法（切入点方法）被增强的那一瞬间叫做织入。

```
public Object invoke(Object proxy, Method method, Object[] args)
        throws Throwable {
    try{
        TransactionManager.startTransaction();//开启事务
        method.invoke(s, args);
        TransactionManager.commit();
    }catch (Exception e) {
        TransactionManager.rollback();
        throw new RuntimeException(e);
    }finally{
        TransactionManager.release();
    }
    return null;
}
```

前置通知：before

切入点

后置通知：after-returnning

异常通知：after-throwning

最终通知：after

整个代码叫做环绕通知的写法