# springAOP技术

AOP:面向切面编程,在不修改源码的基础上可以增加一些新的功能,新的业务逻辑

OOP:面向对象编程

在方法执行的前后记录日志?

代理模式:在不修改源码的基础上可以增加一些新的功能,新的业务逻辑

静态代理:

1. 提供目标对象以及代理对象的接口
2. 编写代理对象,需要把目标对象交给代理对象,代理对象在调用目标对象的方法 前后可以新增业务逻辑

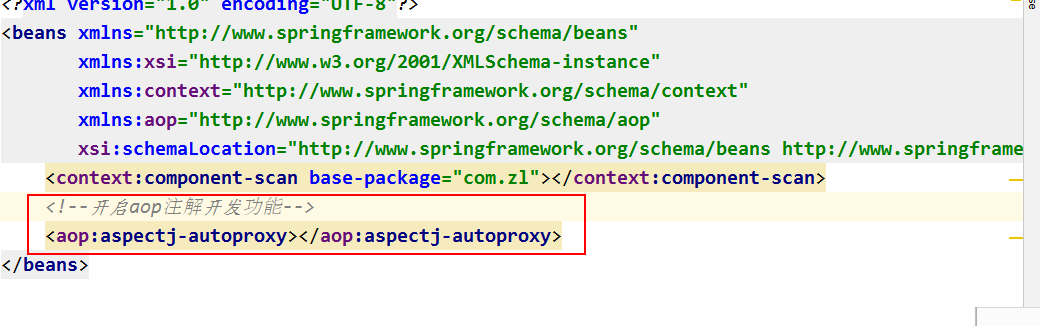
动态代理:

jdk自带的基于反射的动态代理:目标对象和代理必须要实现相同的接口

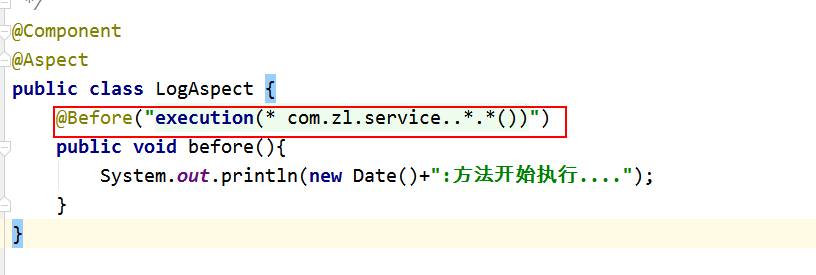
CGLIB基于字节码技术实现动态代理:既可以代理接口又可以代理类

aop开发步骤:

1. 导入spring核心以及springAOP相关的jar包
2. 在主配置文件中开启aop注解开发功能



1. 通过在切面对象的指定通知上面写切点表达式,在指定三个那些



切点表达式:

第一个\*:表示任意修饰符任意返回值

第二个\*:任意类

第三个\*:任意方法

第一个..:当前包以及任意子包

第二个..:任意参数

execution(\* com.zl.service..\*.\*(..)):只要是service包下面的类或则子包中 类的全部任意 返回值任意修饰符任意参数的方法都符合要求

execution(\* com.zl.service..\*.add\*(..)):以add开头的任意方法

!execution(\* com.zl.service..\*.add\*(..)):除了以add开头的任意方法

execution(\* com.zl.service..\*.add\*(..))||execution(\* com.zl.service..\*.delete\*(..)):选中以 add开头或则以delete开头的方法

五种通知类型:

前置通知(@Before):在目标方法执行前执行该通知

后置通知(@After):方法执行完成以后执行该通知

1. 不管方法执行过程中是否出现异常该通知都会执行

返回通知(@AfterReturning):

1. 只有方法正确执行完成以后才会执行返回通知
2. 在通知中可以获取目标方法的返回值

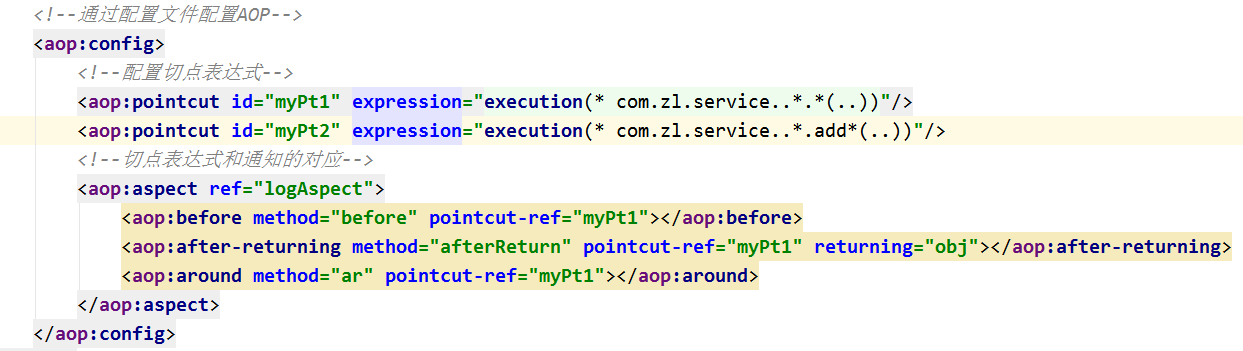
异常通知(@AfterThrowing):

1. 只有方法抛出异常的时候才会触发异常通知
2. 在通知中可以获取目标方法的异常类型

环绕通知:

1. 环绕通知是所有通知类型中功能最为强大的, 能够全面地控制连接点. 甚至可 以控制是否执行连接点
2. 对于环绕通知来说, 连接点的参数类型必须是 ProceedingJoinPoint . 它是 JoinPoint 的子接口, 允许控制何时执行, 是否执行连接点.
3. 在环绕通知中需要明确调用 ProceedingJoinPoint 的 proceed() 方法来执行被代 理的方法. 如果忘记这样做就会导致通知被执行了, 但目标方法没有被执行
4. 环绕通知的方法需要返回目标方法执行之后的结果, 即调用 joinPoint.proceed(); 的返回值, 否则会出现空指针异常
5. 在环绕通知中可以更改目标方法分返回值

通过XML配置AOP:



spring事务管理:

事务的四大特征(ACID)

事务并发导致的问题

事务的传播属性

事务的隔离级别