链滴

ActiveMQ 学习笔记

作者：shiweichn

原文链接：https://ld246.com/article/1535965542057

来源网站：链滴

关于ActiveMQ更多详细的介绍可以参考下面的链接：

https://www.ilanni.com/?p=13543

https://www.cnblogs.com/xinhuaxuan/p/6119912.html

ActiveMQ必须要引用的jar文件，参考下面链接

http://activemq.apache.org/initial-configuration.html

ActiveMQ 的 Queue：

ActiveMQConnectionFactory: 是一个管理对象，通过它来创建连接，这个类实现了 QueueConnectionFactory 和 TopicConnectionFactory。所以可以通过这个对象来创建 Queue连接和Topic连接

Connection：这个是通过 ActiveMQConnectionFactory创建来的连接，通过这个连接可以创建 Session对象。无论是Sender还是Receiver都是通过它来创建。

Session：这个是通过Connection创建来的Session对象，可以作为Sender ，也可以作为 Receiver。

Queue ：这是一个队列，也是ActiveMQ的核心之一，Sender将数据放到这里，Receiver也是从这取出数据。Queue也是通过Session创建，并且Sender和Receiver都要创建一个相同的Queue，然后能协同工作。

MessageProducer：这个是通过Session创建来的，并且在创建此对象是要指定它要发送数据的Queu， 它是数据的生产者（Producer），所以调用send()方法就可以发送数据。

MessageConsumer: 这个是通过Session创建来的，并且在创建此对象是要指定它要读取数据的Que e， 它是数据的消费者（Consumer），将Consumer实现MessageListener接口并重写 onMessage(方法，可以更简单完成接收数据的功能。

注意：

Connection需要通过调用start()方法来进行启动，启动之后才能接收到消息。

Queue方式默认会持久化数据，在data目录下。通过 MessageProducer 的 setDeliveryMode(DeliveryMode.NON_PERSISTENT); 可以设置是否持久化数据。

由于Queue 是 Point-to-Point 方式，所以Queue中的每个数据只能被一个Consumer所接收，

也就是 如果有多个Consumer从同一个Queue中取数据，那么同一条数据仅能被一个Consumer所接，然后其它Consumer读取数据时，只能读取下一条数据。

ActiveMQ 的 Topic：

Topic：是ActiveMQ的核心之一，也是通过Session对象创建的。

Topic方式与Queue方式的主要区别就是：Broker中topic中的消息是以广播的形式发送出去，也就是一条消息将被发送给所有连上该Broker的Consumer。

- 持续订阅：

创建方式和Queue差不多，只是 TopicSubscriber durableSubscriber = session.createDurableSubcriber(topic, "t2");

理论上Consumer可以接收到Topic中的所有数据。

例如： 如果持久订阅者中途宕机，那么Broker会等到持久订阅者再次连上Broker，然后将错过的消再发送给持久订阅者。

造成的影响是：当持久订阅者处于 inactive 状态时，Broker需要为持久订阅者保存消息，如果持久阅者订阅的消息太多则会溢出。(当消息投递成功之后，Broker就可以把消息删除了)

但仅限于中途宕机的情况，如果一个新的Consumer第一次连上一个已经启动了的Broker，那么在此onsumer连上Broker之前的所有消息，Broker不会再发给这个Consumer，无论是否设置消息缓存。

- 非持续订阅：

一般方式：

创建一般方式和Queue基本一样，只是 MessageConsumer messageConsumer = session.creatConsumer(topic);

非持续订阅的一般方式：无论是第一次连上还是中途宕机，只要是在Consumer离线过程中错过的消，当Consumer再连上Broker时，Broker都不会重发那些错过的数据，而是从最新的消息开始发。

所以 此种方式是可靠性不高。

提高非持久订阅者的可靠性 以及 订阅恢复策略

Retroactive Consumers：是一种非持续订阅者，创建方式是在创建Topic的时候指定consumer为retoactive，例如 String topicName = "simon.topic?consumer.retroactive=true"

由于非持续订阅者可能中途宕机或者比Broker运行的要晚，所以可能会漏掉一些消息。此时，可以使一些 恢复策略 来保证非持续订阅的可靠性。例如：

```
    <subscriptionRecoveryPolicy>
      <!-- 缓存最新的100条数据，若中途有Consumer连上Broker，那么Broker将这100条数据给Consumer. -->
      <fixedCountSubscriptionRecoveryPolicy maximumSize="100"/>
    </subscriptionRecoveryPolicy>
```

还有其他一些恢复订阅策略就不一 一介绍了。总之，恢复订阅策略针对的是非持久化的retroactive consumer订阅者而言的。它提高了非持久化消息的可靠性。

http://activemq.apache.org/subscription-recovery-policy.html

之前也了解过一点消息中间件，但却没有实际应用过。这次刚好项目需要，趁此机会，仔细了解了下顺便做点笔记。其实也没啥好记的，官网内容很全，也很容易上手。