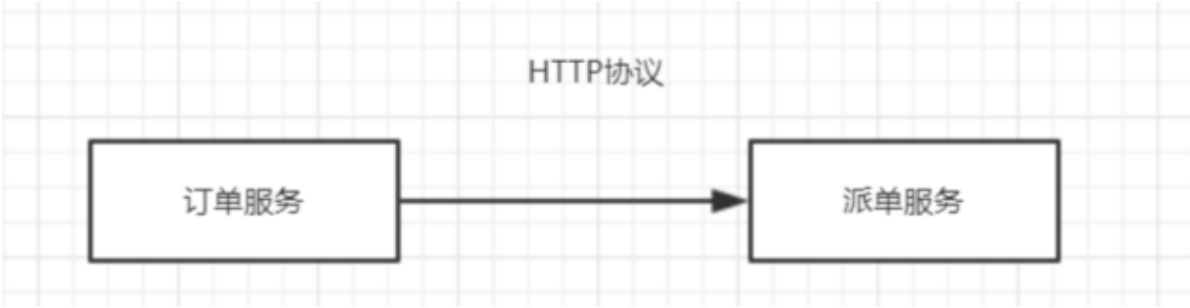
RabbitMQ解决分布式事务问题

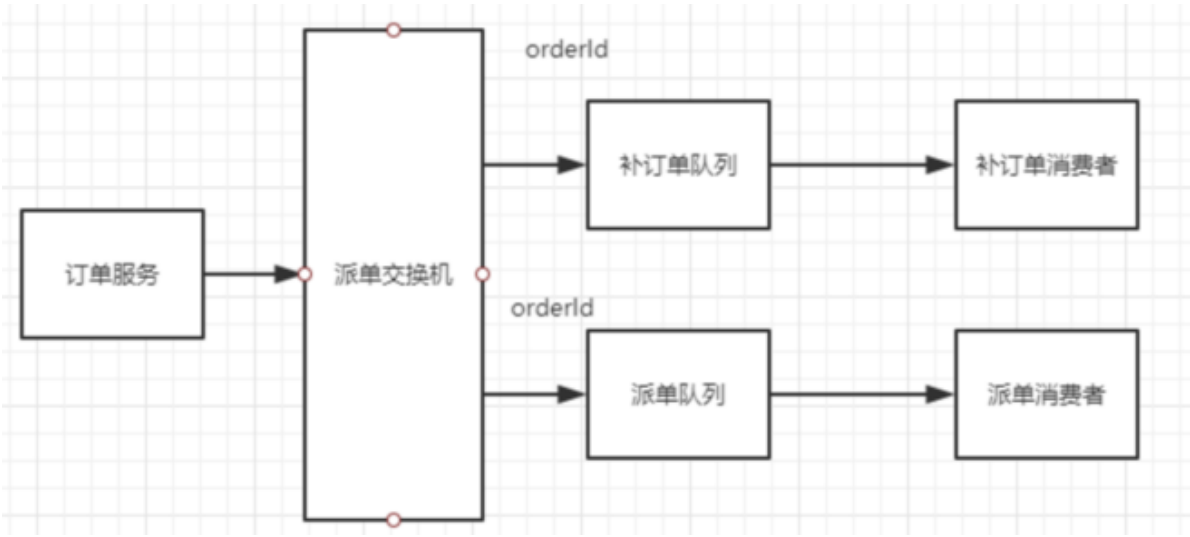
经典案例，以目前流行点外卖的案例，用户下单后，调用订单服务，然后订单服务调用派单系统通知送外卖人员送单，这时候订单系统与派单系统采用MQ异步通讯，保证订单表和派单表必须一致！

用传统的HTTP协议不能解决高并发：



RabbitMQ解决分布式事务原理方案

1. 确保生产者一定要将数据投递到MQ服务器中
    - 生产者采用confirm，确认应答机制
    - 如果失败，生产者进行重试
2. MQ消费者消息能够正常消费消息
    - 采用手动ACK模式，使用补偿机制，注意幂等性问题
3. 采用补单机制
    - 在创建一个补单消费者进行监听，如果订单创建后，又回滚了(数据不一致)，此时需要将订单进行补偿
    - 交换机采用路由键模式，补单队列和派但队列都绑定同一个路由键



派件表：

```sql
CREATE TABLE `platoon` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `order_id` bigint(20) DEFAULT NULL,
  `user_id` bigint(20) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4
```

订单表：

```sql
CREATE TABLE `order_test` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `name` varchar(20) DEFAULT NULL,
  `order_createtime` datetime DEFAULT NULL,
  `order_money` double(10,2) DEFAULT NULL,
  `order_state` int(1) DEFAULT NULL,
  `commodity_id` bigint(20) DEFAULT NULL,
  `order_id` bigint(20) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4
```

生产者

1. 实现接口 implements RabbitTemplate.ConfirmCallback
2. 重写回调方法， 成功、失败的调用

@Override
public void confirm(CorrelationData correlationData, boolean ack, String cause) {

send方法里面调用回调函数：

```
this.rabbitTemplate.setMandatory(true);
this.rabbitTemplate.setConfirmCallback(this);
```

yml需要配置回调机制：

```
###开启消息确认机制 confirms
publisher-confirms: true
publisher-returns: true
```

注意：重试也是有一定次数限制的 如果超过一定次数 就需要进行人工补偿了

代码参照 课件中的源码项目：

- **order-producer** rabbit处理分布式事务处理 ,订单服务 生产者
- **dl-consumer** rabbit处理分布式事务处理 ,派单服务 补单消费者,派单消费者