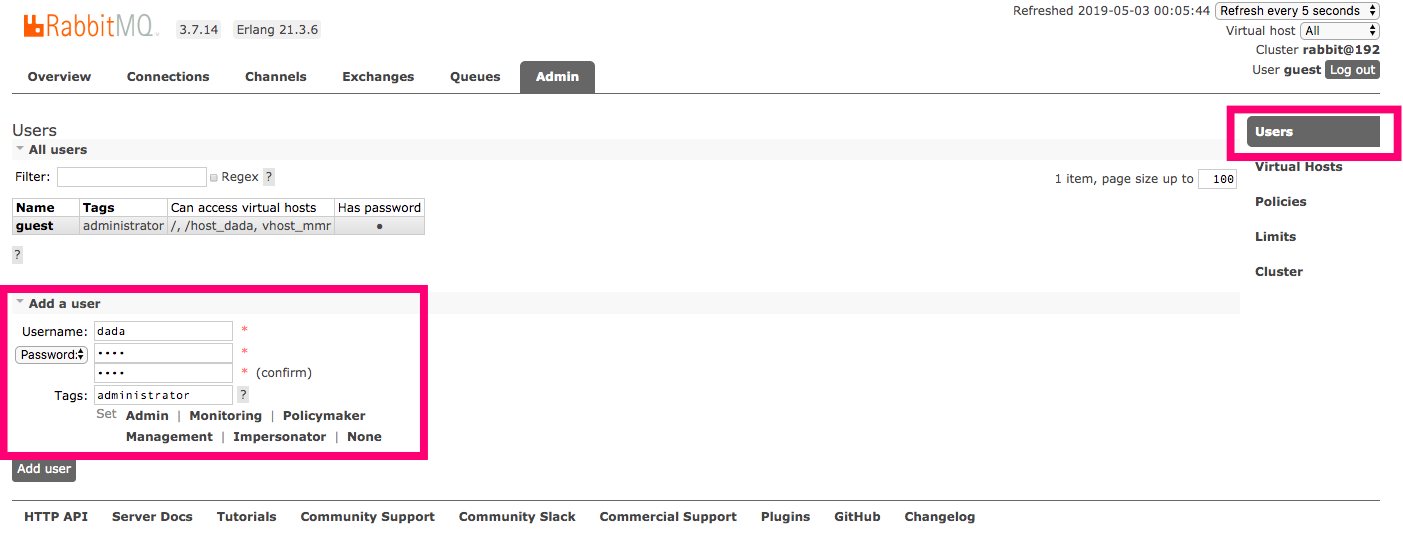
添加用户:

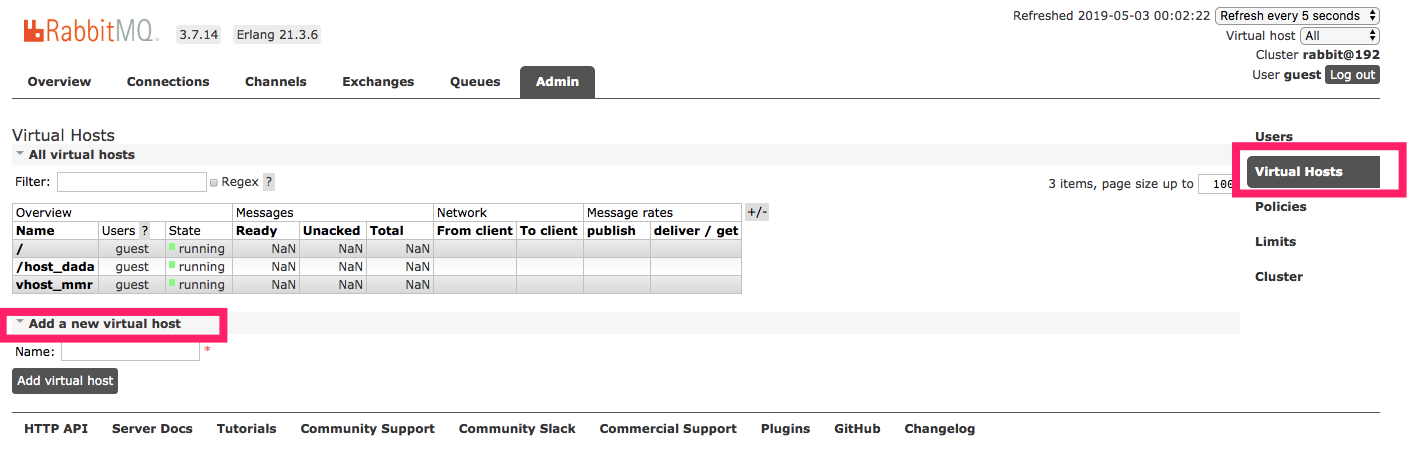


添加用户界面:

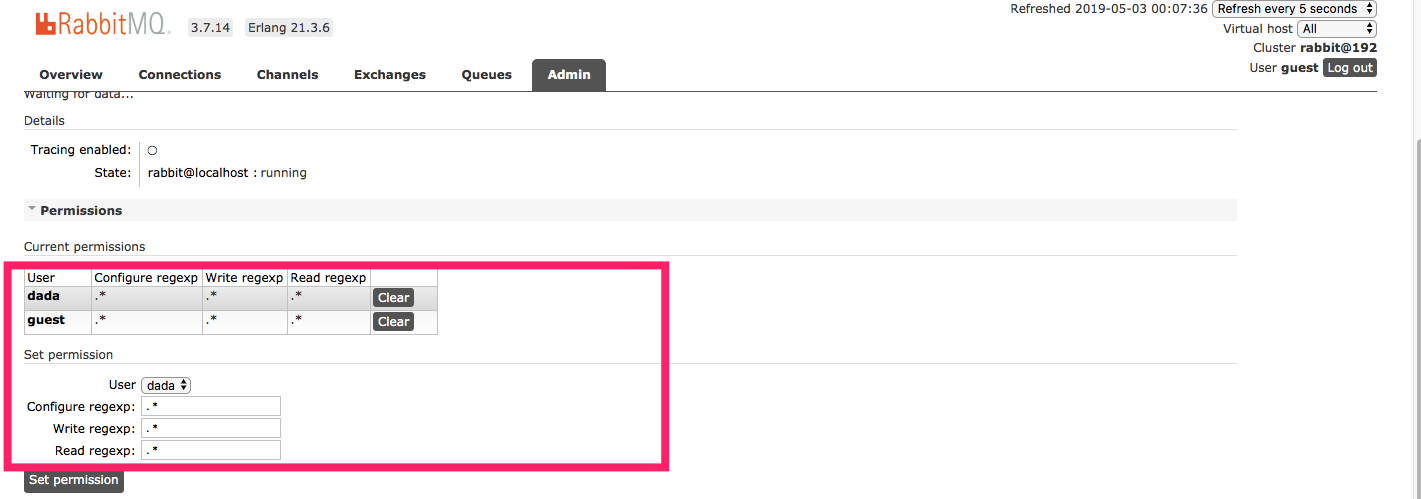
* **virtual hosts:管理**

virtual hosts就相当于mysql的DB

添加一个数据库：



设置权限



* 简单队列

IMG_256

P：消息的生产者

红色：消息队列

C：消费者

三个对象： 生产者 队列(RabbitMq) 消费者

* maven依赖

|  |
| --- |
| ***<?*xml version="1.0" encoding="UTF-8"*?> <*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.dada***</*groupId*>  <*artifactId*>***RabbitMQ***</*artifactId*>  <*version*>***1.0-SNAPSHOT***</*version*>   <*dependencies*>   <*dependency*>  <*groupId*>***com.rabbitmq***</*groupId*>  <*artifactId*>***amqp-client***</*artifactId*>  <*version*>***5.5.2***</*version*>  </*dependency*>   <*dependency*>  <*groupId*>***org.slf4j***</*groupId*>  <*artifactId*>***slf4j-api***</*artifactId*>  <*version*>***1.7.25***</*version*>  </*dependency*>   <*dependency*>  <*groupId*>***log4j***</*groupId*>  <*artifactId*>***log4j***</*artifactId*>  <*version*>***1.2.17***</*version*>  </*dependency*>   <*dependency*>  <*groupId*>***junit***</*groupId*>  <*artifactId*>***junit***</*artifactId*>  <*version*>***4.12***</*version*>  </*dependency*>  </*dependencies*>  </*project*>*** |

* 获取MQ的连接

|  |
| --- |
| */\*\*  \* 连接工厂  \*/* **public class** ConnectionUtil ***{*** */\*\*  \* 获取Mq的连接  \** ***@return*** *\** ***@throws*** *IOException  \** ***@throws*** *TimeoutException  \*/* **public static** Connection getConnection***()* throws** IOException, TimeoutException ***{*** *// 定义一个连接工厂* ConnectionFactory connectionFactory = **new** ConnectionFactory***()***;   *// 设置服务地址* connectionFactory.setHost***(*"127.0.0.1"*)***;   *// 设置端口号* connectionFactory.setPort***(***5672***)***;   *// 设置数据库 Vhost* connectionFactory.setVirtualHost***(*"/vhostnew"*)***;   *// 设置用户名密码* connectionFactory.setUsername***(*"dada"*)***;  connectionFactory.setPassword***(*"dada"*)***;   *// 获取连接* Connection connection = connectionFactory.newConnection***()***;   **return** connection;  ***} }*** |

生产者

|  |
| --- |
| */\*\*  \* 生产者  \*/* **public class** Send ***{* private static final** String ***Queue\_name*** = **"test\_simple\_queque"**;   **public static void** main***(***String***[]*** args***)* throws** Exception ***{*** *// 获取一个连接* Connection connection = ConnectionUtil.*getConnection****()***;   *// 从连接中获取一个通道* Channel channel = connection.createChannel***()***;   *// 声明队列* channel.queueDeclare***(Queue\_name***,**false**,**false**,**false**,**null*)***;   *// 发送的消息* String msg= **"hello Simple"**;   *// 发送* channel.basicPublish***(*""**,***Queue\_name***,**null**,msg.getBytes***())***;   System.***out***.println***(*"----send msg:"** + msg***)***;   *//关闭连接* channel.close***()***;  connection.close***()***;   ***} }*** |

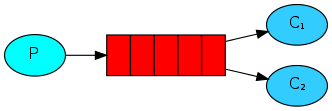
消费者

|  |
| --- |
| */\*\*  \* 消费者消费消息  \*/* **public class** Recv ***{* private static final** String ***Queue\_name*** = **"test\_simple\_queque"**;    **public static void** main***(***String***[]*** args***)* throws** IOException, TimeoutException ***{*** *// 获取连接* Connection connection = ConnectionUtil.*getConnection****()***;   *// 创建频道* Channel channel = connection.createChannel***()***;   *// 声明队列* channel.queueDeclare***(Queue\_name***,**false**,**false**,**false**,**null*)***;   *// 声明监听对象* DefaultConsumer consumer = **new** DefaultConsumer***(***channel***) {*** */\*\*  \* 一旦有消息进入队列 就会触发这个方法  \** ***@param consumerTag*** *\** ***@param envelope*** *\** ***@param properties*** *\** ***@param body*** *数据  \** ***@throws*** *IOException  \*/* @Override  **public void** handleDelivery***(***String consumerTag, Envelope envelope, AMQP.BasicProperties properties, **byte*[]*** body***)* throws** IOException ***{*** String msg = **new** String***(***body***)***;  System.***out***.println***(***msg***)***;  ***}  }***;   *// 监听队列* channel.basicConsume***(Queue\_name***,consumer***)***;    ***} }*** |

* 简单队列的不足：

耦合性高：生产者一一对应消费者（如果有多个消费者消费队列中的消息，这时候就不行了）队列名变更，这时候得同时变更

## **[Work queues](https://www.rabbitmq.com/tutorials/tutorial-two-python.html)工作队列**



一个生产者对应多个消费者

为什么会出现工作队列?

Simple队列是一一对应的，而且我们实际开发，生产者发送消息是毫不费力的，而且一般消费者是跟业务相结合的。消费者接收到消息之后就需要处理，可能需要花费时间，这时候队列就会积压很多消息。

生产者：

|  |
| --- |
| **package** com.dada.work;  **import** com.dada.util.ConnectionUtil; **import** com.rabbitmq.client.Channel; **import** com.rabbitmq.client.Connection;  **import** java.io.IOException; **import** java.util.concurrent.TimeoutException;  */\*\*  \* 工作队列  \*/* **public class** Send ***{* private static final** String ***Queue\_name*** = **"test\_work\_queque"**;    **public static void** main***(***String***[]*** args***)* throws** IOException, TimeoutException, InterruptedException ***{*** *// 获取连接* Connection connection = ConnectionUtil.*getConnection****()***;   *// 获取channel* Channel channel = connection.createChannel***()***;   *// 声明一个队列* channel.queueDeclare***(Queue\_name***,**false**,**false**,**false**,**null*)***;   *// 发送50条消息* **for *(*int** i=0;i<50 ; i++***){*** String msg= **" hello"**+i;  System.***out***.println***(*"send======="**+msg***)***;  channel.basicPublish***(*""**,***Queue\_name***,**null**,msg.getBytes***())***;  Thread.*sleep****(***i\*20***)***;  ***}*** channel.close***()***;  connection.close***()***;  ***} }*** |

消费者1

|  |
| --- |
| **package** com.dada.work;  **import** com.dada.util.ConnectionUtil; **import** com.rabbitmq.client.\*;  **import** java.io.IOException; **import** java.util.concurrent.TimeoutException;  **public class** Recv1 ***{* private static final** String ***Queue\_name*** = **"test\_work\_queque"**;    **public static void** main***(***String***[]*** args***)* throws** IOException, TimeoutException ***{*** Connection connection = ConnectionUtil.*getConnection****()***;  Channel channel = connection.createChannel***()***;  channel.queueDeclare***(Queue\_name***,**false**,**false**,**false**,**null*)***;   *// 定义一个消费者* Consumer consumer = **new** DefaultConsumer***(***channel***) {*** *// 消息到达触发这个方法* @Override  **public void** handleDelivery***(***String consumerTag, Envelope envelope, AMQP.BasicProperties properties, **byte*[]*** body***)* throws** IOException ***{*** String msg = **new** String***(***body,**"utf-8"*)***;  System.***out***.println***(*"[1] Recv msg :::: z"**+msg***)***;  **try *{*** Thread.*sleep****(***20000***)***;  ***}* catch *(***InterruptedException e***) {*** e.printStackTrace***()***;  ***}*finally *{*** System.***out***.println***(*"`1 [down]"*)***;  ***}  }  }***;   **boolean** autoAck= **true**;  channel.basicConsume***(Queue\_name***,autoAck,consumer***)***;  ***} }*** |

消费者2

|  |
| --- |
| **package** com.dada.work;  **import** com.dada.util.ConnectionUtil; **import** com.rabbitmq.client.\*;  **import** java.io.IOException; **import** java.util.concurrent.TimeoutException;  **public class** Recv2 ***{* private static final** String ***Queue\_name*** = **"test\_work\_queque"**;    **public static void** main***(***String***[]*** args***)* throws** IOException, TimeoutException ***{*** Connection connection = ConnectionUtil.*getConnection****()***;  Channel channel = connection.createChannel***()***;  channel.queueDeclare***(Queue\_name***,**false**,**false**,**false**,**null*)***;   *// 定义一个消费者* Consumer consumer = **new** DefaultConsumer***(***channel***) {*** *// 消息到达触发这个方法* @Override  **public void** handleDelivery***(***String consumerTag, Envelope envelope, AMQP.BasicProperties properties, **byte*[]*** body***)* throws** IOException ***{*** String msg = **new** String***(***body,**"utf-8"*)***;  System.***out***.println***(*"[2] Recv msg :::: z"**+msg***)***;  **try *{*** Thread.*sleep****(***10000***)***;  ***}* catch *(***InterruptedException e***) {*** e.printStackTrace***()***;  ***}*finally *{*** System.***out***.println***(*"`2 [down]"*)***;  ***}  }  }***;   **boolean** autoAck= **true**;  channel.basicConsume***(Queue\_name***,autoAck,consumer***)***;  ***} }*** |

默认采用轮询

公平分发：队列一次只发给消费者一个消息，当前消费者处理完毕之后会发rabbitMq一个响应之后rabbitmq才会给分发下一个

消费者

|  |
| --- |
| **package** com.dada.workfair;  **import** com.dada.util.ConnectionUtil; **import** com.rabbitmq.client.\*;  **import** java.io.IOException; **import** java.util.concurrent.TimeoutException;  **public class** Recv1 ***{* private static final** String ***Queue\_name*** = **"test\_work\_quequefair"**;    **public static void** main***(***String***[]*** args***)* throws** IOException, TimeoutException ***{*** Connection connection = ConnectionUtil.*getConnection****()***;  Channel channel = connection.createChannel***()***;  channel.queueDeclare***(Queue\_name***,**false**,**false**,**false**,**null*)***;   *// 设置一次只接收一个队列的消息* channel.basicQos***(***1***)***;   *// 定义一个消费者* Consumer consumer = **new** DefaultConsumer***(***channel***) {*** *// 消息到达触发这个方法* @Override  **public void** handleDelivery***(***String consumerTag, Envelope envelope, AMQP.BasicProperties properties, **byte*[]*** body***)* throws** IOException ***{*** String msg = **new** String***(***body,**"utf-8"*)***;  System.***out***.println***(*"[1] Recv msg :::: z"**+msg***)***;  **try *{*** Thread.*sleep****(***20000***)***;  ***}* catch *(***InterruptedException e***) {*** e.printStackTrace***()***;  ***}*finally *{*** *// 手动应答rabiitMQ 队列*  channel.basicAck**(**envelope.getDeliveryTag**()**,**false)**;  System.***out***.println***(*"`1 [down]"*)***;  ***}  }  }***;   *// 自动应答fasle* **boolean** autoAck= **false**;  channel.basicConsume***(Queue\_name***,autoAck,consumer***)***;  ***} }*** |

生产者

|  |
| --- |
| **package** com.dada.workfair;  **import** com.dada.util.ConnectionUtil; **import** com.rabbitmq.client.Channel; **import** com.rabbitmq.client.Connection; **import** java.io.IOException; **import** java.util.concurrent.TimeoutException;  */\*\*  \* 工作队列  \*/* **public class** Send ***{* private static final** String ***Queue\_name*** = **"test\_work\_quequefair"**;   **public static void** main***(***String***[]*** args***)* throws** IOException, TimeoutException, InterruptedException ***{*** *// 获取连接* Connection connection = ConnectionUtil.*getConnection****()***;   *// 获取channel* Channel channel = connection.createChannel***()***;   *// 声明一个队列* channel.queueDeclare***(Queue\_name***,**false**,**false**,**false**,**null*)***;   */\*\*  \* 每个消费者发送确认消息之前 消息队列不发送下一个消息到消费者，一次只处理一个消息  \* 限制发送给同一个消费者，不超过一个消息  \*/* **int** prefetchCount=1;  channel.basicQos***(***prefetchCount***)***;   *// 发送50条消息* **for *(*int** i=0;i<50 ; i++***){*** String msg= **" hello"**+i;  System.***out***.println***(*"send======="**+msg***)***;  channel.basicPublish***(*""**,***Queue\_name***,**null**,msg.getBytes***())***;  Thread.*sleep****(***i\*20***)***;  ***}*** channel.close***()***;  connection.close***()***;  ***} }*** |

消息应达与消息持久化

**boolean** autoAck= **true;**  
channel.basicConsume***(Queue\_name***,autoAck,consumer***)***;

**boolean autoAck= true;自动确认模式**

**一旦rabbitMq将消息分发给消费者就会从内存中删除**

**这种情况下：如果杀死正在执行的消费者，就会丢失正在处理的消息。**

**boolean autoAck= fasle;手动模式，如果有一个消费者挂掉了，就会交付给其他消费者，rabbbitMQ支持消息应答，告诉rabbitMQ这个消息已经处理完成可以删除了，然后rabbitMQ就会删除内存中的消息。**

**消息应答默认是打开的false**

**如果rabbitMq挂了，我们的消息仍然会丢失！！！！**

**持久化**

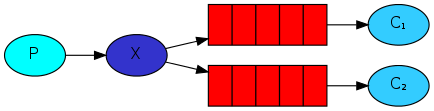
channel.queueDeclare***(Queue\_name***,**false**,**false**,**false**,**null*)***;

Queue.DeclareOk queueDeclare***(***String queue, **boolean** durable, **boolean** exclusive, **boolean** autoDelete,  
 Map***<***String, Object***>*** arguments***)* throws** IOException;

**Durable：持久化**

**已经定义的队列不可更改**

## **[Publish/Subscribe](https://www.rabbitmq.com/tutorials/tutorial-three-python.html)订阅模式**

**模型：**

**解读：**

**1，一个生产者，多个消费者**

**2，每一个消费者都有自己的一个队列**

**3，消费者没有直接把消息发送到队列，而是发送到了交换机(转发器上面 Exchange)**

**4，每个队列都要绑定到交换机上**

**5，生产者发送消息，经过交换机，到达队列，就能实现一个消息被多个消费者消费**

**注册-> 邮件->短信**

**交换机:->**

**生产者:**

|  |
| --- |
| **public class** Send ***{* private static final** String ***exchangeName*** = **"test\_exchange\_fanout"**;   **public static void** main***(***String***[]*** args***)* throws** IOException, TimeoutException ***{*** Connection connection = ConnectionUtil.*getConnection****()***;   Channel channel = connection.createChannel***()***;   *// 声明交换机* channel.exchangeDeclare***(exchangeName***,**"fanout"*)***;*// 分发   // 发送消息* String msg = **"hello ps"**;   channel.basicPublish***(exchangeName***,**""**,**null**,msg.getBytes***())***;   System.***out***.println***(*"sned "**+msg ***)***;   channel.close***()***;   connection.close***()***;   ***} }*** |

**消息哪里去了？丢失了!!!!!!因为交换机没有存储的能力,在rabbitmq中只有队列有存储能力，因为这个时候还没有队列绑定到交换机所以数据会丢失。**

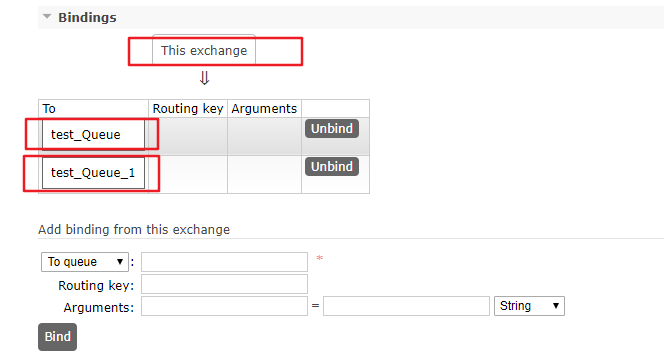
**消费者1**

|  |
| --- |
| **package** com.dada.pb;  **import** com.dada.util.ConnectionUtil; **import** com.rabbitmq.client.\*;  **import** java.io.IOException; **import** java.util.concurrent.TimeoutException;  **public class** Recv1 ***{* private static final** String ***QUEUE\_NAME***=**"test\_Queue"**;   **private static final** String ***exchangeName*** = **"test\_exchange\_fanout"**;   **public static void** main***(***String***[]*** args***)* throws** IOException, TimeoutException ***{*** Connection connection = ConnectionUtil.*getConnection****()***;   Channel channel = connection.createChannel***()***;   *// 声明队列* channel.queueDeclare***(QUEUE\_NAME***,**false**,**false**,**false**,**null*)***;   *// 绑定到交换机或者转发器* channel.queueBind***(QUEUE\_NAME***,***exchangeName***,**""*)***;   channel.basicQos***(***1***)***;   *// 定义一个消费者* Consumer consumer = **new** DefaultConsumer***(***channel***) {*** *// 消息到达触发这个方法* @Override  **public void** handleDelivery***(***String consumerTag, Envelope envelope, AMQP.BasicProperties properties, **byte*[]*** body***)* throws** IOException ***{*** String msg = **new** String***(***body,**"utf-8"*)***;  System.***out***.println***(*"[2] Recv msg :::: z"**+msg***)***;  **try *{*** Thread.*sleep****(***10000***)***;  ***}* catch *(***InterruptedException e***) {*** e.printStackTrace***()***;  ***}*finally *{*** *// 手动应答rabiitMQ 队列* channel.basicAck***(***envelope.getDeliveryTag***()***,**false*)***;  System.***out***.println***(*"`2 [down]"*)***;  ***}  }  }***;   **boolean** autoAck= **false**;  channel.basicConsume***(QUEUE\_NAME***,autoAck,consumer***)***;  ***} }*** |

**消费者2**

|  |
| --- |
| **package** com.dada.pb;  **import** com.dada.util.ConnectionUtil; **import** com.rabbitmq.client.\*;  **import** java.io.IOException; **import** java.util.concurrent.TimeoutException;  **public class** Recv2 ***{* private static final** String ***QUEUE\_NAME***=**"test\_Queue\_1"**;   **private static final** String ***exchangeName*** = **"test\_exchange\_fanout"**;   **public static void** main***(***String***[]*** args***)* throws** IOException, TimeoutException ***{*** Connection connection = ConnectionUtil.*getConnection****()***;   Channel channel = connection.createChannel***()***;   *// 声明队列* channel.queueDeclare***(QUEUE\_NAME***,**false**,**false**,**false**,**null*)***;   *// 绑定到交换机或者转发器* channel.queueBind***(QUEUE\_NAME***,***exchangeName***,**""*)***;   channel.basicQos***(***1***)***;   *// 定义一个消费者* Consumer consumer = **new** DefaultConsumer***(***channel***) {*** *// 消息到达触发这个方法* @Override  **public void** handleDelivery***(***String consumerTag, Envelope envelope, AMQP.BasicProperties properties, **byte*[]*** body***)* throws** IOException ***{*** String msg = **new** String***(***body,**"utf-8"*)***;  System.***out***.println***(*"[1] Recv msg :::: z"**+msg***)***;  **try *{*** Thread.*sleep****(***10000***)***;  ***}* catch *(***InterruptedException e***) {*** e.printStackTrace***()***;  ***}*finally *{*** *// 手动应答rabiitMQ 队列* channel.basicAck***(***envelope.getDeliveryTag***()***,**false*)***;  System.***out***.println***(*"`1 [down]"*)***;  ***}  }  }***;   **boolean** autoAck= **false**;  channel.basicConsume***(QUEUE\_NAME***,autoAck,consumer***)***;  ***} }*** |

**管理界面**

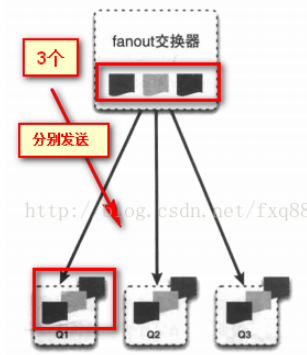
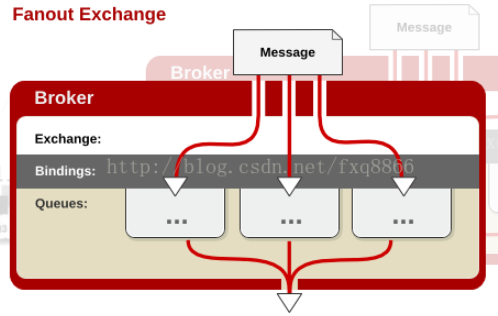
\

* Exchange（交换机 转发器）

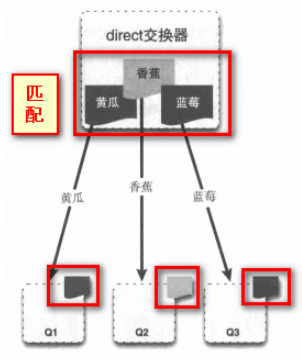
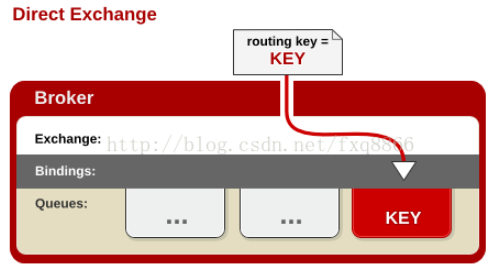
一方面接收生产者的消息，另一方面是向队列推送消息

匿名转发””

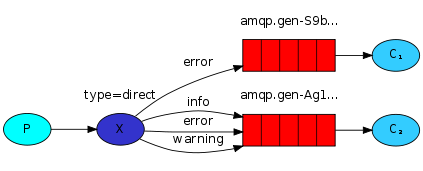
Fanout(不处理路由键)只要与它绑定的队列都能收到消息



* Direct(处理路由键)



* 路由模式：



生产者

|  |
| --- |
| **public class** Send ***{* private static final** String ***EXCHANGE\_NAME***=**"TESTECXCHANGE\_DIRECT"**;   **public static void** main***(***String***[]*** args***)* throws** IOException, TimeoutException ***{*** *// 获取连接* Connection connection = ConnectionUtil.*getConnection****()***;   *// 创建频道* Channel channel = connection.createChannel***()***;   *// 声明Exchangge* channel.exchangeDeclare***(EXCHANGE\_NAME***,**"direct"*)***;   *// 消息* String msg=**"hello direct"**;   *// 路由key // String routerkey="error";* String routerkey=**"info"**;   *// 发送消息* channel.basicPublish***(EXCHANGE\_NAME***,routerkey,**null**,msg.getBytes***())***;   *// 关闭资源* channel.close***()***;  connection.close***()***;  ***} }*** |

消费者1

|  |
| --- |
| **public class** Recv1 ***{* private static final** String ***EXCHANGE\_NAME***=**"TESTECXCHANGE\_DIRECT"**;   **private static final** String ***Queue\_name*** = **"test\_queque\_direct1"**;    **public static void** main***(***String***[]*** args***)* throws** IOException, TimeoutException ***{*** *// 获取连接* Connection connection = ConnectionUtil.*getConnection****()***;   *// 创建频道* Channel channel = connection.createChannel***()***;   *// 声明队列* channel.queueDeclare***(Queue\_name***,**false**,**false**,**false**,**null*)***;   *// 设置手动提交* channel.basicQos***(***1***)***;   *//绑定队列* channel.queueBind***(Queue\_name***,***EXCHANGE\_NAME***,**"error"*)***;  *// 如需多个绑定 // channel.queueBind(Queue\_name,EXCHANGE\_NAME,"error"); // channel.queueBind(Queue\_name,EXCHANGE\_NAME,"error");   // 定义一个消费者* Consumer consumer = **new** DefaultConsumer***(***channel***) {*** *// 消息到达触发这个方法* @Override  **public void** handleDelivery***(***String consumerTag, Envelope envelope, AMQP.BasicProperties properties, **byte*[]*** body***)* throws** IOException ***{*** String msg = **new** String***(***body,**"utf-8"*)***;  System.***out***.println***(*"[1] Recv msg :::: "**+msg***)***;  **try *{*** Thread.*sleep****(***10000***)***;  ***}* catch *(***InterruptedException e***) {*** e.printStackTrace***()***;  ***}*finally *{*** *// 手动应答rabiitMQ 队列* channel.basicAck***(***envelope.getDeliveryTag***()***,**false*)***;  System.***out***.println***(*"`【1】 down"*)***;  ***}  }  }***;   **boolean** autoAck= **false**;  channel.basicConsume***(Queue\_name***,autoAck,consumer***)***;  ***} }*** |

消费者2

|  |
| --- |
| **public class** Recv2 ***{* private static final** String ***EXCHANGE\_NAME***=**"TESTECXCHANGE\_DIRECT"**;   **private static final** String ***Queue\_name*** = **"test\_queque\_direct1"**;    **public static void** main***(***String***[]*** args***)* throws** IOException, TimeoutException ***{*** *// 获取连接* Connection connection = ConnectionUtil.*getConnection****()***;   *// 创建频道* Channel channel = connection.createChannel***()***;   *// 声明队列* channel.queueDeclare***(Queue\_name***,**false**,**false**,**false**,**null*)***;   *// 设置手动提交* channel.basicQos***(***1***)***;   *//绑定队列* channel.queueBind***(Queue\_name***,***EXCHANGE\_NAME***,**"error"*)***;  *//绑定队列* channel.queueBind***(Queue\_name***,***EXCHANGE\_NAME***,**"info"*)***;  *//绑定队列* channel.queueBind***(Queue\_name***,***EXCHANGE\_NAME***,**"warning"*)***;  *// 如需多个绑定 // channel.queueBind(Queue\_name,EXCHANGE\_NAME,"error"); // channel.queueBind(Queue\_name,EXCHANGE\_NAME,"error");   // 定义一个消费者* Consumer consumer = **new** DefaultConsumer***(***channel***) {*** *// 消息到达触发这个方法* @Override  **public void** handleDelivery***(***String consumerTag, Envelope envelope, AMQP.BasicProperties properties, **byte*[]*** body***)* throws** IOException ***{*** String msg = **new** String***(***body,**"utf-8"*)***;  System.***out***.println***(*"[2] Recv msg :::: "**+msg***)***;  **try *{*** Thread.*sleep****(***10000***)***;  ***}* catch *(***InterruptedException e***) {*** e.printStackTrace***()***;  ***}*finally *{*** *// 手动应答rabiitMQ 队列* channel.basicAck***(***envelope.getDeliveryTag***()***,**false*)***;  System.***out***.println***(*"`【2】 down"*)***;  ***}  }  }***;   **boolean** autoAck= **false**;  channel.basicConsume***(Queue\_name***,autoAck,consumer***)***;  ***} }*** |