关键技术点记录:

1. **消息选择器的使用**

百度资料：

一、序言

       消息大多数情况都是发送到broker 的，在知道Destination 的情况下，都可以消费，因此有些情况下需要我们将消息分组、隔离，或则指定A消息，只能有A消费者消费等等情况，这里做个大概的介绍和实例。

二、实例场景

       我们通过有时候我们需要一个queue/topic 通道，然后发送消息，但是我们要对不同消费者接受的消息进行限制，或者说过滤，就可以使用这种情况。

       我这里建立一个queue,分别发送 300条A,B消息，分别有消费者A,B接收

三、代码实例

       基本连接代码这里就就不贴了，建议前的

       发送者代码：

**Java代码  [收藏代码](javascript:void())**

1. Destination send\_destination = session.createQueue("order\_queue");
2. MessageProducer producer = session.createProducer(send\_destination);
3. **for**(**int** i =0;i<300;i++){
4. // 创建一个文本消息
5. TextMessage message =  session.createTextMessage("A-张三-"+i);
6. // 这里我们分别设置对应的消息信息，当成是一组消息
7. message.setStringProperty("JMSXGroupID","A");
8. producer.send(message);
10. TextMessage message1 =  session.createTextMessage("B-李四-"+i);
11. message1.setStringProperty("JMSXGroupID","B");
12. producer.send(message1);
13. }

   消费者代码：

**Java代码  [收藏代码](javascript:void())**

1. Destination destination = session.createQueue("order\_queue");
2. // 创建消费者
3. MessageConsumer consumer = session.createConsumer(destination,"JMSXGroupID='A'");
4. consumer.setMessageListener(**new** MessageListener() {
5. @Override
6. **public** **void** onMessage(Message message) {
7. TextMessage textMessage = (TextMessage) message;
8. **try** {
9. System.out.println("A："+textMessage.getText());
10. } **catch** (JMSException e) {
11. e.printStackTrace();
12. }
13. }
14. });

   消费者B

**Java代码  [收藏代码](javascript:void())**

1. // 指定接收消息的地方
2. Destination destination = session.createQueue("order\_queue");
3. // 创建消费者
4. MessageConsumer consumer = session.createConsumer(destination,"JMSXGroupID='B'");
5. consumer.setMessageListener(**new** MessageListener() {
6. @Override
7. **public** **void** onMessage(Message message) {
8. TextMessage textMessage = (TextMessage) message;
9. **try** {
10. System.out.println("B:"+textMessage.getText());
11. } **catch** (JMSException e) {
12. e.printStackTrace();
13. }
14. }
15. });

    然后开启A,B消费者监听，启动发送者，那么就能看到消息分别消费了

    同时Selector 支持一些表达式的过滤，比如可以写成：JMSXGroupID = 'A' or JMSXGroupID = 'B'

 更多可参考：http://activemq.apache.org/features.html

四、其他情况分析

       1.如果A挂掉，那么B收消息会收到影响，因为在同一个队列，A 积压的消息量越多，那么B收到最后消息的时间越长。道理很简单，A挂了，B前面把大部分消息收到了，后面只有少量B的消息以及积压的A消息，分配几率一定的情况下，B发送的时间就拖久了。

       2.如果消费者不指定"JMSXGroupID='A'" 这个，那么会默认收取未收到的所有消息，他会默认当成一个分组了。

       3.这种模式和应答模式结合的时候，有朋友测试会卡主，但是我一直没重现~。~ 希望多测测

五、小结

        1.AMQ 提供的这个东西还是有一定用处，虽然可以减少了我们的通道的数量，同样的会照成通道压力过大，小范围的消息是可以的。

         2.有问题请指出，有其他的场景欢迎分享，或者需要满足其他场景的可以一起讨论。

1. **队列的超时设置(无法设置队列超时)，信息大小设置**

其他：

[Per Destination Policies](http://activemq.apache.org/per-destination-policies.html)

译文：每个目的地的配置项如下：

目的地公共配置项

The properties you can set on a Destination are as follows:

|  |  |  |
| --- | --- | --- |
| Common property | default | description |
| producerFlowControl | true | the producer will slow down and eventually block if no resources(e.g. memory) are available on the broker. If this is off messages get off-lined to disk to prevent memory exhaustion  producerFlowControl表示是否监控流量，默认为true，如果设置为false，消息就会存在磁盘中以防止内存溢出；memoryLimit表示在producerFlowControl=”true”的情况下，消息存储在内存中最大量，当消息达到这个值时，ActiveMQ会减慢消息的产生甚至阻塞。 |
| enableAudit | true | tracks duplicate messages (which can occur in failover for non-persistent messages)；  跟踪重复消息（在非持久化消息传递将发生错误） |
| useCache | true | persistent messages are cached for fast retrieval from store:  缓存持久化的信息 以便能尽快从存储提取(暂时未用到信息持久化) |
| maxPageSize | 200 | maximum number of messages to page from store at a time  控制进入内存中的消息数量。用于大数据传输 |
| maxBrowsePageSize | 400 | maximum number of messages to page from store for a browser  增加控制使用当前消费者列表的预取值，用于大数据传输 |
| memoryLimit | n/a | The memory limit for a given destination. This acts as a child to the overall broker memory specified by the[**<systemUsage>'s memoryLimit attribute**](http://activemq.apache.org/producer-flow-control.html). There is no default for this value; it simply acts as a child to the overall broker memory until the broker memory is exhausted.  目的地存储的大小限制，这个作为broker整体存储的子配置项，（broker整体存储大小在<systemUsage>的memoryLimit 属性里配置），此属性并没有默认值。 |
| minimumMessageSize  最小信息大小 | 1024 | for non-serialized messages (embedded broker) - the assumed size of the message used for memory usage calculation. Serialized messages used the serialized size as the basis for the memory calculation  对于非序列化信息，信息假设大小，用于存储计算。序列化信息使用序列化的大小作为存储计算的依据。 |
| cursorMemoryHighWaterMark | 70 | the percentage (%) tipping point at which a system memory limit will cause a cursor to block or spool to disk  系统存储的临界点，超过会引起阻塞或者暂时将信息写到磁盘。 |
| storeUsageHighWaterMark | 100 | the percentage (%) tipping point at which a system usage store limit will cause a sent to block  信息存储允许使用的最大百分比。 |
| prioritizedMessages | false | have the store respect message priority  根据消息的优先级进行存储 |
| advisoryForConsumed | false | send an advisory message when a message is consumed by a client  当客户端消费完信息后发送一条报告信息。 |
| advisoryForDelivery | false | send an advisory message when a message is sent to a client  当信息发送给客户端后发送一条报告 |
| advisoryForSlowConsumers | false | send an advisory message if a consumer is deemed slow  当一个消费者被认定缓慢之后，发一条公告。 |
| advsioryForFastProducers | false | send an advisory message if a producer is deemed fast  当一个生产者被认定过快之后，发一条公告。 |
| advisoryWhenFull | false | send an advisory message when a limit (memory,store,temp disk) is full  当存储被占满之后发一条公告。 |
| includeBodyForAdvisory | false | includes the body of the original message that triggered the advisory as part of the dataStructure field in the advisory message (when applicable). Normally the message body is cleared.  保留原始报文的主体部分，最为报告信息的一部分。  通常情况下，消息的主题会被清掉。  和上面advisoryForConsumed和advisory**ForDelivery选项配合使用** |
| gcInactiveDestinations | false | delete inactive destination  回收无效的destination |
| inactiveTimoutBeforeGC | 5000 | inactivity period (in ms) before destination is considered inactive  被认定为无效destination的超时时间 |
| usePrefetchExtension | true | the prefetch extension is used when a message is delivered but not acked, such that the broker can dispatch another message (e.g., prefetch == 0), the idea being that there will always be prefetch num messages pending. It also allows a transaction batch to exceed the prefetch value.  预取机制，即当一条信息被发送后并未被确认，代理服务仍可以传送另一条信息给相同消费者， |
| slowConsumerStrategy | null | sets the strategy for handling slow consumers. see [abortSlowConsumerStrategy](http://svn.apache.org/repos/asf/activemq/trunk/activemq-core/src/main/java/org/apache/activemq/broker/region/policy/AbortSlowConsumerStrategy.java)  为处理消息缓慢的消费者设定策略 |
| maxDestinations | -1 | (v5.12) if 0 or greater, sets the maximum number of destinations that can be created. This is useful to limit the number of hierarchical destinations that can be created under a wildcard destination.  在同一个destination建立虚拟Destinations的最大数量。 |

队列特有配置项

Additional properties for a Queue

|  |  |  |
| --- | --- | --- |
| Queue only property | default | description |
| useConsumerPriority | true | use the priority of a consumer when dispatching messages from a Queue  根据消费者的优先级派送信息 |
| strictOrderDispatch | false | if true queue will not round robin consumers, but it'll use a single one until its prefetch buffer is full  如果设定为true，消息将会一直派发给一个消费者，直到它的预取缓存器被占满 |
| optimizedDispatch | false | don't use a separate thread for dispatching from a Queue  调度优化，不使用独立线程从队列调度信息。 |
| lazyDispatch | false | only page in from store the number of messages that can be dispatched at time  增加控制使用当前消费者列表的预取值；大数据传输使用 |
| consumersBeforeDispatchStarts | 0 | when the first consumer connects, wait for specified number of consumers before message dispatching starts  第一个消费者连接后，等待到指定消费者都连接后派送信息。 |
| timeBeforeDispatchStarts | 0 | when the first consumer connects, wait for specified time (in ms) before message dispatching starts  第一个消费者连接后，等待到指定时间后派送信息。 |
| queuePrefetch | n/a | sets the prefetch for consumers that are using the default value  为消费者设置预取信息的默认值 |
| expireMessagesPeriod | 30000 | the period (in ms) of checks for message expiry on queued messages, value of 0 disables  检查排队信息到期的时间。设定为0，不可用 |
| persistJMSRedelivered | false | (v 5.10) if true, before a persistent message is dispatched by the broker for the first time, the message is rewritten to reflect the possible delivery.  This ensures the message JMSRedelivered header is a reliable indication of possible duplicate delivery.  持久信息 |

使用方法：

<policyEntry queue=">" producerFlowControl="true" memoryLimit="1mb">

  <pendingQueuePolicy>

    <vmQueueCursor/>

  </pendingQueuePolicy>

</policyEntry>

1. **消息重发设置，以及死亡队列的设置**

消息重新发送：

在以下三种情况中，消息会被再次传送给消费者：

1．在使用事务的Session中，调用rollback()方法；

2．在使用事务的Session中，调用commit()方法之前就关闭了Session;

3．在Session中使用CLIENT\_ACKNOWLEDGE签收模式，并且调用了recover()方法。

消息过期设置有两种方法：

1， MessageProducer.setTimeToLive()

所有用此种方法发送的消息都公用。

2，MessageProducer.send() ，只对当前发送的消息有效！

配置死亡队列：

<destinationPolicy>

<policyMap>

<policyEntries>

<policyEntry queue=">">

<deadLetterStrategy>

<individualDeadLetterStrategy queuePrefix="DLQ." useQueueForQueueMessages="true" processExpired="false"

processNonPersistent="false"/>

</deadLetterStrategy>

</policyEntry>

</policyEntries>

</policyMap>

</destinationPolicy>

1. **存储设置**

设置消息在内存、磁盘中存储的大小，配置如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <systemUsage>    <systemUsage>        <memoryUsage>            <memoryUsage limit="20 mb"/>        </memoryUsage>        <storeUsage>            <storeUsage limit="1 gb"/>        </storeUsage>        <tempUsage>            <tempUsage limit="100 mb"/>        </tempUsage>    </systemUsage>  </systemUsage> |

memoryUsage表示ActiveMQ使用的内存，这个值要大于等于destinationPolicy中设置的所有队列的内存之和。

storeUsage表示持久化存储文件的大小。

tempUsage表示非持久化消息存储的临时内存大小。