--动态代理能干什么？

可以在不改变原来目标方法功能的前提下，在代理中增强自己的功能代码

程序开发中：

如我开发一个项目，有一个功能是其他人的（公司其他部门或者小组人员写的），我可以拿来用。

```
GoNeng.class,GoNeng gn = new GongNeng();,gn.print();
```

用完之后发现还有缺点，无法满足目前我这个项目的需求，我要在gn.print()执行之后，需要自己增加代码

而用代理实现print()的调用，再增加自己的代理，不用改变原来GongNeng类文件

--JDK动态代理要求目标类必须实现接口，若没有实现接口，需要使用cglib动态代理

--1、什么是代理？

如代购、商家、科学上网中的IP代理、中介等等，都可以理解为代理

例如：

有一所美国大学，这所大学向全世界招生。

那么留学中介，就是一个代理

留学中介（代理）作用：帮助美国这所大学进行招生，中介就是该所大学的代理，代替了学校完成招生的功能

--所以"代理"的特点：

1）、中介（代理）和学校要做的事情是一致的：招生

2）、中介是学校的代理，学校是求学者（客户端）的目标

3）、流程：求学者->(通过)中介(的学校介绍，办理入学手续)->(进入)美国大学

4）、中介要帮助求学者完成进入美国大学，就需要向求学者收取手续费

--一定找中介的理由？

1）、专业的而且方便

2）、求学者不能访问到学校。换句话说，求学者没有能力访问或者该学校拒绝个人访问

再例如我们买东西都是向商家买，商家卖，可以说商家是某个商品或者是商品厂家的代理，个人买东西是无法与厂家进行

买卖的，接触不到厂家

--2、在Java开发中的代理

开发中有一个a类，这个类需要调用c类中的方法，来完成某个功能。但是由于某些原因c类拒绝a类的访问调用。

此时a--不能调用c的方法

我们就在a和c之间创建一个b作为c的代理，b可以访问c。

就实现了：a->(访问)b->(访问)c，a间接访问了c中的方法

--实际例子：

登录和注册功能都需要验证码的情况，验证码是手机短信，发送短信是底层没法实现的，项目也没必要开发发送短信

的功能。

而运营商如中国移动、中国联通、中国电信可以发短信。

如果说注册时个人要求运营商给发送一条短信作为验证码，这是不实际的。那么如何实现发送验证码的步骤呢？

运行商是有子公司或者是关联公司的，它们代替运行商面向社会提供短信发送功能。

那么就有：张山项目需要发送短信->运营商子公司、关联公司（代理）->运营商

--开发中的这种模式叫做：代理模式。

指的是为其他对象提供一种代理以控制对某个对象的访问。某些情况下，一个对象不能或者无法直接引用另外一个对象，那么提供一个代理对象让客户类和目标对象之间起到中介的作用。

使用代理对象是为了在不修改目标对象功能的基础上，增强主页务逻辑。

--客户类真正想要访问的对象是目标对象，而实际上真正访问的是代理对象

--作用：

1）、功能增强。客户类通过访问代理类达到访问目标类，完成原本要实现的功能外，还可以在代理类实现目标

类基础上新增其他的功能或业务逻辑

2）、控制访问。代理类不让你访问目标类。

--3、实现代理的方式

1）、--静态代理

1. 代理类是自己手工实现的，自己创建一个Java类，定义为代理类

2. 由于是自己定义的代理类，代理的目标类就是被固定了的，确定了的。

--优缺点：

优：实现简单，容易理解

缺：

--在项目中目标类和代理类很多时，有以下的缺点

1）、当目标类增多时，需要自己写的代理类就会成倍的增加，降低开发效率

2）、当接口中功能增加或者修改后，会影响众多的实现类，厂家类和代理类都需要修改。影响较多，后期维护难度

增加

--模拟一个用户购买U盘的行为：

用户，客户端类

商家，代理类

U盘厂家，目标类

商家和厂家都是卖U盘的，完成的功能是一致的

--实现步骤：

创建一个普通的Java项目

1. 创建一个卖U盘的接口UsbSell，定义卖U盘的动作，即卖U盘的方法sell(int amount),amount表示一次购买的

数量，返回float类型，是返回价格。

2. 创建厂家目标类，实现上面的接口。假设是金士顿厂家UsbKingFactory，实现sell方法，定义一个U盘的价格是

85。当然可以根据购买的数量amount定义U盘的价格，这里就不做那么麻烦了。这里直接返回85.0f。不接受用户单

独购买。

3. 创建一个商家代理类，代理金士顿U盘销售，也需要实现接口。假设商家是TaoTao，在该类中要声明代理的具体

厂家是谁:"private UsbSell factory = new UsbKingFactory();",然后实现sell方法，在该方法中调用目标类

中的sell方法："float price = factory.sell(amount);",意思是向厂家发送订单，告诉厂家我买了U盘，厂家

发货，返回的是厂家定的价格。但是商家卖给用户需要加价："price = price + 25;"这就相当于是代理类调用目

标类方法后给客户端类实现某个功能（用户购买到U盘）的前提下的功能增强，这个sell方法返回price，就是商家

卖给客户的U盘的价格。

--新增

代理商家可以不仅有TaoBao的，还可以有WeiShang的，这个WeiShang代理商也实现UsbSell接口，实现sell方法调

用UsbKingFactory的sell方法，可以定义向用户出售的价格与TaoBao的不一样

当用户需要购买闪迪厂家的U盘时，我们需要创建闪迪厂家的代理类，有TaoBao的有WeiShang的

--从中可以看出，代理类需要完成的功能：

1）、目标类中方法的调用

为什么？

因为代理类中没有完成业务功能的代码，只有调用目标类中完成功能的方法才能实现代理。

好比说用户想留学，代理就得找学校，代理本身是没有让用户留学的能力的，只能找能让用户留学的学校，即

目标类

2）、功能增强

虽然代理本身不具有完成某功能的能力，但是可以不改变目标类功能的基础上增加额外的功能，这就是功能增强

2）、--动态代理

当使用静态代理时目标类很多时，可以换成动态代理，来避免静态代理的缺点

使用动态代理，即使项目的目标类很多：

1.生成的代理类也可以很少

2.修改接口中的方法也不会影响代理类

--原理：在程序执行过程中使用JDK的反射机制，创建代理类对象，然后动态的指定目标类

动态代理可以看作一种创建java对象的能力，自己不用创建一个如TaoBao.java的代理类，就能创建这个代理类的

对象

--例如：

通常创建一个Java对象：

1.创建一个Java类，编译为class文件

2.使用该类的构造方法，创建类的对象

--动态：动态的意思是在程序执行过程中，调用JDK中的方法才能创建对象。程序不执行的时候这个能力无法作用的

3）、--动态代理的实现

有两种实现方式：

1、--JDK动态代理（理解）：

使用Java反射包中的类和接口实现动态代理的功能

Java反射包：java.lang.reflect

需要使用反射包中的一个接口和两个类：

接口：java.lang.reflect.InvocationHandler

类：java.lang.reflect.Method（反射机制中学过）

类：java.lang.reflect.Proxy

2、--CGLIB动态代理（了解）：

cglib是第三方的工具库，创建代理对象

原理是继承，cglib通过继承目标类，创建子类，然后再子类中重写目标类中同名的方法，实现功能修改

因为cglib是继承，重写方法。要求目标类能够继承，方法可以重写，所以目标类不能是final修饰，方

法也不能是final修饰，对目标类要求还是比较宽松的。cglib也可以在很多框架中使用，mybatis、

spring等等。

复习反射机制中Method类：

这个类是反射中获取类中的方法。

第一步：获取某个类的.class，如User，其中有个play(String sports)方法

```
Class userClass = Class.forName("User类的根路径中的完整类名");
```

或者

```
Class userClass = xxx.yyy.ttt.User.class
```

第二步：用Class中getDeclareMethod(String name, Class<?>... parameterTypes)方法根据User中的

方法名，获取Method对象

```
Method method = userClass.getDeclareMethod("play",String.class)
```

第三步：创建User类对象

```
User user = userClass.newInstance();
```

第四步：执行play方法

--Method中的方法：Object invoke(Object obj, Object... args)

（需要一个对象：user，方法中要的参数：run，以及返回值：自定义，和方法：method）

这里返回一个Object类型，方便测试

```
            Object obj = method.invoke(user,"run");
```

反射包中三个类作用：

1）、InvocationHandler接口：里面只有一个invoke()方法

该方法表示：代理对象要执行的功能代码。代理类要完成的功能代码写在invoke方法中

代理类要写的代码有：

调用目标类的方法，执行目标类方法的功能

功能增强。调用目标类方法的同时，增加功能

方法原型：

--Object invoke(Object proxy, Method method, Object[] args)

参数解析：

Object proxy：JDK创建的代理对象，我们无需赋值

Method method：JDK通过反射获取到目标类中方法，由JDK提供的Method对象，我们不用赋值

Object[] args：目标类方法执行时需要的参数，JDK提供，无需我们赋值

--InvocationHandler接口作用就是代理要做的什么。

--该接口用法：

1.创建一个实现类实现该接口InvocationHandler

2.重写invoke方法，把原来静态代理中代理类要完成的功能，写到该方法中

2）、Method类：代表方法，即目标类中实现功能的方法

作用：通过Method执行某个目标类中的方法，method.invoke(目标类实例化对象,方法所需参数);

在上面重写的invoke方法中使用JDK提供的获取了目标类中方法的Method对象调用执行目标类中的方法

3）、Proxy类：该类是一个核心对象，作用是用来创建代理对象的。

一般我们创建对象都是用关键字new

而Proxy类中的一个方法可以可以创建对象，底层可能是new也可能是反射创建对象。

该类中方法如下：

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

是一个静态方法，作用就是创建代理对象，等同于静态代理中代理类对象的创建

--方法中参数解释：

1、--ClassLoader loader：类加载器对象。负责向内存中加载对象的。

使用反射机制获取对象的ClassLoader，例如类a，获取其类加载器对象

先获取a类的class，用Class中的getClassLoader()

ClassLoader loader = a.Class.getClassLoader();

--这里获取的是目标对象的类加载器

2、 --Class<?>[] interfaces：是一个接口对象，是目标类实现的接口，通过反射获取，Class类型的

因为JDK动态代理一定要目标类实现接口才可以使用，原因就在这

3、--InvocationHandler h:很明显。需要传的是我们自己实现InvocationHandler接口的类的对象。

包含的是代理类要完成的功能

--该方法的返回值就是一个代理对象

--动态代理具体实现步骤：

1、创建接口，定义目标类要完成的功能

2、创建目标类实现接口

3、创建InvocationHandler接口的实现类，重写invoke方法，该重写发方法完成代理类的功能

包括：调用目标类方法和功能增强

4、使用Proxy类中的静态方法创建代理对象。

--注意要把该方法的返回值转为目标类的实现接口类型，这样该代理对象就可以.目标类中的方法。多态

1.6动态代理

动态代理是指代理类对象在程序运行时由 JVM 根据反射机制动态生成的。动态代理不需要定义代理类的.java 源文件。

动态代理其实就是 jdk 运行期间，动态创建 class 字节码并加载到 JVM。

动态代理的实现方式常用的有两种：使用 JDK 代理代理，与通过 CGLIB 动态代理。

1.6.1 jdk 的动态代理

jdk 动态代理是基于 Java 的反射机制实现的。使用 jdk 中接口和类实现代理对象的动态创建。

Jdk 的动态要求目标对象必须实现接口，这是 java 设计上的要求。

从 jdk1.3 以来，java 语言通过 java.lang.reflect 包提供三个类支持代理模式 Proxy, Method 和 InovcationHandler。

（1） InvocationHandler 接口

InvocationHandler 接口叫做调用处理器，负责完调用目标方法，并增强功能。

通过代理对象执行目标接口中的方法，会把方法的调用分派给调用处理器(InvocationHandler)的实现类，执行实现类中的 invoke()方法，我们需要把功能代理写在 invoke

1.6.3 cgLib 代理

CGLIB(Code Generation Library)是一个开源项目。是一个强大的，高性能，高质量的 Code 生成类库，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。

使用 JDK 的 Proxy 实现代理，要求目标类与代理类实现相同的接口。若目标类不存在接口，则无法使用该方式实现。

但对于无接口的类，要为其创建动态代理，就要使用 CGLIB 来实现。CGLIB 代理的生成原理是生成目标类的子类，而子类是增强过的，这个子类对象就是代理对象。所以，使用 CGLIB 生成动态代理，要求目标类必须能够被继承，即不能是 final 的类。

cglib 经常被应用在框架中，例如 Spring ，Hibernate 等。Cglib 的代理效率高于 Jdk。对于 cglib 一般的开发中并不使用。做了一个了解就可以。

动态代理具体实现步骤：

1、创建接口，定义目标类要完成的功能

创建一个java工程idea-dongtaidaili-proxy，以上面静态代理的用户买U盘为例。

创建接口com.proxy.service.UsbSell

```
package com.proxy.service;

//目标接口
public interface UsbSell {

    //定义卖U盘的方法
    float sell(int amount);

}
```

2、创建目标类实现接口

创建一个卖U盘的金士顿厂家类com.proxy.factory.UsbKingSell，实现目标接口

```
package com.proxy.factory;

import com.proxy.service.UsbSell;

//目标类：金士顿U盘厂家
public class UsbKingSell implements UsbSell {
    @Override
    public float sell(int amount) {
        //目标类方法：功能是出售U盘，每个85
        System.out.println("目标类中sell方法执行");
        return 85f;
    }
}
```

3、创建InvocationHandler接口的实现类，重写invoke方法，该重写发方法完成代理类的功能

包括：调用目标类方法和功能增强

创建com.proxy.handler.MyHandler类实现反射包中的InvocationHandler接口，在重写的方法编写代理类要完成的功能

```
package com.proxy.handler;

import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Method;

//实现InvocationHandler接口，完成代理类要做的功能
public class MyHandler implements InvocationHandler {

    //因为是动态代理，目标类是不固定的，这里创建一个目标类对象的获取变量
    //通过构造方法传不同的目标类对象参数
    private Object target = null;

    public MyHandler(Object target) {
        this.target = target;
```

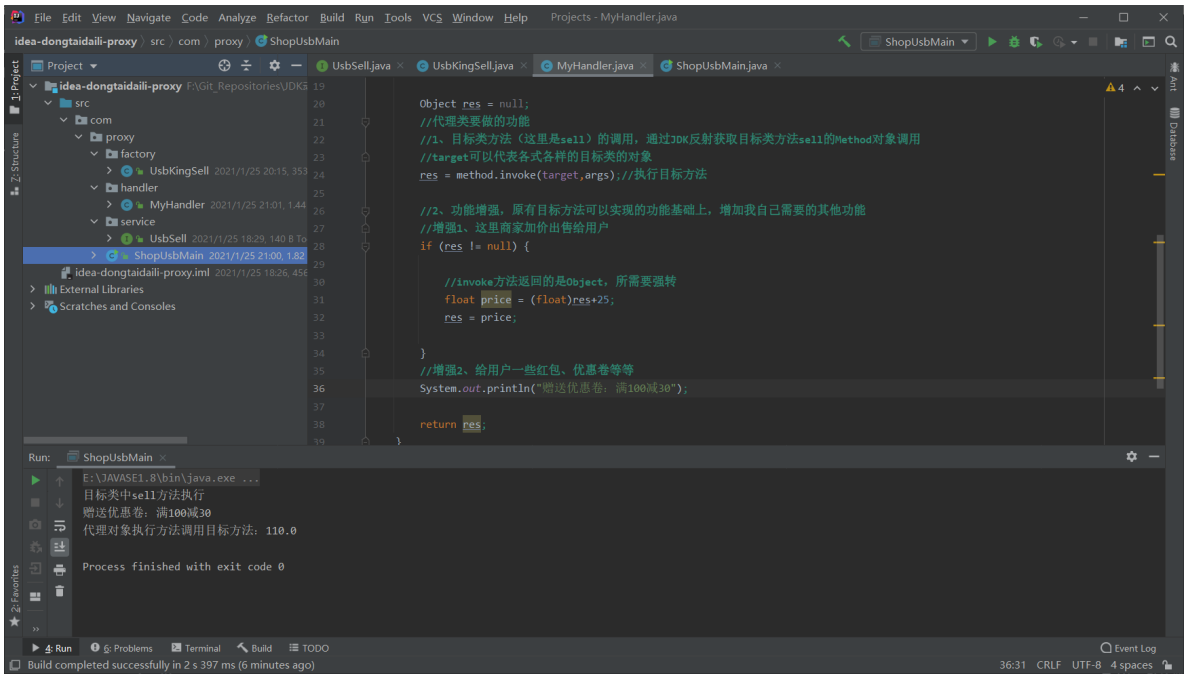
```
    }

    @Override
    public Object invoke(Object proxy, Method method, Object[] args) throws
Throwable {

        Object res = null;
        //代理类要做的功能
        //1、目标类方法（这里是sell）的调用，通过JDK反射获取目标类方法sell的Method对象调用
        //target可以代表各式各样的目标类的对象
        res = method.invoke(target,args);//执行目标方法

        //2、功能增强，原有目标方法可以实现的功能基础上，增加我自己需要的其他功能
        //增强1、这里商家加价出售给用户
        if (res != null) {

            //invoke方法返回的是Object，所需要强转
            float price = (float)res+25;
            res = price;

        }
        //增强2、给用户一些红包、优惠卷等等
        System.out.println("优惠卷：满100减30");

        return res;
    }
}
```

4、使用Proxy类中的静态方法创建代理对象。

创建一个测试类com.proxy.ShopUsbMain，表示用户买U盘。其中需要创建一个目标类对象，接口类型的。

```
package com.proxy;

import com.proxy.factory.UsbKingSell;
import com.proxy.handler.MyHandler;
import com.proxy.service.UsbSell;

import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Proxy;

public class ShopUsbMain {

    public static void main(String[] args) {

        //创建代理对象，完成功能
        //第一步、创建一个目标类对象,父类（接口）类型定义
        // 作用：
        // 1、为下面创建InvocationHandler实现类对象时作为参数传进去，
        // 2、通过反射，获取目标类的类加载器，作为创建代理对象方法的参数
        // 3、通过反射，获取目标类的实现接口UsbSell的class
        UsbSell usbFactory = new UsbKingSell();
```

```
        //第二步、创建InvocationHandler接口实现类对象，把目标类对象传进去
        //作用：下面调用创建代理对象的方法中作为InvocationHandler对象参数传进去
        InvocationHandler myHandler = new MyHandler(usbFactory);//传谁，下面代码就给谁创建代理对象

        //第三步、创建代理对象,转换为目标类的实现接口类型（多态）
        //其中参数目标类实现接口的class获取可以用反射获取，接口可实现多个，方法返回的是一个String数组
        // 该参数的作用就是使返回值能够强制转换成目标类的实现接口类型，然后可以.sell()方法
        UsbSell proxy = (UsbSell) Proxy.newProxyInstance(usbFactory.getClass().getClassLoader(),

 usbFactory.getClass().getInterfaces(),
                                                myHandler);
        //JDK动态代理创建的代理对象
        //System.out.println(proxy);//报错

 System.out.println("proxy:"+proxy.getClass().getName());//proxy:com.sun.proxy.$Proxy0

        //第四步、通过代理执行方法
        float price = proxy.sell(1);
        System.out.println("代理对象执行方法调用目标方法："+price);

    }

}
```



通过上面测试可以看出，我们只需要通过固定的代码，创建不同实现接口的目标类对象，就可以让程序自动帮我们创建代理对象，执行代理对象的方法调用目标类的方法。当我们修改接口中的方法时，只需要改动目标类就可以了。而当前目标类中的功能没法满足业务时，直接在InvocationHandler接口的实现类中添加额外的功能进行业务增强，不需要每个代理类都添加，不改变原功能的基础上。