Spring源码分析

# Spring中用到AOP的地方

Authentication权限认证

Logging日志

Transctions Manager事务

Lazy Loading懒加载

Context Process上下文处理

Error Handler错误跟踪（异常捕获机制）

Cache缓存处理

# 设计模式

## 代理模式

特点：

1. 执行者，被代理者
2. 对于被代理人来说，这件事是一定要做的，但是自己又不想做或没有时间做，找代理
3. 需要获取到被代理人的个人资料(执行者持有被带离对象的引用)

原理：

1. 拿到被带离对象的引用，然后获取它的接口
2. JDK代理重新生成一个类，同时实现我们给的代理对象锁实现的接口
3. 把被代理对象的引用也拿到了
4. 重新动态生成一个class字节码
5. 然后编译

总结：

做了一件什么事？字节码重组

可以在每一个方法调用之前加一些代码，在方法调用之后再加一些代码

AOP：是无代理、日志监听

Service方法

开启一个事务

事务的执行是由我们自己的代码完成的

监听到是否有异常，可能需要根据异常的类型来决定这个事务是否要回滚还是继续提交

事务要关闭

IP4  
FF.FF.FF.FF  
255.255.255.255  
0.0.0.0  
  
IP6  
FFFF.FFFF.FFFF.FFFF.FFFF.FFFF  
2^16-1.2^16-1.2^16-1.2^16-1.2^16-1.2^16-1  
0.0.0.0.0.0

## 工厂模式

生产者消费者要区分开

消费者来说：产品保质期有没有过？产品使用效果怎么样？产品价格便不便宜？不关心：面包生产工艺

## 委派模式

## 策略模式