# 前提

.什么是消息中间件 ？都是用在没有返回值的前提下使用

消息中间件利用高效可靠的消息传递机制进行平台无关的数据交流，并基于数据通信来进行分布式系统的集成。通过提供消息传递和消息排队模型，它可以在分布式环境下扩展进程间的通信。对于消息中间件，常见的角色大致也就有Producer（生产者）、Consumer（消费者）

常见的消息中间件产品:

（1）ActiveMQ

ActiveMQ 是Apache出品，最流行的，能力强劲的开源消息总线。ActiveMQ 是一个完全支持JMS1.1和J2EE 1.4规范的 JMS Provider实现。我们在本次课程中介绍 ActiveMQ的使用。

（2）RabbitMQ

AMQP协议的领导实现，支持多种场景。淘宝的MySQL集群内部有使用它进行通讯，OpenStack开源云平台的通信组件，最先在金融行业得到运用。

（3）ZeroMQ

史上最快的消息队列系统

（4）Kafka

Apache下的一个子项目 。特点：高吞吐，在一台普通的服务器上既可以达到10W/s的吞吐速率；完全的分布式系统。适合处理海量数据。

# JDK的jms介绍

JMS（Java Messaging Service）是Java平台上有关面向消息中间件的技术规范，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、接收消息的接口简化企业应用的开发。

JMS本身只定义了一系列的接口规范，是一种与厂商无关的 API，用来访问消息收发系统。它类似于 JDBC(java Database Connectivity)：这里，JDBC 是可以用来访问许多不同关系数据库的 API，而 JMS 则提供同样与厂商无关的访问方法，以访问消息收发服务。许多厂商目前都支持 JMS，包括 IBM 的 MQSeries、BEA的 Weblogic JMS service和 Progress 的 SonicMQ，这只是几个例子。 JMS 使您能够通过消息收发服务（有时称为消息中介程序或路由器）从一个 JMS 客户机向另一个 JML 客户机发送消息。消息是 JMS 中的一种类型对象，由两部分组成：报头和消息主体。报头由路由信息以及有关该消息的元数据组成。消息主体则携带着应用程序的数据或有效负载。

JMS 定义了五种不同的消息正文格式，以及调用的消息类型，允许你发送并接收以一

些不同形式的数据，提供现有消息格式的一些级别的兼容性。

• TextMessage--一个字符串对象

• MapMessage--一套名称-值对

• ObjectMessage--一个序列化的 Java 对象

• BytesMessage--一个字节的数据流

• StreamMessage -- Java 原始值的数据流

# 使用步骤

## 安装和启动

（1）官方网站下载：http://activemq.apache.org/

（2）解压此文件

tar zxvf apache-activemq-5.12.0-bin.tar.gz

（3）为apache-activemq-5.12.0目录赋权

chmod 777 apache-activemq-5.12.0

（4）进入apache-activemq-5.12.0\bin目录

（5）赋与执行权限

chmod 755 activemq

-------------------------------------- 知识点小贴士 --------------------------

linux 命令chmod 755的意思

chmod是Linux下设置文件权限的命令，后面的数字表示不同用户或用户组的权限。

一般是三个数字：

第一个数字表示文件所有者的权限

第二个数字表示与文件所有者同属一个用户组的其他用户的权限

第三个数字表示其它用户组的权限。

权限分为三种：读（r=4），写（w=2），执行（x=1） 。 综合起来还有可读可执行（rx=5=4+1）、可读可写（rw=6=4+2）、可读可写可执行(rwx=7=4+2+1)。

所以，chmod 755 设置用户的权限为：

1.文件所有者可读可写可执行 --7

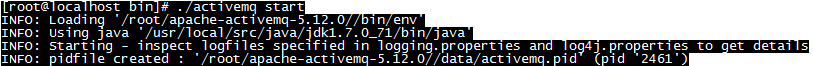
2.与文件所有者同属一个用户组的其他用户可读可执行 --5

3.其它用户组可读可执行 --5

启动

命令 ./activemq start

出现下列提示表示成功！



假设服务器地址为106.12.106.21 ，打开浏览器输入地址

http:// 106.12.106.21:8161/ 即可进入ActiveMQ管理页面

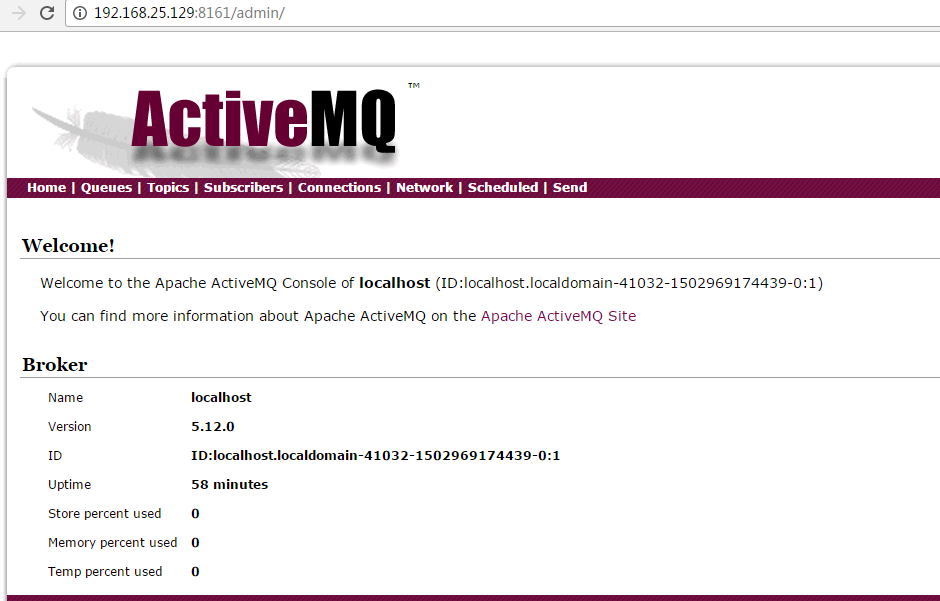
点击进入管理页面 点击manager---------



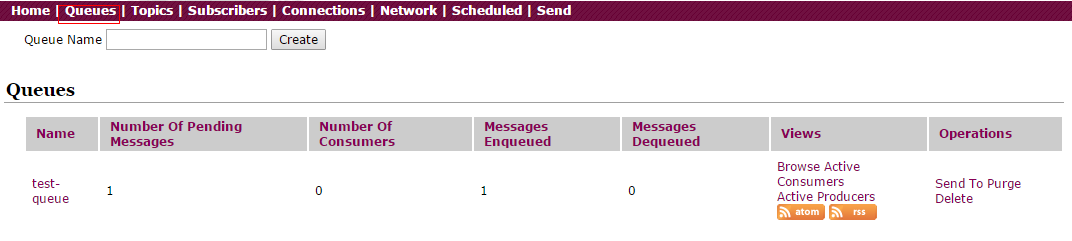
输入用户名和密码 均为 admin



进入主界面



点对点消息列表：



列表各列信息含义如下：

Number Of Pending Messages ：等待消费的消息 这个是当前未出队列的数量。

Number Of Consumers ：消费者 这个是消费者端的消费者数量

Messages Enqueued ：进入队列的消息 进入队列的总数量,包括出队列的。

Messages Dequeued ：出了队列的消息 可以理解为是消费这消费掉的数量。

## 原生步骤

### 点对点模式步骤

producer生产者发送信息至activeMQ

第一步：导包

<dependencies>

<dependency>

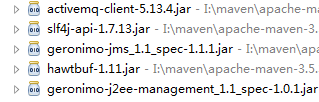
<groupId>org.apache.activemq</groupId>

<artifactId>activemq-client</artifactId>

<version>5.13.4</version>

</dependency>

</dependencies>



第二步：main方法写步骤

public class QueueProducer {

public static void main(String[] args) throws JMSException {

//1.创建连接工厂 注意：8161端口是界面中的端口，连接activeMq端口是：61616，tcp协议

//ActiveMQConnectionFactory ctionFactory= new ActiveMQConnectionFactory("tcp://106.12.106.21:61616");

//建议用jdk提供的jms规范接口接收：

ConnectionFactory connectionFactory = new ActiveMQConnectionFactory("tcp://106.12.106.21:61616");

//2、创建连接

Connection connection = connectionFactory.createConnection();

//3、启动连接

connection.start();

//4、获取session(会话对象)

//参数1：是否启动事务：(和数据事务一样的有回滚等)，

//参数2：消息的确认方式：枚举类型有四个参数:1\AUTO\_AXKNO。。：自动确认,2\CLIENT\_\*客户端手动确认3\DUPS\_\*自动批量确认4\SESSION\_\*事务提交并确认

Session session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);

//5、创建队列对象,参数：字符串名称（自定义）这个名称消费者那边要用到

Queue queue = session.createQueue("test-wwl");

//6、创建消息生产者对象,小知识：按F4可以查看对象的继承结构

MessageProducer producer = session.createProducer(queue);

//7、创建消息对象(消息类型)：1、TextMessage--字符串对象2、MapMessage--键值对3、ObjectMessage--一个序列化的 Java 对象4、BytesMessage--一个字节的数据流5、StreamMessage -- Java 原始值的数据流

TextMessage textMessage = session.createTextMessage("欢迎来到神奇的activeMQ");

//8、最关键的：发送消息

producer.send(textMessage);

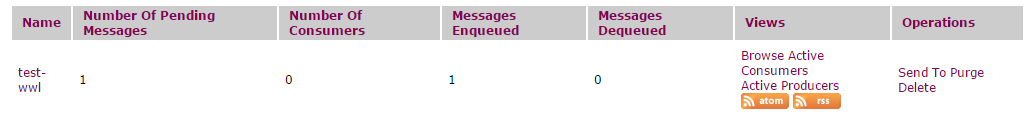
//9、关闭资源

producer.close();

session.close();

connection.close();

执行后打开ActiveMQ页面在点对点就会有一条未消费的记录出现



消费者消费取出消息

导包之类的都是一样的

直接代码：代码中1到5步和生产者生产消息一致

1到4步省略………

//5、创建队列对象,参数：取值需要和生产者名称一致

Queue queue = session.createQueue("test-wwl");

//6、创建消费者对象

MessageConsumer consumer = session.createConsumer(queue);

//7、设置监听

consumer.setMessageListener(new MessageListener() {

@Override

public void onMessage(Message message) {

TextMessage textMessage=(TextMessage) message;

try {

System.out.println("提取消息："+textMessage.getText());

} catch (JMSException e) {

e.printStackTrace();

}

}

});

//8、等待键盘输入

System.in.read();

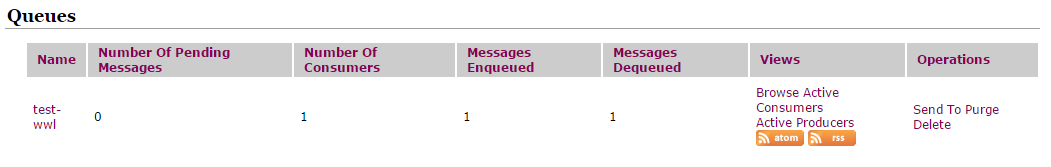
//9、关闭资源

consumer.close();

session.close();

connection.close();

执行后打开ActiveMQ页面在点对点就会有一条未消费的记录出现



测试

把消费者先关掉，连续执行两次生产者代码：会出现Number Of Pending：2个记录，Number Of Consumers：显示0个消费者，把消费者打开就会有执行2次消息

结论：如果Number Of Pending有多条记录，没有消费者，就会等待，一旦有消费者就会全部执行，如果多个消费者，只会是第一个连接消费者取走，其它消费者不会得到，这就是点对点

### 发布/订阅模式步骤

producer生产者发送信息至activeMQ

第一步：导包

<dependencies>

<dependency>

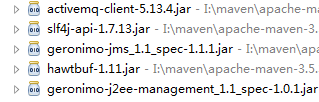
<groupId>org.apache.activemq</groupId>

<artifactId>activemq-client</artifactId>

<version>5.13.4</version>

</dependency>

</dependencies>



第二步：main方法写步骤 和点对点模式只有第五步不一样，其它都一致

public class QueueProducer {

public static void main(String[] args) throws JMSException {

//1.创建连接工厂 注意：8161端口是界面中的端口，连接activeMq端口是：61616，tcp协议

//ActiveMQConnectionFactory ctionFactory= new ActiveMQConnectionFactory("tcp://106.12.106.21:61616");

//建议用jdk提供的jms规范接口接收：

ConnectionFactory connectionFactory = new ActiveMQConnectionFactory("tcp://106.12.106.21:61616");

//2、创建连接

Connection connection = connectionFactory.createConnection();

//3、启动连接

connection.start();

//4、获取session(会话对象)

//参数1：是否启动事务：(和数据事务一样的有回滚等)，

//参数2：消息的确认方式：枚举类型有四个参数:1\AUTO\_AXKNO。。：自动确认,2\CLIENT\_\*客户端手动确认3\DUPS\_\*自动批量确认4\SESSION\_\*事务提交并确认

Session session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);

//5、创建主题对象

Topic topic = session.createTopic("test-wwl2");

//6、创建消息生产者对象,小知识：按F4可以查看对象的继承结构

MessageProducer producer = session.createProducer(topic);

//7、创建消息对象(消息类型)：1、TextMessage--字符串对象2、MapMessage--键值对3、ObjectMessage--一个序列化的 Java 对象4、BytesMessage--一个字节的数据流5、StreamMessage -- Java 原始值的数据流

TextMessage textMessage = session.createTextMessage("点对点和发布订阅者模式只有第五步不一样其它都一样");

//8、最关键的：发送消息

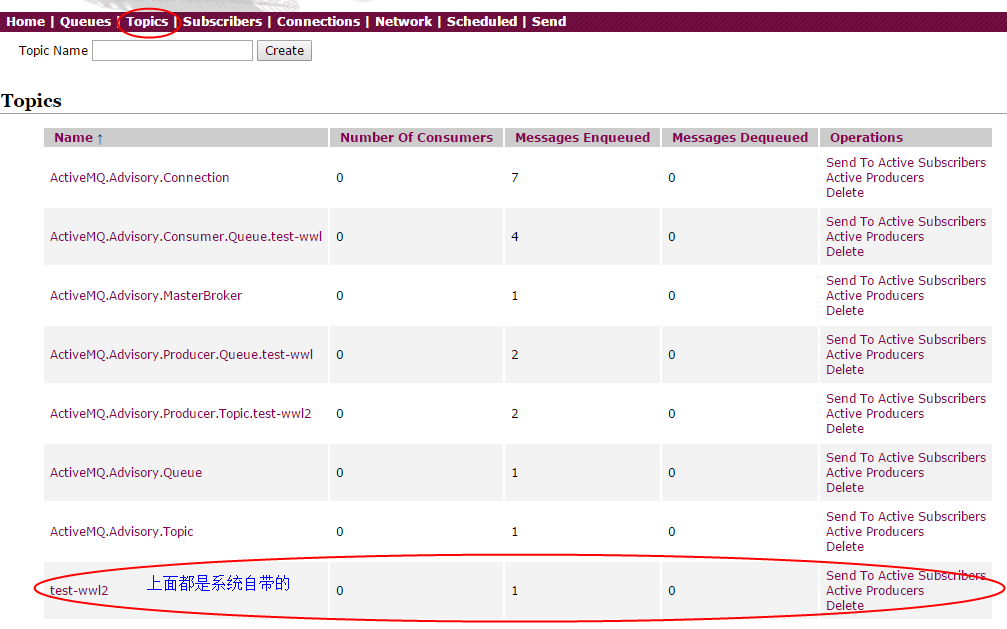
producer.send(textMessage);

//9、关闭资源 producer.close();

session.close();

connection.close();

执行后打开ActiveMQ页面在topics就会有一条未消费的记录出现



消费者消费取出消息

导包和上面一样

//1.创建连接工厂 注意：8161端口是界面中的端口，连接activeMq端口是：61616，tcp协议

//ActiveMQConnectionFactory activeMQConnectionFactory= new ActiveMQConnectionFactory("tcp://106.12.106.21:61616");

//建议用jdk提供的jms规范接口接收：

ConnectionFactory connectionFactory = new ActiveMQConnectionFactory("tcp://106.12.106.21:61616");

//2、创建连接

Connection connection = connectionFactory.createConnection();

//3、启动连接

connection.start();

//4、获取session(会话对象)

//参数1：是否启动事务：(和数据事务一样的有回滚等)，

//参数2：消息的确认方式：枚举类型有四个参数:1\AUTO\_AXKNO。。：自动确认,2\CLIENT\_\*客户端手动确认3\DUPS\_\*自动批量确认4\SESSION\_\*事务提交并确认

Session session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);

//5、创建主题对象

Topic topic = session.createTopic("test-wwl2");

//6、创建消息生产者对象,小知识：按F4可以查看对象的继承结构

MessageProducer producer = session.createProducer(topic);

//7、创建消息对象(消息类型)：1、TextMessage--字符串对象2、MapMessage--键值对3、ObjectMessage--一个序列化的 Java 对象4、BytesMessage--一个字节的数据流5、StreamMessage -- Java 原始值的数据流

TextMessage textMessage = session.createTextMessage("点对点和发布订阅者模式只有第五步不一样其它都一样");

//8、最关键的：发送消息

producer.send(textMessage);

//9、关闭资源

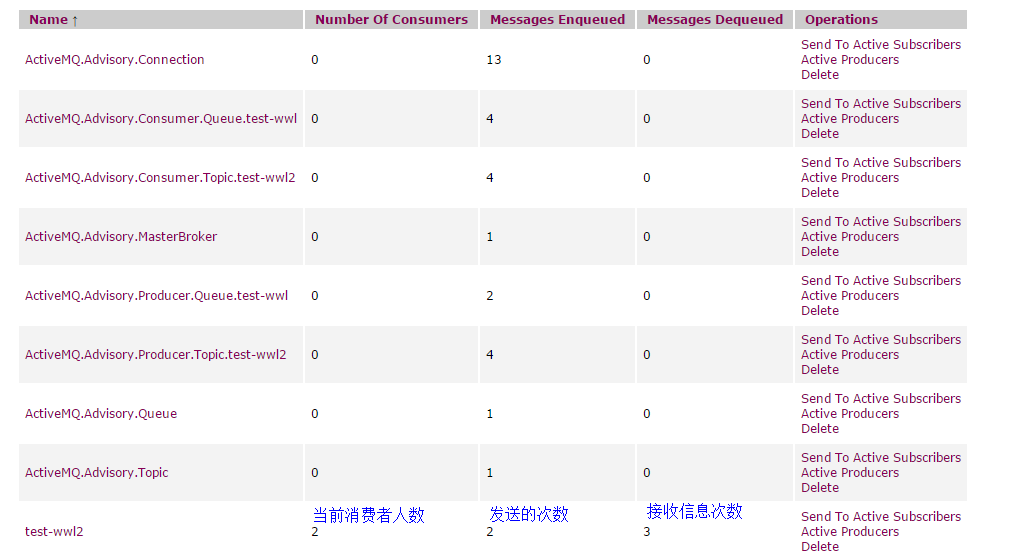
producer.close();

session.close();

connection.close();

测试

发送一个消息所有消费者都能收到信息，就和广播一样



## 整合spring

### 点对点模式步骤

生产者发送信息

第一步 导包

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jms</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.9</version>

</dependency>

<dependency>

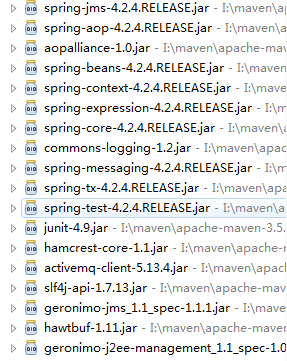
<groupId>org.apache.activemq</groupId>

<artifactId>activemq-client</artifactId>

<version>5.13.4</version>

</dependency>

</dependencies>



第二步 spring配置文件的配置

<context:component-scan base-package="cn.itcast.demo"/>

<!—第一步 真正可以产生Connection的ConnectionFactory，由对应的 JMS服务厂商提供-->

<bean id="targetConnectionFactory" class="org.apache.activemq.ActiveMQConnectionFactory">

<property name="brokerURL" value="tcp://106.12.106.21:61616"/>

</bean>

<!—第二步 Spring用于管理真正的ConnectionFactory的ConnectionFactory 需导入第一步-->

<bean id="connectionFactory" class="org.springframework.jms.connection.SingleConnectionFactory">

<!-- 目标ConnectionFactory对应真实的可以产生JMS Connection的ConnectionFactory -->

<property name="targetConnectionFactory" ref="targetConnectionFactory"/>

</bean>

<!—第三步 Spring提供的JMS工具类，它可以进行消息发送、接收等 需导入第二步-->

<bean id="jmsTemplate" class="org.springframework.jms.core.JmsTemplate">

<!-- 这个connectionFactory对应的是我们定义的Spring提供的那个ConnectionFactory对象 -->

<property name="connectionFactory" ref="connectionFactory"/>

</bean>

<!—第四步 这个是队列目的地，点对点的 文本信息 id名称随便定义，要在类中使用到，-->

<bean id="queueTextDestination" class="org.apache.activemq.command.ActiveMQQueue">

<!-- 此名称相当于原生第五步的名称，也是activemq页面中的name -->

<constructor-arg value="queue\_wwl"/>

</bean>

<!—第四步 这个是订阅模式 文本信息 不用需要注释不可以和上面点对点同时打开-->

<bean id="topicTextDestination" class="org.apache.activemq.command.ActiveMQTopic">

<constructor-arg value="topic\_wwl"/>

</bean>

第三步 代码编写

点对点方式发送文本信息

@Component

public class QueueProducer {

//注意 这两个变量名称不能随便定义要和配置文件定义的id一致

@Resource

private JmsTemplate jmsTemplate; 配置文件第三步的id名称

@Resource

private Destination queueTextDestination; 配置文件第四步的id名称

public void sendText(final String text) {

jmsTemplate.send(queueTextDestination,new MessageCreator() {

@Override

public Message createMessage(Session session) throws JMSException {

return session.createTextMessage(text);

}

});

}

}

第四步 测试：使用spring的test包配合junit进行测试

@RunWith(SpringJUnit4ClassRunner.class)

@ContextConfiguration(locations="classpath:applicationContext-jms-producer.xml")

public class Test {

//注入自己写的类，配置文件有进行扫描包

@Autowired

private QueueProducer queueProducer;

//调用自己写的方法

@org.junit.Test

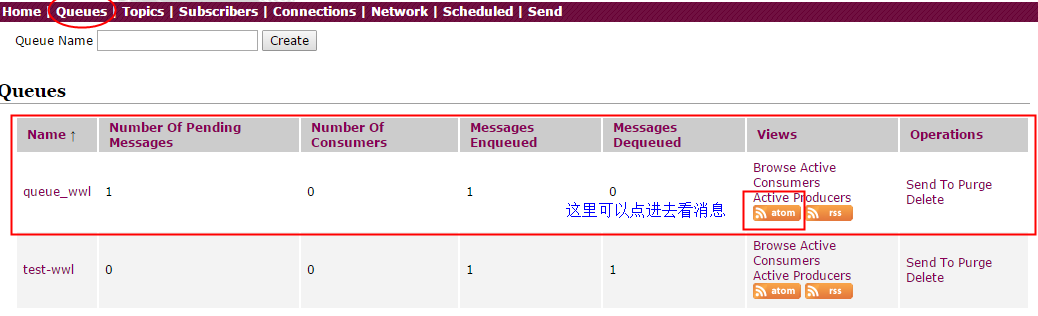
public void testSend() {

queueProducer.sendText("呵呵发送了信息了");

}

}

执行成功后图



消费者消费信息

第一步 导包 不变

第二步 spring配置文件的配置

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:context="http://www.springframework.org/schema/context" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:amq="http://activemq.apache.org/schema/core"

xmlns:jms="http://www.springframework.org/schema/jms"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd">

<!--第一步 真正可以产生Connection的ConnectionFactory，由对应的 JMS服务厂商提供-->

<bean id="targetConnectionFactory" class="org.apache.activemq.ActiveMQConnectionFactory">

<property name="brokerURL" value="tcp://106.12.106.21:61616"/>

</bean>

<!--第二步 Spring用于管理真正的ConnectionFactory的ConnectionFactory 属性需要第一步的-->

<bean id="connectionFactory" class="org.springframework.jms.connection.SingleConnectionFactory">

<!-- 目标ConnectionFactory对应真实的可以产生JMS Connection的ConnectionFactory -->

<property name="targetConnectionFactory" ref="targetConnectionFactory"/>

</bean>

<!--第三步 这个是队列目的地，点对点的 文本信息-->

<bean id="queueTextDestination" class="org.apache.activemq.command.ActiveMQQueue">

<!-- 这个需要在acticeMq页面中name一致-->

<constructor-arg value="queue\_wwl"/>

</bean>

<!--第四步 自己要写的监听类 -->

<bean id="myMessageListener" class="cn.itcast.demo.MyMessageListener"></bean>

<!—第五步 消息监听容器 -->

<bean class="org.springframework.jms.listener.DefaultMessageListenerContainer">

<!--需要属性：第二步的 工厂、第三步队列目的地、第四步 自己写的监听类 -->

<property name="connectionFactory" ref="connectionFactory" />

<property name="destination" ref="queueTextDestination" />

<property name="messageListener" ref="myMessageListener" />

</bean>

</beans>

第三步 编写监听类myMessageListener

import javax.jms.JMSException;

import javax.jms.Message;

import javax.jms.MessageListener;

import javax.jms.TextMessage;

//实现MessageListener接口

public class MyMessageListener implements MessageListener{

public void onMessage(Message message) {

TextMessage text=(TextMessage) message;

try {

System.out.println("接收到的信息："+text.getText());

} catch (JMSException e) {

e.printStackTrace();

}

}

}

第四步测试

@RunWith(SpringJUnit4ClassRunner.class)

@ContextConfiguration(locations="classpath:applicationContext-jms-consumer.xml")

public class Test {

@org.junit.Test

public void testSend() throws IOException {

//其实呢什么也不用写，为了显示效果让他停留下来

System.in.read();

}

}

### 发布/订阅模式步骤

生产者发送信息

大部分都和点对点模式一致

不同之处：spring配置文件

变成这个类

<!—第四步 这个是订阅模式 文本信息 不用需要注释不可以和上面点对点同时打开-->

<bean id="topicTextDestination" class="org.apache.activemq.command.ActiveMQTopic">

<constructor-arg value="topic\_wwl"/>

</bean>

类中引用这个id名称

//注意 这两个变量名称不能随便定义要和配置文件定义的id一致

@Resource

private JmsTemplate jmsTemplate; 配置文件第三步的id名称

@Resource

private Destination topicTextDestination; 配置文件第四步的id名称

public void sendText(final String text) {

jmsTemplate.send(topicTextDestination,new MessageCreator() {

@Override

public Message createMessage(Session session) throws JMSException {

return session.createTextMessage(text);

}

});

}

消费者消费信息

大部分都和点对点模式一致

不同之处 ：spring配置文件

<!--第三步 这个是队列目的地，点对点的 文本信息-->

<bean id="topicTextDestination" class="org.apache.activemq.command.ActiveMQTopic">

<!-- 这个需要在acticeMq页面中name一致-->

<constructor-arg value="queue\_wwl"/>

</bean>

<!-- 消息监听容器 -->

<bean class="org.springframework.jms.listener.DefaultMessageListenerContainer">

<!--需要属性：第二步的 工厂、第三步队列目的地、第四步 自己写的监听类 -->

<property name="connectionFactory" ref="connectionFactory" />

<property name="destination" ref=" topicTextDestination " />

<property name="messageListener" ref="myMessageListener" />

</bean>

其它都一样的