### 消息确认机制首先分两块，有发送方（生产者）确认和消费者确认。

### 生产者确认消息

生产者确认消息是为了保证发送方的消息不丢失，为了保证消息不丢失，RabbitMQ提供了两种方式：

①事务机制

②发送方确认机制

事务机制有点类似于spring的编程式事务，需要先通过channel.txSelect将信道设为事务模式，channel.exCommit用于提交事务，channel.rxRollback用于回滚事务，但是事务机制会严重降低性能，所以慎用。

发送方确认机制是指生产者发送消息给Broker并持久化之后，Broker会返回一个确认ACK给生产者，告诉生产者这条消息已经持久化了，但是注意Broker并不会知道告诉你哪些消息没有持久化，因为Broker也不知道你到底有多少消息，因为有可能有的消息甚至没到达Broker就丢失了，所以要想知道哪些消息没有持久化，就需要先记录所有消息，记录成功的消息，然后就可以得出失败的消息。

如何开启发送方确认机制，生产者将信道设置为confirm模式，进入这个模式后，所有在该信道上发布的消息都会拥有一个唯一的id，从1开始，一旦消息持久化后，Broker会返回一个确认（包含消息的唯一id）。但是其实持久化和发送方确认是两回事，不是说要开启了持久化才能使用确认机制，没设置持久化就只需要消息到达Broker就能返回确认，不需要持久化而已。

事务机制是同步的，如果发送方确认机制不设置为异步或者批量确认，其实两者的QPS不会差太多。

1. 单条确认

生产者发送一条后需等待Broker发送确认ack，如果发送多条仍然使用单条确认，那么和事务机制就差不多了。

1. 批量确认

生产者发送一批消息后进行确认，批量确认可以提高效率，但是如果出现未确认的消息，无法确认消息具体是哪一条，生产者需要重发这一批次的消息，会带来重复消息，如果消息经常丢失，批量确认的性能会不升反降。

1. 异步确认

异步确认的实现需要在客户端提供的addConfirmListener方法中添加ConfirmListener这个回调接口，这个接口包含两个方法，handleAck，和handleNack，在这两个方法中都包含一个唯一有序的deliverTag，我们需要维护一个未确认的消息的有序集合，发送消息时记录下所有消息，然后确认成功就删除消息。