RocketMQ的幂等性保障

什么是幂等性

那什么是幂等性呢，说人话就是当多次重复请求时，接口能够保证与预期相符的结果。 我们举个例子来说明，例如我们设计了一个为员工涨薪的接口,本次请求发送后为1号员工涨薪500元。

```
PUT https://edu.lagou.com/employee/salary
{"id" : "1","incr_salary":500}
```

如果你是一个新人没有考虑幂等特性，可能会这么写，伪代码如下：

```
//查询1号员工数据
Employee employee = employeeService.selectById(1);
//更新工资
employee.setSalary(employee.getSalary() + incrSalary);
//执行更新语句
employeeService.update(employee)
```

对于这段代码，单独执行没有任何问题，但是我们要注意，在分布式环境下，为了保证消息的高可靠性，往往客户端会采用重试或者消息补偿的形式重复发送同一个请求，那在这种情况下这段代码就会出严重问题，每一个重复请求被发送到服务器都会让该员工工资加500，最终该员工工资会大于要求实际应收工资。

消息队列 RocketMQ 消费者在接收到消息以后，有必要根据业务上的唯一 Key 对消息做幂等处理的必要性。

消费幂等的必要性

在互联网应用中，尤其在网络不稳定的情况下，消息队列 RocketMQ 的消息有可能会出现重复，这个重复简单可以概括为以下情况：

发送时消息重复

当一条消息已被成功发送到服务端并完成持久化，此时出现了网络闪断或者客户端宕机，导致服务端对客户端应答失败。 如果此时生产者意识到消息发送失败并尝试再次发送消息，消费者后续会收到两条内容相同并且 Message ID 也相同的消息。

投递时消息重复

消息消费的场景下，消息已投递到消费者并完成业务处理，当客户端给服务端反馈应答的时候网络闪断。 为了保证消息至少被消费一次，消息队列 RocketMQ 的服务端将在网络恢复后再次尝试投递之前已被处理过的消息，消费者后续会收到两条内容相同并且 Message ID 也相同的消息。

负载均衡时消息重复（包括但不限于网络抖动、Broker 重启以及订阅方应用重启）

当消息队列 RocketMQ 的 Broker 或客户端重启、扩容或缩容时，会触发 Rebalance，此时消费者可能会收到重复消息。

幂等性处理方式

因为 Message ID 有可能出现冲突（重复）的情况，所以真正安全的幂等处理，不建议以 Message ID 作为处理依据。 最好的方式是以业务唯一标识作为幂等处理的关键依据，而业务的唯一标识可以通过消息 Key 进行设置：

```
Message message = new Message();
message.setKey("order_110010");
SendResult sendResult = producer.send(message);
```

订阅方收到消息时可以根据消息的 Key 进行幂等处理：

```
consumer.subscribe("ons_test", "*", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        String key = message.getKey();
        //if(order_110010状态为"等待支付")
            //处理扣款
            //返回支付成功响应
        //else if(order_110010状态为"已支付")
            //返回支付成功响应
    }
});
```

状态机