# 71\_代码实战: 基于 Spring 事务保存订单数据,解决一致性问题

**儒猿架构官网上线**,内有石杉老师架构课最新大纲,儒猿云平台详细介绍,敬请浏 览

官网: <u>www.ruyuan2020.com</u> (建议 PC 端访问)

#### 1、开篇

上节课先通过复习事务 ACID 的特性,对事务的本质有所了解,然后返回创建订单的业务流程中,对照事务的特性将符合其特性的三个操作:创建订单、使用优惠券和扣除积分放到一个事务中处理。本节课会通过代码实战的方式,看看如何通过Spring 事务实现以上三个操作的事务处理,解决数据一致性的问题。今天课程的内容包括以下几个部分:

- Spring 实现事务方式
- 创建 Consumer、Coupon、Order 的 Mapper 文件
- 创建 Consumer、Coupon、Order 的服务文件
- 定义 Spring 事务
- 创建 OrderController

### 2、Spring 实现事务方式

上节课说明了我们需要使用事务解决数据一致性的问题,然后引出 Spring 的事务处理模式。Spring 有两种事务管理方式分别是:编程式事务管理和声明式事务管理。编程式事务管理使用 TransactionTemplate 或者直接使用底层的

PlatformTransactionManager。编程式事务管理的特点是需要手动写入添加操作, 并且通过代码提交事务。

而与之不同的声明式事务管理,却是建立在 AOP 之上的。声明式事务管理的本质是对方法前后进行拦截,然后在目标方法开始之前创建事务,在执行完目标方法之

后根据执行情况提交或者回滚事务。其最大优点就是不通过编程的方式管理事务,也就是不需要在业务逻辑中掺杂事务代码,只需在配置文件中做做好配置即可。显然声明式事务管理要优于编程式事务管理,也是 Spring 倡导的非侵入式的开发方式。因此,本案例中也会使用声明式事务管理,接下来的代码编写会带大家一起熟悉,如何通过声明式事务实现创建订单业务的事务管理。

## 3、创建 Consumer、Coupon、Order 的 Mapper 文件

在介绍 Spring 事务管理之前,先介绍一下三个操作访问数据的 Mapper 文件,也就是对应三个业务:扣除积分、使用优惠券和创建订单。与这些操作对应的其实就是修改数据库中的三张业务表,这里依次给大家介绍。

如图 1 所示, 在 resources/mapper 目录下面创建 consumerMapper.xml 文件,其中配置了对 t\_consumer 表的操作。我们需要注意的是 deductCredits 这个数据表操作,它通过 update 的语句对 t\_consumer 表 中的 credits (积分) 字段进行操作,从而达到扣减积分的目的。

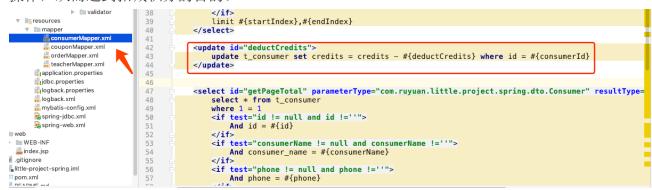


图 1 创建 consumerMapper.xml

看完了扣除积分,再来看使用优惠券,如图 2 所示,在 resources/mapper 目录下的 couponMapper.xml 文件中依旧定义了与 t\_consumer\_coupon 表的操作,这张数据表主要是描述消费者与优惠券之间的关系。其中 updateStatusByld 主要更新t consumer coupon 中 status,通过对优惠券状态的修改实现"使用优惠券"操作。

```
mapper
                                   43
                                                   <insert id="add" parameterType="com.ruyuan.little.project.spring.dto.Coupon">
                   service
                                    44
                                                       INSERT INTO t_coupon
                   utils u
                                                       (id, coupon_name, phone, credits, sex, birthday, order_times,
    total_prices, open_id, email, address)
VALUES (#{id},#{couponName},#{phone},#{credits},#{sex},#{birthday},
                                     45
                 validator
                                     46
    resources
       mapper
                                     48
                                                            #{orderTimes},#{totalPrices},#{openId},#{email},#{address});
          consumerMapper.xml
                                     49
                                                  </insert>
          couponMapper.xml
                                     50
          arderMapper.xml
                                     51
                                                  <update id="updateStatusById">
          teacherMapper.xml
                                     52
                                                       update t_consumer_coupon set status = #{status} where id = #{id}
        application.properties
                                     53
       📊 jdbc.properties
                                     54
55
       logback.properties
                                                   <insert id="receiveCoupon" parameterType="com.ruyuan.little.project.spring.dto.</pre>
       logback.xml
                                    56
                                                            useGeneratedKeys="true" keyProperty="id">
       mybatis-config.xml
                                     57
                                                       INSERT INTO
        spring-jdbc.xml
                                     58
                                                        t_consumer_coupon(
       spring-web.xml
                                     59
                                                       consumer_id,
web
```

图 2 couponMapper.xml 文件

最后来看 orderMapper.xml 中对创建订单的操作,同样是在 resources/mapper 目录下,如图 3 所示,orderMapper.xml 中进行 add 操作的定义,它会往 t\_order 表中插入一条记录包括:消费者 ld、消费者姓名、教师 ld、教师姓名、课程、开始日期、结束日期、课程价格等信息,显然是用来创建订单的。

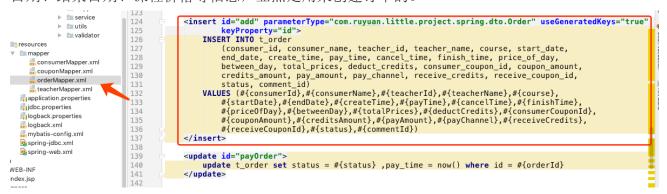


图 3 orderMapper.xml 文件

#### 4、创建 Consumer、Coupon、Order 的服务文件

有了三个操作的数据库访问文件以后,再来看三个操作对应的服务。如图 4 所示,

在 spring/service 目录下面创建 ConsumerService、CouponService、

OrderService 的接口和 ConsumerServiceImpl、CouponServiceImpl、

OrderServiceImpl 的实现,这些服务分别实现了积分扣除、使用优惠券和创建订单的功能。

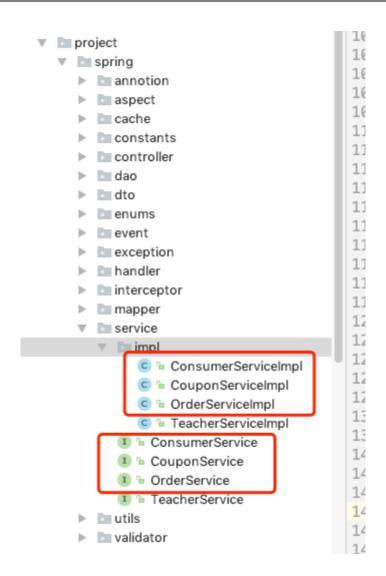


图 4 三个操作的接口和服务

图 5 所示,在 ConsumerServiceImpl 中的 orderCreateInformDeductCredits 方法

传入 order 参数,调用 consumerMapper 中的 deductCreadits 方法扣除积分。

```
60
61 ©
aspect
                                                       public int orderCreateInformDeductCredits(Order order) {
   if (order.getDeductCredits() <= 0) {
      return 0;
}</pre>
 cache
                                       62
63
64
65
66
67
68
69
70
71
constants
controller
dao
                                                             Consumer consumer = consumerMapper.findById(order.getConsumerId());
if (consumer.getCredits() < order.getDeductCredits()) {
    throw new BusinessException(EducationBusinessErrorCodeEnum.CREDITS_NOT_ENOUGH.getMsg());</pre>
dto
 enums
event
exception
                                                              return consumerMapper.deductCredits(order.getDeductCredits(), order.getConsumerId());
handler
interceptor
mapper
                                                       @Override
public void updateLoginTimes(Integer id) { consumerMapper.updateLoginTimes(id); }
service
ConsumerServiceImpl
       © 1 CouponServiceImpl
© 1 OrderServiceImpl
                                        78 of
                                                        public int getPageTotal(Consumer consumer) { return consumerMapper.getPageTotal(consumer); }
       © 1 TeacherServiceImpl
```

图 5 扣除积分

如图 6 所示,在 CouponServiceImpl 中的 usedCoupon 方法传入 order 参数,通



图 6 使用优惠券

如图 7 所示,在 OrderServiceImpl 中的 createOrder 方法传入 order 参数,进行 创建订单的操作。在方法体中先通过 order 的 getPayAmount 方法计算出总共要支付的金额,然后设置订单的相关信息包括:支付状态和课程名称。然后依次执行下订单、使用优惠券和扣除积分三个操作,这个三个操作分别来自三个不同的服务

```
(订单、优惠券、消费)。
dto
                                        duverride
                            64 🜒
                                        public void createOrder(Order order) {
enums
                            65
                                            // 根据订单金额获取积分
event
                            66
                                            BigDecimal payAmount = order.getPayAmount();
exception
                                            if (payAmount != null) {
                            67
nandler 🗎
                                                 order.setReceiveCredits(payAmount.intValue());
                            68
intercepto
                            69
mapper mapper
                            70
                                            order.setStatus(OrderStatusEnum.WAITING_FOR_PAY.getStatus());
service
                                            order.setStatusName(OrderStatusEnum.getDescByStatus(order.getStatus()));
order.setCourseName(CourseTypeEnum.getNameByCode(order.getCourse()));
ConsumerServiceImpl
                            73
                                               下订单
    CouponServiceImpl
                                            orderMapper.add(order);
OrderServiceImpl
                            75
                                             // 使用优惠券
    TeacherServiceImpl
                            76
                                            couponService.usedCoupon(order);
  ConsumerService
                            77
                                               扣除积分
  © CouponService
                            78
                                            consumerService.orderCreateInformDeductCredits(order);
  OrderService
                            79
  TeacherService
```

图 7 创建订单

如果使用编程式的事务管理,调用次数的三个操作时需要加入事务控制代码,如果使用声明式事务管理,就需要对配置文件进行修改。

### 5、定义 Spring 事务

为了实现 Spring 的声明式事务管理我们来到 spring-jdbc.xml 文件,它位于resources 目录下面。文件中定义了三个节点,我们从上往下逐一介绍:txManager 中定义了

"org.springframework.jdbc.datasource.DataSourceTransactionManager"这个类是用来管理事务的类,在 property 属性中会引用 dataSource,也就是我们之前定义的数据库连接池。

txAdvice 中定义 txManager 为自己的事务处理模式,在 attributes 的节点中定义了对方法的处理。这个节点使用的 tx: advice,advice 之前我们介绍过就是 Spring AOP 中需要加强的部分,也就是具体实现事务的那一部分。其中通过定义 tx: method 节点来制定那些方法需要执行事务。其中"get\*"和"find\*",表明以这些字符开头的方法使用采用优化的只读事务。因为这里方法基本都是读取操作,一般而言这类操作都不会放到事务中。接下来是对"\*"的定义,也就是对所有的方法进行事务处理。并且定义了"rollback-for"为"java.lang.Exception",意思是当事务执行过程中遇到这种类型的异常会将事务回滚。同时也定义了"no-rollback-for",也就是发生"com.ruyuan.little.project.exception.NoNullbackException"的时候事务不会回滚。最后是 aop:config 的部分,在 pointcut 的部分定义切入点,

Expression 中定义了"execution(\*

com.ruyuan.little.project.spring.service.\*Service.\*(..))"意思是在包名包含了"\*com.ruyuan.little.project.spring.service.Service."的 jar 包都适用于这个切入点。切入点中定义了 advice-ref 为增强的方法,也就是事务的处理方法,以及切入点的引用 serviceOperation。



图 8 spring-jdbc.xml

#### 6.创建 OrderController

至此 Spring 事务定义就完成了,接下来看看创建订单入口的 Controller 定义。如图 9 所示,在 spring/controller 目录下面有 OrderController 类,其中关注 add 方法,路径也是/add,方法体内直接调用 orderService 中的 createOrder 方法传入 order 对象。

```
@RequestMapping("/order")
                                23
                                24
                                       public class OrderController {
                                25
                                26
                                           /**
                                27
                                            * 订单service管理组件
an
                                28
                                            */
ttle
                                           @Autowired
                                29
project
                                30
                                           private OrderService orderService;
🛮 🖿 spring
                                31
 annotion
                                32
                                            /**
 aspect
                                33
                                            * 添加订单
 cache
                                34
 constants
                                35
                                            * @param order 订单信息
  controller
                                36
                                            * @return 结果
      © • ConsumerController
                                37
      © • HealthController
                                           @PostMapping("/add")
                                38
      © 1 JdbcDemoControlle
                                39 👰
                                           public CommonResponse add(@Valid Order order) {
      C G OrderController
                                40
                                                orderService.createOrder(order);
      😊 🖫 TeacherController
                                41
                                                return CommonResponse.success(order);
 dao
                                42
  ▶ 🛅 dto
                               43
 enums
```

图 9 OrderController 定义

#### 7、总结

本节课是 Spring 事务的代码实施课,首先介绍了 Spring 实现事务方式,然后在现有的项目中创建 Consumer、Coupon、Order 的 Mapper 文件,以及对应的 Consumer、Coupon、Order 的服务文件,接下来通过配置文件定义 Spring 事务 的 Advice 和 Pointcuts,最后创建 OrderController 作为创建订单的入口。 下节课一起验证一下系统的下单逻辑是否正确?这里将代码给大家,下期见,拜 拜。