66_技术挑战：如何解决订单、优惠券、积分的数据一致性问题？

1、开篇

上节课通过 Spring 自定义验证的方法，将 Order 中 statrtDate 和 endDate 的验证方式进行类改造。分别进行了 IsDate 注释接口定义，DateValidator 类定义，并将 IsDate 注释到 Order 类的 statrtDate 和 endDate 字段上。本节课会思考，如何解决订单、优惠券、积分的数据一致性问题。今天课程的内容包括以下几个部分：

- 事务的特性
- 创建订单流程中的数据一致性问题

2、事务的特性

在开始对订单流程中数据一致性进行思考之前，我们先来对事物和事物的特性进行复习。

事务（Transaction）是并发控制单位，是用户定义的一个操作序列，这个操作序列中包括一个或者多个操作，这些操作要么都做要么都不做，对于整个操作序列而言是一个不可分割的工作单位。

事务具有四个特征：原子性（ Atomicity ）、一致性（ Consistency ）、隔离性（ Isolation ）和持续性（ Durability ）。这四个特性简称为 ACID 特性。

前面提到了，事务操作序列中的操作要么全部执行，要么都不执行； 如果事务没有原子性的保证，那么在发生系统 故障的情况下，数据库就有可能处于不一致状态。 因而，事务的原子性与一致性是密切相关的。下面将事务的特性逐一为大家介绍：

- 原子性（Atomicity），是指事务包含的所有操作要么全部成功，要么全部失败回滚。
- 一致性（Consistency），是指让数据从一个状态变换到另一个状态，也就是说一个事务执行的前后都必须处于一致性状态。拿转账来说，账户 A、B 无论转账多少次，两个账户的金额之和都是一个一定的数额。
- 隔离性（Isolation），是指在多个请求同时访问数据时，每个请求对于数据的操作都是独立的，不会受到其他请求的影响，也就是并且请求之间是隔离的。
- 持久性（Durability），是指一个事务一旦被提交了，那么对数据的改变是永久的。

3、创建订单流程中的数据一致性问题

在复习完 ACID 的理论以后，再回头看看创建订单流程中是否存在事务，或者说是否存在数据一致性的问题。回到本周第一节课中的创建订单流程图，如图 1 所示，在图中我们发现支付启动支付订单流程之后，系统会做如下操作：创建订单、使用优惠券、扣除积分。这三个操作在消费者有优惠券和积分的时候，会按照先后执行。这个三个操作分别位于订单服务、优惠券服务以及消费者服务中，并且会访问不同的数据库表。从业务的角度来看，这三个操作都在完成创建订单并且为订单付款，也就是我们前面提到的事务。这三个操作要么不完成，如果完成要一起完成，不可能出现创建了订单，但是没有使用优惠券或者没有扣除积分的情况，满足原子性特性。同样用户支付的金额加上优惠券优惠的金额以及积分兑换的金额是等于最终的课程后，满足一致性特性。三个操作对数据的操作也是独立的，满足隔离性特性。最后，三个操作对数据会造成永久的改变，满足持久性特性。鉴于以上分析，我们将这三个操作用虚线框起来，把他们放到一个事务中处理。

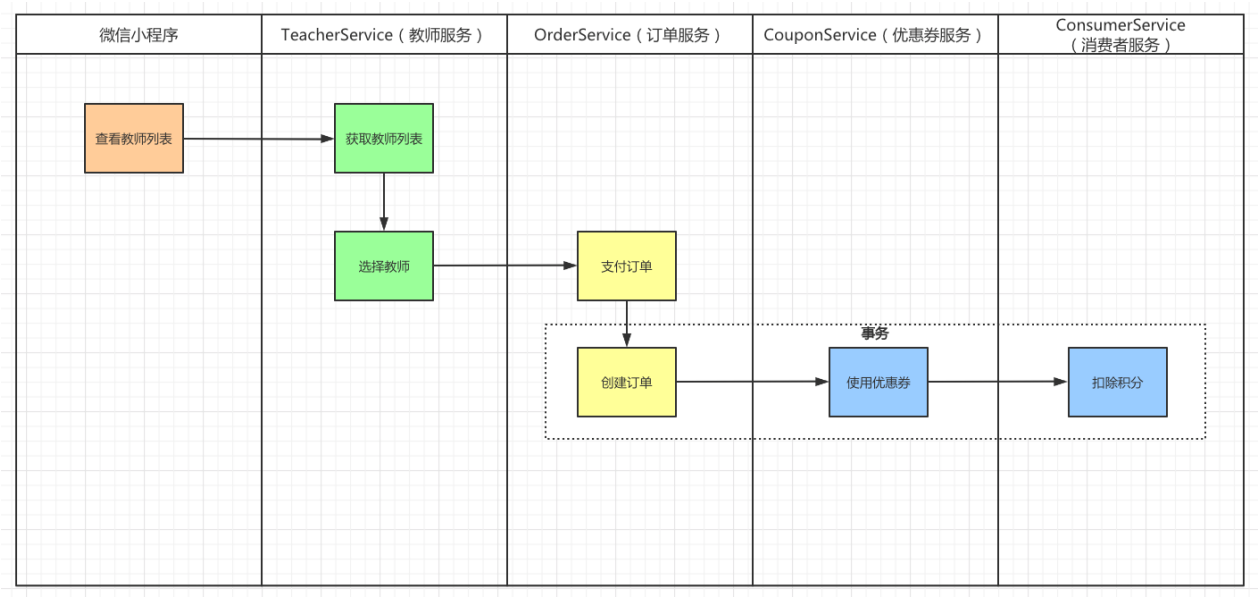

图 1 创建订单流程中的事务

4、总结

本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。下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。下期见，拜拜。