# 基于Redis的消息队列之发布订阅模式

## 1、pom.xml

与我之前发表的文章《基于Redis的消息队列之生产消费者模式》内容一样，不写了。

## 2、redis.properties

同上。

## 3、applicationContext.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd*  *"*>  <context:component-scan base-package=*"org.leo.ssm"* />  <!-- 属性文件读入 -->  <context:property-placeholder location=*"classpath:redis.properties"* />  <import resource=*"applicationContext-redis.xml"* />  </beans> |

## 4、applicationContext-redis.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:redis=*"http://www.springframework.org/schema/redis"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/redis*  *http://www.springframework.org/schema/redis/spring-redis-1.0.xsd"*>  <bean id=*"poolConfig"* class=*"redis.clients.jedis.JedisPoolConfig"*>  <property name=*"maxIdle"* value=*"${redis.maxIdle}"* />  <property name=*"maxTotal"* value=*"${redis.maxTotal}"* />  <property name=*"maxWaitMillis"* value=*"${redis.maxWaitMillis}"* />  <property name=*"testOnBorrow"* value=*"${redis.testOnBorrow}"* />  </bean>  <bean id=*"redisConnectionFactory"*  class=*"org.springframework.data.redis.connection.jedis.JedisConnectionFactory"*  p:host-name=*"${redis.host}"* p:port=*"${redis.port}"* p:password=*"${redis.pwd}"*  p:pool-config-ref=*"poolConfig"* />  <bean id=*"redisTemplate"* class=*"org.springframework.data.redis.core.RedisTemplate"*>  <property name=*"connectionFactory"* ref=*"redisConnectionFactory"* />  <property name=*"keySerializer"*>  <bean  class=*"org.springframework.data.redis.serializer.StringRedisSerializer"* />  </property>  <property name=*"valueSerializer"*>  <bean  class=*"org.springframework.data.redis.serializer.StringRedisSerializer"*></bean>  </property>  </bean>  <!--配置订阅者1 -->  <bean id=*"listener"* class=*"org.leo.ssm.listener.MessageDelegateListenerImpl"* />  <!--配置订阅者2 -->  <bean id=*"listener2"* class=*"org.leo.ssm.listener.MessageDelegateListener2Impl"* />  <!--配置订阅者接收的消息是String类型，不配置的话，默认是JdkSerializationRedisSerializer -->  <bean id=*"stringSerializer"*  class=*"org.springframework.data.redis.serializer.StringRedisSerializer"* />  <!--MessageListenerAdapter默认的频道名是String，消息是jdk -->  <redis:listener-container connection-factory=*"redisConnectionFactory"*>  <!-- 如果方法名为"handleMessage"，method的属性可以跳过不配置，凡是频道为hello开头，比如hello1、helloaa的交由listener2进行处理 -->  <redis:listener ref=*"listener2"* serializer=*"stringSerializer"*  topic=*"hello\*"* />  <!--凡是频道为java或tv开头的消息交由listener处理 -->  <redis:listener ref=*"listener"* serializer=*"stringSerializer"*  method=*"handleMessage"* topic=*"java tv\*"* />  </redis:listener-container>  <!--如果是低版本的jar，想多配几个listener，或者一个listener多订阅几个频道都会报错 -->  <!--顺便说一句，不用redis:listener-container，而用普通bean的配置方法也能成功，只是本文档就不再多写了 -->  </beans> |

## 5、订阅者

|  |
| --- |
| **package** org.leo.ssm.listener;  **public** **class** MessageDelegateListenerImpl **implements** MessageDelegateListener {  @Override  **public** **void** handleMessage(String msg) {  // 处理逻辑  **if** (**null** == msg || "".equals(msg)) {  System.***out***.println("Listener1 receive msg is null");  } **else** {  System.***out***.println("Listener1 receive msg:" + msg);  }  }  } |

接口就不附上了，另一个订阅者代码跟基本一致，也不附上了。

## 6、发布者

在我之前发表的《基于Redis的消息队列之生产消费者模式》中有RedisQueueDaoImpl、RedisQueueService两个接口和实现类，以及测试类，分别添加如下方法。

Dao：

|  |
| --- |
| **public** **void** sendMsg(String channel, String value) {  redisTemplate.convertAndSend(channel, value);  } |

Service：

|  |
| --- |
| **public** **void** sendMsg(String channel, String value) {  redisQueueDao.sendMsg(channel, value);  } |

测试类：

|  |
| --- |
| **public** **void** testPub() {  **for** (**int** i = 0; i < 10; i++) {  redisQueueService.sendMsg("java", "测试" + i);  redisQueueService.sendMsg("hello1", "测试" + i);  }  redisQueueService.sendMsg("tv1", "OK");  } |

将订阅者工程打包，启动Tomcat，运行测试类，结果如下：

注意，因为是多线程，所以订阅者接收并打印的信息是无序的。

|  |
| --- |
| Listener2 receive msg:测试0  Listener1 receive msg:测试0  Listener2 receive msg:测试2  Listener1 receive msg:测试4  Listener2 receive msg:测试3  Listener1 receive msg:测试3  Listener1 receive msg:测试2  Listener1 receive msg:测试1  Listener2 receive msg:测试1  Listener2 receive msg:测试4  Listener1 receive msg:测试5  Listener1 receive msg:测试9  Listener1 receive msg:测试8  Listener2 receive msg:测试8  Listener1 receive msg:测试7  Listener2 receive msg:测试7  Listener1 receive msg:测试6  Listener2 receive msg:测试6  Listener2 receive msg:测试5  Listener2 receive msg:测试9  Listener1 receive msg:OK |