#### Spring Boot

##### 之面向切面编程（AOP）-读写锁

问题描述

Spring Boot 2.2.5

JDK 8

面向切面编程打破了面向过程（函数式）、面向对象程序设计的代码流程，在一些场景下可以发挥意向不到的妙用。例如：

1. 失败重试
2. 事务管理
3. 【读写锁】

心得 面向切面编程类似于虚拟化中的“横向扩展”，可以起到非侵入式改造、代码去重的效果

解决

AbstractReadWriteLockAspect.java

@Aspect  
public abstract class AbstractReadWriteLockAspect {  
  
 private ReentrantReadWriteLock lock = new ReentrantReadWriteLock();  
  
 @Pointcut  
 protected abstract void read();  
  
 @Pointcut  
 protected abstract void write();  
  
 @Around("read()")  
 public void doRead(ProceedingJoinPoint joinPoint) {  
 try {  
 lock.readLock().lock();  
 joinPoint.proceed();  
 } catch (Throwable throwable) {  
 throwable.printStackTrace();  
 } finally {  
 lock.readLock().unlock();  
 }  
 }  
  
 @Around("write()")  
 public void doWrite(ProceedingJoinPoint joinPoint) {  
 try {  
 lock.writeLock().lock();  
 joinPoint.proceed();  
 } catch (Throwable throwable) {  
 throwable.printStackTrace();  
 } finally {  
 lock.writeLock().unlock();  
 }  
 }  
}

笔记

1. 面向切面编程“三件套”
   1. 新建注解
   2. 定义切面
      1. Aspect
      2. Pin point
      3. Advice
   3. 添加注解
2. @target: Limits matching to join points (the execution of methods when using Spring AOP) where the class of the executing object has an annotation of the given type.
3. 完成项目地址：

<https://github.com/MariaLikesFish/spring-aop-read-write-lock>

□