**Rabbitmq的个人理解和总结**

消息队列就是指在应用间传递的数据。

是一种应用间的通信方式，消息发送后可以立即返回，有消息系统确保消息可靠传递。

**Rabbitmq的组件：**

message,publisher,exchange,binding,queue,connection,channel,consumer,broker。

生产者把消息发布到exchange上，通过binding使用rootingkey,将exchange和队列进行绑定（queues）,消费者监听queues，获取消息。

**AMQP的三大组件：**

交换器 (Exchange)：消息代理服务器中用于把消息路由到队列的组件。

队列 (Queue)：用来存储消息的数据结构，位于硬盘或内存中。

绑定 (Binding)：一套规则，告知交换器消息应该将消息投递给哪个队列。

**AMQP协议的三层：**

****Module Layer****:协议最高层，主要定义了一些客户端调用的命令，客户端可以用这些命令实现自己的业务逻辑。

****Session Layer****:中间层，主要负责客户端命令发送给服务器，再将服务端应答返回客户端，提供可靠性同步机制和错误处理。

****TransportLayer****:最底层，主要传输二进制数据流，提供帧的处理、信道服用、错误检测和数据表示等。

**不同的分发策略：四种：direct,fanout,topic,headers。**

direct：完全匹配，单播模式，routing key 和binding key 完全一直，交换器将消息发送到绑定的队列中。

fanout:发到fanout类型交换器的消息都会分到所有绑定的队列上。像子网广播。所有fanout类型转发消息速度最快。

topic:topic交换器通过模式匹配分配消息的路由键属性，队列绑定一个模式，路由键和绑定键的的字符串切分成单词，binding key 中匹配的方式#.和.#。

**RabbitMQ 客户端中与事务机制相关的方法有三个:**

channel.txSelect  用于将当前的信道设置成事务模式。

channel . txCommit 用于提交事务 。

channel . txRollback 用于事务回滚,如果在事务提交执行之前由于 RabbitMQ 异常崩溃或者其他原因抛出异常,通过txRollback来回滚。

## **生产者生产消息**

1.Producer先连接到Broker,建立连接Connection,开启一个信道(Channel)。

2.Producer声明一个交换器并设置好相关属性。

3.Producer声明一个队列并设置好相关属性。

4.Producer通过路由键将交换器和队列绑定起来。

5.Producer发送消息到Broker,其中包含路由键、交换器等信息。

6.相应的交换器根据接收到的路由键查找匹配的队列。

7.如果找到，将消息存入对应的队列，如果没有找到，会根据生产者的配置丢弃或者退回给生产者。

8.关闭信道。

9.管理连接。

## **消费者接收消息**

1.Consumer连接到Broker,建立连接Connection,开启一个信道(Channel)。

2.向Broker请求消费响应的队列中消息，可能会设置响应的回调函数。

3.等待Broker回应并投递相应队列中的消息，接收消息。

4.消费者确认收到的消息,ack。

5.RabbitMq从队列中删除已经确定的消息。

6.关闭信道。

7.关闭连接。

## **生产者将消息可靠投递到MQ**

1.Client发送消息给MQ

2.MQ将消息持久化后，发送Ack消息给Client，此处有可能因为网络问题导致Ack消息无法发送到Client，那么Client在等待超时后，会重传消息；

3.Client收到Ack消息后，认为消息已经投递成功。

## **MQ将消息可靠投递到消费者**

1.MQ将消息push给Client（或Client来pull消息）

2.Client得到消息并做完业务逻辑

3.Client发送Ack消息给MQ，通知MQ删除该消息，此处有可能因为网络问题导致Ack失败，那么Client会重复消息，这里就引出消费幂等的问题；

4.MQ将已消费的消息删除

## **RabbitMQ消息队列的高可用**

RabbitMQ 有三种模式：单机模式，普通集群模式，镜像集群模式。

****单机模式****：就是demo级别的，一般就是你本地启动了玩玩儿的，没人生产用单机模式

****普通集群模式****：意思就是在多台机器上启动多个RabbitMQ实例，每个机器启动一个。

****镜像集群模式****：这种模式，才是所谓的RabbitMQ的高可用模式，跟普通集群模式不一样的是，你创建的queue，无论元数据(元数据指RabbitMQ的配置数据)还是queue里的消息都会存在于多个实例上，然后每次你写消息到queue的时候，都会自动把消息到多个实例的queue里进行消息同步。

**与springboot的整合：**

导入依赖：

<!-- rabbitmq依赖 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-amqp</artifactId>

</dependency>

添加配置

RabbitMQ

spring.rabbitmq.host=39.106.128.50

spring.rabbitmq.port=5672

spring.rabbitmq.username=guest

spring.rabbitmq.password=guest

spring.rabbitmq.virtual-host=/

消费者数量

spring.rabbitmq.listener.simple.concurrency=10

spring.rabbitmq.listener.simple.max-concurrency=10

消费者每次从队列中获取的消息数量

spring.rabbitmq.listener.simple.prefetch=1

消费者自动启动

spring.rabbitmq.listener.simple.auto-startup=true

消费失败，自动重新入队

spring.rabbitmq.listener.simple.default-requeue-rejected=true

启用发送重试

spring.rabbitmq.template.retry.enabled=true

spring.rabbitmq.template.retry.initial-interval=1000

spring.rabbitmq.template.retry.max-attempts=3

spring.rabbitmq.template.retry.max-interval=10000

spring.rabbitmq.template.retry.multiplier=1.0

生产者发送消息：

@Component

public class Producer {

@Autowired

RabbitTemplate rabbitTemplate;

public void produce() {

String message=”this is a producer”;

rabbitTemplate.convertAndSend(RabbitKeys.QUEUE\_PLAY, message);

}

}

消费者接收消息

@Component

public class Consumer {

@RabbitHandler

@RabbitListener(queue=RabbitKeys.QUEUE\_PLAY)

public void process(String message) {

System.out.println("消费者消费消息=====" + message);

}

}