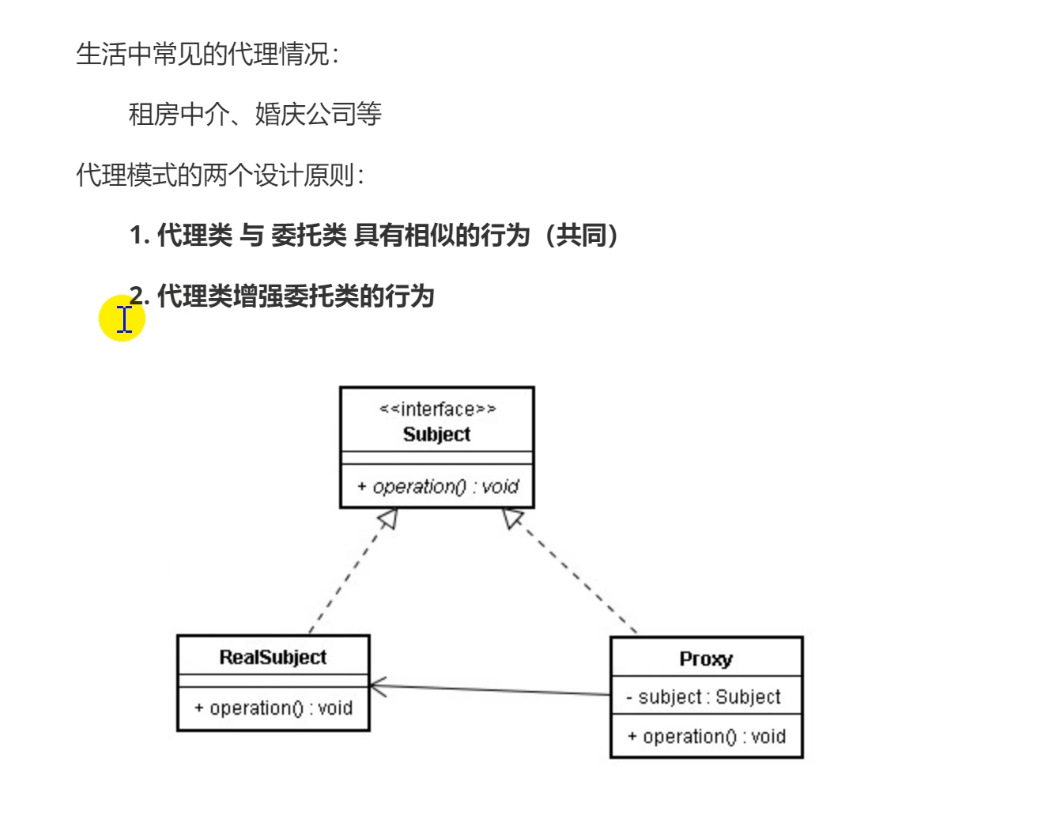
# 代理模式

婚庆公司 租房中介

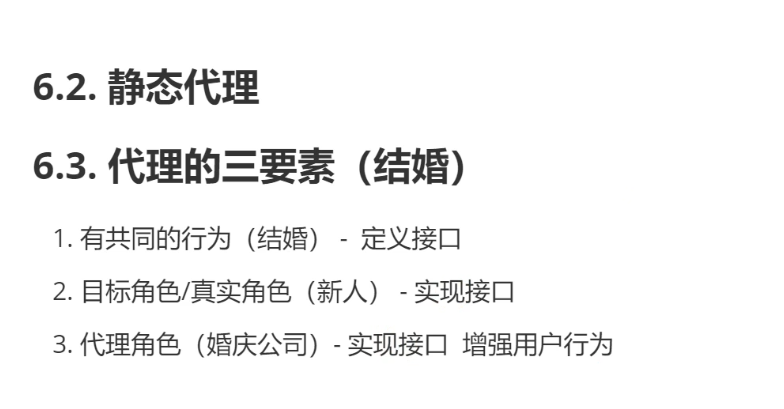


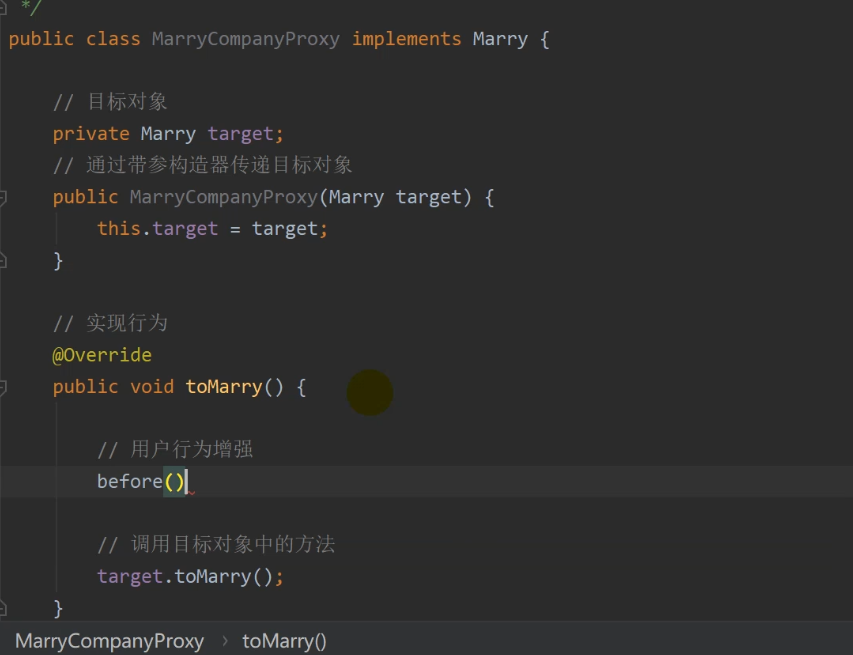
过滤 做一些增强行为



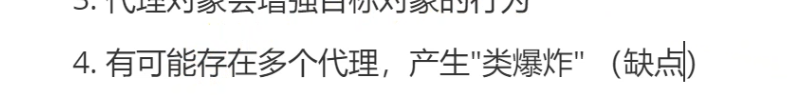


## 静态代理



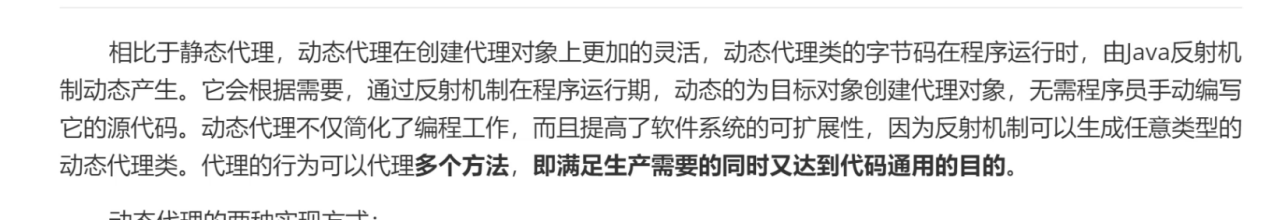






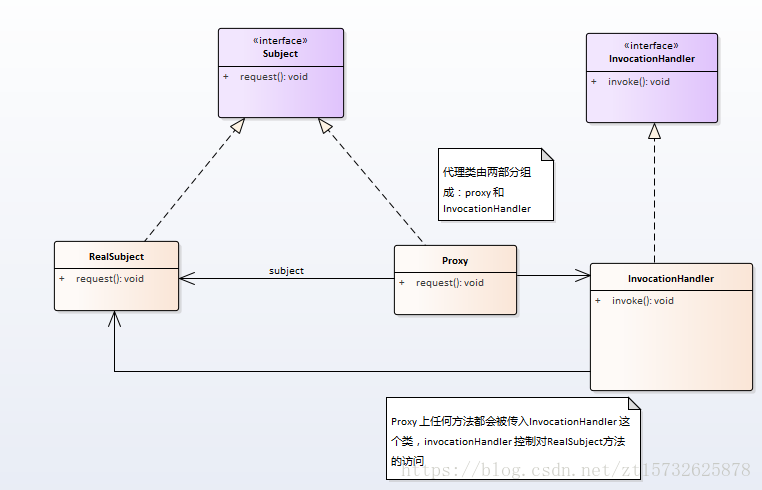
代理的作用是高内聚 其他琐碎功能交给代理 ioc 是降低耦合

## 动态代理



代理类通过反射 创建委托类对象





代理类由proxy.newproxyInstance 产生

|  |
| --- |
| Returns a proxy instance for the specified interfaces that dispatches method invocations to the specified invocation handler. |

参数 proxyclassloader realclass interface obj， handler

# 动态代理步骤

1.先实现 invocationhandler接口的类 重写 invoke

Invocationhandler 需要重写invoke 方法 增强功能

1. 使用proxy.newproxyInstance生成代理类

代理类由proxy.newproxyInstance 产生

|  |
| --- |
| Returns a proxy instance for the specified interfaces that dispatches method invocations to the specified invocation handler. |

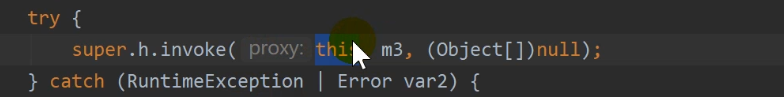
参数 proxyclassloader target interface obj， handler

Invocationhandler 中有target

生成的代理类中有handler

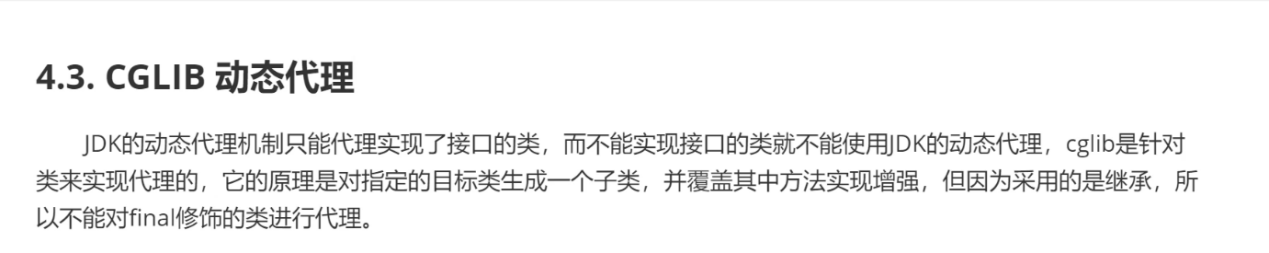
调用real object的方法

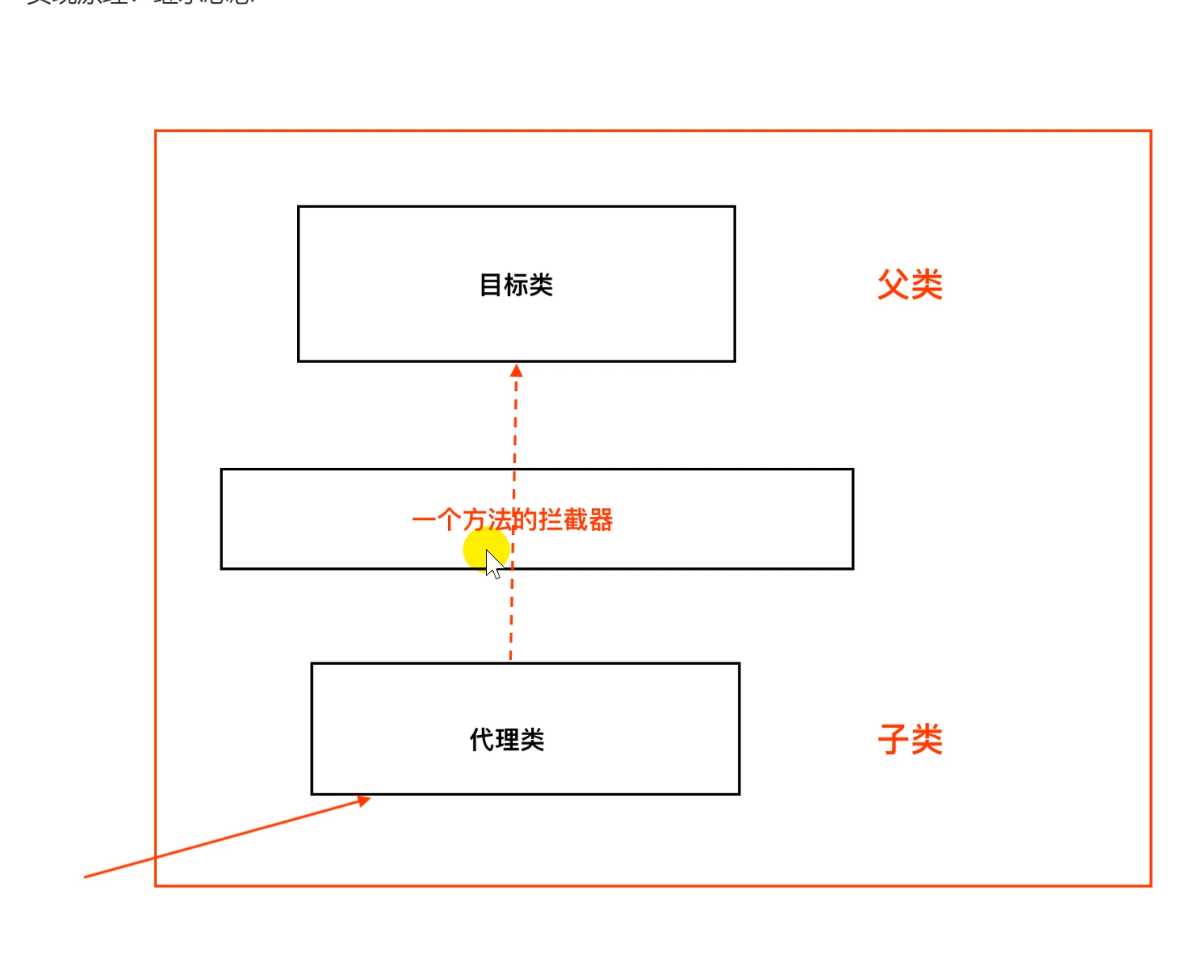
生成的代理类继承了Proxy，并实现了target interface，代理类调用target interface方法的时候，会调用父类Proxy 中的handler，从而调用handler中的invoke方法



Invoke 中的object 就是调用方法中返回的值

# CGLIB动态代理





实现Method interceptor 接口 该接口继承callback 类

在mymethodInterceptor 中重写 intercept方法

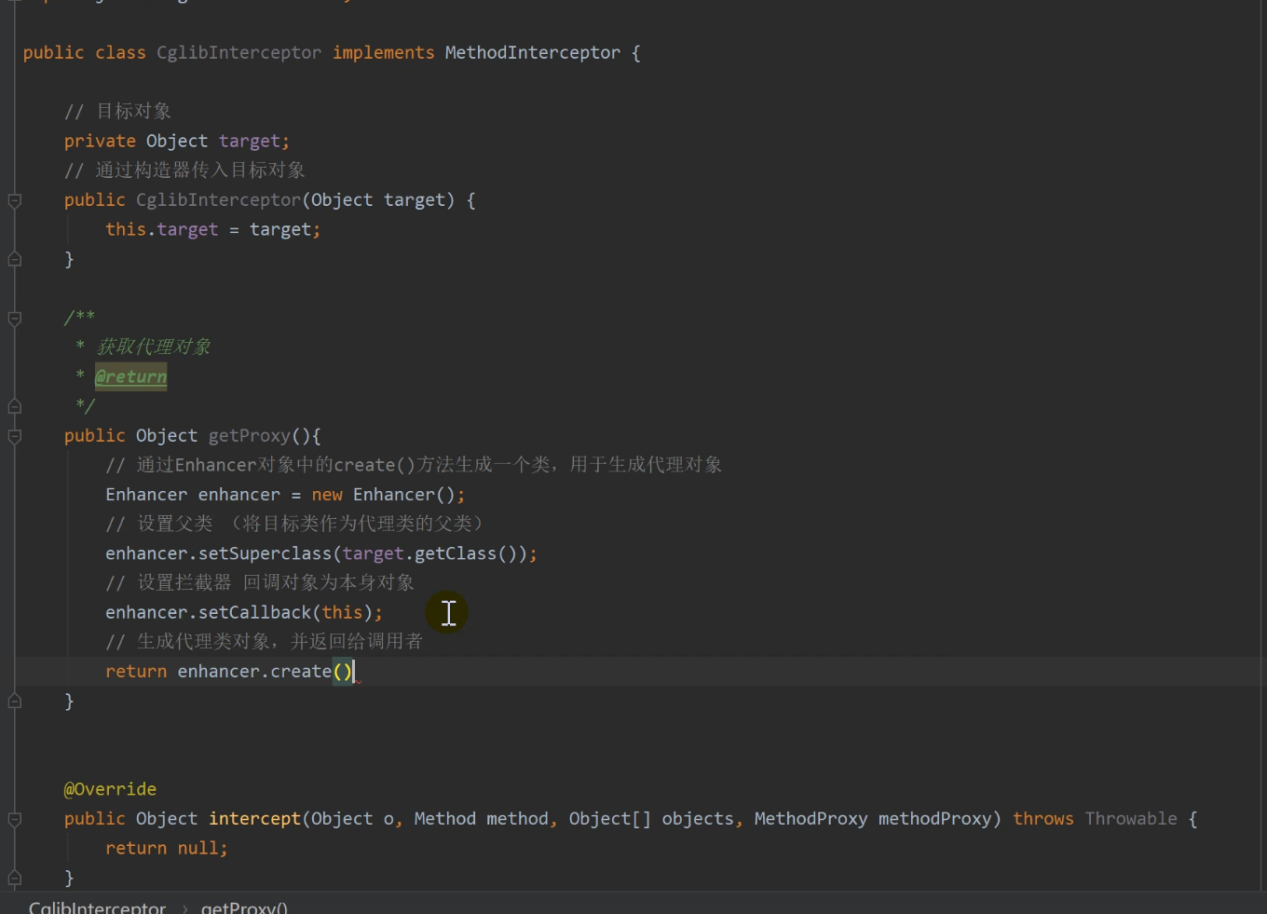
Setcallback 理解为回调代理子类的方法 设置这个规则

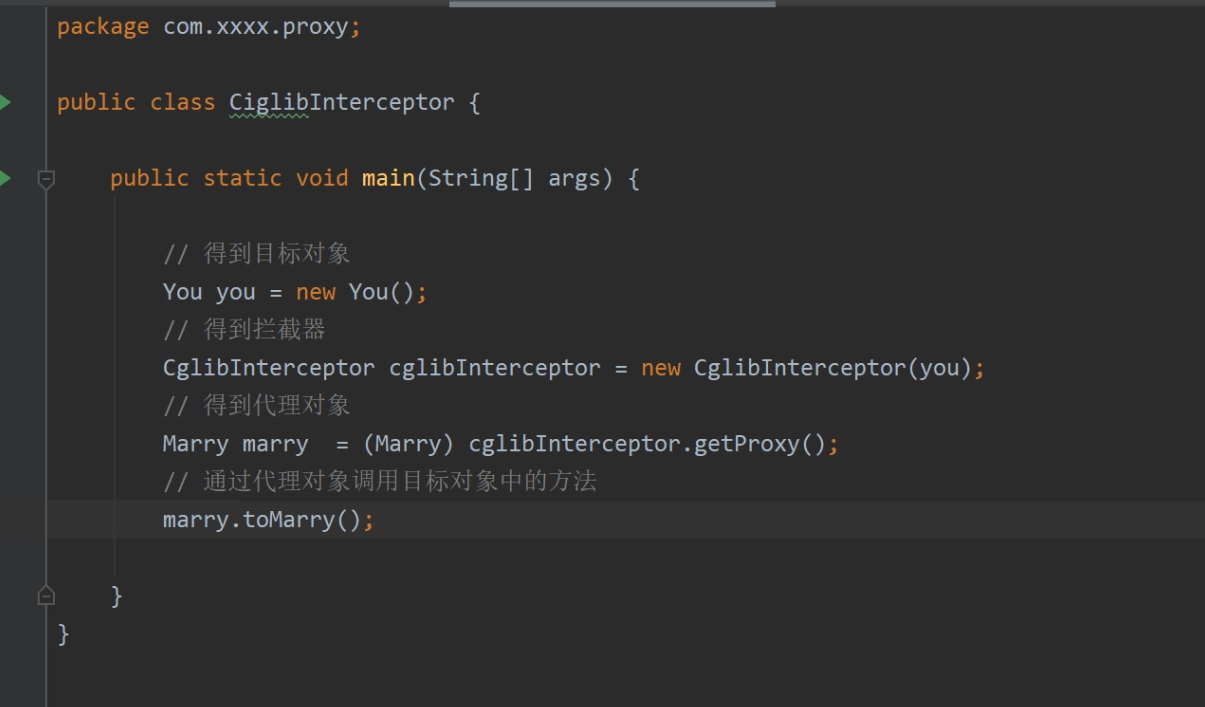
Enhacer中有callback 方法

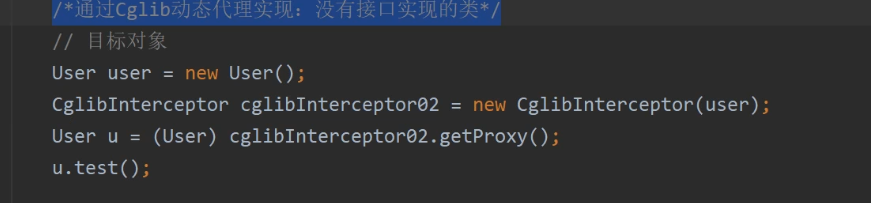
Methodinterceptor 接口继承了CallBack

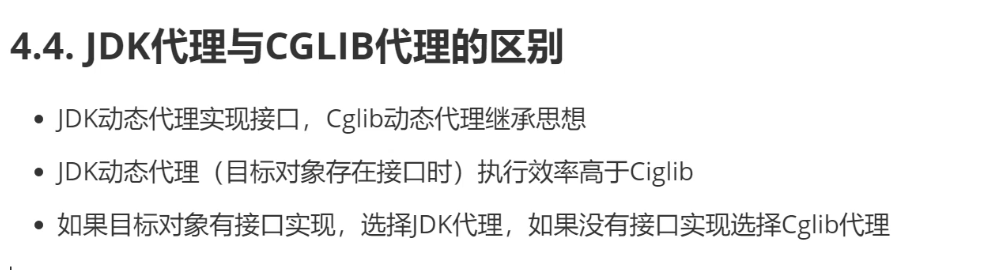
所以myinterceptor 本身就是一个callback

里氏替换原则



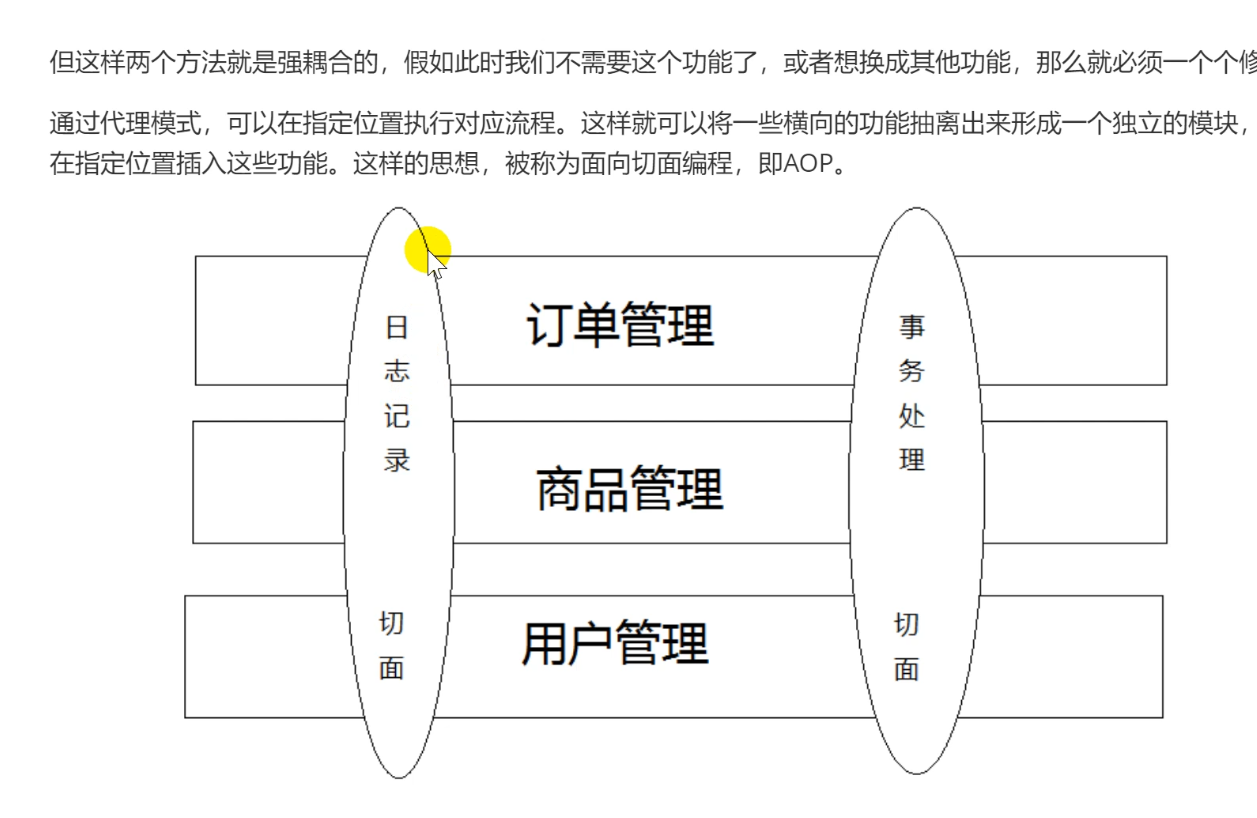


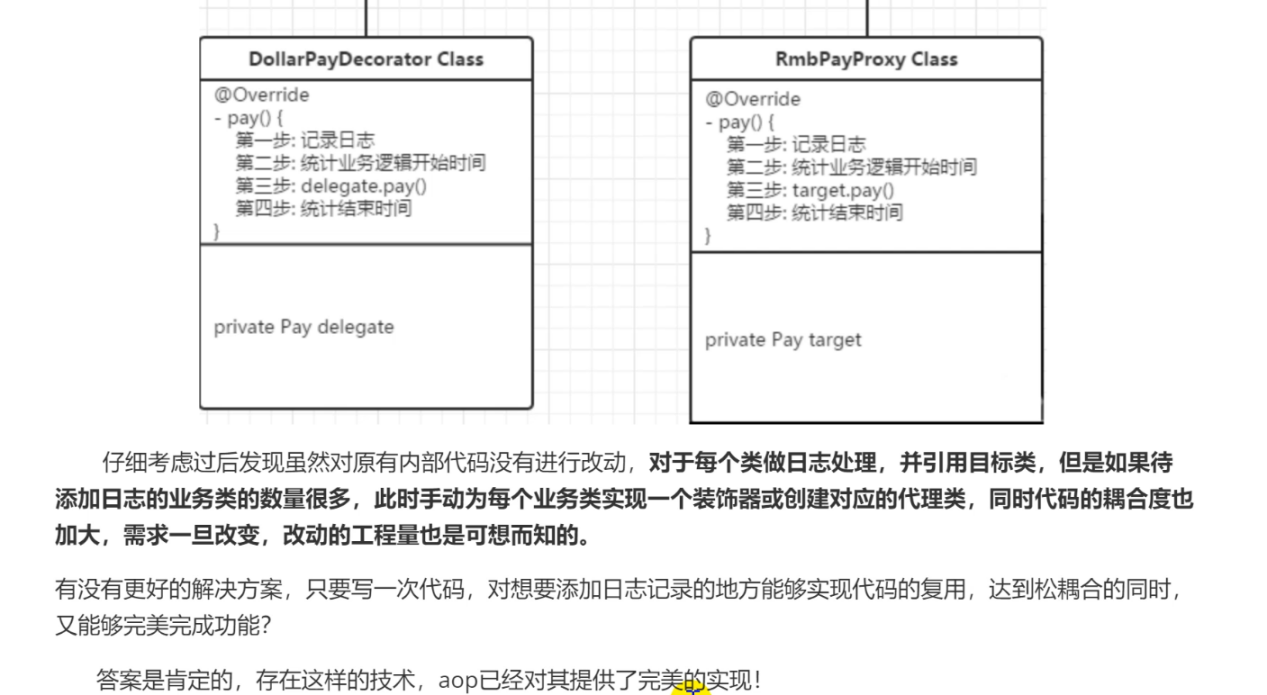


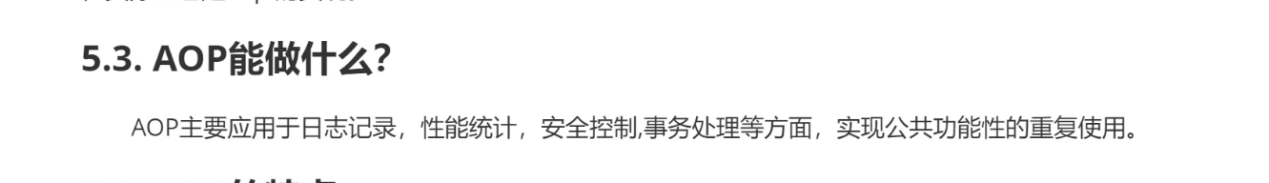


# 为什么要使用代理模式

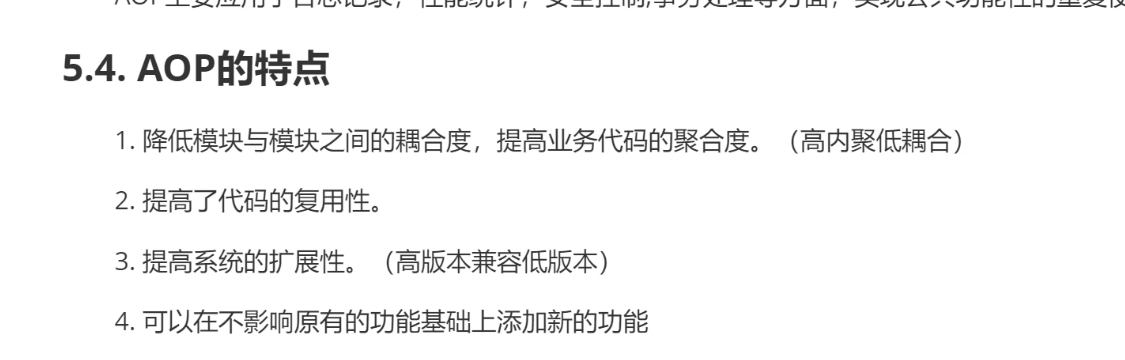








# Aop的特点，概念

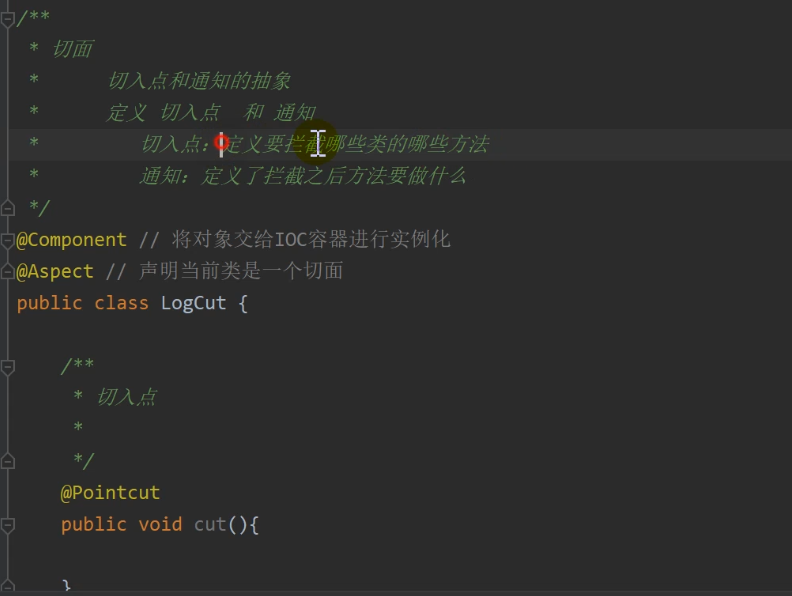




# Aop 的实现



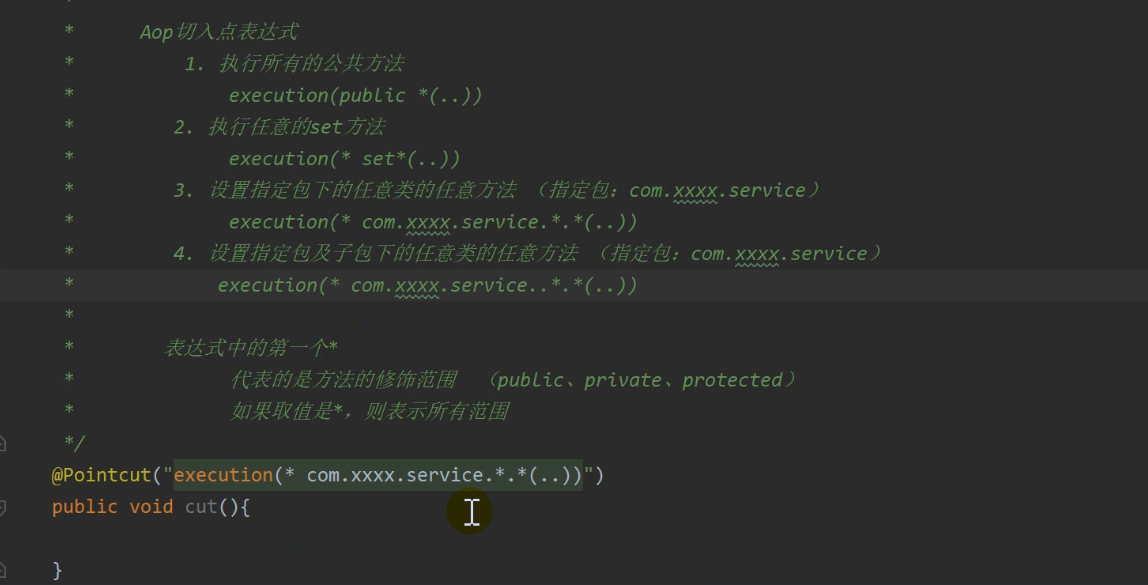




## 切入点：

第一个\* 方法修饰范围

第二个\* 表示包，第三个类 第四个方法



## 前置通知



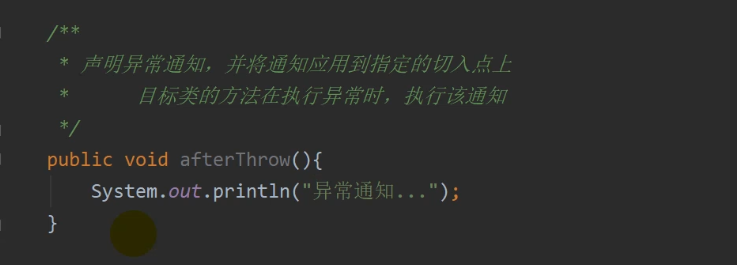
## 返回通知：



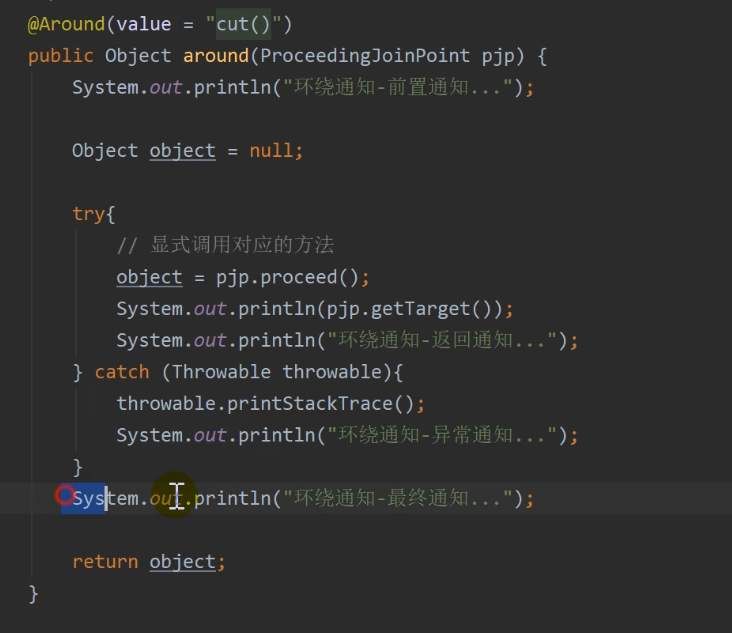
## 最终通知



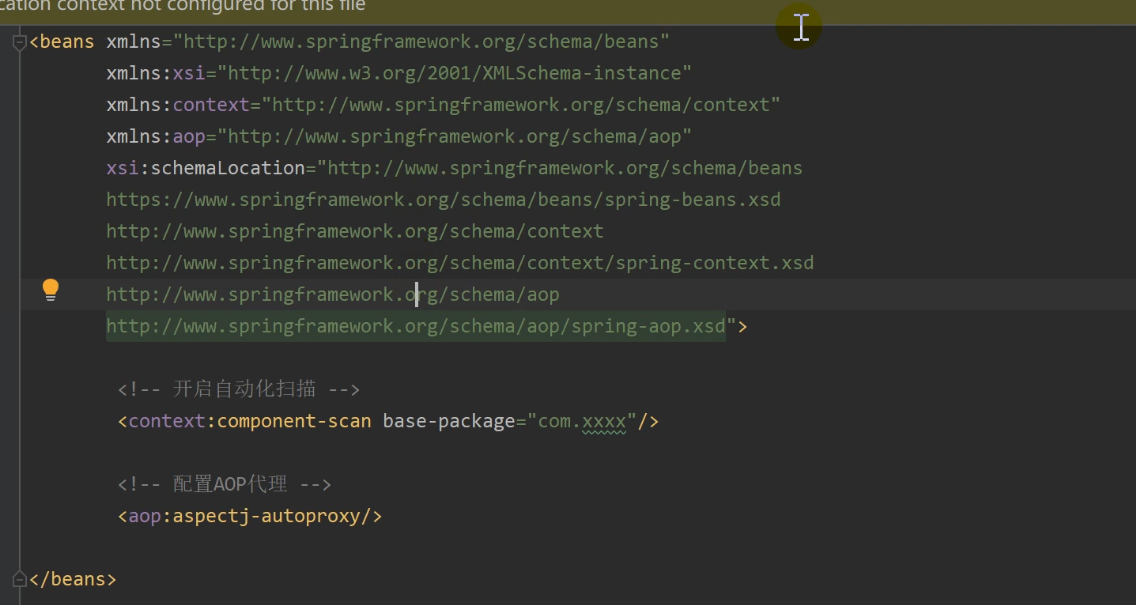
## 异常通知



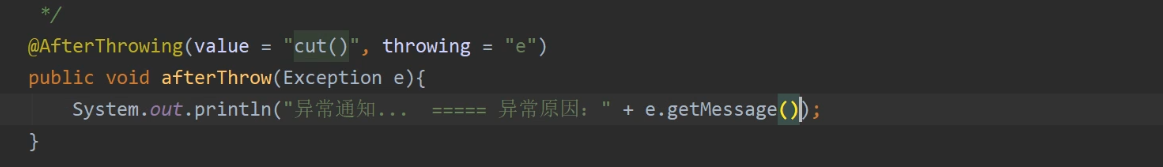
## 环绕通知：



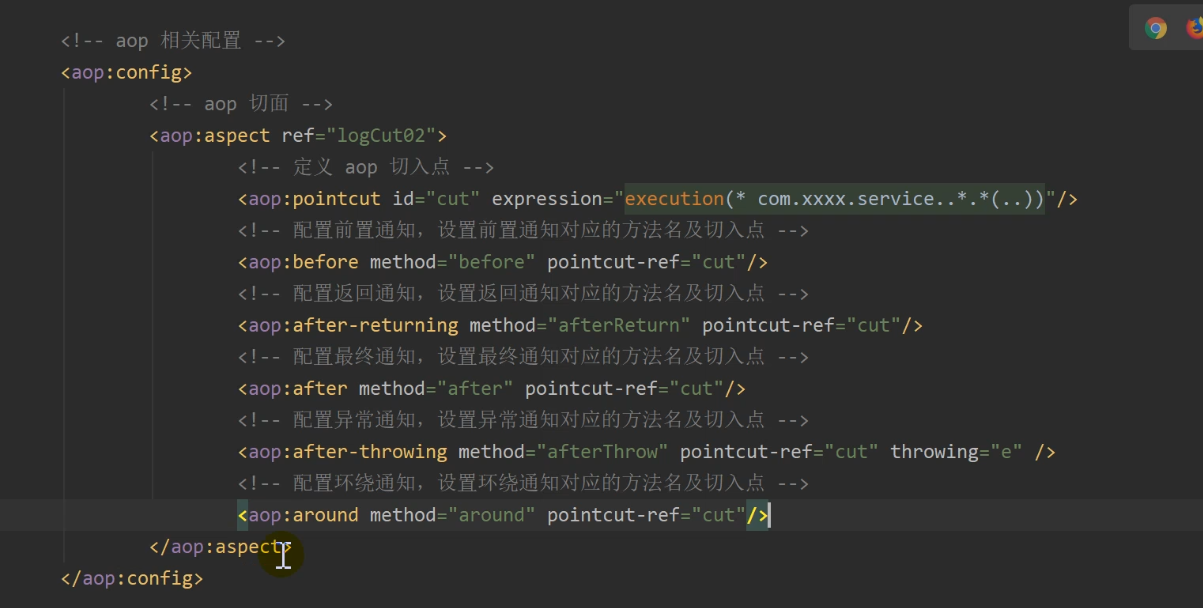
## Xml 中开启aop 自动代理



## 异常通知 可以打印异常信息



# XML 实现AOP



# Aop 优点：

高内聚 低耦合 代码

高版本兼容低版本 提高代码的复用

