链滴

JMS 与 Active MQ 入门、Spring 整合 Active MQ

作者：DongXiaokai0819

原文链接：https://ld246.com/article/1582636993099

来源网站：链滴

JMS与Active MQ入门

JMS是什么

JMS（Java Messaging Service）是Java平台上有关面向消息中间件的技术规范，实际上是一套api
它便于消息系统中的Java应用程序进行消息交换,并且通过提供标准的产生、发送、接收消息的接口简
企业应用的开发，ActiveMQ而是这个规范的一个具体实现。JMS是一种与厂商无关的 API，用来访
收发系统消息，它类似于JDBC.

JMS对象模型

- **连接工厂**：连接工厂负责创建一个JMS连接。
- **JMS连接**：JMS连接（Connection）表示JMS客户端和服务器端之间的一个活动的连接，是由客
端通过调用连接工厂的方法建立的。
- **JMS会话**：JMS会话（Session）表示JMS客户与JMS服务器之间的会话状态。JMS会话建立在JM
连接上，表示客户与服务器之间的一个会话线程。
- **JMS目的/ Broker**：客户用来指定它生产的消息的目标和它消费的消息的来源的对象，一个消息中
件的实例。
- **JMS生产者和消费者**：生产者（Message Producer）和消费者（Message Consumer）对象由Se
sion对象创建，用于发送和接收消息。

JMS中的消息

JMS 消息由以下三部分组成：

- **消息头**：每个消息头字段都有相应的getter 和setter 方法。
- **消息属性**：如果需要除消息头字段以外的值，那么可以使用消息属性。
- **消息体**：JMS 定义的消息类型有TextMessage、MapMessage、BytesMessage、StreamMessag
和 ObjectMessage。ActiveMQ也有对应的实现。

JMS消息模型

上次的《基于异步消息模式的通信》中已对此有所介绍，这里再详细说一下：

Point-to-Point(P2P) / 点对点

    消息通过称为队列的
个虚拟通道来进行交换。队列是生产者发送消息的目的地和接受者消费消息的消息源。

     每条消息通仅会传送给
个接受者。可能会有多个接受者在一个队列中侦听，但是每个队列中的消息**只能被队列中的一个接受
消费。**

     消息存在先后顺序。
个队列会按照消息服务器将消息放入队列中的顺序，把它们传送给消费者当消息已被消费时，就会从
列头部将它们删除。

    每个消息

有一个消费者（Consumer）(即一旦被消费，消息就不再在消息队列中)

    发送者发送了消息之
，不管接收者有没有正在运行，它不会影响到消息被发送到队列

    接收者在成功接收消
之后需向队列应答成功

    如果希望发送的每个
息都应该被成功处理的话，使用这个P2P模式。

Topic/ 主题（发布订阅(Pub/Sub)）

    消息生产者（发布）
消息发布到topic中，同时有多个消息消费者（订阅）消费该消息。和点对点方式不同，发布到topic
消息会**被所有订阅者消费。**

    如果你希望发送的消
可以不被做任何处理、或者被一个消息者处理、或者可以被多个消费者处理的话，那么可以采用topi
模型

消息的消费方式

1. **同步消费**：通过调用 消费者的receive 方法从目的地中显式提取消息。receive 方法可以一直阻塞
消息到达。

2. **异步消费**：客户可以为消费者注册一个消息监听器，以定义在消息到达时所采取的动作。

Active MQ入门

Active MQ 下载与启动

- 下载： http://activemq.apache.org/components/classic/download/，下载后解压

- 启动：运行bin目录下的activemq.bat即可。Linux下操作类似（进入bin目录运行./activemq start
动，./activemq stop关闭）

    运行后在浏览器中访问
ttp://127.0.0.1/admin，即可看到ActiveMQ的管理控制台

ActiveMQ中，**61616**为服务端口，**8161**为管理控制台端口。

使用原生Active MQ

新建一个maven项目，导入Active MQ的pom依赖。

```
<dependency>
   <groupId>org.apache.activemq</groupId>
   <artifactId>activemq-all</artifactId>
   <version>5.8.0</version>
</dependency>
```

消息生产端

```
public class JmsProducer {

    /*默认连接用户名*/
    private static final String USERNAME
            = ActiveMQConnection.DEFAULT_USER;
    /* 默认连接密码*/
    private static final String PASSWORD
            = ActiveMQConnection.DEFAULT_PASSWORD;
    /* 默认连接地址*/
    private static final String BROKEURL
            = ActiveMQConnection.DEFAULT_BROKER_URL;
    private static final int SENDNUM = 3;

    public static void main(String[] args) {
        /* 连接工厂*/
        ConnectionFactory connectionFactory;
        /* 连接*/
        Connection connection = null;
        /* 会话*/
        Session session;
        /* 消息的目的地*/
        Destination destination;
        /* 消息的生产者*/
        MessageProducer messageProducer;

        /* 实例化连接工厂*/
        connectionFactory = new ActiveMQConnectionFactory(USERNAME,PASSWORD,
                BROKEURL);
        try {
            /* 通过连接工厂获取连接*/
            connection = connectionFactory.createConnection();
            /* 启动连接*/
            connection.start();
            /* 创建session
            * 第一个参数表示是否使用事务，第二次参数表示是否自动确认*/
            session = connection.createSession(false,
                    Session.AUTO_ACKNOWLEDGE);
            /* 创建一个名为HelloWorld消息队列*/
            //destination = session.createTopic("HelloActiveMq");
            destination = session.createQueue("HelloActiveMqQueue");
            /* 创建消息生产者*/
            messageProducer = session.createProducer(destination);
            /* 循环发送消息*/
            for(int i=0;i<SENDNUM;i++){
                String msg = "发送消息"+i+" "+System.currentTimeMillis();
                TextMessage textMessage = session.createTextMessage(msg);
                System.out.println("标准用法:"+msg);
                messageProducer.send(textMessage);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            if(connection!=null){
                try {
```

```
                connection.close();
            } catch (JMSException e) {
                e.printStackTrace();
            }
        }

    }
  }
}
```

消息消费端

```
public class JmsConsumer {
    /*默认连接用户名*/
    private static final String USERNAME
            = ActiveMQConnection.DEFAULT_USER;
    /* 默认连接密码*/
    private static final String PASSWORD
            = ActiveMQConnection.DEFAULT_PASSWORD;
    /* 默认连接地址*/
    private static final String BROKEURL
            = ActiveMQConnection.DEFAULT_BROKER_URL;

    public static void main(String[] args) {
        /* 连接工厂*/
        ConnectionFactory connectionFactory;
        /* 连接*/
        Connection connection = null;
        /* 会话*/
        Session session;
        /* 消息的目的地*/
        Destination destination;
        /* 消息的消费者*/
        MessageConsumer messageConsumer;

        /* 实例化连接工厂*/
        connectionFactory
                = new ActiveMQConnectionFactory(USERNAME,PASSWORD,BROKEURL);

        try {
            /* 通过连接工厂获取连接*/
            connection = connectionFactory.createConnection();
            /* 启动连接*/
            connection.start();
            /* 创建session*/
            session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
            /* 创建一个名为HelloWorld消息队列*/
            //destination = session.createTopic("HelloActiveMq");
            destination = session.createQueue("HelloActiveMqQueue");
            /* 创建消息消费者*/
            messageConsumer = session.createConsumer(destination);
            Message message;
            while((message = messageConsumer.receive())!=null){
                System.out.println(((TextMessage)message).getText());
```

```
            }
        } catch (JMSException e) {
            e.printStackTrace();
        }finally {
            if(connection!=null){
                try {
                    connection.close();
                } catch (JMSException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

消息异步消费端

```
public class JmsConsumerAsyn {
    /*默认连接用户名*/
    private static final String USERNAME
            = ActiveMQConnection.DEFAULT_USER;
    /* 默认连接密码*/
    private static final String PASSWORD
            = ActiveMQConnection.DEFAULT_PASSWORD;
    /* 默认连接地址*/
    private static final String BROKEURL
            = ActiveMQConnection.DEFAULT_BROKER_URL;

    public static void main(String[] args) {
        /* 连接工厂*/
        ConnectionFactory connectionFactory;
        /* 连接*/
        Connection connection = null;
        /* 会话*/
        Session session;
        /* 消息的目的地*/
        Destination destination;
        /* 消息的消费者*/
        MessageConsumer messageConsumer;

        /* 实例化连接工厂*/
        connectionFactory
                = new ActiveMQConnectionFactory(USERNAME,PASSWORD,BROKEURL);

        try {
            /* 通过连接工厂获取连接*/
            connection = connectionFactory.createConnection();
            /* 启动连接*/
            connection.start();
            /* 创建session*/
            session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
            /* 创建一个名为HelloWorld消息队列*/
            //destination = session.createTopic("HelloActiveMq");
            destination = session.createQueue("HelloActiveMqQueue");
```

```
            /* 创建消息消费者*/
            messageConsumer = session.createConsumer(destination);
            /* 设置消费者监听器，监听消息*/
            messageConsumer.setMessageListener(new MessageListener() {
                public void onMessage(Message message) {
                    try {
                        System.out.println(((TextMessage)message).getText());
                    } catch (JMSException e) {
                        e.printStackTrace();
                    }
                }
            });
        } catch (JMSException e) {
            e.printStackTrace();
        }
    }
}
```

Active MQ在Spring中的使用

1、添加依赖

    首先我们先要搭建一个
pring 的maven项目。

    然后我们在pom文件
添加active mq的依赖与：

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jms</artifactId>
    <version>4.3.11.RELEASE</version>
</dependency>
```

2、配置文件applicationContext.xml

命名空间的添加

```
xmlns:amq="http://activemq.apache.org/schema/core"
http://activemq.apache.org/schema/core
http://activemq.apache.org/schema/core/activemq-core-5.8.0.xsd
```

消费者的命名空间要再额外添加如下：

```
xmlns:jms="http://www.springframework.org/schema/jms"
http://www.springframework.org/schema/jms
http://www.springframework.org/schema/jms/spring-jms-4.0.xsd
```

ActiveMQ 连接工厂

```
<amq:connectionFactory id="amqConnectionFactory"
        brokerURL="tcp://127.0.0.1:61616" userName="" password="" />
```

Spring Caching连接工厂

```
<!-- Spring用于管理真正的ConnectionFactory的ConnectionFactory -->
<bean id="connectionFactory"
      class="org.springframework.jms.connection.CachingConnectionFactory">
    <property name="targetConnectionFactory" ref="amqConnectionFactory"></property>
    <property name="sessionCacheSize" value="100"></property>
</bean>
```

3、消息生产者配置以及代码的编写

Spring JmsTemplate 的消息生产者

```
<!-- 定义JmsTemplate的Queue类型 -->
<bean id="jmsQueueTemplate" class="org.springframework.jms.core.JmsTemplate">
    <constructor-arg ref="connectionFactory"></constructor-arg>
    <!-- 队列模式-->//true为发布订阅模式
    <property name="pubSubDomain" value="false"></property>
</bean>
```

然后便可以使用JmsTemplate

```
@Autowired
@Qualifier("jmsQueueTemplate")
private JmsTemplate jmsTemplate;

public void send(String queueName,final String message){
    jmsTemplate.send(queueName, new MessageCreator() {
        public Message createMessage(Session session) throws JMSException {
            Message msg = session.createTextMessage(message);
            //TODO  应答
            return msg;
            }
        });
```

3、消息消费者配置以及代码的编写

定义Queue监听器

```
<jms:listener-container destination-type="queue" container-type="default"
                        connection-factory="connectionFactory" acknowledge="auto">
    <jms:listener destination="test.queue" ref="queueReceiver1"></jms:listener>
    <jms:listener destination="test.queue" ref="queueReceiver2"></jms:listener>
</jms:listener-container>
```

queueReceiver1的编写

```
@Component
public class QueueReceiver1 implements MessageListener {
    public void onMessage(Message message) {
        try {
            String textMsg = ((TextMessage)message).getText();
            System.out.println("QueueReceiver1 accept msg : "+textMsg);
```

```
        } catch (JMSException e) {
            e.printStackTrace();
        }
    }
}
```