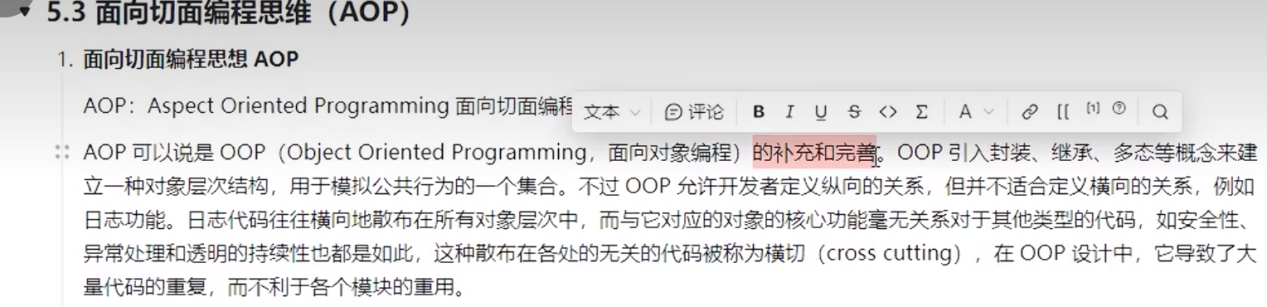
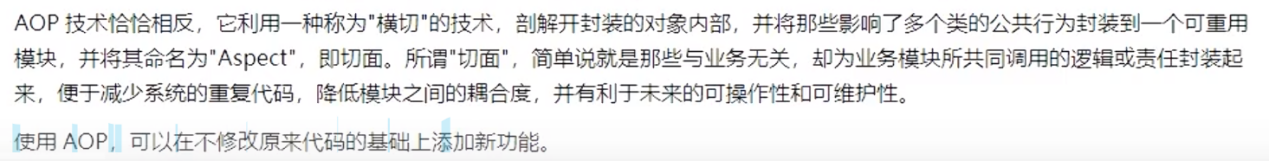
Spring-AOP

Aop面向切面编程，而之前大多都是oop面向对象编程，而aop并不是独立的一种，是基于oop面向对象进行优化改良的，aop其主要的思想就是将重复的代码和核心目标代码**分离解耦**。而SpringAOP框架是基于AOP思想实现的，底层的原理是**动态代理**。





代理：

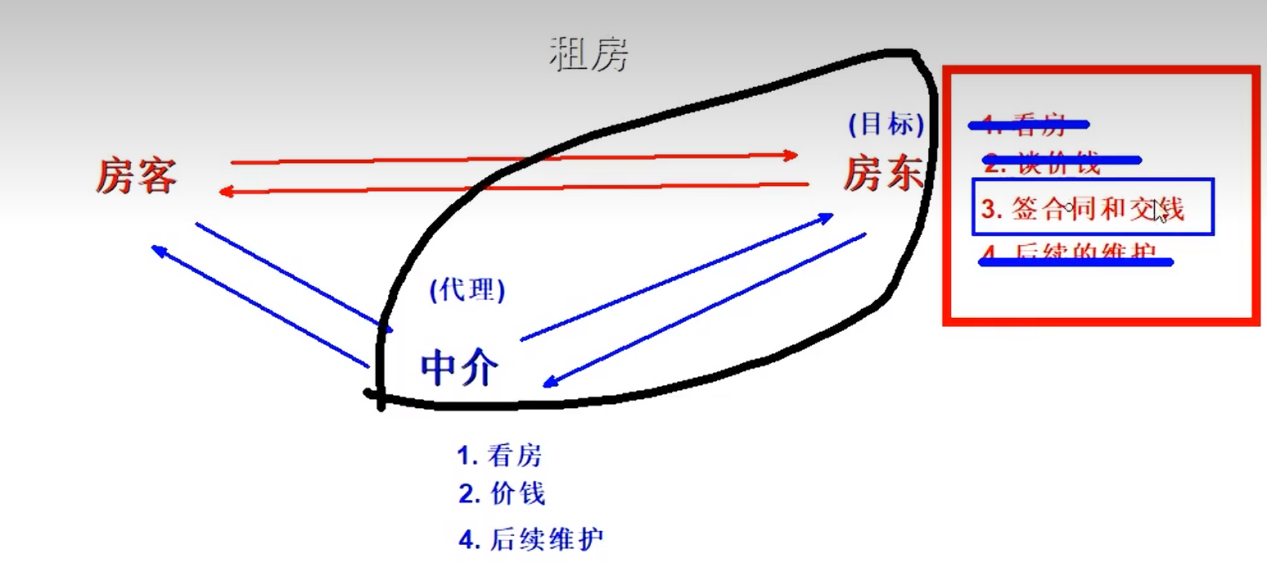
实现代理和被代理的类实现同一个接口，在代理对象中编写目标对象重复的代码并且调用被代理对象中的方法。被代理对象专注实现核心的代码逻辑

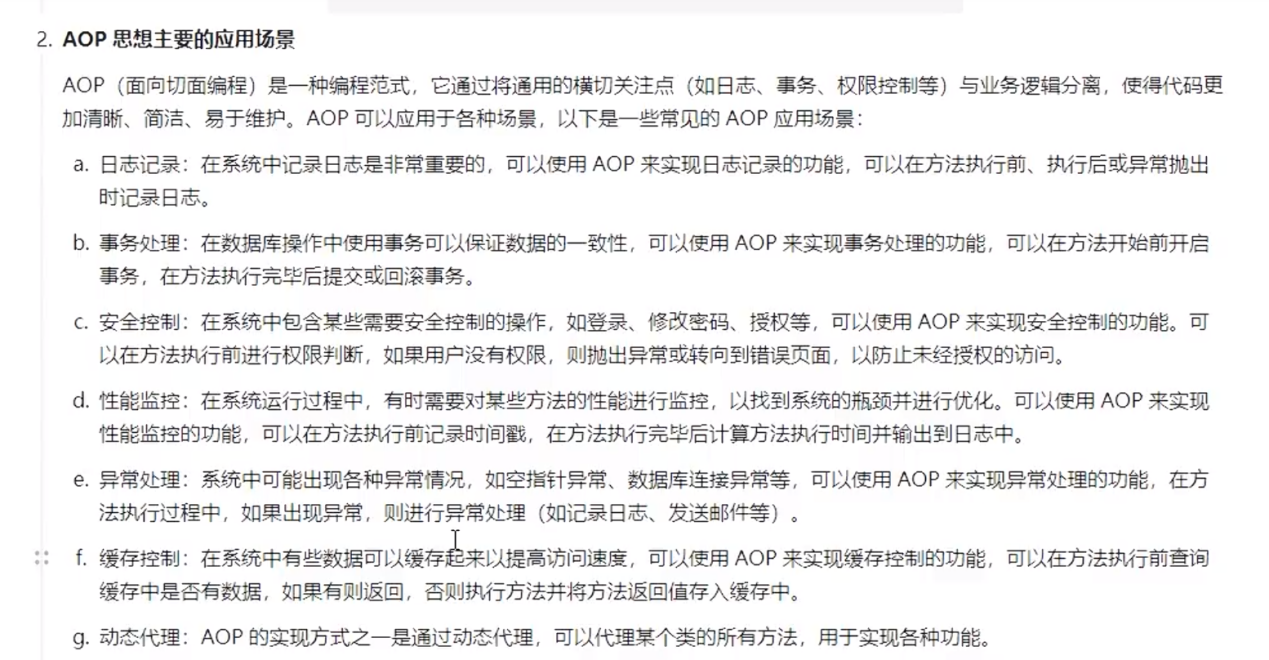
分为静态代理和动态代理。

而动态代理又有两种技术实现：

JDK原生：需要被代理的对象必须实现接口，他会根据目标类接口自动生成动态代理对象,代理对象和目标对象实现相同的接口

Cglib:第三方，并且已经集成到spring中，不需要目标对象实现接口，会根据目标对象生成一个子类对象，通过继承被代理的目标类实现代理





1.通知：

每一个横切关注点要做的是都需要一个方法来实现，这样的方法叫做通知方法。

前置通知：在被代理的目标方法前执行。

返回通知；在被代理的目标方法成功执行后执行

异常通知：在被代理的目标方法出现异常后执行

后置通知：在被代理的目标方法后执行，目标方法执行成功与否都会执行该通知

环绕通知：在被代理的目标方法前后执行



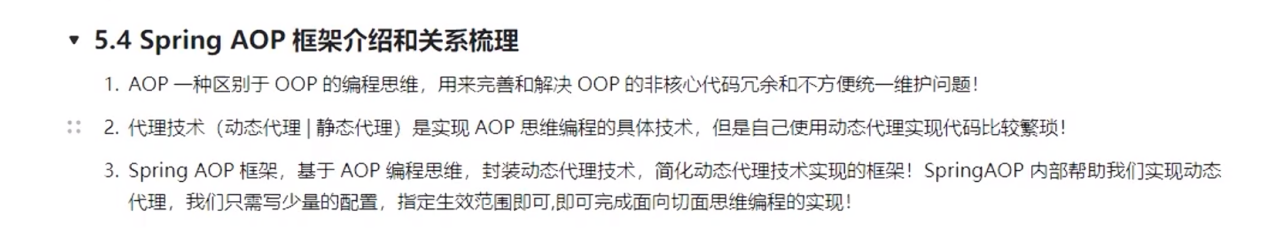
1. 连接点：

可以目标类中的方法，可以被切入的方法都可以是连接点

1. 切点：

被选中的切入的连接点，切点就是连接点，但连接点不一定是切点

1. 切面：切点+织入+通知



使用AOP框架需要导入的依赖:



不过spring-aop依赖已经通过spring-context依赖传递过来了不需要单独导入，只需要引入spring-aspects依赖

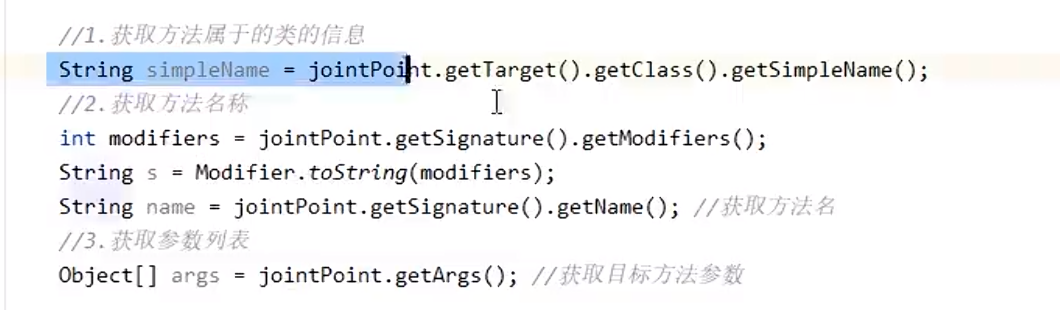
如何使用AOP框架：

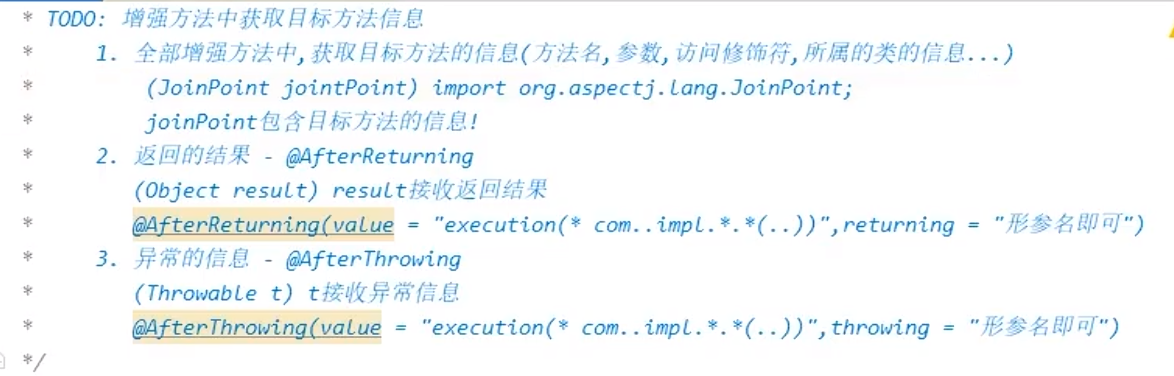
1. 正常写IOC组件、配置文件和核心业务
2. 定义切面类（增强类）
3. 切面类的配置
4. 开启aop注解的支持（在配置文件xml中配置或配置类中配置）

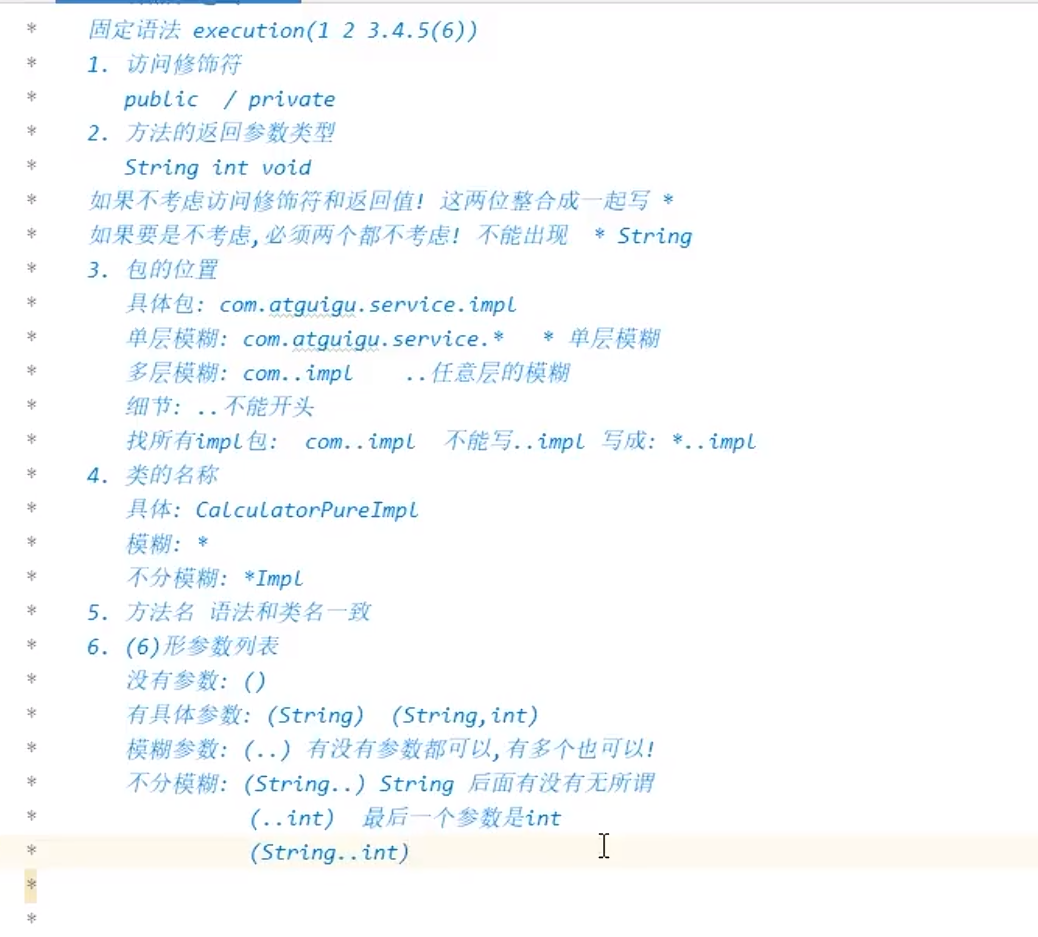
Xml配置aop注解的支持：<aop:aspectj-autoproxy/>

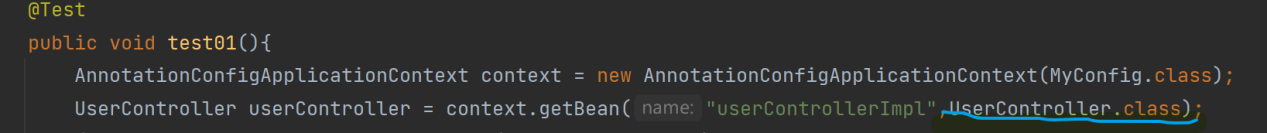
配置类配置aop注解支持:在配置类上加上注解@EnableAspectJAutoProxy

注：aop切面类的也必须加入IOC容器中









获取bean必须用接口.class去拿bean如果使用了代理，不然会报错



XML配置aop了解即可