RabbitMQ如何保证消息的可靠性

1、保证消息不丢失(三步)

- 开启事务(不推荐)
- 开启confirm(推荐)
- 开启RabbitMQ持久化(交换机、队列、消息)
- 关闭RabbitMQ自动ack(改成手动)

2、保证消息不重复消费

幂等性(每个消息用一个唯一标识来区分，消费前先判断标识有没有被消费过，若已消费过，则直接ACK)

3、RabbitMQ如何保证消息的顺序性

将消息放入同一个交换机，交给同一个队列，这个队列只有一个消费者，消费者只允许同时开启一个线程

4、RabbitMQ消息重试机制

消费者在消费消息的时候，如果消费者业务逻辑出现程序异常，这时候应该如何处理?

使用消息重试机制(SpringBoot默认3次消息重试机制)

如何合适选择重试机制?
消费者取到消息后，调用第三方接口，接口无法访问，需要使用重试机制
消费者取到消息后，抛出数据转换异常，不需要重试机制，需要发布者进行解决。

5、SpringBoot消息重试机制

@EnableRetry注解：表示启用重试机制(value表示哪些异常需要触发重试，maxAttempts设置最大重试次数，delay表示重试的延迟时间，multiplier表示上一次延时时间是这一次的倍数)
eg、@Retryable(value = Exception.class, maxAttempts = 3, backoff = @Backoff(delay = 2000, multiplier = 1.5))

@Recover注解：当重试次数达到设置的最大次数的时候，程序还是执行异常，调用的回调函数。

6、RabbitMQ死信队列

死信队列是当消息在一个队列因为下列原因:
a、消息被拒绝(basic.reject或basic.nack)并且requeue=false.
b、消息TTL过期
c、队列达到最大长度(队列满了，数据无法添加到mq中)
变成了 “死信队列” 后被重新投递(publish)到另一个Exchange，然后重新消费。说白了就是没有被消费的消息换个地方重新被消费

7、RabbitMQ解决分布式事务

经典案例，以目前流行的外卖为例，用户下单后，调用订单服务，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。

RabbitMQ解决分布式事务原理
答案：采用最终一致性原理

需要保证以下三要素:
a、确保生产者一定要将数据投递到MQ服务器中(采用MQ消息确认机制)
b、确保消费者能够正确消费消息，采用手动ACK模式(注意重试、幂等性问题)
c、如何保证第一个事务先执行，采用补偿机制，在创建一个补单消费者进行监听，如果订单没有创建成功，进行补单。(如果第一个事务中出错，补单消费者会在重新执行一次第一个事务，例如第一个事务是添加订单表，如果失败在补单的时候重新生成订单记录，由于订单号唯一，所以不会重复)

8、RabbitMQ保证消息不丢失的具体方案

前提:
(1)开启confirm
(2)开启RabbitMQ的持久化(交换机、队列、消息)
(3)关闭RabbitMQ的自动ack(改成手动)
(4)配置消费重试次数，消费重试间隔时间等

涉及到的技术点:
MQ、Redis、定时任务

8.1、保证投放消息不丢失

(1)先将消息放入生产者Redis(此时消息的状态为未投放)，再放入队列
(2)根据confirm(ReturnCallback和ConfirmCallback)的结果来确定消息是否投递成功，
投递成功的，修改生产者redis中消息的投递状态为已投递
投递失败的消息将会放入失败的Redis，并从生产者Redis中删除，由定时任务定期扫描并重新投递

(3)生产者Redis定时任务
生产者Redis定时任务专门扫描生产者Redis中存放了一定时间，但是状态还是未投放的消息
此消息会被认为已经投递，但是没有任何反馈结果(由于不可知因素，导致没有ReturnCallback，也没有ConfirmCallback)，
此类消息被扫描到后，会放入失败的Redis，并从生产者Redis中删除，由定时任务定期扫描并重新投递
(4)还需要一个专门的定时任务扫描生产者Redis中存放了很久，仍然未消费的数据(状态为已投递)，此类消息被扫描到后，会放入失败的Redis，并从生产者Redis中删除，由定时任务定期扫描并重新投递
(5)扫描失败的Redis的定时任务都遵循一条原则，一条消息最多被重新投递三次，若投递了三次仍然失败，则记录日志，记录到数据库，不会再投递，需要人工干预处理

8.2、保证消费消息不丢失

(1)消费者取到消息后，从消息中取出唯一标识，先判断此消息有没有被消费过，若已消费过，则直接ACK(避免重复消费)
(2)正常处理成功后，将生产者Redis中的此消息删除，并ACK(告诉server端此消息已成功消费)
(3)遇到异常时，捕获异常，验证自己在消息中设定的重试次数是否超过阀值，若超过，则放入死信队列，若未超过，则向将消息中的重试次数加1，抛出自定义异常，进入重试机制
(4)有专门的消费者用于处理死信队列中消费多次仍未消费成功的数据，可以记录日志，入库，人工干预处理