Atiit rabbit mq 读取发送范例

所有mq选型特点

目录

[1. Activemq rabbit mq 区别 1](#_Toc27529)

[2. Activemq rabbit mq 区别ati总结 1](#_Toc24620)

[3. 发送范例 2](#_Toc7071)

[3.1. 接受 6](#_Toc12711)

[3.2. 查询 10](#_Toc6427)

[4. Ref 14](#_Toc7686)

# Activemq rabbit mq 区别

1. ActiveMQ/ApolloMQ  
　　优点：老牌的消息队列，使用Java语言编写。对JMS支持最好，采用多线程并发，资源消耗比较大。如果你的主语言是Java，可以重点考虑。  
　　缺点：由于历史悠久，历史包袱较多，版本更新很缓慢。集群模式需要依赖Zookeeper实现。最新架构的产品被命名为Apollo，号称下一代ActiveMQ，目前案例较少。  
  
2. RocketMQ/Kafka   
　　优点：专为海量消息传递打造，主张使用拉模式，天然的集群、HA、负载均衡支持。话说还是那句话，适合不适合看你有没有那么大的量。  
　　缺点：所谓鱼和熊掌不可兼得，放弃了一些消息中间件的灵活性，使用的场景较窄，需关注你的业务模式是否契合，否则山寨变相使用很别扭。除此之外，RocketMQ没有.NET下的客户端可用。RocketMQ身出名门，但使用者不多，生态较小，毕竟消息量能达到这种体量的公司不多，你也可以直接去购买阿里云的消息服务。Kafka生态完善，其代码是用Scala语言写成，可靠性比RocketMQ低一些。  
  
　　3. RabbitMQ  
　　优点：生态丰富，使用者众，有很多人在前面踩坑。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。  
　　缺点：Erlang代码你Hold得住不? 虽然Erlang是天然集群化的，但RabbitMQ在高可用方面

# Activemq rabbit mq 区别ati总结

Amq有队列名称概念。。Rmq使用topic概念来区分。。

# 发送范例

*package* org.chwin.firefighting.apiserver.MQTT;  
  
*import* com.aliyun.openservices.ons.api.Message;  
*import* com.aliyun.openservices.ons.api.Producer;  
*import* com.aliyun.openservices.ons.api.SendResult;  
*import* com.aliyun.openservices.ons.api.ONSFactory;  
*import* com.aliyun.openservices.ons.api.PropertyKeyConst;  
*import* org.apache.commons.io.FileUtils;  
  
*import* java.io.File;  
*import* java.io.IOException;  
*import* java.util.Properties;  
*public class* MQTTSendMsg {  
 *public static void* main(String[] args) *throws* IOException {  
 Properties properties = *new* Properties();  
 *// 您在控制台创建的 Group ID GID\_IOT\_HTTP\_PRO* properties.put(PropertyKeyConst.GROUP\_ID, "GID\_TCP\_FIRE\_PRO1");  
 *// 鉴权用 AccessKeyId，在阿里云服务器管理控制台创建* properties.put(PropertyKeyConst.AccessKey,"");  
 *// 鉴权用 AccessKeySecret，在阿里云服务器管理控制台创建* properties.put(PropertyKeyConst.SecretKey, "");  
 *// 设置 TCP 接入域名，进入控制台的实例管理页面，在页面上方选择实例后，在实例信息中的“获取接入点信息”区域查看* properties.put(PropertyKeyConst.NAMESRV\_ADDR,"ernet.aliyuncs.com:80");  
 Producer producer = ONSFactory.*createProducer*(properties);  
 *// 在发送消息前，必须调用 start 方法来启动 Producer，只需调用一次即可* producer.start();  
 String topicname = "T\_Fire\_Alarm";  
 *//循环发送消息  
 while*(*true*){  
 String mqMsg= FileUtils.*readFileToString*(*new* File("D:\\00wkspc\\v4master\\fire\\apiserver\\src\\main\\java\\org\\chwin\\firefighting\\apiserver\\warning\\iot.json"));  
  
 Message msg = *new* Message( *//  
 // 在控制台创建的 Topic，即该消息所属的 Topic 名称 T\_Hidden\_Warn* topicname,  
 *// Message Tag,  
 // 可理解为 Gmail 中的标签，对消息进行再归类，方便 Consumer 指定过滤条件在消息队列 RocketMQ 服务器过滤* "TagA",  
 *// Message Body  
 // 任何二进制形式的数据，消息队列 RocketMQ 不做任何干预，  
 // 需要 Producer 与 Consumer 协商好一致的序列化和反序列化方式* mqMsg.getBytes());  
 *// 设置代表消息的业务关键属性，请尽可能全局唯一，以方便您在无法正常收到消息情况下，可通过控制台查询消息并补发  
 // 注意：不设置也不会影响消息正常收发* msg.setKey("ORDERID\_101");  
 *// 发送消息，只要不抛异常就是成功  
 // 打印 Message ID，以便用于消息发送状态查询* SendResult sendResult = producer.send(msg);  
 System.out.println("Send Message success. Message ID is: " + sendResult.getMessageId());  
 *break*;  
 }  
 *// 在应用退出前，可以销毁 Producer 对象  
 // 注意：如果不销毁也没有问题* producer.shutdown();  
 }  
}

## 接受

*package* org.chwin.firefighting.apiserver.MQTT;  
  
*import* com.aliyun.openservices.ons.api.Action;  
*import* com.aliyun.openservices.ons.api.ConsumeContext;  
*import* com.aliyun.openservices.ons.api.Consumer;  
*import* com.aliyun.openservices.ons.api.Message;  
*import* com.aliyun.openservices.ons.api.MessageListener;  
*import* com.aliyun.openservices.ons.api.ONSFactory;  
*import* com.aliyun.openservices.ons.api.PropertyKeyConst;  
*import* com.sun.xml.internal.bind.v2.runtime.output.SAXOutput;  
*import* org.apache.commons.io.FileUtils;  
*import* org.chwin.firefighting.apiserver.core.util.DateUtil;  
  
*import* java.io.File;  
*import* java.io.IOException;  
*import* java.util.Properties;  
*//import java.util.function.Consumer;  
  
public class* MQTTRecvMsg {  
 *public static void* main(String[] args) {  
 String queruoName\_topic = "T\_Fire\_Alarm";  
 java.util.function.Consumer csmr=*new* java.util.function.Consumer() {  
 @Override  
 *public void* accept(Object msgtxt) {  
 System.out.println(msgtxt);  
 *try* {  
 *// DateUtil.* FileUtils.*writeStringToFile*(*new* File("d:\\0mq\\"+DateUtil.*getDateStr4file*()+".txt"),msgtxt.toString());  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 };  
  
 *ini*(queruoName\_topic, csmr);  
 }  
  
 *public static void* ini(String queruoName, java.util.function.Consumer csmr) {  
 Properties properties = *new* Properties();  
 *// 您在控制台创建的 Group ID jeig recv send byyo* properties.put(PropertyKeyConst.GROUP\_ID, "GID\_TCP\_FIRE\_CON1");  
 *// 鉴权用 AccessKeyId，在阿里云服务器管理控制台创建* properties.put(PropertyKeyConst.AccessKey, "");  
 *// 鉴权用 AccessKeySecret，在阿里云服务器管理控制台创建* properties.put(PropertyKeyConst.SecretKey, "");  
 *// 设置 TCP 接入域名，进入控制台的实例管理页面，在页面上方选择实例后，在实例信息中的“获取接入点信息”区域查看* properties.put(PropertyKeyConst.NAMESRV\_ADDR,"http://s.mq-internet.aliyuncs.com:80");  
 Consumer consumer = ONSFactory.*createConsumer*(properties);  
  
 *//topic tag ??* consumer.subscribe(queruoName, "\*", *new* MessageListener() {  
 @Override  
 *public* Action consume(Message message, ConsumeContext context) {  
 *// System.out.println("Receive: " + new String(message.getBody()));* csmr.accept(*new* String(message.getBody()));  
 *return* Action.CommitMessage;  
 }  
 });  
 consumer.start();  
 System.out.println("Consumer Started");  
 }  
}

## 查询

*package org.chwin.firefighting.apiserver.MQTT;  
//  
//import com.aliyuncs.DefaultAcsClient;  
//import com.aliyuncs.IAcsClient;  
//import com.aliyuncs.exceptions.ClientException;  
//import com.aliyuncs.ons.model.v20190214.OnsMessagePageQueryByTopicRequest;  
//import com.aliyuncs.ons.model.v20190214.OnsMessagePageQueryByTopicResponse;  
//import com.aliyuncs.profile.DefaultProfile;  
//import com.aliyuncs.profile.IClientProfile;  
//  
//import java.io.UnsupportedEncodingException;  
//import java.util.Base64;  
//import java.util.List;  
//  
//public class MQTTQueryMsg {  
// public static void main(String[] args) {  
// /\*\*  
// \*Open API 的接入点，设置为目标 Region  
// \*/  
// String regionId = "mq-internet-access";  
// /\*\*  
// \*鉴权使用的 AccessKeyId，由阿里云管理控制台获取  
// \*/  
// String accessKey = "";  
// /\*\*  
// \*鉴权使用的 AccessKeySecret，由阿里云管理控制台获取  
// \*/  
// String secretKey = "";  
// IClientProfile profile = DefaultProfile.getProfile(regionId, accessKey, secretKey);  
// IAcsClient iAcsClient = new DefaultAcsClient(profile);  
// OnsMessagePageQueryByTopicRequest request = new OnsMessagePageQueryByTopicRequest();  
// request.setPreventCache(System.currentTimeMillis());  
// request.setInstanceId("Mc4");  
// request.setTopic("T\_Hidden\_Warn");  
// request.setBeginTime(System.currentTimeMillis() - 24 \* 3600 \* 1000 \* 10);  
// request.setEndTime(System.currentTimeMillis());  
// request.setCurrentPage(1);  
// request.setPageSize(100);  
//  
// try {  
// OnsMessagePageQueryByTopicResponse response = iAcsClient.getAcsResponse(request);  
// OnsMessagePageQueryByTopicResponse.MsgFoundDo msgFoundDo = response.getMsgFoundDo();  
// List<OnsMessagePageQueryByTopicResponse.MsgFoundDo.OnsRestMessageDo> msgFoundList = msgFoundDo.getMsgFoundList();  
//  
//  
// for (OnsMessagePageQueryByTopicResponse.MsgFoundDo.OnsRestMessageDo msg : msgFoundList  
// ) {  
// String body = msg.getBody();  
// if(null != body){  
// Base64.Decoder decoder = Base64.getDecoder();  
//  
// try {  
// String gbk = new String(decoder.decode(body.getBytes()), "UTF-8");  
// System.out.println(gbk);  
// } catch (UnsupportedEncodingException e) {  
// }  
// }  
// }  
//  
// } catch (ClientException e) {  
// e.printStackTrace();  
// }  
// }  
//}*

# Ref

Atitit mq读取队列信息 范例

Activemq的