# 一、Consumer 批量消费(推模式)

可以通过

```
consumer.setConsumeMessageBatchMaxSize(10);//每次拉取10条
```

## 这里需要分为2种情况

- Consumer端先启动
- Consumer端后启动. 正常情况下:应该是Consumer需要先启动

注意:如果broker采用推模式的话,consumer先启动,会一条一条消息的消费,consumer后启动会才用批量消费

Consumer端先启动

## 1, Consumer.java

```
package quickstart;
import java.util.List;
import com.alibaba.rocketmq.client.consumer.DefaultMQPushConsumer;
import com.alibaba.rocketmq.client.consumer.listener.ConsumeConcurrentlyContext;
import com.alibaba.rocketmq.client.consumer.listener.ConsumeConcurrentlyStatus;
import com.alibaba.rocketmq.client.consumer.listener.MessageListenerConcurrently;
import com.alibaba.rocketmq.client.exception.MQClientException;
import com.alibaba.rocketmq.common.consumer.ConsumeFromWhere;
import com.alibaba.rocketmq.common.message.MessageExt;
* Consumer,订阅消息
public class Consumer {
   public static void main(String[] args) throws InterruptedException, MQClientException {
       DefaultMQPushConsumer consumer = new DefaultMQPushConsumer("please_rename_unique_group_name_4");
       consumer.setNamesrvAddr("192.168.100.145:9876;192.168.100.146:9876");
       consumer.setConsumeMessageBatchMaxSize(10);
        * 设置Consumer第一次启动是从队列头部开始消费还是队列尾部开始消费<br/>
        * 如果非第一次启动,那么按照上次消费的位置继续消费,(消费顺序消息的时候设置)
       consumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME FROM FIRST OFFSET);
       consumer.subscribe("TopicTest", "*");
       consumer.registerMessageListener(new MessageListenerConcurrently() {
           public ConsumeConcurrentlyStatus consumeMessage(List<MessageExt> msgs, ConsumeConcurrentlyContext context) {
               try {
                   System.out.println("msgs的长度" + msgs.size());
                   System.out.println(Thread.currentThread().getName() + " Receive New Messages: " + msgs);
               } catch (Exception e) {
                   e.printStackTrace();
                   return ConsumeConcurrentlyStatus.RECONSUME LATER;
               return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
       });
       consumer.start();
       System.out.println("Consumer Started.");
}
```

由于这里是Consumer先启动,所以他回去轮询MQ上是否有订阅队列的消息,由于每次producer插入一条,Consumer就拿一条所以测试结果如下(每次size都是1)

```
msqs的长度1
ConsumeMessageThread_4 Receive New Messages: [MessageExt [queueId=0, storeSize=138, queueOffset=248, sysFlag=0, bornTimestamp=1487682278750, bornHost=
msqs的长度1
ConsumeMessageThread_3 Receive New Messages: [MessageExt [queueId=1, storeSize=138, queueOffset=248, sysFlag=0, bornTimestamp=1487682278777, bornHost=
ConsumeNessageThread_1 Receive New Messages: [MessageExt [queueId=2, storeSize=138, queueOffset=248, sysFlag=0, bornTimestamp=1487682278789, bornHost=
msgs的长度1
       MessageThread_5 Receive New Messages: [MessageExt [queueId=3, storeSize=138, queueOffset=248, sysFlag=0, bornTimestamp=1487682278817, bornHost=
msqs的长度1
ConsumeMessageThread_6 Receive New Messages: [MessageExt [queueId=0, storeSize=138, queueOffset=248, sysFlag=0, bornTimestamp=1487682278835, bornHost=
ConsumeMessageThread 7 Receive New Messages: [MessageExt [queueId=1, storeSize=138, queueOffset=248, sysFlag=0, bornTimestamp=1487682278847, bornHost=
msgs的长度1
       .
MessageThread 8 Receive New Messages: [MessageExt [queueId=2, storeSize=138, queueOffset=248, sysFlag=0, bornTimestamp=1487682278866, bornHost=
msqs的长度1
        lessageThread_10 Receive New Messages: [MessageExt [queueId=3, storeSize=138, queueOffset=248, sysFlag=0, bornTimestamp=1487682278880, bornHost:
msgs的长度1
ConsumeMessageThread 9 Receive New Messages: [MessageExt [queueId=0, storeSize=138, queueOffset=249, sysFlag=0, bornTimestamp=1487682278895, bornHost=
ConsumeMessageThread_12 Receive New Messages: [MessageExt [queueId=1, storeSize=138, queueOffset=249, sysFlag=0, bornTimestamp=1487682278908, bornHost:
msgs的长度1
        MessageThread_13 Receive New Messages: [MessageExt [queueId=2, storeSize=138, queueOffset=249, sysFlag=0, bornTimestamp=1487682278925, bornHost
msqs的长度1
ConsumeMessageThread 11 Receive New Messages: [MessageExt [queueId=3, storeSize=138, queueOffset=249, sysFlag=0, bornTimestamp=1487682278949, bornHost:
msgs的长度1
ConsumeMessageThread 16 Receive New Messages: [MessageExt [queueId=0, storeSize=138, queueOffset=249, sysFlag=0, bornTimestamp=1487682278959, bornHost:
msgs的长度1
ConsumeMessageThread_15 Receive New Messages: [MessageExt [queueId=1, storeSize=138, queueOffset=249, sysFlag=0, bornTimestamp=1487682278969, bornHost
msgs的长度1
        LessageThread_14 Receive New Messages: [MessageExt [queueId=2, storeSize=138, queueOffset=249, sysFlag=0, bornTimestamp=1487682278986, bornHost
msgs的长度1
ConsumeMessageThread 17 Receive New Messages: [MessageExt [queueId=3, storeSize=138, queueOffset=249, sysFlag=0, bornTimestamp=1487682279003, bornHost:
```

# 2、Consumer端后启动,也就是Producer先启动

由于这里是Consumer后启动,所以MQ上也就堆积了一堆数据,Consumer的

consumer.setConsumeMessageBatchMaxSize(10);//每次拉取10条

# 所以这段代码就生效了<u>测试</u>结果如下(每次size最多是10):

```
msgs的长度10
ConsumeMessageThread 6 Receive New Messages: [MessageExt [queueId
msgs的长度10
ConsumeMessageThread 6 Receive New Messages: [MessageExt [queueId
msas的长度1
ConsumeMessageThread 6 Receive New Messages: [MessageExt [queueId
msgs的长度1
ConsumeMessageThread 6 Receive New Messages: [MessageExt [queueId
ConsumeMessageThread 16 Receive New Messages: [MessageExt [queueI
ConsumeMessageThread 2 Receive New Messages: [MessageExt [queueId
ConsumeMessageThread 13 Receive New Messages: [MessageExt [queueI
ConsumeMessageThread 10 Receive New Messages: [MessageExt [queueI
ConsumeMessageThread 4 Receive New Messages: [MessageExt [queueId
ConsumeMessageThread_14 Receive New Messages: [MessageExt [queueI
ConsumeMessageThread_8/Receive_New_Messages://[MessageExt [queueId
ConsumeMessageThread 17 Receive New Messages: [MessageExt [queueI
ConsumeMessageThread_1 Receive New Messages: [MessageExt [queueId
ConsumeMessageThread_12 Receive New Messages: [MessageExt [queueI
ConsumeMessageThread 7 Receive New Messages: [MessageExt [queueId
ConsumeMessageThread 20 Receive New Messages: [MessageExt [queueI
ConsumeMessageThread 15 Receive New Messages: [MessageExt [queue]
ConsumeMessageThread 5 Receive New Messages: [MessageExt [queueId
ConsumeMessageThread 19 Receive New Messages: [MessageExt [queueI
msgs的长度10
msgs的长度10
ConsumeMessageThread 11 Receive New Messages: [MessageExt [queueI
ConsumeMessageThread 3 Receive New Messages: [MessageExt [queueId
```

## 二、消息重试机制:消息重试分为2种

- 1、Producer端重试
- 2、Consumer端重试

## 1、Producer端重试

也就是Producer往MQ上发消息没有发送成功,我们可以设置发送失败重试的次数,发送并触发回调函数

```
//设置重试的次数
           producer.setRetryTimesWhenSendFailed(3);
           producer.start();
           //创建一条消息
          Message msg = new Message("PushTopic", "push", "1", "我是一条普通消息".getBytes());
           SendResult result = producer.send(msg);
           //发送,并触发回调函数
           producer.send(msg, new SendCallback() {
              @Override
              //成功的回调函数
              public void onSuccess(SendResult sendResult) {
                  System.out.println(sendResult.getSendStatus());
                  System.out.println("成功了");
              @Override
              //出现异常的回调函数
              public void onException(Throwable e) {
              System.out.println("失败了"+e.getMessage());
           });
```

```
[sendStatus=SEND_OK, msgId=COA8649200002A9F000000000042C26, messageQueue=MessageQueue
[sendStatus=SEND_OK, msgId=COA8649200002A9F000000000042C26, messageQueue=MessageQueue
[sendStatus=SEND_OK, msgId=COA8649100002A9F000000000042C98, messageQueue=MessageQueue
[sendStatus=SEND_OK, msgId=COA8649100002A9F0000000000042C92, messageQueue=MessageQueue
[sendStatus=SEND_OK, msgId=COA8649100002A9F000000000000000042C92, messageQueue=MessageQueue
SendResult
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-b, queueId=3], qu
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-a, queueId=0], [topic=TopicTest, brokerName=broker-a, queueId=1],
SendResult
SendResult
                  [sendStatus=SEND_OK,
[sendStatus=SEND_OK,
                                                    msgId=COA8649100002A9F0000000000002D22, messageQueue=MessageQueue
msgId=COA8649100002A9F0000000000042D2C, messageQueue=MessageQueue
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-a, queueId=1], [topic=TopicTest, brokerName=broker-a, queueId=2],
SendResult
SendResult
                   [sendStatus=SEND OK,
                                                    msqId=C0A8649100002A9F0000000000042E36, messageQueue=MessageQueue
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-a, queueId=3],
SendResult
SendResult
                  [sendStatus=SEND_OK,
[sendStatus=SEND_OK,
                                                   msgId=C0A8649200002A9F0000000000042CB0, msgId=C0A8649200002A9F00000000000042D3A,
                                                                                                                     messageQueue=MessageQueue
messageQueue=MessageQueue
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-b, [topic=TopicTest, brokerName=broker-b,
                                                                                                                                                                                                                                 queueId=1],
SendResult
                  [sendStatus=SEND OK,
                                                   msgId=C0A8649200002A9F0000000000042DC4, messageQueue=MessageQueue
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-b, queueId=2],
                                                                                                                                                                                                                                 queueId=3],
queueId=0],
                  [sendStatus=SEND_OK,
[sendStatus=SEND_OK,
                                                    msgId=C0A8649200002A9F0000000000042E4E,
msgId=C0A8649100002A9F0000000000042EC0,
                                                                                                                     messageQueue=MessageQueue
messageQueue=MessageQueue
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-b, [topic=TopicTest, brokerName=broker-a,
SendResult
                                                    msqId=C0A8649100002A9F0000000000042F4A, messageQueue=MessageQueue
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-a, queueId=1], [topic=TopicTest, brokerName=broker-a, queueId=2], [topic=TopicTest, brokerName=broker-a, queueId=3],
SendResult
                  [sendStatus=SEND OK,
SendResult
                  [sendStatus=SEND_OK,
                                                    msqId=C0A8649100002A9F0000000000042FD4,
                                                                                                                      messageQueue=MessageQueue
                   [sendStatus=SEND_OK,
                                                    msgId=C0A8649100002A9F00000000004305E,
                                                                                                                     messageQueue=MessageQueue
                                                   msgId=C0A8649200002A9F0000000000042ED8,
SendResult
                  [sendStatus=SEND OK,
                                                                                                                     messageQueue=MessageQueue
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-b,
                                                                                                                                                                                                                                 queueId=0],
                                                                                                                                                                                                                                 queueId=1],
SendResult
                   sendStatus=SEND OK,
                                                   msgId=C0A8649200002A9F0000000000042F62,
                                                                                                                     messageQueue=MessageQueue
messageQueue=MessageQueue
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-b, [topic=TopicTest, brokerName=broker-b,
                                                    msgId=COA8649200002A9F000000000042FEC, messageQueue=MessageQueue

cgId=COA8649200002A9F0000000000043076, messageQueue=MessageQueue
SendResult
                   [sendStatus=SEND OK
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-b, queueId=3],
SendResult
                                                   msgId=c9A8649100002A9F0000000000430E8, messageQueue=MessageQueue
msgId=C0A8649100002A9F000000000043172, messageQueue=MessageQueue
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-a, queueId=0], [topic=TopicTest, brokerName=broker-a, queueId=1], [topic=TopicTest, brokerName=broker-a, queueId=2],
SendResult
                  [sendStatus=SEND OK,
                  [sendStatus=SEND_OK msgId=C0A8649100002A9F000000000043172, messageQueue=MessageQueue
[sendStatus=SEND_OK msgId=C0A8649100002A9F0000000000431FC, messageQueue=MessageQueue
SendResult
SendResult
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-a, queueId=3], [topic=TopicTest, brokerName=broker-b, queueId=0], [topic=TopicTest, brokerName=broker-b, queueId=1],
                                                   msgId=C0A8649100002A9F000000000043286, messageQueue=MessageQueue
msgId=C0A8649200002A9F000000000043100, messageQueue=MessageQueue
SendResult
                  [sendStatus=SEND OK.
SendResult
                  [sendStatus=SEND_OK, sendStatus=SEND_OK,
                                                   msgId=COA8649200002A9F000000000003100, messageQueue=MessageQueue
msgId=COA8649200002A9F000000000004318A, messageQueue=MessageQueue
SendResult
                                                                                                                                                                 [topic=TopicTest, brokerName=broker-b, queueId=2], [topic=TopicTest, brokerName=broker-b, queueId=3],
SendResult
                  [sendStatus=SEND OK.
                                                   msgId=C0A8649200002A9F0000000000043214, messageOueue=MessageOueue
                  [sendStatus=SEND_OK msgId=C0A8649200002A9F0000000004329E, messageQueue=MessageQueue
[sendStatus=SEND_OK msgId=C0A8649100002A9F00000000043310, messageQueue=MessageQueue
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-b, queueId=3], [topic=TopicTest, brokerName=broker-a, queueId=0],
SendResult
SendResult
SendResult
                  [sendStatus=SEND OK.
                                                   msgId=C0A8649100002A9F000000000004339A,
                                                                                                                     messageOueue=MessageOueue
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-a,
                                                                                                                                                                                                                                 gueueId=11.
                                                                                                                                                                  [topic=TopicTest, brokerName=broker-a, [topic=TopicTest, brokerName=broker-a,
SendResult
                   [sendStatus=SEND_OK, msgId=C0A8649100002A9F0000000000043424,
                                                                                                                     messageQueue=MessageQueue
                                                                                                                                                                                                                                 queueId=2],
SendResult
                  [sendStatus=SEND_OK] msgId=C0A8649100002A9F000000000434AE, messageQueue=MessageQueue
[sendStatus=SEND_OK] msgId=C0A8649200002A9F000000000043328, messageQueue=MessageQueue
                                                                                                                                                                                                                                 queueId=3],
                                                                                                                                                                 [topic=TopicTest, brokerName=broker-b, queueId=0], qu
SendResult
                  [sendStatus=SEND_OK, msgId=C0A8649200002A9F00000000000433B2, messageQueue=MessageQueue [topic=TopicTest, brokerName=broker-b, queueId=1], q=[sendStatus=SEND_OK, msgId=C0A8649200002A9F00000000004343C, messageQueue=MessageQueue [topic=TopicTest, brokerName=broker-b, queueId=2], q=
SendResult
```

## 2、Consumer端重试

#### 2.1、exception的情况,一般重复16次 10s、30s、1分钟、2分钟、3分钟等等

上面的代码中消费异常的情况返回

return ConsumeConcurrentlyStatus.RECONSUME\_LATER;//重试

正常则返回:

return ConsumeConcurrentlyStatus.CONSUME\_SUCCESS;//成功

```
package quickstart;

import java.util.List;

import com.alibaba.rocketmq.client.consumer.DefaultMQPushConsumer;
import com.alibaba.rocketmq.client.consumer.listener.ConsumeConcurrentlyContext;
import com.alibaba.rocketmq.client.consumer.listener.ConsumeConcurrentlyStatus;
import com.alibaba.rocketmq.client.consumer.listener.MessageListenerConcurrently;
import com.alibaba.rocketmq.client.exception.MQClientException;
```

```
import com.alibaba.rocketmq.common.consumer.ConsumeFromWhere;
import com.alibaba.rocketmq.common.message.MessageExt;
* Consumer, 订阅消息
public class Consumer {
   public static void main(String[] args) throws InterruptedException, MQClientException {
       DefaultMQPushConsumer consumer = new DefaultMQPushConsumer("please rename unique group name 4");
       consumer.setNamesrvAddr("192.168.100.145:9876;192.168.100.146:9876");
       consumer.setConsumeMessageBatchMaxSize(10);
       * 设置Consumer第一次启动是从队列头部开始消费还是队列尾部开始消费<br>
       * 如果非第一次启动,那么按照上次消费的位置继续消费
       consumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME FROM FIRST OFFSET);
       consumer.subscribe("TopicTest", "*");
       consumer.registerMessageListener(new MessageListenerConcurrently() {
          try {
                 // System.out.println("msgs的长度" + msgs.size());
                 System.out.println(Thread.currentThread().getName() + " Receive New Messages: " + msgs);
                 for (MessageExt msg : msgs) {
                     String msgbody = new String(msg.getBody(), "utf-8");
                     if (msgbody.equals("Hello RocketMQ 4")) {
                         System.out.println("=====错误=====");
                         int a = 1 / 0;
                  }
              } catch (Exception e) {
                  e.printStackTrace();
                  if (msgs.get(0).getReconsumeTimes() == 3) {
                     //记录日志
                     return ConsumeConcurrentlyStatus.CONSUME SUCCESS;// 成功
                  }else{
                  return ConsumeConcurrentlyStatus.RECONSUME_LATER;// 重试
              return ConsumeConcurrentlyStatus.CONSUME SUCCESS;// 成功
       });
       consumer.start();
       System.out.println("Consumer Started.");
}
```

## 打印结果:

```
java.lang.ArithmeticException: / by zero
  ConsumeMessageThread 5 Receive New Messages: [MessageExt [queueId=0, storeSize=136, queueOffset=503, sys!
  guickstart.Consumer$1.consumeMessage(Consumer.java:57)
         at com.alibaba.rocketmq.client.impl.consumer.ConsumeMessageConcurrentlyService$ConsumeRequest.rur
         at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:471)
         at java.util.concurrent.FutureTask$Sync.innerRun(FutureTask.java:334)
         at java.util.concurrent.FutureTask.run(FutureTask.java:166)
         at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
         at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
         at java.lang.Thread.run(Thread.java:724)
  ConsumeMessageThread_6 Receive New Messages: [MessageExt [queueId=1, storeSize=136, queueOffset=503, sys!
  ConsumeMessageThread 7 Receive New Messages: [MessageExt [queueId=2, storeSize=136, queueOffset=503, sys!
  ConsumeMessageThread 8 Receive New Messages: [MessageExt [queueId=3, storeSize=136, queueOffset=503, sys]
  ConsumeMessageThread_9 Receive New Messages: [MessageExt [queueId=0, storeSize=136, queueOffset=507, sys!
  ConsumeMessageThread_10 Receive New Messages: [MessageExt [queueId=1, storeSize=136, queueOffset=507, sy:
  java.lang.ArithmeticException: / by zeroConsumeMessageThread_11 Receive New Messages: [MessageExt [queue]
  ======错误======
         at quickstart.ConsumerS1.consumeMessage(Consumer.java:57)
         \verb|at com.alibaba.rocketmq.client.impl.consumer.ConsumeMessageConcurrentlyService\\§ConsumeRequest.rur|
         at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:471)
         at java.util.concurrent.FutureTask$Sync.innerRun(FutureTask.java:334)
         at java.util.concurrent.FutureTask.run(FutureTask.java:166)
         at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
         at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
                   6437, reconsumeTimes=0, preparedTransactionOffset=0, toString()=Message [topic=TopicTest,
     reconsumeTimes=0, preparedTransactionOffset=0, toString()=Message [topic=TopicTest, f
070,l
                                            重试次数
                                 http://blog.csdn.net/u010634288
3152, reconsumeTimes=0, preparedTransactionOffret=0, toString()=Message [topic=TopicTest,
2618, reconsumeTimes=0, preparedTransactionOffsat=0, toString()=Message [topic=TopicTest,
972, reconsumeTimes=0, preparedTransactionOffset=0, toString()=Message [topic=TopicTest, f 109, reconsumeTimes=0, preparedTransactionOffset=0, toString()=Message [topic=TopicTest, f
77195, reconsumeTimes=0, preparedTransactionOffset=0, toString()=Message [topic=TopicTest,
commitLogOffset=278259, bodyCRC=601994070, reconsumeTimes=1, preparedTransactionOffset=0,
假如超过了多少次之后我们可以让他不再重试记录 日志。
if(msgs.get(0).getReconsumeTimes()==3){
//记录日志
return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;// 成功
```

## 2.2超时的情况,这种情况MQ会无限制的发送给消费端。

就是由于网络的情况,MQ发送数据之后,Consumer端并没有收到导致超时。也就是消费端没有给我返回return 任何状态,这样的就认为没有到达Consumer端。

这里模拟Producer只发送一条数据。consumer端暂停1分钟并且不发送接收状态给MQ

```
package model;
import java.util.List;

import com.alibaba.rocketmq.client.consumer.DefaultMQPushConsumer;
import com.alibaba.rocketmq.client.consumer.listener.ConsumeConcurrentlyContext;
import com.alibaba.rocketmq.client.consumer.listener.ConsumeConcurrentlyStatus;
import com.alibaba.rocketmq.client.consumer.listener.MessageListenerConcurrently;
import com.alibaba.rocketmq.client.exception.MQClientException;
```

```
import com.alibaba.rocketmq.common.consumer.ConsumeFromWhere;
import com.alibaba.rocketmq.common.message.MessageExt;
* Consumer, 订阅消息
public class Consumer {
   public static void main(String[] args) throws InterruptedException, MQClientException {
       DefaultMQPushConsumer consumer = new DefaultMQPushConsumer("message consumer");
       consumer.setNamesrvAddr("192.168.100.145:9876;192.168.100.146:9876");
       consumer.setConsumeMessageBatchMaxSize(10);
        * 设置Consumer第一次启动是从队列头部开始消费还是队列尾部开始消费<br>
        * 如果非第一次启动,那么按照上次消费的位置继续消费
       consumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME FROM FIRST OFFSET);
       consumer.subscribe("TopicTest", "*");
       consumer.registerMessageListener(new MessageListenerConcurrently() {
           try {
                   // 表示业务处理时间
                   System.out.println("======开始暂停=======");
                   Thread.sleep(60000);
                   for (MessageExt msg : msgs) {
                       System.out.println(" Receive New Messages: " + msg);
               } catch (Exception e) {
                   e.printStackTrace();
                   return ConsumeConcurrentlyStatus.RECONSUME LATER;// 重试
               return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;// 成功
           }
       });
       consumer.start();
       System.out.println("Consumer Started.");
}
🚜 Servers 📮 Console 🌣 🖶 Progress 💂 Devices 📮 LogCat 🦹 Markers
                                                                                       <terminated > Consumer (1) [Java Application] D:\JDK1.7_64\jre\bin\javaw.exe (2017年2月22日 上午12:10:45)
   SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder" SLF4J: Defaulting to no-operation (NOP) logger implementation
   SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
   Consumer Started.
      =====开始暂停====
                                             Consumer端停止 之后
                                                                                       👭 Servers 📃 Console 🖾 🔫 Progress 📱 Devices 📮 LogCat 🦹 Markers
  Consumer2 [Java Application] D:\JDK1.7_64\jre\bin\javaw.exe (2017年2月22日 上午12:10:51)
  SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
  SLF4J: Defaulting to no-operation (NOP) logger implementation
  SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
  Consumer Started.
   Receive New Messages: MessageExt [queueId=0, storeSize=136, queueOffset=513, sysFlag=0, bornTimestamp=1487693459294,
                              Consumer2端就马上收到消息了
```

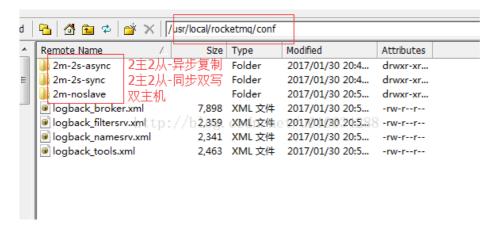
# 三、消费模式

# rocketMQ默认是集群消费,我们可以通过在Consumer来支持广播消费

consumer.setMessageModel(MessageModel.BROADCASTING);// 广播消费

```
package model;
import java.util.List;
import com.alibaba.rocketmq.client.consumer.DefaultMQPushConsumer;
import com.alibaba.rocketmq.client.consumer.listener.ConsumeConcurrentlyContext;
import com.alibaba.rocketmq.client.consumer.listener.ConsumeConcurrentlyStatus;
import com.alibaba.rocketmq.client.consumer.listener.MessageListenerConcurrently;
import com.alibaba.rocketmq.client.exception.MQClientException;
import com.alibaba.rocketmq.common.consumer.ConsumeFromWhere;
import com.alibaba.rocketmq.common.message.MessageExt;
import com.alibaba.rocketmq.common.protocol.heartbeat.MessageModel;
* Consumer,订阅消息
public class Consumer2 {
   public static void main(String[] args) throws InterruptedException, MQClientException {
       DefaultMQPushConsumer consumer = new DefaultMQPushConsumer("message_consumer");
       consumer.setNamesrvAddr("192.168.100.145:9876;192.168.100.146:9876");
       consumer.setConsumeMessageBatchMaxSize(10);
       consumer.setMessageModel(MessageModel.BROADCASTING);// 广播消费
       \verb|consumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME\_FROM\_FIRST\_OFFSET);|
       consumer.subscribe("TopicTest", "*");
       consumer.registerMessageListener(new MessageListenerConcurrently() {
           try {
                  for (MessageExt msg : msgs) {
                      System.out.println(" Receive New Messages: " + msg);
               } catch (Exception e) {
                  e.printStackTrace();
                  return ConsumeConcurrentlyStatus.RECONSUME LATER; // 重试
              return ConsumeConcurrentlyStatus.CONSUME SUCCESS;// 成功
       });
       consumer.start();
       System.out.println("Consumer Started.");
   }
}
```

# 四、conf下的配置文件说明



异步复制和同步双写主要是主和从的关系。消息需要实时消费的,就需要采用主从模式部署

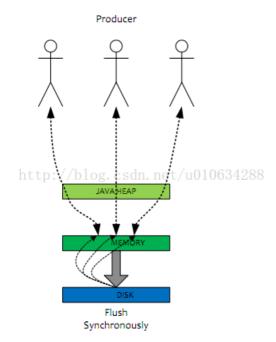
异步复制:比如这里有一主一从,我们发送一条消息到主节点之后,这样消息就算从producer端发送成功了,然后通过异步复制的方法将数据复制 到从节点

同步双写:比如这里有一主一从,我们发送一条消息到主节点之后,这样消息就并不算从producer端发送成功了,需要通过同步双写的方法将数据同步到从节点后, 才算数据发送成功。

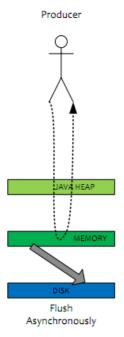
如果rocketMq才用双master部署,Producer往MQ上写入20条数据 其中Master1中拉取了12条。 Master2中拉取了8条,这种情况下,Master1宕机,那么我们消费数据的时候,只能消费到Master2中的8条,Master1中的12条默认持久化,不会丢失消息,需要Master1恢复之后这12条数据才能继续被消费,如果想保证消息实时消费,就才用双Master双Slave的模式

# 五、刷盘方式

同步刷盘:在消息到达MQ后,RocketMQ需要将数据持久化,同步刷盘是指数据到达内存之后,必须刷到commitlog日志之后才算成功,然后返回producer数据已经发送成功。



异步刷盘:,同步刷盘是指数据到达内存之后,返回producer说数据已经发送成功。,然后再写入commitlog日志。



# commitlog:

commitlog就是来存储所有的元信息,包含消息体,类似于<u>MySQL</u>、<u>Oracle</u>的redolog,所以主要有CommitLog在,Consume Queue即使数据丢失,仍然可以恢复出来。

consumequeue:记录数据的位置,以便Consume快速通过consumequeue找到commitlog中的数据