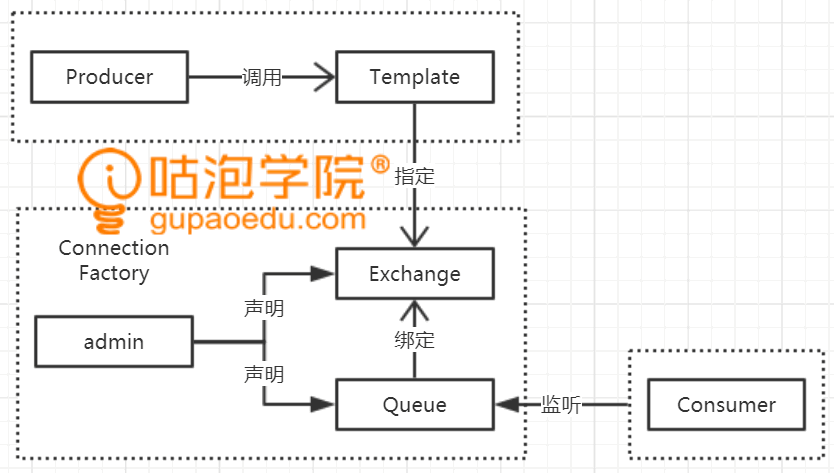
**Spring整合RabbitMQ （JDK 1.8环境）**

# 创建Maven工程

pom.xml添加依赖

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.gupaoedu</groupId>  <artifactId>spring-rabbitmq</artifactId>  <version>0.0.1-SNAPSHOT</version>  <properties>  <!-- spring版本号 -->  <spring.version>4.3.14.RELEASE</spring.version>  <!-- log4j日志文件管理包版本 -->  <slf4j.version>1.6.6</slf4j.version>  <log4j.version>1.2.12</log4j.version>  <!-- junit版本号 -->  <junit.version>4.10</junit.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-core</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  <version>${spring.version}</version>  </dependency>  <!--rabbitmq依赖 -->  <dependency>  <groupId>org.springframework.amqp</groupId>  <artifactId>spring-rabbit</artifactId>  <version>1.3.5.RELEASE</version>  </dependency>  <!-- 日志文件管理包 -->  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>${log4j.version}</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>${slf4j.version}</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>${slf4j.version}</version>  </dependency>  <!-- log end -->  <!--单元测试依赖 -->  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>${junit.version}</version>  <scope>test</scope>  </dependency>  </dependencies>  </project> |

# 二、rabbitMQ.xml配置



放在src/main/resource下

|  |
| --- |
| <?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:rabbit=*"http://www.springframework.org/schema/rabbit"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/rabbit*  *http://www.springframework.org/schema/rabbit/spring-rabbit-1.2.xsd"*>  <!--配置connection-factory，指定连接rabbit server参数 -->  <rabbit:connection-factory id=*"connectionFactory"* virtual-host=*"/"* username=*"guest"* password=*"guest"* host=*"127.0.0.1"* port=*"5672"* />    <!--通过指定下面的admin信息，当前producer中的exchange和queue会在rabbitmq服务器上自动生成 -->  <rabbit:admin id=*"connectAdmin"* connection-factory=*"connectionFactory"* />    <!--######分隔线######-->  <!--定义queue -->  <rabbit:queue name=*"MY\_FIRST\_QUEUE"* durable=*"true"* auto-delete=*"false"* exclusive=*"false"* declared-by=*"connectAdmin"* />  <!--定义direct exchange，绑定MY\_FIRST\_QUEUE -->  <rabbit:direct-exchange name=*"MY\_DIRECT\_EXCHANGE"* durable=*"true"* auto-delete=*"false"* declared-by=*"connectAdmin"*>  <rabbit:bindings>  <rabbit:binding queue=*"MY\_FIRST\_QUEUE"* key=*"FirstKey"*>  </rabbit:binding>  </rabbit:bindings>  </rabbit:direct-exchange>  <!--定义rabbit template用于数据的接收和发送 -->  <rabbit:template id=*"amqpTemplate"* connection-factory=*"connectionFactory"* exchange=*"MY\_DIRECT\_EXCHANGE"* />  <!--消息接收者 -->  <bean id=*"messageReceiver"* class=*"com.gupaoedu.consumer.FirstConsumer"*></bean>  <!--queue listener 观察 监听模式 当有消息到达时会通知监听在对应的队列上的监听对象 -->  <rabbit:listener-container connection-factory=*"connectionFactory"*>  <rabbit:listener queues=*"MY\_FIRST\_QUEUE"* ref=*"messageReceiver"* />  </rabbit:listener-container>    <!--定义queue -->  <rabbit:queue name=*"MY\_SECOND\_QUEUE"* durable=*"true"* auto-delete=*"false"* exclusive=*"false"* declared-by=*"connectAdmin"* />  <!-- 将已经定义的Exchange绑定到MY\_SECOND\_QUEUE，注意关键词是key -->  <rabbit:direct-exchange name=*"MY\_DIRECT\_EXCHANGE"* durable=*"true"* auto-delete=*"false"* declared-by=*"connectAdmin"*>  <rabbit:bindings>  <rabbit:binding queue=*"MY\_SECOND\_QUEUE"* key=*"SecondKey"*></rabbit:binding>  </rabbit:bindings>  </rabbit:direct-exchange>  <!-- 消息接收者 -->  <bean id=*"receiverSecond"* class=*"com.gupaoedu.consumer.SecondConsumer"*></bean>  <!-- queue litener 观察 监听模式 当有消息到达时会通知监听在对应的队列上的监听对象 -->  <rabbit:listener-container connection-factory=*"connectionFactory"*>  <rabbit:listener queues=*"MY\_SECOND\_QUEUE"* ref=*"receiverSecond"* />  </rabbit:listener-container>  <!--######分隔线######-->  <!--定义queue -->  <rabbit:queue name=*"MY\_THIRD\_QUEUE"* durable=*"true"* auto-delete=*"false"* exclusive=*"false"* declared-by=*"connectAdmin"* />  <!-- 定义topic exchange，绑定MY\_THIRD\_QUEUE，注意关键词是pattern -->  <rabbit:topic-exchange name=*"MY\_TOPIC\_EXCHANGE"* durable=*"true"* auto-delete=*"false"* declared-by=*"connectAdmin"*>  <rabbit:bindings>  <rabbit:binding queue=*"MY\_THIRD\_QUEUE"* pattern=*"#.Third.#"*></rabbit:binding>  </rabbit:bindings>  </rabbit:topic-exchange>  <!--定义rabbit template用于数据的接收和发送 -->  <rabbit:template id=*"amqpTemplate2"* connection-factory=*"connectionFactory"* exchange=*"MY\_TOPIC\_EXCHANGE"* />  <!-- 消息接收者 -->  <bean id=*"receiverThird"* class=*"com.gupaoedu.consumer.ThirdConsumer"*></bean>  <!-- queue litener 观察 监听模式 当有消息到达时会通知监听在对应的队列上的监听对象 -->  <rabbit:listener-container connection-factory=*"connectionFactory"*>  <rabbit:listener queues=*"MY\_THIRD\_QUEUE"* ref=*"receiverThird"* />  </rabbit:listener-container>    <!--######分隔线######-->  <!--定义queue -->  <rabbit:queue name=*"MY\_FOURTH\_QUEUE"* durable=*"true"* auto-delete=*"false"* exclusive=*"false"* declared-by=*"connectAdmin"* />  <!-- 定义fanout exchange，绑定MY\_FIRST\_QUEUE 和 MY\_FOURTH\_QUEUE -->  <rabbit:fanout-exchange name=*"MY\_FANOUT\_EXCHANGE"* auto-delete=*"false"* durable=*"true"* declared-by=*"connectAdmin"* >  <rabbit:bindings>  <rabbit:binding queue=*"MY\_FIRST\_QUEUE"*></rabbit:binding>  <rabbit:binding queue=*"MY\_FOURTH\_QUEUE"*></rabbit:binding>  </rabbit:bindings>  </rabbit:fanout-exchange>    <!-- 消息接收者 -->  <bean id=*"receiverFourth"* class=*"com.gupaoedu.consumer.FourthConsumer"*></bean>  <!-- queue litener 观察 监听模式 当有消息到达时会通知监听在对应的队列上的监听对象 -->  <rabbit:listener-container connection-factory=*"connectionFactory"*>  <rabbit:listener queues=*"MY\_FOURTH\_QUEUE"* ref=*"receiverFourth"* />  </rabbit:listener-container>  </beans> |

# 三、applicationContext.xml配置

放在src/main/resource下

|  |
| --- |
| <?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:context=*"http://www.springframework.org/schema/context"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.1.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.1.xsd"*>  <import resource=*"classpath\*:rabbitMQ.xml"* />  <!-- 扫描指定package下所有带有如 @Controller,@Service,@Resource 并把所注释的注册为Spring Beans -->  <context:component-scan base-package=*"com.gupaoedu.\*"* />  <!-- 激活annotation功能 -->  <context:annotation-config />  <!-- 激活annotation功能 -->  <context:spring-configured />  </beans> |

# 四、log4j.properties

放在src/main/resouces下

|  |
| --- |
| log4j.rootLogger=INFO,consoleAppender,fileAppender  log4j.category.ETTAppLogger=DEBUG, ettAppLogFile  log4j.appender.consoleAppender=org.apache.log4j.ConsoleAppender  log4j.appender.consoleAppender.Threshold=TRACE  log4j.appender.consoleAppender.layout=org.apache.log4j.PatternLayout  log4j.appender.consoleAppender.layout.ConversionPattern=%-d{yyyy-MM-dd HH:mm:ss SSS} ->[%t]--[%-5p]--[%c**{1}**]--%m%n  log4j.appender.fileAppender=org.apache.log4j.DailyRollingFileAppender  log4j.appender.fileAppender.File=d:/log/rabbitmq/debug1.log  log4j.appender.fileAppender.DatePattern='\_'yyyy-MM-dd'.log'  log4j.appender.fileAppender.Threshold=TRACE  log4j.appender.fileAppender.Encoding=BIG5  log4j.appender.fileAppender.layout=org.apache.log4j.PatternLayout  log4j.appender.fileAppender.layout.ConversionPattern=%-d{yyyy-MM-dd HH:mm:ss SSS}-->[%t]--[%-5p]--[%c**{1}**]--%m%n  log4j.appender.ettAppLogFile=org.apache.log4j.DailyRollingFileAppender  log4j.appender.ettAppLogFile.File=d:/log/rabbitmq/ettdebug.log  log4j.appender.ettAppLogFile.DatePattern='\_'yyyy-MM-dd'.log'  log4j.appender.ettAppLogFile.Threshold=DEBUG  log4j.appender.ettAppLogFile.layout=org.apache.log4j.PatternLayout  log4j.appender.ettAppLogFile.layout.ConversionPattern=%-d{yyyy-MM-dd HH\:mm\:ss SSS}-->[%t]--[%-5p]--[%c**{1}**]--%m%n |

# 五、编写生产者

MessageProducer.java

放在com.gupaoedu.producer package下面

|  |
| --- |
| **package** com.gupaoedu. producer;  **import** org.slf4j.Logger;  **import** org.slf4j.LoggerFactory;  **import** org.springframework.amqp.core.AmqpTemplate;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.beans.factory.annotation.Qualifier;  **import** org.springframework.stereotype.Service;  /\*\*  \* 消息生产者  \* **@author** Qingshan <br>  \* Better late than never. <br>  \*/  @Service  **public** **class** MessageProducer {  **private** Logger logger = LoggerFactory.*getLogger*(MessageProducer.**class**);    @Autowired  @Qualifier("amqpTemplate")  **private** AmqpTemplate amqpTemplate;    @Autowired  @Qualifier("amqpTemplate2")  **private** AmqpTemplate amqpTemplate2;  /\*\*  \* 演示三种交换机的使用  \*  \* **@param** message  \*/  **public** **void** sendMessage(Object message) {  logger.info("Send message:" + message);    // amqpTemplate 默认交换机 MY\_DIRECT\_EXCHANGE  // amqpTemplate2 默认交换机 MY\_TOPIC\_EXCHANGE  // Exchange 为 direct 模式，直接指定routingKey  amqpTemplate.convertAndSend("FirstKey", "[Direct,FirstKey] "+message);  amqpTemplate.convertAndSend("SecondKey", "[Direct,SecondKey] "+message);    // Exchange模式为topic，通过topic匹配关心该主题的队列  amqpTemplate2.convertAndSend("msg.Third.send","[Topic,msg.Third.send] "+message);    // 广播消息，与Exchange绑定的所有队列都会收到消息，routingKey为空  amqpTemplate2.convertAndSend("MY\_FANOUT\_EXCHANGE",**null**,"[Fanout] "+message);  }  } |

# 六、编写消费者

这里编写了四个消费者，**实现MessageListener接口，重写onMessage()方法**

放在com.gupaoedu.consumer 下面

第一个消费者 FirstConsumer

|  |
| --- |
| **package** com.gupaoedu.consumer;  **import** org.slf4j.Logger;  **import** org.slf4j.LoggerFactory;  **import** org.springframework.amqp.core.Message;  **import** org.springframework.amqp.core.MessageListener;  **public** **class** FirstConsumer **implements** MessageListener {  **private** Logger logger = LoggerFactory.*getLogger*(FirstConsumer.**class**);  @Override  **public** **void** onMessage(Message message) {  logger.info("The first consumer received message : " + message);  }  } |

第二个消费者 SecondConsumer

|  |
| --- |
| **package** com.gupaoedu.consumer;  **import** org.slf4j.Logger;  **import** org.slf4j.LoggerFactory;  **import** org.springframework.amqp.core.Message;  **import** org.springframework.amqp.core.MessageListener;  **public** **class** SecondConsumer **implements** MessageListener {  **private** Logger logger = LoggerFactory.*getLogger*(SecondConsumer.**class**);  @Override  **public** **void** onMessage(Message message) {  logger.info("The second consumer received message : " + message);  }  } |

第三个消费者 ThirdConsumer

|  |
| --- |
| **package** com.gupaoedu.consumer;  **import** org.slf4j.Logger;  **import** org.slf4j.LoggerFactory;  **import** org.springframework.amqp.core.Message;  **import** org.springframework.amqp.core.MessageListener;  **public** **class** ThirdConsumer **implements** MessageListener {  **private** Logger logger = LoggerFactory.*getLogger*(ThirdConsumer.**class**);  @Override  **public** **void** onMessage(Message message) {  logger.info("The third cosumer received message : " + message);  }  } |

第四个消费者 FourthConsumer

|  |
| --- |
| **package** com.gupaoedu.consumer;  **import** org.slf4j.Logger;  **import** org.slf4j.LoggerFactory;  **import** org.springframework.amqp.core.Message;  **import** org.springframework.amqp.core.MessageListener;  **public** **class** FourthConsumer **implements** MessageListener {  **private** Logger logger = LoggerFactory.*getLogger*(FourthConsumer.**class**);  @Override  **public** **void** onMessage(Message message) {  logger.info("The fourth consumer received message : " + message);  }  } |

# 七、编写测试类

src/test/java下新建package

RabbitTest.java

|  |
| --- |
| **package** com.gupaoedu.test;  **import** org.junit.Test;  **import** org.springframework.context.ApplicationContext;  **import** org.springframework.context.support.ClassPathXmlApplicationContext;  **import** com.gupaoedu.producer.MessageProducer;  **public** **class** RabbitTest {  **private** ApplicationContext context = **null**;  @Test  **public** **void** sendMessage() {  context = **new** ClassPathXmlApplicationContext("applicationContext.xml");  MessageProducer messageProducer = (MessageProducer) context.getBean("messageProducer");  **int** k = 100;  **while** (k > 0) {  messageProducer.sendMessage("第" + k + "次发送的消息");  k--;  **try** {  Thread.*sleep*(1000);  } **catch** (Exception e) {  e.printStackTrace();  }  }  }  } |