

RabbitMQ入门案例 - Topic模式

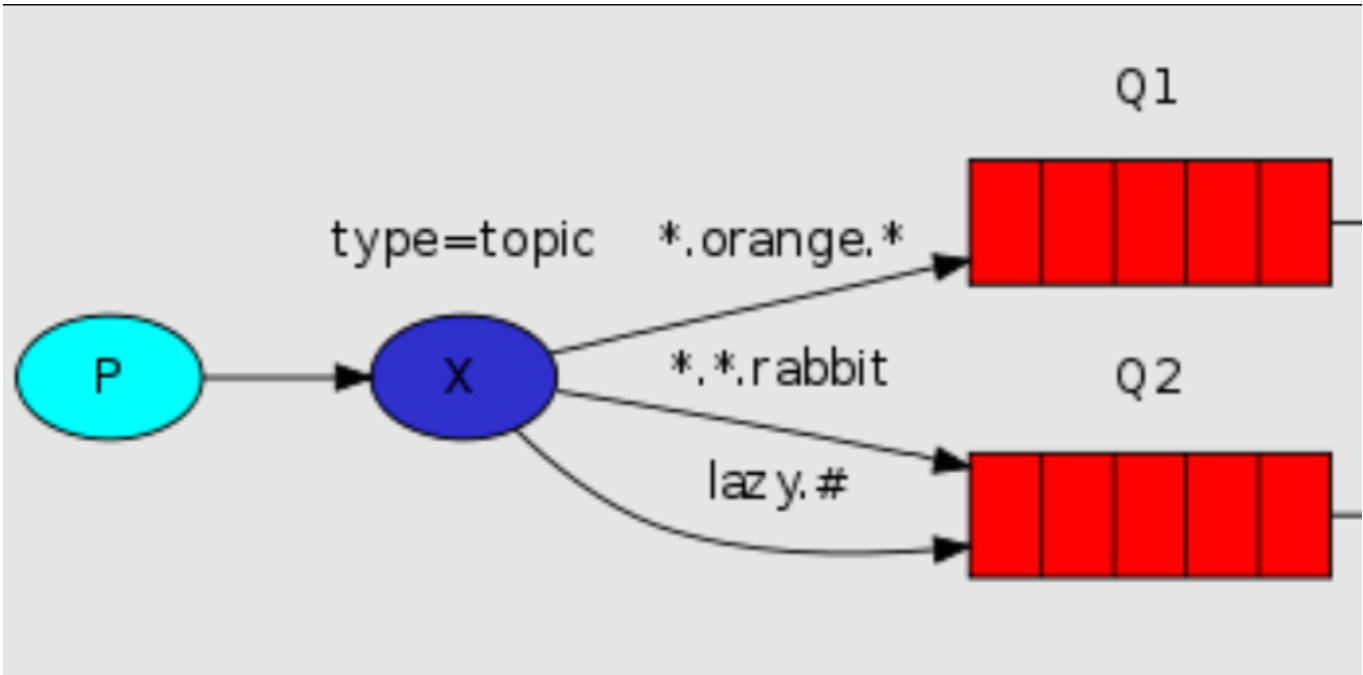
飞哥 VIP 分类: 学习笔记 创建时间: 2021/03/02 19:19 字体 皮肤

RabbitMQ支持消息的模式

参考官网: <https://www.rabbitmq.com/getstarted.html>

01、RabbitMQ的模式之Topic模式

图解



01-1、发布订阅模式具体实现

- web操作查看视频
- 类型: topic
- 特点: Topic模式是direct模式上的一种叠加, 增加了模糊路由RoutingKey的模式。

生产者

[RabbitMQ支持消息的模式](#)[01、RabbitMQ的模式之Topic模式](#) ▾[01-1、发布订阅模式具体实现](#)

```
1. package com.xuexiangban.rabbitmq.routing;
2.
3. import com.rabbitmq.client.Channel;
4. import com.rabbitmq.client.Connection;
5. import com.rabbitmq.client.ConnectionFactory;
6.
7. /**
8.  * @author: 学相伴-飞哥
9.  * @description: Producer 简单队列生产者
10.  * @Date : 2021/3/2
11.  */
12. public class Producer {
13.
14.     public static void main(String[] args) {
15.
16.         // 1: 创建连接工厂
17.         ConnectionFactory connectionFactory = new ConnectionFactory();
18.         // 2: 设置连接属性
19.         connectionFactory.setHost("47.104.141.27");
20.         connectionFactory.setPort(5672);
21.         connectionFactory.setVirtualHost("/");
22.         connectionFactory.setUsername("admin");
23.         connectionFactory.setPassword("admin");
24.
25.         Connection connection = null;
26.         Channel channel = null;
27.         try {
28.             // 3: 从连接工厂中获取连接
29.             connection = connectionFactory.newConnection("生产者");
30.             // 4: 从连接中获取通道channel
31.             channel = connection.createChannel();
32.             // 6: 准备发送消息的内容
33.             String message = "你好, 学相伴!!!";
34.
35.
36.             String exchangeName = "topic-exchange";
37.             String routingKey1 = "com.course.order";//都可以收到 queue-1 queue-2
38.             String routingKey2 = "com.order.user";//都可以收到 queue-1 queue-3
39.             // 7: 发送消息给中间件rabbitmq-server
40.             // @params1: 交换机exchange
41.             // @params2: 队列名称/routingkey
42.             // @params3: 属性配置
43.             // @params4: 发送消息的内容
44.             channel.basicPublish(exchangeName, routingKey1, null, message.getBytes());
45.
46.             System.out.println("消息发送成功!");
47.         } catch (Exception ex) {
48.             ex.printStackTrace();
49.             System.out.println("发送消息出现异常...");
50.         } finally {
51.             // 7: 释放连接关闭通道
52.             if (channel != null && channel.isOpen()) {
53.                 try {
54.                     channel.close();
55.                 } catch (Exception ex) {
56.                     ex.printStackTrace();
57.                 }
58.             }
59.             if (connection != null) {
60.                 try {
61.                     connection.close();
62.                 } catch (Exception ex) {
63.                     ex.printStackTrace();
64.                 }
65.             }
66.         }
67.     }
68. }
```



消费者

[RabbitMQ支持消息的模式](#)

[01、RabbitMQ的模式之Topic模式](#) ▾

[01-1、发布订阅模式具体实现](#)



[RabbitMQ支持消息的模式](#)[01、RabbitMQ的模式之Topic模式](#) [01-1、发布订阅模式具体实现](#)

```
1. package com.xuexiangban.rabbitmq.routing;
2.
3. import com.rabbitmq.client.*;
4.
5. import java.io.IOException;
6.
7. /**
8.  * @author: 学相伴-飞哥
9.  * @description: Consumer
10.  * @Date : 2021/3/2
11.  */
12. public class Consumer {
13.
14.     private static Runnable runnable = () -> {
15.         // 1: 创建连接工厂
16.         ConnectionFactory connectionFactory = new ConnectionFactory();
17.         // 2: 设置连接属性
18.         connectionFactory.setHost("47.104.141.27");
19.         connectionFactory.setPort(5672);
20.         connectionFactory.setVirtualHost("/");
21.         connectionFactory.setUsername("admin");
22.         connectionFactory.setPassword("admin");
23.
24.         //获取队列的名称
25.         final String queueName = Thread.currentThread().getName();
26.
27.         Connection connection = null;
28.         Channel channel = null;
29.         try {
30.             // 3: 从连接工厂中获取连接
31.             connection = connectionFactory.newConnection("生产者");
32.             // 4: 从连接中获取通道channel
33.             channel = connection.createChannel();
34.             // 5: 申明队列queue存储消息
35.             /*
36.              * 如果队列不存在，则会创建
37.              * Rabbitmq不允许创建两个相同的队列名称，否则会报错。
38.              *
39.              * @params1: queue 队列的名称
40.              * @params2: durable 队列是否持久化
41.              * @params3: exclusive 是否排他，即是否私有的，如果为true,会对当前队列加
42.              * 锁，并且连接自动关闭
43.              * @params4: autoDelete 是否自动删除，当最后一个消费者断开连接之后是否自
44.              * 动删除
45.              * @params5: arguments 可以设置队列附加参数，设置队列的有效期，消息的最大
46.              * 期等等。
47.              * */
48.             // 这里如果queue已经被创建过一次了，可以不需要定义
49.             //channel.queueDeclare("queue1", false, false, false, null);
50.
51.             // 6: 定义接受消息的回调
52.             Channel finalChannel = channel;
53.             finalChannel.basicConsume(queueName, true, new DeliverCallback() {
54.                 @Override
55.                 public void handle(String s, Delivery delivery) throws IOException {
56.                     System.out.println(queueName + " : 收到消息是 : " + new String(d
57.                         ecode.getBytes("UTF-8")));
58.                 }
59.             }, new CancelCallback() {
60.                 @Override
61.                 public void handle(String s) throws IOException {
62.
63.                 }
64.             });
65.
66.             System.out.println(queueName + " : 开始接受消息");
67.             System.in.read();
68.         } catch (Exception ex) {
69.             ex.printStackTrace();
70.         }
```

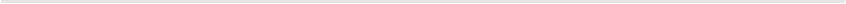


```
66.         System.out.println("发送消息出现异常...");
67.     } finally {
68.         // 7: 释放连接关闭通道
69.         if (channel != null && channel.isOpen()) {
70.             try {
71.                 channel.close();
72.             } catch (Exception ex) {
73.                 ex.printStackTrace();
74.             }
75.         }
76.         if (connection != null && connection.isOpen()) {
77.             try {
78.                 connection.close();
79.             } catch (Exception ex) {
80.                 ex.printStackTrace();
81.             }
82.         }
83.     }
84. };
85.
86. public static void main(String[] args) {
87.     // 启动三个线程去执行
88.     new Thread(runnable, "queue-1").start();
89.     new Thread(runnable, "queue-2").start();
90.     new Thread(runnable, "queue-3").start();
91. }
92. }
```

[RabbitMQ支持消息的模式](#)

[01、RabbitMQ的模式之Topic模式](#) 

[01-1、发布订阅模式具体实现](#)



[关于我们](#) | [加入我们](#) | [联系我们](#) | [帮助中心](#)

Copyright © 广东学相伴网络科技有限公司 [粤ICP备 - 2020109190号](#)

