# ActiveMQ分享5--使用Spring JMS收发消息

每次收发消息都要写许多重复的代码，Spring 为我们提供了更为方便的方式，这就是Spring JMS。我们通过一个例子展开讲述。包括队列、主题消息的收发相关的Spring配置、代码、测试。

       本例中，消息的收发都写在了一个工程里。

### 1.使用maven管理依赖包

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | < dependencies >       < dependency >           < groupId >junit</ groupId >           < artifactId >junit</ artifactId >           < version >4.12</ version >           < scope >test</ scope >       </ dependency >       < dependency >           < groupId >org.apache.activemq</ groupId >           < artifactId >activemq-all</ artifactId >           < version >5.11.0</ version >       </ dependency >       < dependency >           < groupId >org.springframework</ groupId >           < artifactId >spring-jms</ artifactId >           < version >4.1.4.RELEASE</ version >       </ dependency >       < dependency >           < groupId >org.springframework</ groupId >           < artifactId >spring-test</ artifactId >           < version >4.1.4.RELEASE</ version >       </ dependency >  </ dependencies > |

### 2.队列消息的收发

#### 2.1Spring配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | <? xml  version = "1.0"  encoding = "UTF-8" ?>  < beans  xmlns = "<http://www.springframework.org/schema/beans>"  [xmlns:xsi](http://xmlnsxsi) = "<http://www.w3.org/2001/XMLSchema-instance>"       xsi:schemaLocation="http://www.springframework.org/schema/beans  <http://www.springframework.org/schema/beans/spring-beans.xsd>">         <!-- 配置JMS连接工厂 -->       < bean  id = "connectionFactory"  class = "org.apache.activemq.ActiveMQConnectionFactory" >           < property  name = "brokerURL"  value = "failover:(<tcp://localhost:61616>)"  />       </ bean >         <!-- 定义消息队列（Queue） -->       < bean  id = "queueDestination"  class = "org.apache.activemq.command.ActiveMQQueue" >           <!-- 设置消息队列的名字 -->           < constructor-arg >               < value >queue1</ value >           </ constructor-arg >       </ bean >         <!-- 配置JMS模板（Queue），Spring提供的JMS工具类，它发送、接收消息。 -->       < bean  id = "jmsTemplate"  class = "org.springframework.jms.core.JmsTemplate" >           < property  name = "connectionFactory"  ref = "connectionFactory"  />           < property  name = "defaultDestination"  ref = "queueDestination"  />           < property  name = "receiveTimeout"  value = "10000"  />       </ bean >         <!--queue消息生产者 -->       < bean  id = "producerService"  class = "guo.examples.mq02.queue.ProducerServiceImpl" >           < property  name = "jmsTemplate"  ref = "jmsTemplate" ></ property >       </ bean >         <!--queue消息消费者 -->       < bean  id = "consumerService"  class = "guo.examples.mq02.queue.ConsumerServiceImpl" >           < property  name = "jmsTemplate"  ref = "jmsTemplate" ></ property >       </ bean > |

#### 2.2消息生产者代码

从下面的代码可以出，使用Spring JMS，可以减少重复代码（接口类ProducerService代码省略）。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | package  guo.examples.mq02.queue;    import  javax.jms.Destination;  import  javax.jms.JMSException;  import  javax.jms.Message;  import  javax.jms.Session;    import  org.springframework.jms.core.JmsTemplate;  import  org.springframework.jms.core.MessageCreator;    public  class  ProducerServiceImpl  implements  ProducerService {       private  JmsTemplate jmsTemplate;       /\*\*      \* 向指定队列发送消息      \*/     public  void  sendMessage(Destination destination,  final  String msg) {       System.out.println( "向队列"  + destination.toString() +  "发送了消息------------"  + msg);       jmsTemplate.send(destination,  new  MessageCreator() {         public  Message createMessage(Session session)  throws  JMSException {           return  session.createTextMessage(msg);         }       });     }    /\*\*    \* 向默认队列发送消息    \*/     public  void  sendMessage( final  String msg) {       String destination =  jmsTemplate.getDefaultDestination().toString();       System.out.println( "向队列"  +destination+  "发送了消息------------"  + msg);       jmsTemplate.send( new  MessageCreator() {         public  Message createMessage(Session session)  throws  JMSException {           return  session.createTextMessage(msg);         }       });       }       public  void  setJmsTemplate(JmsTemplate jmsTemplate) {       this .jmsTemplate = jmsTemplate;     }    } |

#### 2.3消息消费者代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | package  guo.examples.mq02.queue;    import  javax.jms.Destination;  import  javax.jms.JMSException;  import  javax.jms.TextMessage;    import  org.springframework.jms.core.JmsTemplate;    public  class  ConsumerServiceImpl  implements  ConsumerService {         private  JmsTemplate jmsTemplate;         /\*\*        \* 接受消息        \*/       public  void  receive(Destination destination) {           TextMessage tm = (TextMessage) jmsTemplate.receive(destination);           try  {               System.out.println( "从队列"  + destination.toString() +  "收到了消息：\t"                       + tm.getText());           }  catch  (JMSException e) {               e.printStackTrace();           }       }         public  void  setJmsTemplate(JmsTemplate jmsTemplate) {           this .jmsTemplate = jmsTemplate;       }    } |

#### 2.4队列消息监听

接受消息的时候，可以不用2.3节中的方式，Spring JMS同样提供了消息监听的模式，下面给出对应的配置和代码。

**Spring配置**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | <!-- 定义消息队列（Queue），我们监听一个新的队列，queue2 -->       < bean  id = "queueDestination2"  class = "org.apache.activemq.command.ActiveMQQueue" >           <!-- 设置消息队列的名字 -->           < constructor-arg >               < value >queue2</ value >           </ constructor-arg >       </ bean >         <!-- 配置消息队列监听者（Queue），代码下面给出，只有一个onMessage方法 -->       < bean  id = "queueMessageListener"  class = "guo.examples.mq02.queue.QueueMessageListener"  />         <!-- 消息监听容器（Queue），配置连接工厂，监听的队列是queue2，监听器是上面定义的监听器 -->       < bean  id = "jmsContainer"           class = "org.springframework.jms.listener.DefaultMessageListenerContainer" >           < property  name = "connectionFactory"  ref = "connectionFactory"  />           < property  name = "destination"  ref = "queueDestination2"  />           < property  name = "messageListener"  ref = "queueMessageListener"  />       </ bean > |

**监听类代码**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | package  guo.examples.mq02.queue;    import  javax.jms.JMSException;  import  javax.jms.Message;  import  javax.jms.MessageListener;  import  javax.jms.TextMessage;    public  class  QueueMessageListener  implements  MessageListener {           //当收到消息时，自动调用该方法。       public  void  onMessage(Message message) {           TextMessage tm = (TextMessage) message;           try  {               System.out.println( "ConsumerMessageListener收到了文本消息：\t"                       + tm.getText());           }  catch  (JMSException e) {               e.printStackTrace();           }       }    } |

### 3.主题消息收发

     在使用Spring JMS的时候，主题（Topic）和队列消息的主要差异体现在JmsTemplate中"pubSubDomain"是否设置为True。如果为True，则是Topic；如果是false或者默认，则是queue。

|  |  |
| --- | --- |
| 1 | < property  name = "pubSubDomain"  value = "true"  /> |

#### 3.1Spring配置

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | <!-- 定义消息主题（Topic） -->       < bean  id = "topicDestination"  class = "org.apache.activemq.command.ActiveMQTopic" >           < constructor-arg >               < value >guo\_topic</ value >           </ constructor-arg >       </ bean >       <!-- 配置JMS模板（Topic），pubSubDomain="true"-->       < bean  id = "topicJmsTemplate"  class = "org.springframework.jms.core.JmsTemplate" >           < property  name = "connectionFactory"  ref = "connectionFactory"  />           < property  name = "defaultDestination"  ref = "topicDestination"  />           < property  name = "pubSubDomain"  value = "true"  />           < property  name = "receiveTimeout"  value = "10000"  />       </ bean >       <!--topic消息发布者 -->       < bean  id = "topicProvider"  class = "guo.examples.mq02.topic.TopicProvider" >           < property  name = "topicJmsTemplate"  ref = "topicJmsTemplate" ></ property >       </ bean >       <!-- 消息主题监听者 和 主题监听容器 可以配置多个，即多个订阅者 -->       <!-- 消息主题监听者(Topic) -->       < bean  id = "topicMessageListener"  class = "guo.examples.mq02.topic.TopicMessageListener"  />       <!-- 主题监听容器 （Topic） -->       < bean  id = "topicJmsContainer"           class = "org.springframework.jms.listener.DefaultMessageListenerContainer" >           < property  name = "connectionFactory"  ref = "connectionFactory"  />           < property  name = "destination"  ref = "topicDestination"  />           < property  name = "messageListener"  ref = "topicMessageListener"  />       </ bean > |

#### 3.2消息发布者

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | package  guo.examples.mq02.topic;    import  javax.jms.Destination;  import  javax.jms.JMSException;  import  javax.jms.Message;  import  javax.jms.Session;    import  org.springframework.jms.core.JmsTemplate;  import  org.springframework.jms.core.MessageCreator;    public  class  TopicProvider {         private  JmsTemplate topicJmsTemplate;         /\*\*        \* 向指定的topic发布消息        \*        \* @param topic        \* @param msg        \*/       public  void  publish( final  Destination topic,  final  String msg) {             topicJmsTemplate.send(topic,  new  MessageCreator() {               public  Message createMessage(Session session)  throws  JMSException {                   System.out.println( "topic name 是"  + topic.toString()                           +  "，发布消息内容为:\t"  + msg);                   return  session.createTextMessage(msg);               }           });       }         public  void  setTopicJmsTemplate(JmsTemplate topicJmsTemplate) {           this .topicJmsTemplate = topicJmsTemplate;       }    } |

#### 3.3消息订阅者（监听）

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | package  guo.examples.mq02.topic;    import  javax.jms.JMSException;  import  javax.jms.Message;  import  javax.jms.MessageListener;  import  javax.jms.TextMessage;  /\*\*    \*和队列监听的代码一样。    \*/  public  class  TopicMessageListener  implements  MessageListener {         public  void  onMessage(Message message) {           TextMessage tm = (TextMessage) message;           try  {               System.out.println( "TopicMessageListener \t"  + tm.getText());           }  catch  (JMSException e) {               e.printStackTrace();           }       }    } |

### 4.测试

#### 4.1 测试代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114 | package  guo.examples.mq02;    import  javax.jms.Destination;    import  guo.examples.mq02.queue.ConsumerService;  import  guo.examples.mq02.queue.ProducerService;  import  guo.examples.mq02.topic.TopicProvider;    import  org.junit.Test;  import  org.junit.runner.RunWith;  import  org.springframework.beans.factory.annotation.Autowired;  import  org.springframework.beans.factory.annotation.Qualifier;  import  org.springframework.test.context.ContextConfiguration;  import  org.springframework.test.context.junit4.SpringJUnit4ClassRunner;    /\*\*    \* 测试Spring JMS    \*    \* 1.测试生产者发送消息    \*    \* 2. 测试消费者接受消息    \*    \* 3. 测试消息监听    \*    \* 4.测试主题监听    \*    \*/  @RunWith (SpringJUnit4ClassRunner. class )  // ApplicationContext context = new  // ClassPathXmlApplicationContext("applicationContext.xml");  @ContextConfiguration ( "/applicationContext.xml" )  public  class  SpringJmsTest {         /\*\*        \* 队列名queue1        \*/       @Autowired       private  Destination queueDestination;         /\*\*        \* 队列名queue2        \*/       @Autowired       private  Destination queueDestination2;         /\*\*        \* 主题 guo\_topic        \*/       @Autowired       @Qualifier ( "topicDestination" )       private  Destination topic;         /\*\*        \* 主题消息发布者        \*/       @Autowired       private  TopicProvider topicProvider;         /\*\*        \* 队列消息生产者        \*/       @Autowired       @Qualifier ( "producerService" )       private  ProducerService producer;         /\*\*        \* 队列消息生产者        \*/       @Autowired       @Qualifier ( "consumerService" )       private  ConsumerService consumer;         /\*\*        \* 测试生产者向queue1发送消息        \*/       @Test       public  void  testProduce() {           String msg =  "Hello world!" ;           producer.sendMessage(msg);       }         /\*\*        \* 测试消费者从queue1接受消息        \*/       @Test       public  void  testConsume() {           consumer.receive(queueDestination);       }         /\*\*        \* 测试消息监听        \*        \* 1.生产者向队列queue2发送消息        \*        \* 2.ConsumerMessageListener监听队列，并消费消息        \*/       @Test       public  void  testSend() {           producer.sendMessage(queueDestination2,  "Hello China~~~~~~~~~~~~~~~" );       }         /\*\*        \* 测试主题监听        \*        \* 1.生产者向主题发布消息        \*        \* 2.ConsumerMessageListener监听主题，并消费消息        \*/       @Test       public  void  testTopic()  throws  Exception {           topicProvider.publish(topic,  "Hello T-To-Top-Topi-Topic!" );       }    } |

#### 4.2 测试结果

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | topic name 是topic: //guo\_topic ，发布消息内容为:    Hello T-To-Top-Topi-Topic!  TopicMessageListener   Hello T-To-Top-Topi-Topic!  向队列queue: //queue2 发送了消息------------Hello China~~~~~~~~~~~~~~~  ConsumerMessageListener收到了文本消息： Hello China~~~~~~~~~~~~~~~  向队列queue: //queue1 发送了消息------------Hello world!  从队列queue: //queue1 收到了消息： Hello world! |