ActiveMq

|  |
| --- |
| https://timgsa.baidu.com/timg?image&quality=80&size=b9999_10000&sec=1514915844615&di=1af970bc55fc0450e5df77af9f45d5eb&imgtype=0&src=http%3A%2F%2Fimg2.shangxueba.com%2Fimg%2Fku%2F20140828%2F13%2F75C2E1277E5AFD23A3EDBFFBD4A3AEE9.png  Message Broker就是消息实例 |

# 1 介绍

Mq：消息中间件

activeMq：Apache

rabbitmq：AMQP，即Advanced Message Queuing Protocol

kafka：Apache Kafka，大数据

ZeroMQ：简单快速，是个类似于Socket的一系列接口

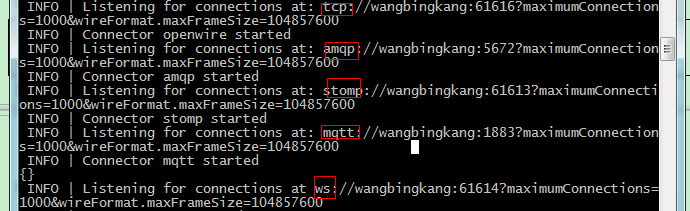
MetaMQ(RocketMQ)：阿里开源消息中间件MetaQ(RocketMQ)

# 2 安装

Windows和linux通用（到activeMq安装根目录执行）

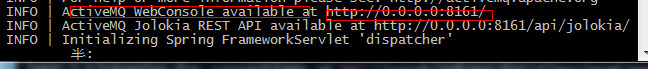
|  |
| --- |
| 1 activeMq start  2 activeMq stop |

执行完后通过DOS发现有好多监听，说明activeMq支持好多方式，



但是我们一般传输的都是string类型的消息，所以使用tcp足够了

下面还发现了http的web控制台路径，即可以在浏览器打开的



ActiveMQ WebConsole available at http://0.0.0.0:8161/（通过localhost:8161即可访问）

登录进去后点击Manage ActiveMQ broker；会让你输入用户名和密码，都是admin

Linux安装注意事项

|  |
| --- |
| Active集成控制台页面跳转需要配置Hosts文件  1 etc/hosts  配置网络映射地址列表  2 etc/sysconfig/network  配置主机名 |

# 3 MQ(队列)测试代码

Pom依赖信息

|  |
| --- |
| <dependency>  <groupId>org.apache.activemq</groupId>  <artifactId>activemq-all</artifactId>  <version>5.11.2</version>  </dependency> |

发布队列消息到mq

|  |
| --- |
| ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(url);  Connection connection = connectionFactory.createConnection();  connection.start();  Session session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);  Queue queue = session.createQueue(application);  MessageProducer producer = session.createProducer(queue);  TextMessage textMessage = session.createTextMessage(message);  producer.send(textMessage);  producer.close();  session.close();  connection.close(); |

消费消息端服务：

|  |
| --- |
| ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(url);  Connection connection = connectionFactory.createConnection();  connection.start();  Session session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);  Queue queue = session.createQueue(application);  MessageConsumer consumer = session.createConsumer(queue);  // 接收消息  consumer.setMessageListener(new MessageListener() {  @Override  public void onMessage(Message message) {  // 打印结果  TextMessage textMessage = (TextMessage) message;  String text= textMessage.getText();  System.out.println(text);  }  });  // 等待接收消息  System.in.read(); |

# 4 MQ(订阅模式)测试代码

Pom依赖信息

|  |
| --- |
| <dependency>  <groupId>org.apache.activemq</groupId>  <artifactId>activemq-all</artifactId>  <version>5.11.2</version>  </dependency> |

发布事件消息到mq

|  |
| --- |
| ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(url);  Connection connection = connectionFactory.createConnection();  connection.start();  Session session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);  Topic topic = session.createTopic(application);  MessageProducer producer = session.createProducer(topic);  TextMessage textMessage = session.createTextMessage(message);  producer.send(textMessage);  producer.close();  session.close();  connection.close(); |

消费消息端服务：

|  |
| --- |
| ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(url);  Connection connection = connectionFactory.createConnection();  connection.start();  Session session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);  Topic topic = session.createTopic(application);  MessageConsumer consumer = session.createConsumer(topic);  // 接收消息  consumer.setMessageListener(new MessageListener() {  @Override  public void onMessage(Message message) {  // 打印结果  TextMessage textMessage = (TextMessage) message;  String text = textMessage.getText();  System.out.println(text);  }  });  System.out.println("订阅消费者n。。。。");  // 等待接收消息  System.in.read();  } |

# 5spring整合mq发布任务消息

Pom

|  |
| --- |
| <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jms</artifactId>  <version>4.0.0.RELEASE</version>  </dependency>  <dependency>  <groupId>org.apache.activemq</groupId>  <artifactId>activemq-all</artifactId>  <version>5.11.2</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  <version>4.0.0.RELEASE</version>  </dependency> |

在spring容器中创建MQ工厂

|  |
| --- |
| <!-- jsm消息工厂 -->  <bean id="targetConnectionFactory" class="org.apache.activemq.ActiveMQConnectionFactory">  <property name="brokerURL" value="tcp://localhost:61616" />  </bean>  <bean id="connectionFactory"  class="org.springframework.jms.connection.SingleConnectionFactory">  <!--产生JMS Connection的ConnectionFactory -->  <property name="targetConnectionFactory" ref="targetConnectionFactory" />  </bean> |

MQ执行模板

|  |
| --- |
| <!-- 执行消息任务的模板 -->  <bean id="jmsTemplate" class="org.springframework.jms.core.JmsTemplate">  <property name="connectionFactory" ref="connectionFactory" />  </bean>  <bean id="queueDestination" class="org.apache.activemq.command.ActiveMQQueue">  <constructor-arg value="queue" />  </bean>  <bean id="topicDestination" class="org.apache.activemq.command.ActiveMQTopic">  <constructor-arg value="topic" />  </bean> |

消息产生端测试代码：即消息任务发布的系统程序

|  |
| --- |
| // 发送mq消息  jmsTemplate.send(destination, new MessageCreator() {  @Override  public Message createMessage(Session session) throws JMSException {  return session.createTextMessage("send activemq message");  }  }); |

# 6 spring整合mq执行消息任务

通过实现MessageListener创建监听类，获得消息任务信息

MyMessageListener

|  |
| --- |
| import javax.jms.MessageListener  public class MyMessageListener implements MessageListener {  @Override  public void onMessage(Message message) {  // 消息内容  TextMessage textMessage = (TextMessage) message;  String text = textMessage.getText();  System.out.println(text);  }  } |

配置监听到spring容器中

|  |
| --- |
| <bean id="myMessageListener" class="MyMessageListener"/>  <!-- 消息监听容器 -->  <bean class="org.springframework.jms.listener.DefaultMessageListenerContainer">  <property name="connectionFactory" ref="connectionFactory" />  <property name="destination" ref="queueDestination" />  <property name="messageListener" ref="myMessageListener" />  </bean>  <!--订阅模式监听容器-->  <bean class="org.springframework.jms.listener.DefaultMessageListenerContainer">  <property name="connectionFactory" ref="connectionFactory" />  <property name="destination" ref="topicDestination" />  <property name="messageListener" ref="myMessageListener" />  </bean> |