3.6：消费端如何做限流量

一:什么是消费端的限流

场景:首先，我们迎来了订单的高峰期，在mq的broker上堆积了成千上万条消息没有处理，这个时候，我们随便打开了消费者，就会出现下面请

如此多的消息瞬间推送给消费者，我们的消费者不能处理这么多消息 就会导致消费者出现巨大压力，甚至服务器崩溃

解决方案:

rabbitmq 提供一个钟qos（服务质量保证）,也就是在关闭了消费端的自动ack的前提下，我们可以设置阈值（出队）的消息数没有被确认（手动确认），那么就不会推送消息过来.

限流的级别(consumer级别或者是channel级别)

实现的方式 void BasicQos(uint prefetchSize,ushort prefetchCount ,bool global)

uint prefetchSize ：**指定的是设定消息的大小**(rabbitmq还没有该功能，所以一般是填写0表示不限制)

ushort perfetchCount：**表示设置消息的阈值，每次过来几条消息(一般是填写1 一条 一条的处理消息)**

bool global：**表示是channel级别的还是 consumer的限制**(channel的限制rabbitmq 还没有该功能)

代码演示:（生产者）

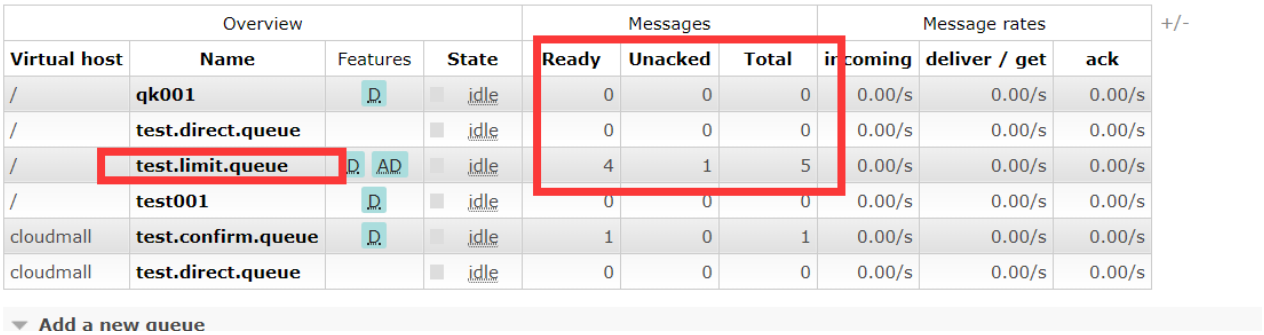
```
public class producter {
public static void main(String[] args) throws IOException, TimeoutException {
 ConnectionFactory connectionFactory = new ConnectionFactory();
 connectionFactory.setVirtualHost("/");
 connectionFactory.setHost("47.104.128.12");
 connectionFactory.setPort(5672);
Connection connection = connectionFactory.newConnection();
Channel channel = connection.createChannel();
//发送五条消息
for(int i=0;i<5;i++) {
 channel.basicPublish("test.limit.exchange","test.limit.key",false,null,"自定义消费端消息".getBytes());
 }
}
```

消费者:

```
public static void main(String[] args) throws IOException, TimeoutException {
 ConnectionFactory connectionFactory = new ConnectionFactory();
 connectionFactory.setVirtualHost("/");
 connectionFactory.setHost("47.104.128.12");
 connectionFactory.setPort(5672);
Connection connection = connectionFactory.newConnection();
Channel channel = connection.createChannel();
channel.exchangeDeclare("test.limit.exchange","direct",true,true,false,null);
 channel.queueDeclare("test.limit.queue",true,false,true,null);
 channel.queueBind("test.limit.queue","test.limit.exchange","test.limit.key");
//gloabl设置为ture 那么就是channel级别的限流，若为false 就是consumer级别的限制流量
 channel.basicQos(0,1,false);
//关闭自动签收
 channel.basicConsume("test.limit.queue",false,new AngleCustomConsumer(channel));
}


自定义消费者监听
public class AngleCustomConsumer extends DefaultConsumer {

    private Channel channel;
    /**
     * Constructs a new instance and records its association to the passed-in channel.
     *
     * @param channel the channel to which this consumer is attached
     */
    public AngleCustomConsumer(Channel channel) {
        super(channel);
        this.channel = channel;
    }

    /**
     * 处理消息
     * @param consumerTag
     * @param envelope
     * @param properties
     * @param body
     * @throws IOException
     */
    public void handleDelivery(String consumerTag,Envelope envelope,AMQP.BasicProperties properties, byte[] bo
            throws IOException
    {
        System.out.println("自定义的消息消费端");
        System.out.println("consumerTag="+consumerTag);
        System.out.println("envelope="+envelope);
        System.out.println("properties="+properties);
        System.out.println("body="+new String(body));

        //消费端的手动签收,假如关闭手动签收，也关闭自动签收，那么消费端只会接收到一条消息
        //channel.basicAck(envelope.getDeliveryTag(),false);
```

```
    }
}
```

运行结果:



Overview				Messages			Message rates			+/-
Virtual host	Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
/	qk001	D	idle	0	0	0	0.00/s	0.00/s	0.00/s	
/	test.direct.queue		idle	0	0	0	0.00/s	0.00/s	0.00/s	
/	test.limit.queue	D AD	idle	4	1	5	0.00/s	0.00/s	0.00/s	
/	test001	D	idle	0	0	0	0.00/s	0.00/s	0.00/s	
cloudmall	test.confirm.queue	D	idle	1	0	1	0.00/s	0.00/s	0.00/s	
cloudmall	test.direct.queue		idle	0	0	0	0.00/s	0.00/s	0.00/s	

▼ Add a new queue