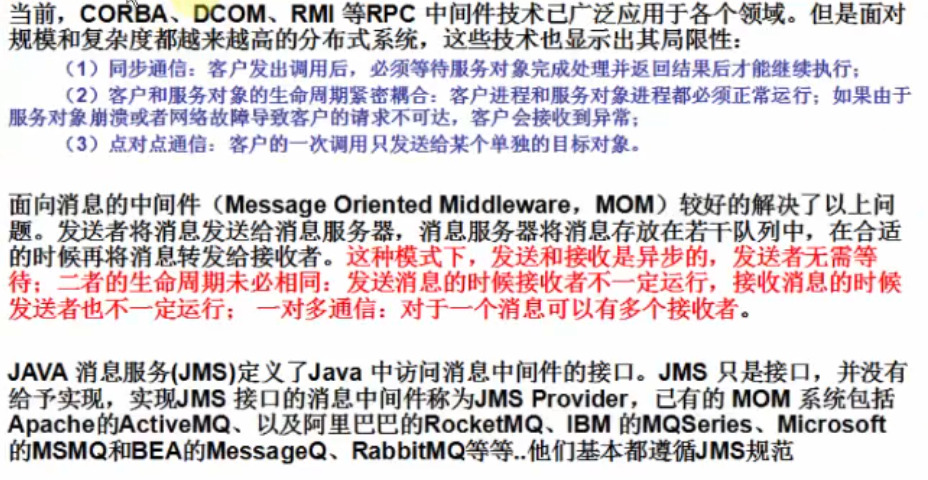
ActiveMQ

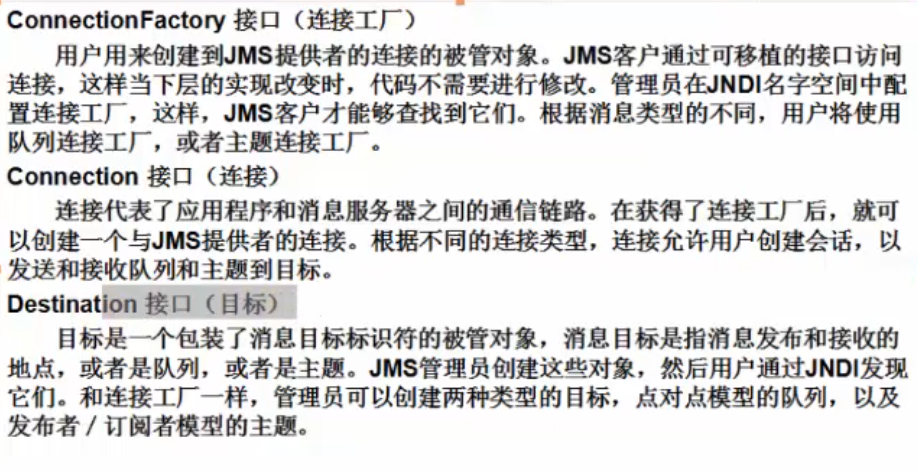
# ActiveMQ简介和入门

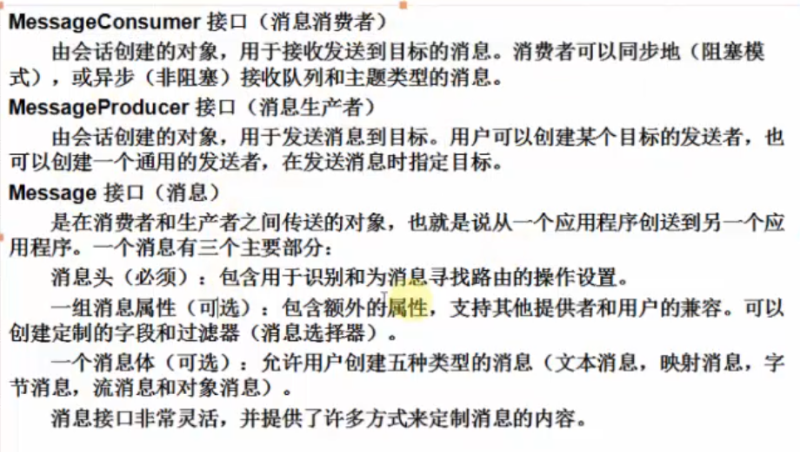
## 背景&JMS概述

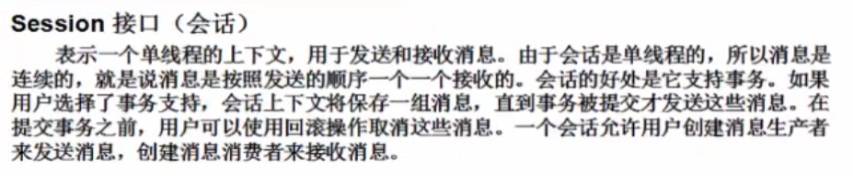


## JMS术语

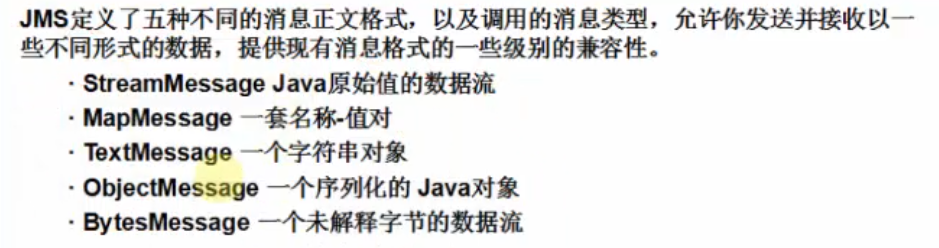




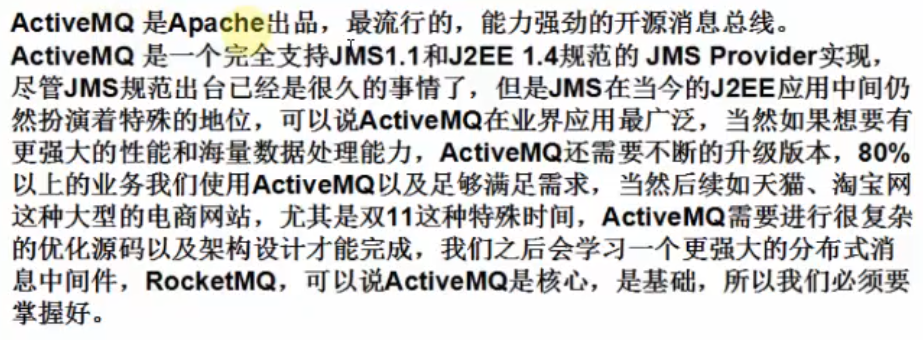




## JMS消息格式



## ActiveMQ简介



## ActiveMQ的安装

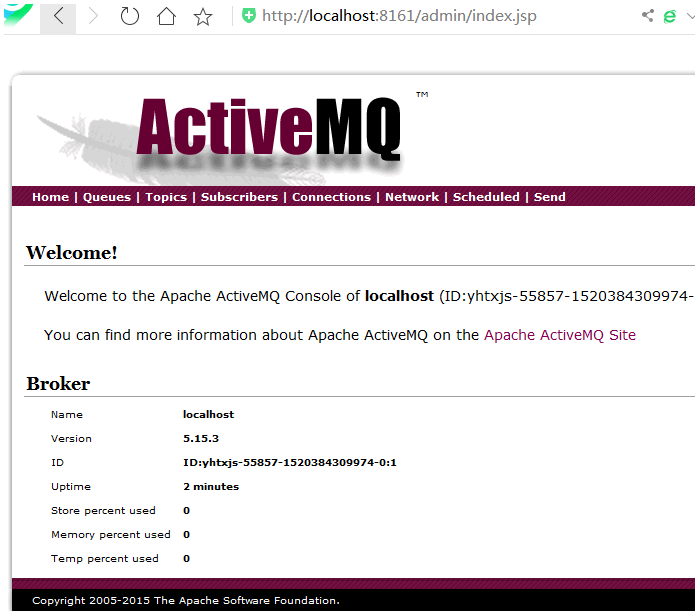
去官网下载<http://activemq.apache.org/>

我们下载最新的版本5.15.3

先使用window版本

ActiveMQ解压后动后的目录





# ActiveMQ的使用

## ActiveMQ-HelloWorld



### JMSProducer

**package** com.guigu.activemq.helloworld;

**import** javax.jms.Connection;

**import** javax.jms.ConnectionFactory;

**import** javax.jms.DeliveryMode;

**import** javax.jms.Destination;

**import** javax.jms.JMSException;

**import** javax.jms.MessageProducer;

**import** javax.jms.Session;

**import** javax.jms.TextMessage;

**import** org.apache.activemq.ActiveMQConnectionFactory;

/\*\*

\* <p>project\_name:ActiveMQProvider</p>

\* <p>package\_name:com.guigu.activemq.helloworld.JMSProducer</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年3月7日上午9:19:59 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**public** **class** JMSProducer {

**public** **static** **void** main(String[] args) {

**try** {

// 1.建立ConnectionFactory工厂对象 ，需要填入用户名 密码 以及连接的地址 都使用默认即可

// 默认端口号是tcp://localhost:61616

ConnectionFactory connectionFactory = **new** ActiveMQConnectionFactory(ActiveMQConnectionFactory.***DEFAULT\_USER***,

ActiveMQConnectionFactory.***DEFAULT\_PASSWORD***, "tcp://localhost:61616");

// 2.通过ConnetionFactory工厂 创建一个Connection连接

// 并且调用Connection的start方法开启连接

Connection connection = connectionFactory.createConnection();

connection.start();

// 3.通过Connection对象创建Session会话 上下文件环境对象 用于接受消息 参数1代表 是否是开启事务 参数2签收方式

// 一般使用自动签收

Session session = connection.createSession(Boolean.***FALSE***, Session.***AUTO\_ACKNOWLEDGE***);

// 4.通过sesssion创建Destination对象 指的是一个客户端用来指定生产消息目标和消费消息来源的对象

Destination destination = session.createQueue("HelloWolrd\_guigu");

// 5.通过Session对象 创建消息的发送和接受对象(生产者和消费者)

MessageProducer producer = session.createProducer(destination);

// 6.我们可以使用MessageProducer对象的setDeliveryMode方法设置持久化特性 和非持久化特性

producer.setDeliveryMode(DeliveryMode.***NON\_PERSISTENT***);

// 发送消息到activemq消息服务器

**for** (**int** i = 0; i < 10; i++) {

TextMessage msg = session.createTextMessage("我是消息内容:" + i + "activemq is so easy");

producer.send(msg);

System.***out***.println("发送消息：" + msg.getText());

Thread.*sleep*(1000);

}

// 7.释放连接即可

**if** (connection != **null**) {

connection.close();

}

} **catch** (JMSException e) {

e.printStackTrace();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

### JMSConsumer

**package** com.guigu.activemq.helloworld;

**import** javax.jms.Connection;

**import** javax.jms.ConnectionFactory;

**import** javax.jms.DeliveryMode;

**import** javax.jms.Destination;

**import** javax.jms.JMSException;

**import** javax.jms.MessageConsumer;

**import** javax.jms.MessageProducer;

**import** javax.jms.Session;

**import** javax.jms.TextMessage;

**import** org.apache.activemq.ActiveMQConnectionFactory;

/\*\*

\* <p>project\_name:ActiveMQProvider</p>

\* <p>package\_name:com.guigu.activemq.helloworld.JMSConsumer</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年3月7日上午10:16:43 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**public** **class** JMSConsumer {

**public** **static** **void** main(String[] args) {

**try** {

// 1.建立ConnectionFactory工厂对象 ，需要填入用户名 密码 以及连接的地址 都使用默认即可

// 默认端口号是tcp://localhost:61616

ConnectionFactory connectionFactory = **new** ActiveMQConnectionFactory(ActiveMQConnectionFactory.***DEFAULT\_USER***,

ActiveMQConnectionFactory.***DEFAULT\_PASSWORD***, "tcp://localhost:61616");

// 2.通过ConnetionFactory工厂 创建一个Connection连接

// 并且调用Connection的start方法开启连接

Connection connection = connectionFactory.createConnection();

connection.start();

// 3.通过Connection对象创建Session会话 上下文件环境对象 用于接受消息 参数1代表 是否是开启事务 参数2签收方式

// 一般使用自动签收

Session session = connection.createSession(Boolean.***FALSE***, Session.***AUTO\_ACKNOWLEDGE***);

// 4.通过sesssion创建Destination对象 指的是一个客户端用来指定生产消息目标和消费消息来源的对象

Destination destination = session.createQueue("HelloWolrd\_guigu");

// 5.通过Session对象 创建消息的发送和接受对象(生产者和消费者)

MessageConsumer consumer = session.createConsumer(destination);

**while**(**true**) {

TextMessage msg =(TextMessage) consumer.receive();

**if**(msg==**null**) {

**break**;

}

System.***out***.println("收到内容:"+msg.getText());

}

// 7.释放连接即可

**if** (connection != **null**) {

connection.close();

}

} **catch** (JMSException e) {

e.printStackTrace();

}

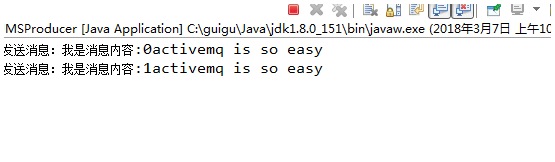
}

}

验证

1. 首先启动ActiveMQ 在浏览器上<http://localhost:8161/amdin>

2、运行生产者



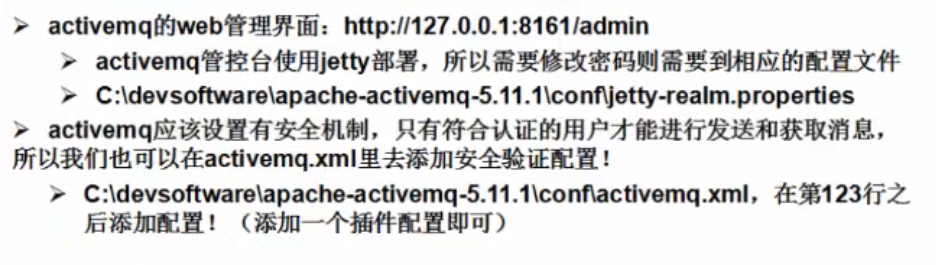
查看ActiveMQ服务器相关的信息

1. 运行消费者

查看ActiveMQ服务器相关的信息s



## ActiveMQ安全机制

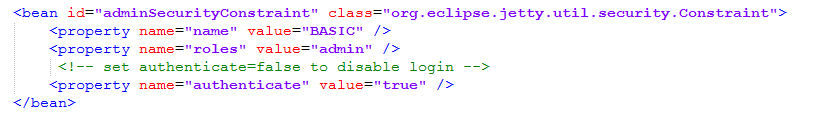


**一、设置后台管理密码**

**A）修改jetty.xml开启密码验证**

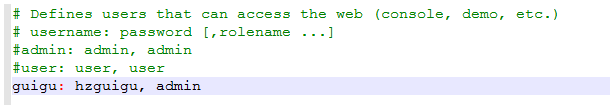
**C:\guigu\apache-activemq-5.15.3-bin\apache-activemq-5.15.3\conf\jetty.xml文件**

在低版本中默认是false是没有开启的 。



**B) 修改控制台登录用户名和密码 也可也不修改 默认就是admin admin**

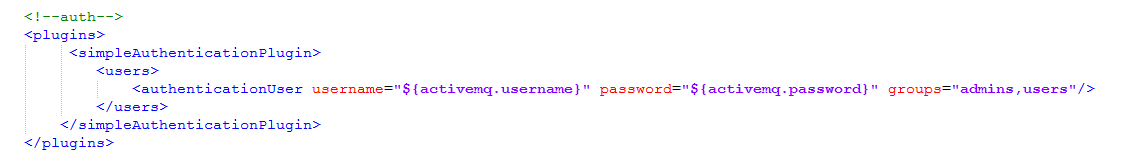
jetty-realm.properties



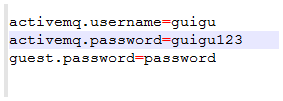
**二、消息生产者和消息消费者密码认证**

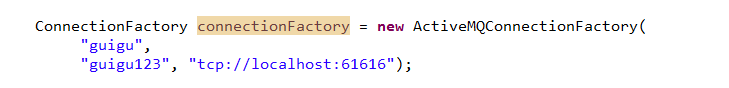
**A)添加权限认证**

C:\guigu\apache-activemq-5.15.3-bin\apache-activemq-5.15.3\conf\activemq.xml

 **B)设置用户名和密码**

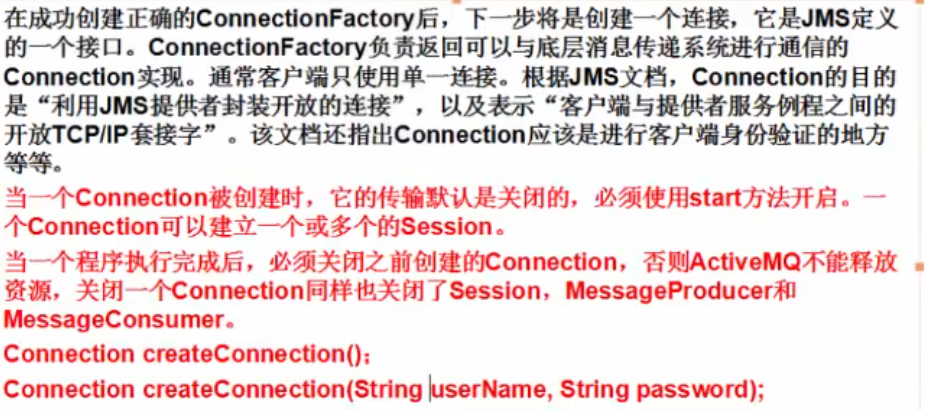
credentials.properties





## API详解

### Connection方法的使用



### Session方法的使用



**JMSProducer**

// 3.通过Connection对象创建Session会话 上下文件环境对象 用于接受消息 参数1代表 是否是开启事务 参数2签收方式

// 一般使用自动签收

// Session session = connection.createSession(Boolean.FALSE, Session.AUTO\_ACKNOWLEDGE);

//开启事务 需要手动提交 签收模式为 手动签收

Session session = connection.createSession(Boolean.***TRUE***, Session.***CLIENT\_ACKNOWLEDGE***);

// 4.通过sesssion创建Destination对象 指的是一个客户端用来指定生产消息目标和消费消息来源的对象

Destination destination = session.createQueue("HelloWolrd\_guigu");

// 5.通过Session对象 创建消息的发送和接受对象(生产者和消费者)

MessageProducer producer = session.createProducer(destination);

// 6.我们可以使用MessageProducer对象的setDeliveryMode方法设置持久化特性 和非持久化特性

producer.setDeliveryMode(DeliveryMode.***NON\_PERSISTENT***);

// 发送消息到activemq消息服务器

**for** (**int** i = 0; i < 10; i++) {

TextMessage msg = session.createTextMessage("我是消息内容:" + i + "activemq is so easy");

producer.send(msg);

System.***out***.println("发送消息：" + msg.getText());

Thread.*sleep*(1000);

}

//事务开启 必须手动提交事务

session.commit();

**JMSConsumer**

// 4.通过sesssion创建Destination对象 指的是一个客户端用来指定生产消息目标和消费消息来源的对象

Destination destination = session.createQueue("HelloWolrd\_guigu");

// 5.通过Session对象 创建消息的发送和接受对象(生产者和消费者)

MessageConsumer consumer = session.createConsumer(destination);

**while**(**true**) {

TextMessage msg =(TextMessage) consumer.receive();

**if**(msg==**null**) {

**break**;

}

//手动确认接收消息

**msg.acknowledge();**

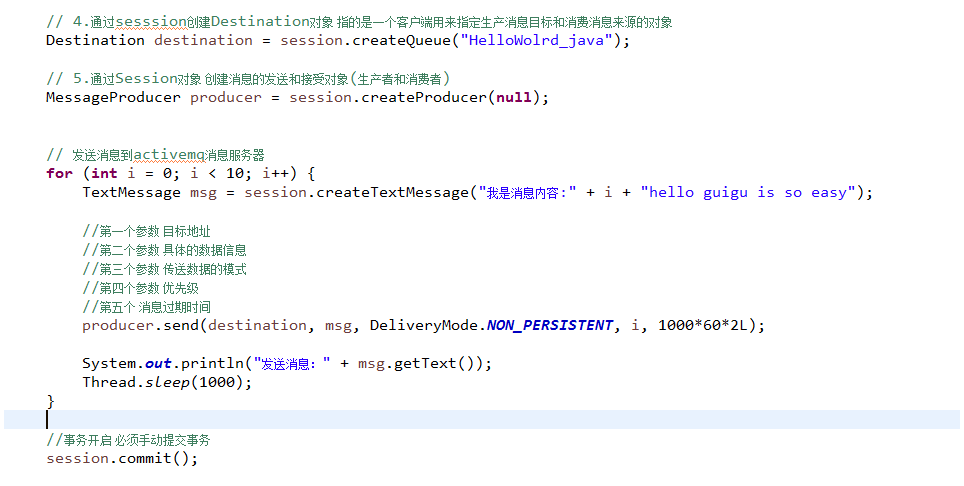
System.***out***.println("收到内容:"+msg.getText());

}

### MessageProducer



**JMSProducer2**



**JMSConsumer2**



### MessageConsumer（配置消息选择器）



**package** com.guigu.activemq.messageselector;

/\*\*

\* <p>project\_name:ActiveMQProvider</p>

\* <p>package\_name:com.guigu.activemq.messageselector.Producer</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年3月7日下午1:57:41 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**import** javax.jms.Connection;

**import** javax.jms.ConnectionFactory;

**import** javax.jms.DeliveryMode;

**import** javax.jms.Destination;

**import** javax.jms.JMSException;

**import** javax.jms.MapMessage;

**import** javax.jms.MessageProducer;

**import** javax.jms.Session;

**import** org.apache.activemq.ActiveMQConnectionFactory;

**public** **class** Producer {

// 1.连接工厂

**private** ConnectionFactory connectionFactory;

// 2.连接对象

**private** Connection connection;

// 3.Sesssion对象

**private** Session session;

// 4.定义生产者

**private** MessageProducer messageProducer;

**public** Producer() {

**try** {

**this**.connectionFactory = **new** ActiveMQConnectionFactory("tcp://localhost:61616");

**this**.connection = connectionFactory.createConnection("guigu", "guigu123");

**this**.connection.start();

// 设置自动签收模式

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

**this**.messageProducer = session.createProducer(**null**);

} **catch** (JMSException e) {

e.printStackTrace();

}

}

**public** Session getSession() {

**return** **this**.session;

}

**public** **void** send1() {

**try** {

Destination destination =**this**.session.createQueue("first");

MapMessage msg1 =session.createMapMessage();

msg1.setString("name", "guiguhz");

msg1.setInt("age", 30);

//设置用于消息过滤的条件

msg1.setStringProperty("name", "gghz");

msg1.setIntProperty("age", 30);

msg1.setStringProperty("color", "blue");

MapMessage msg2 =session.createMapMessage();

msg2.setString("name", "zhangsan");

msg2.setInt("age", 25);

//设置用于消息过滤的条件

msg2.setStringProperty("name", "zs");

msg2.setIntProperty("age", 25);

msg2.setStringProperty("color", "white");

MapMessage msg3 =session.createMapMessage();

msg3.setString("name", "guiguhz");

msg3.setInt("age", 35);

//设置用于消息过滤的条件

msg3.setStringProperty("name", "zl");

msg3.setIntProperty("age", 35);

msg3.setStringProperty("color", "black");

//发送消息到消息中间件

**this**.messageProducer.send(destination,msg1,DeliveryMode.***NON\_PERSISTENT***,4,1000\*60\*10);

**this**.messageProducer.send(destination,msg2,DeliveryMode.***NON\_PERSISTENT***,4,1000\*60\*10);

**this**.messageProducer.send(destination,msg3,DeliveryMode.***NON\_PERSISTENT***,4,1000\*60\*10);

} **catch** (JMSException e) {

e.printStackTrace();

}

}

**public** **static** **void** main(String[] args) {

Producer p1 =**new** Producer();

p1.send1();

}

}

**package** com.guigu.activemq.messageselector;

**import** javax.jms.Connection;

**import** javax.jms.ConnectionFactory;

**import** javax.jms.Destination;

**import** javax.jms.JMSException;

**import** javax.jms.MapMessage;

**import** javax.jms.Message;

**import** javax.jms.MessageConsumer;

**import** javax.jms.MessageListener;

**import** javax.jms.Session;

**import** javax.jms.TextMessage;

**import** org.apache.activemq.ActiveMQConnectionFactory;

/\*\*

\* <p>project\_name:ActiveMQProvider</p>

\* <p>package\_name:com.guigu.activemq.messageselector.Consumer</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年3月7日下午2:14:03 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**public** **class** Consumer {

// 1.连接工厂

**private** ConnectionFactory connectionFactory;

// 2.连接对象

**private** Connection connection;

// 3.Sesssion对象

**private** Session session;

// 4.定义消费者对象

**private** MessageConsumer messageConsumer;

// 5.目的地

**private** Destination destination;

// 定义消息选择器规则

**public** **final** String SELECTOR\_1 = "age > 28 and color='black'";

**public** Consumer() {

**try** {

**this**.connectionFactory = **new** ActiveMQConnectionFactory("tcp://localhost:61616");

**this**.connection = connectionFactory.createConnection("guigu", "guigu123");

**this**.connection.start();

// 设置自动签收模式

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

**this**.destination = session.createQueue("first");

**this**.messageConsumer = session.createConsumer(destination, SELECTOR\_1);

} **catch** (JMSException e) {

e.printStackTrace();

}

}

**public** Session getSession() {

**return** **this**.session;

}

// 用于监听消息队列的消息

**class** MyListener **implements** MessageListener {

@Override

**public** **void** onMessage(Message message) {

System.***out***.println("-------33");

**try** {

// 首先判断数据类型

**if** (message **instanceof** MapMessage) {

System.***out***.println("-------11");

MapMessage result = (MapMessage) message;

System.***out***.println(result.toString());

System.***out***.println(result.getString("name"));

System.***out***.println(result.getInt("age"));

// System.out.println(result.getInt("color"));

}

} **catch** (JMSException e) {

e.printStackTrace();

}

}

}

//用于异步监听消息

**public** **void** receiver() {

**try** {

messageConsumer.setMessageListener(**new** MyListener());

System.***out***.println("-------22");

} **catch** (JMSException e) {

e.printStackTrace();

}

}

**public** **static** **void** main(String[] args) {

Consumer consumer =**new** Consumer();

consumer.receiver();

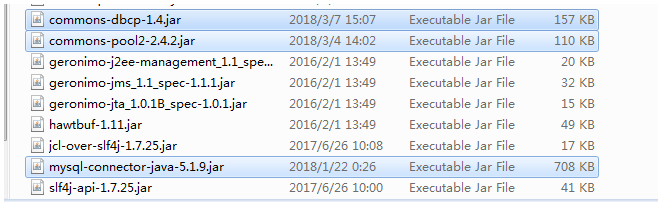
}

}

## ActiveMQ持久化

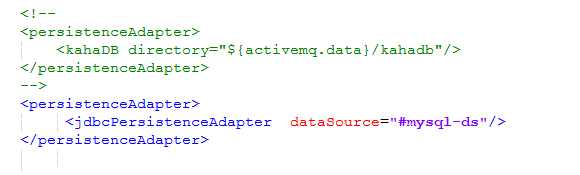
**一）把ActiveMQ 持久化到mysql**

1、把数据库的jar包 放入到activemq的lib目录下



1. 修改activemq的conf目录下的activemq.xml配置文件

2.1修改原来默认的持久化方式改为mysql 默认的kahadab注释



2.2连接mysql的配置

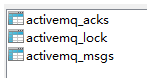


1. 重启activemq服务器即可
2. 配置持久化操作



当消息没有被消费会同步暂存到mysql数据库内。

1. 数据库的内容



activemq\_msgs用于存储消息，Queue和Topic都存储在这个表中：

ID：自增的数据库主键

CONTAINER：消息的Destination

MSGID\_PROD：消息发送者客户端的主键

MSG\_SEQ：是发送消息的顺序，MSGID\_PROD+MSG\_SEQ可以组成JMS的MessageID

EXPIRATION：消息的过期时间，存储的是从1970-01-01到现在的毫秒数

MSG：消息本体的Java序列化对象的二进制数据

PRIORITY：优先级，从0-9，数值越大优先级越高

activemq\_acks用于存储订阅关系。如果是持久化Topic，订阅者和服务器的订阅关系在这个表保存：

主要的数据库字段如下：

CONTAINER：消息的Destination

SUB\_DEST：如果是使用Static集群，这个字段会有集群其他系统的信息

CLIENT\_ID：每个订阅者都必须有一个唯一的客户端ID用以区分

SUB\_NAME：订阅者名称

SELECTOR：选择器，可以选择只消费满足条件的消息。条件可以用自定义属性实现，可支持多属性AND和OR操作

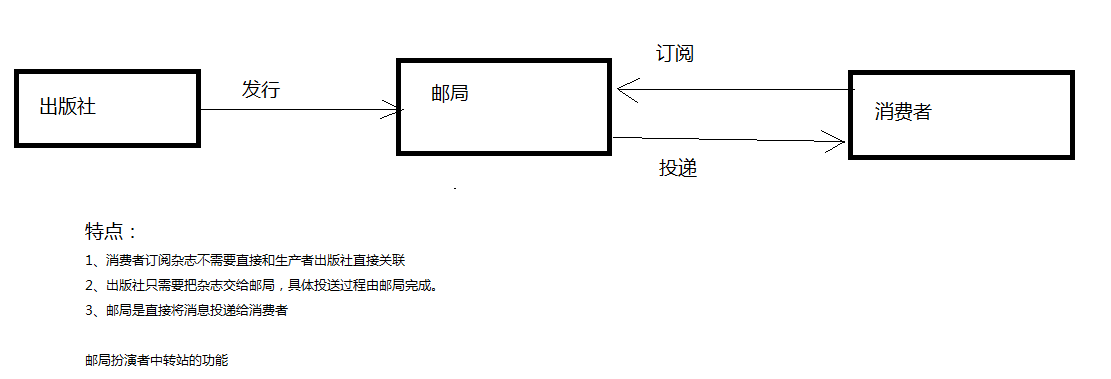
LAST\_ACKED\_ID：记录消费过的消息的ID。

表activemq\_lock在集群环境中才有用，只有一个Broker可以获得消息，称为Master Broker，

其他的只能作为备份等待Master Broker不可用，才可能成为下一个Master Broker。

这个表用于记录哪个Broker是当前的Master Broker。

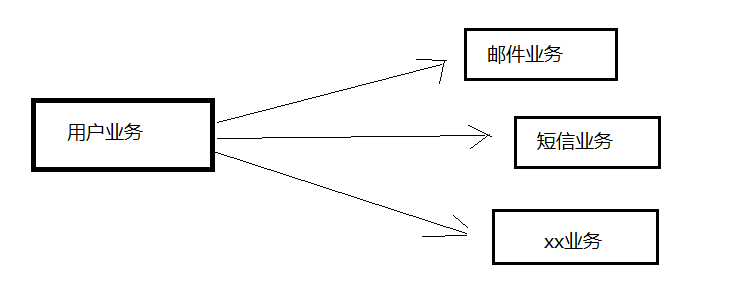
## ActiveMQ-发布订阅消息模式



**二）发布—订阅模式**

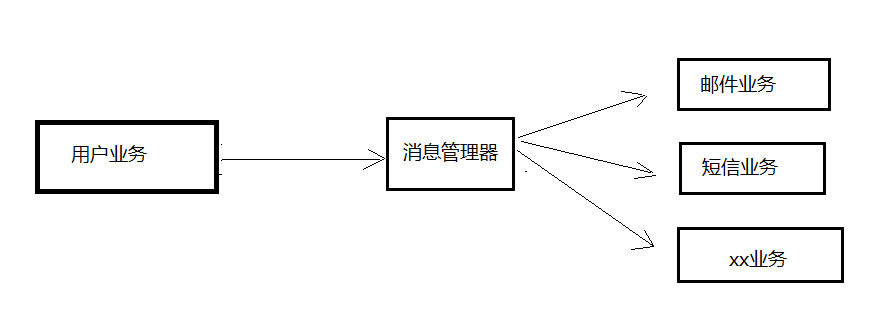
网站注册新用户，发送一封激活邮件到用户的邮箱，用户点击该链接才能注册到该网站。

随着业务的变化 要求注册需要发送短信激活。

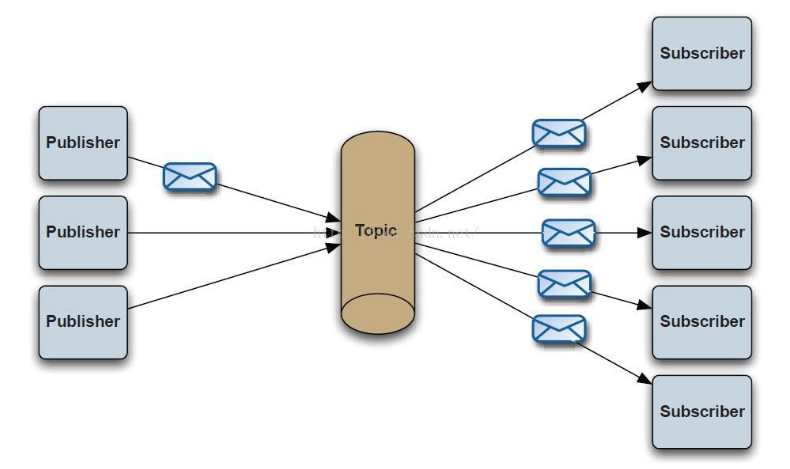


注册成功之后还依赖其他业务。这个业务还是不确定的。。。。

1-N业务需求。



N-N业务需求



**package** com.guigu.activemq.publishsub;

**import** javax.jms.Connection;

**import** javax.jms.ConnectionFactory;

**import** javax.jms.Destination;

**import** javax.jms.JMSException;

**import** javax.jms.MessageProducer;

**import** javax.jms.Session;

**import** javax.jms.TextMessage;

**import** org.apache.activemq.ActiveMQConnectionFactory;

/\*\*

\* <p>project\_name:ActiveMQProvider</p>

\* <p>package\_name:com.guigu.activemq.publishsub.Publish</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年3月8日上午8:54:20 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**public** **class** Publish {

// 1.连接工厂

**private** ConnectionFactory connectionFactory;

// 2.连接对象

**private** Connection connection;

// 3.Sesssion对象

**private** Session session;

// 4.定义生产者

**private** MessageProducer messageProducer;

**public** Publish() {

**try** {

**this**.connectionFactory = **new** ActiveMQConnectionFactory("tcp://localhost:61616");

**this**.connection = connectionFactory.createConnection("guigu", "guigu123");

**this**.connection.start();

// 设置自动签收模式

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

**this**.messageProducer = session.createProducer(**null**);

} **catch** (JMSException e) {

e.printStackTrace();

}

}

**public** **void** sendMessage() **throws** Exception {

Destination destination = session.createTopic("Topic001");

TextMessage msg = session.createTextMessage("我是消息的内容.....new");

messageProducer.send(destination, msg);

**if** (connection != **null**) {

connection.close();

}

}

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

Publish publish = **new** Publish();

publish.sendMessage();

}

}

**package** com.guigu.activemq.publishsub;

**import** javax.jms.Connection;

**import** javax.jms.ConnectionFactory;

**import** javax.jms.Destination;

**import** javax.jms.JMSException;

**import** javax.jms.Message;

**import** javax.jms.MessageConsumer;

**import** javax.jms.MessageListener;

**import** javax.jms.Session;

**import** javax.jms.TextMessage;

**import** org.apache.activemq.ActiveMQConnectionFactory;

**import** com.guigu.activemq.messageselector.Consumer;

/\*\*

\* <p>project\_name:ActiveMQProvider</p>

\* <p>package\_name:com.guigu.activemq.publishsub.Subscribe1</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年3月8日上午8:58:33 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**public** **class** Subscribe1 {

// 1.连接工厂

**private** ConnectionFactory connectionFactory;

// 2.连接对象

**private** Connection connection;

// 3.Sesssion对象

**private** Session session;

// 4.定义消费者对象

**private** MessageConsumer messageConsumer;

// 5.目的地

**private** Destination destination;

**public** Subscribe1() {

**try** {

**this**.connectionFactory = **new** ActiveMQConnectionFactory("tcp://localhost:61616");

**this**.connection = connectionFactory.createConnection("guigu", "guigu123");

**this**.connection.start();

// 设置自动签收模式

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

} **catch** (JMSException e) {

e.printStackTrace();

}

}

**public** **void** receive() **throws** Exception{

Destination destination =session.createTopic("Topic001");

messageConsumer =session.createConsumer(destination);

messageConsumer.setMessageListener(**new** MessageListener() {

@Override

**public** **void** onMessage(Message message) {

**if**(message **instanceof** TextMessage) {

**try** {

TextMessage msg =(TextMessage) message;

System.***out***.println("subcriber1 received message:"+msg.getText());

} **catch** (JMSException e) {

e.printStackTrace();

}

}

}

});

}

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

Subscribe1 sb1 =**new** Subscribe1();

sb1.receive();

}

}

**package** com.guigu.activemq.publishsub;

**import** javax.jms.Connection;

**import** javax.jms.ConnectionFactory;

**import** javax.jms.Destination;

**import** javax.jms.JMSException;

**import** javax.jms.Message;

**import** javax.jms.MessageConsumer;

**import** javax.jms.MessageListener;

**import** javax.jms.Session;

**import** javax.jms.TextMessage;

**import** org.apache.activemq.ActiveMQConnectionFactory;

/\*\*

\* <p>project\_name:ActiveMQProvider</p>

\* <p>package\_name:com.guigu.activemq.publishsub.Subscribe1</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年3月8日上午8:58:33 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**public** **class** Subscribe2 {

// 1.连接工厂

**private** ConnectionFactory connectionFactory;

// 2.连接对象

**private** Connection connection;

// 3.Sesssion对象

**private** Session session;

// 4.定义消费者对象

**private** MessageConsumer messageConsumer;

// 5.目的地

**private** Destination destination;

**public** Subscribe2() {

**try** {

**this**.connectionFactory = **new** ActiveMQConnectionFactory("tcp://localhost:61616");

**this**.connection = connectionFactory.createConnection("guigu", "guigu123");

**this**.connection.start();

// 设置自动签收模式

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

} **catch** (JMSException e) {

e.printStackTrace();

}

}

**public** **void** receive() **throws** Exception{

Destination destination =session.createTopic("Topic001");

messageConsumer =session.createConsumer(destination);

messageConsumer.setMessageListener(**new** MessageListener() {

@Override

**public** **void** onMessage(Message message) {

**if**(message **instanceof** TextMessage) {

**try** {

TextMessage msg =(TextMessage) message;

System.***out***.println("subcriber2 received message:"+msg.getText());

} **catch** (JMSException e) {

e.printStackTrace();

}

}

}

});

}

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

Subscribe2 sb2 =**new** Subscribe2();

sb2.receive();

}

}

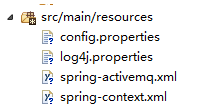
## ActiveMQ的具体业务使用

### ActiveMQProvider

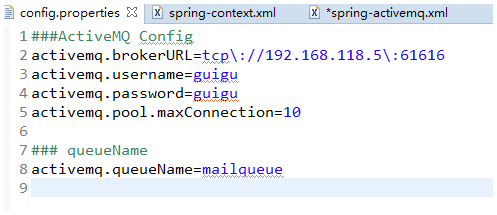
#### 搭建环境

。。。

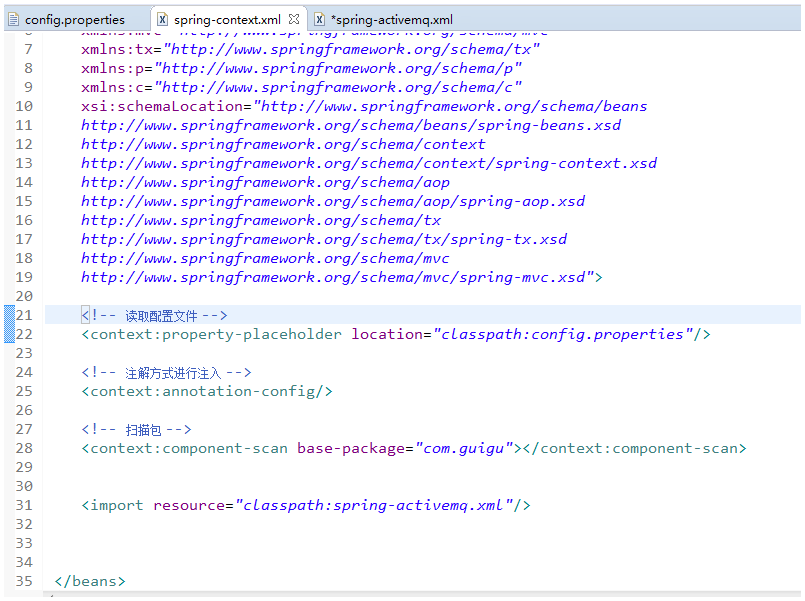
#### 编写配置文件



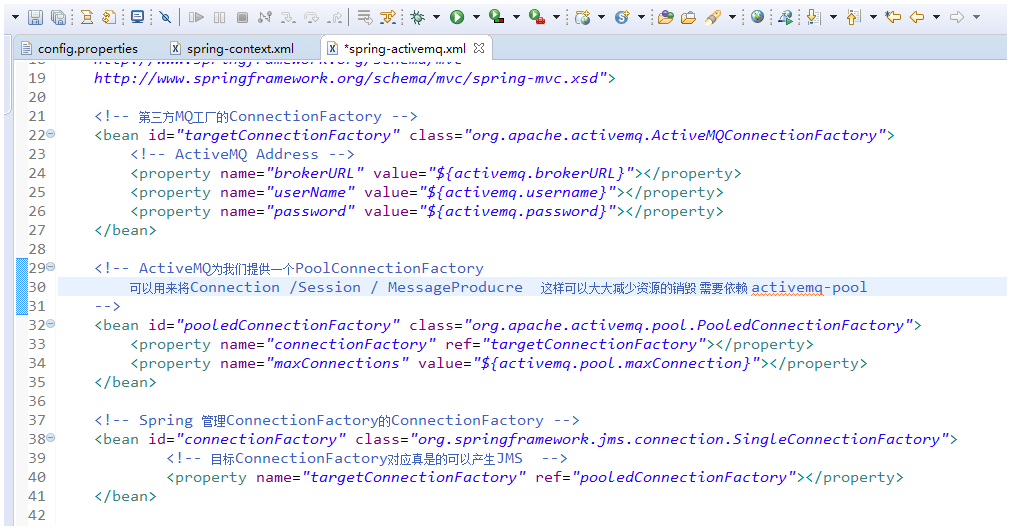
##### Config.properties

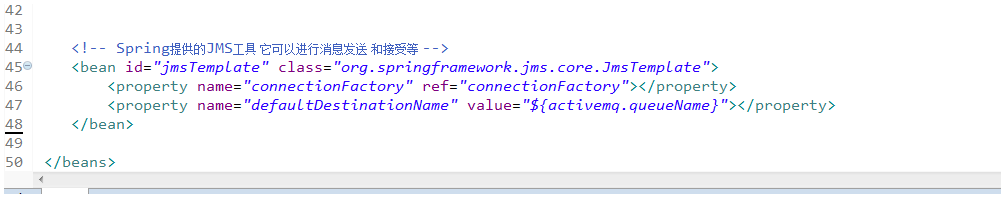


##### Spring-context.xml



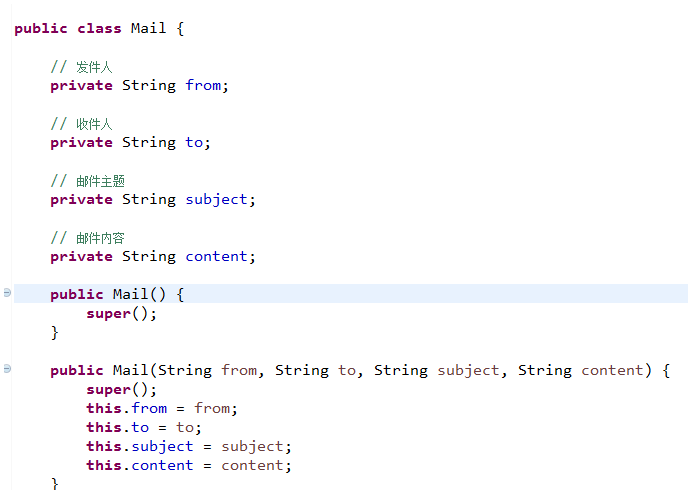
##### Spring-activemq.xml





#### 编写代码

##### 实体类



##### 生产者



##### 测试代码

### ActiveMQConsumer

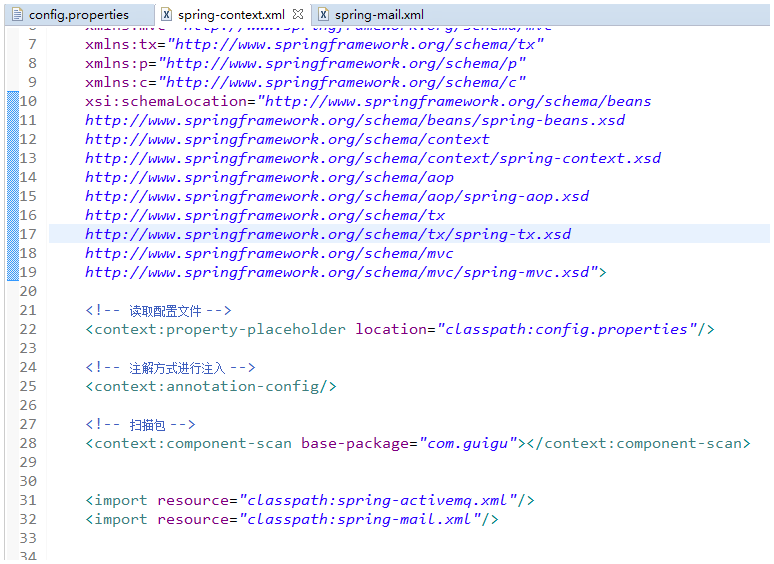
#### 搭建环境

#### 编写配置文件

##### Config.properties



##### spring-context



##### Spring-activemq.xml





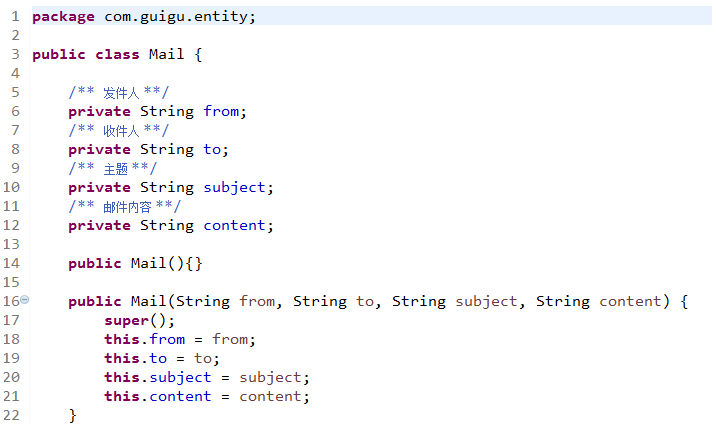
##### Spring-mail.xml





#### 编写代码

##### Mail



##### MailQueueMessageListener

**package** com.guigu.mq;

**import** javax.jms.Destination;

**import** javax.jms.Message;

**import** javax.jms.Session;

**import** javax.jms.TextMessage;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.jms.core.JmsTemplate;

**import** org.springframework.jms.listener.SessionAwareMessageListener;

**import** org.springframework.stereotype.Component;

**import** com.alibaba.fastjson.JSONObject;

**import** com.guigu.entity.Mail;

**import** com.guigu.service.MailService;

@Component

**public** **class** MailQueueMessageListener **implements** SessionAwareMessageListener<Message> {

@Autowired

**private** JmsTemplate jmsTemplate;

@Autowired

**private** Destination mailQueue;

@Autowired

**private** MailService mailService;

**public** **synchronized** **void** onMessage(Message message, Session session) {

**try** {

TextMessage msg = (TextMessage) message;

**final** String ms = msg.getText();

System.***out***.println("收到消息：" + ms);

//转换成相应的对象

Mail mail = JSONObject.*parseObject*(ms, Mail.**class**);

**if** (mail == **null**) {

**return**;

}

**try** {

//执行发送业务

mailService.mailSend(mail);

} **catch** (Exception e) {

e.printStackTrace();

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

##### MailService

**package** com.guigu.service;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.mail.MailException;

**import** org.springframework.mail.SimpleMailMessage;

**import** org.springframework.mail.javamail.JavaMailSender;

**import** org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor;

**import** org.springframework.stereotype.Service;

**import** com.guigu.entity.Mail;

@Service("mailService")

**public** **class** MailService {

@Autowired

**private** JavaMailSender mailSender;

@Autowired

**private** SimpleMailMessage simpleMailMessage;

@Autowired

**private** ThreadPoolTaskExecutor threadPool;

**public** **void** mailSend(**final** Mail mail) {

threadPool.execute(**new** Runnable() {

**public** **void** run() {

**try** {

simpleMailMessage.setFrom(simpleMailMessage.getFrom());

simpleMailMessage.setTo(mail.getTo());

simpleMailMessage.setSubject(mail.getSubject());

simpleMailMessage.setText(mail.getContent());

mailSender.send(simpleMailMessage);

} **catch** (MailException e) {

e.printStackTrace();

**throw** e;

}

}

});

}

}

# ActiveMQ集群

## 部署方案

### 配置Zookeeper

ZooKeeper是一个分布式的，开放源码的分布式应用程序协调服务，是Google的Chubby一个开源的实现，是Hadoop和Hbase的重要组件。它是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、域名服务、分布式同步、组服务等。

**1)ssZookeeper的部署方案**

|  |  |  |  |
| --- | --- | --- | --- |
| 主机IP | 消息端口号 | 通信端口号 | 节点目录/usr/local/sofeware |
| 192.168.118.6 | 2181 | 2888:3888 | Zookeeper |
| 192.168.118.9 | 2181 | 2888:3888 | Zookeeper |
| 192.168.118.10 | 2181 | 2888:3888 | Zookeeper |

**2)把相关的程序进行安装**

2.1）上传文件zookeeper-3.4.11.tar.gz 文件到/usr/local/sofeware目录下

2.2）解压文件

tar -zxvf zookeeper-3.4.11.tar.gz -C /usr/local/sofeware/

2.3）修改文件夹的名称

mv zookeeper-3.4.11 zookeeper

**3）修改相关的配置文件**

Vim /etc/profile

三个服务器上都进行相关的修改即可

export ZOOKEEPER\_HOME=/usr/local/sofeware/zookeeper

export PATH=.:$JAVA\_HOME/bin:$ZOOKEEPER\_HOME/bin:$PATH

让配置文件立即生效

source /etc/profile

**4）修改Zookeeper相关的配置文件**

[root@J2001 sofeware]# ***cd zookeeper***

[root@J2001 zookeeper]# **ll**

总用量 1596

drwxr-xr-x. 2 502 games 149 11月 2 02:52 bin

-rw-r--r--. 1 502 games 87943 11月 2 02:47 build.xml

drwxr-xr-x. 2 502 games 77 11月 2 02:52 conf

drwxr-xr-x. 10 502 games 130 11月 2 02:47 contrib

drwxr-xr-x. 2 502 games 4096 11月 2 02:54 dist-maven

drwxr-xr-x. 6 502 games 4096 11月 2 02:52 docs

-rw-r--r--. 1 502 games 1709 11月 2 02:47 ivysettings.xml

-rw-r--r--. 1 502 games 8197 11月 2 02:47 ivy.xml

drwxr-xr-x. 4 502 games 4096 11月 2 02:52 lib

-rw-r--r--. 1 502 games 11938 11月 2 02:47 LICENSE.txt

-rw-r--r--. 1 502 games 3132 11月 2 02:47 NOTICE.txt

-rw-r--r--. 1 502 games 1585 11月 2 02:47 README.md

-rw-r--r--. 1 502 games 1770 11月 2 02:47 README\_packaging.txt

drwxr-xr-x. 5 502 games 47 11月 2 02:47 recipes

drwxr-xr-x. 8 502 games 211 11月 2 02:52 src

-rw-r--r--. 1 502 games 1478279 11月 2 02:49 zookeeper-3.4.11.jar

-rw-r--r--. 1 502 games 195 11月 2 02:52 zookeeper-3.4.11.jar.asc

-rw-r--r--. 1 502 games 33 11月 2 02:49 zookeeper-3.4.11.jar.md5

-rw-r--r--. 1 502 games 41 11月 2 02:49 zookeeper-3.4.11.jar.sha1

[root@J2001 zookeeper]# **cd conf**

[root@J2001 conf]# **ll**

总用量 12

-rw-r--r--. 1 502 games 535 11月 2 02:47 configuration.xsl

-rw-r--r--. 1 502 games 2161 11月 2 02:47 log4j.properties

-rw-r--r--. 1 502 games 922 11月 2 02:47 zoo\_sample.cfg

[root@J2001 conf]#  **mv zoo\_sample.cfg zoo.cfg**

[root@J2001 conf]# **ll**

总用量 12

-rw-r--r--. 1 502 games 535 11月 2 02:47 configuration.xsl

-rw-r--r--. 1 502 games 2161 11月 2 02:47 log4j.properties

-rw-r--r--. 1 502 games 922 11月 2 02:47 zoo.cfg

[root@J2001 conf]# **vim zoo.cfg**

修改

**dataDir=/usr/local/sofeware/zookeeper/data**

添加

**server.0=192.168.118.6:2888:3888**

**server.1=192.168.118.9:2888:3888**

**server.2=192.168.118.10:2888:3888**

[root@J2001 conf]# **cd ..**

[root@J2001 zookeeper]# **mkdir data**

[root@J2001 zookeeper]# **ll**

[root@J2001 data]# vim myid

在**server.0=192.168.118.6 这个主机的文件myid中添加 一个 0**

在**server.1=192.168.118.9 这个主机的文件myid中添加 一个 1**

在**server.2=192.168.118.10 这个主机的文件myid中添加 一个 2**

**五)启动服务**

[root@J2001 data]# **zkServer.sh start 启动zookeeper服务**

ZooKeeper JMX enabled by default

Using config: /usr/local/sofeware/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

[root@J2001 data]# **zkServer.sh status 查看状态**

ZooKeeper JMX enabled by default

Using config: /usr/local/sofeware/zookeeper/bin/../conf/zoo.cfg

Error contacting service. **It is probably not running.**

[root@J2001 data]# **systemctl stop firewalld 关闭防火墙**

[root@J2001 data]# **zkServer.sh restart 重启zookeeper服务器**

ZooKeeper JMX enabled by default

Using config: /usr/local/sofeware/zookeeper/bin/../conf/zoo.cfg

ZooKeeper JMX enabled by default

Using config: /usr/local/sofeware/zookeeper/bin/../conf/zoo.cfg

Stopping zookeeper ... STOPPED

ZooKeeper JMX enabled by default

Using config: /usr/local/sofeware/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

[root@J2001 data]# **zkServer.sh status 查看状态**

ZooKeeper JMX enabled by default

Using config: /usr/local/sofeware/zookeeper/bin/../conf/zoo.cfg

Mode: **follower**

[root@J2001 data]# zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /usr/local/sofeware/zookeeper/bin/../conf/zoo.cfg

Mode: **leader**

### 配置ActiveMQ环境

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 主机IP | 集群端口号 | 消息端口号 | 控制台端口号 | 节点目录 |
| 192.168.118.6 | 62621 | 61616 | 8161 | /usr/local/sofeware/activemq1 |
| 192.168.118.9 | 62622 | 61616 | 8161 | /usr/local/sofeware/activemq2 |
| 192.168.118.10 | 62623 | 61616 | 8161 | /usr/local/sofeware/activemq3 |

1. **上传和解压文件**

tar -zxvf apache-activemq-5.15.3-bin.tar.gz -C /usr/local/sofeware/

分别命名三个服务器上的文件 为 activemq1 activemq2 和activemq3

mv apache-activemq-5.15.3 activemq1

**二）修改配置文件**

ActiveMQ集群

1. 集群中名称必须一致 borkerName 每个节点服务器名称必须相同

<broker xmlns="http://activemq.apache.org/schema/core" brokerName="**activemq-cluste**r" dataDirectory="${activemq.data}">

2）修改zkAdderss hostname zkPath

<persistenceAdapter>

<replicatedLevelDB

directory="${activemq.data}/leveldb"

replicas="3"

bind="tcp://0.0.0.0:62621"

zkAddress="192.168.118.6:2181,192.168.118.9:2181,192.168.118.10:2181"

hostname="192.168.118.6"

zkPath="/activemq/leveldb-stores"

/>

</persistenceAdapter>

第二个节点的修改

<persistenceAdapter>

<replicatedLevelDB

directory="${activemq.data}/leveldb"

replicas="3"

bind="tcp://0.0.0.0:62622"

zkAddress="192.168.118.6:2181,192.168.118.9:2181,192.168.118.10:2181"

hostname="192.168.118.6"

zkPath="/activemq/leveldb-stores"

/>

</persistenceAdapter>

第三个节点的配置和修改

<persistenceAdapter>

<replicatedLevelDB

directory="${activemq.data}/leveldb"

replicas="3"

bind="tcp://0.0.0.0:62623"

zkAddress="192.168.118.6:2181,192.168.118.9:2181,192.168.118.10:2181"

hostname="192.168.118.6"

zkPath="/activemq/leveldb-stores"

/>

</persistenceAdapter>

**三）启动服务**

Linux 🡪Shell编程自学

第一步 启动zookeeper集群 命令 zkServer.sh start 重启zkServer restart

第二步 启动ActiveMQ集群 按照顺序启动即可

[root@J2001 conf]# **cd ..**

[root@J2001 activemq3]# **cd bin/**

[root@J2001 bin]# **activemq start**

INFO: Loading '/usr/local/sofeware/activemq3//bin/env'

INFO: Using java '/usr/bin/java'

INFO: Starting - inspect logfiles specified in logging.properties and log4j.properties to get details

INFO: pidfile created : '/usr/local/sofeware/activemq3//data/activemq.pid' (pid '8590')

[root@J2001 bin]#

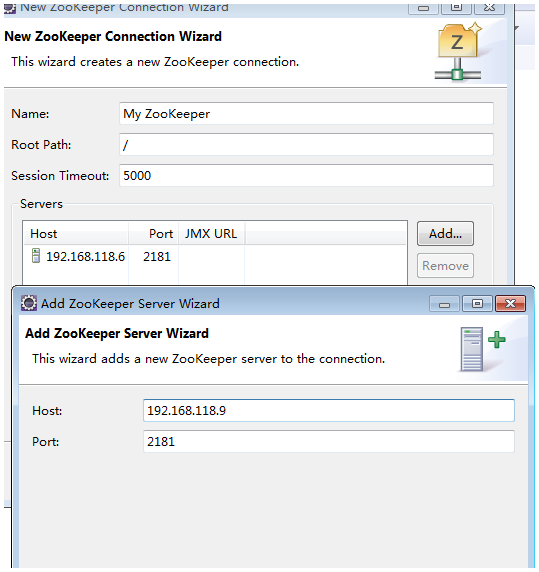
**四）查看日志信息**

如果没有错误说明集群正确启动成功

tail -f /usr/local/sofeware/activemq3/data/activemq.log

可以通过Eclipse的插件进行相关的操作

添加 url http://www.massedynamic.org/eclipse/updates/



## 连接测试