RabbitMQ入门案例 - fanout模式

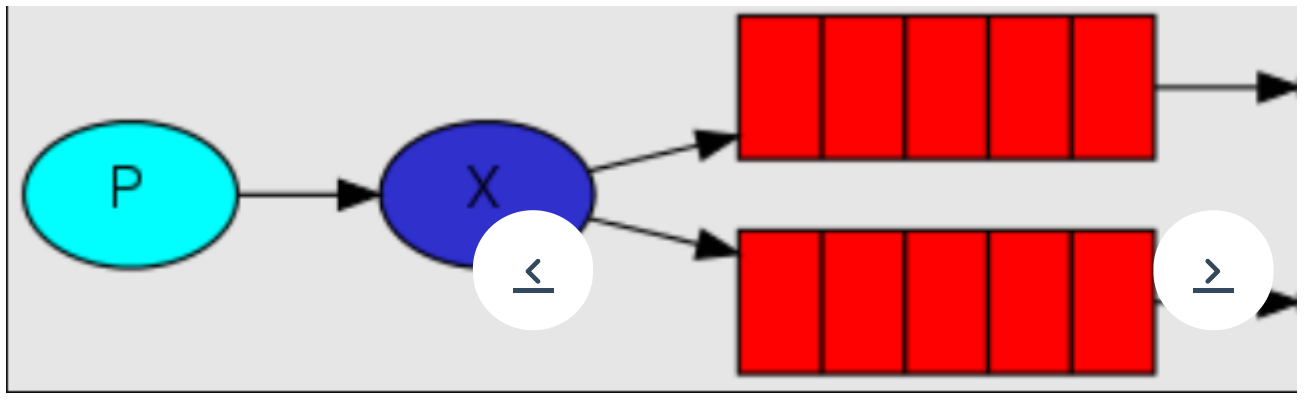
飞哥 VIP 分类：学习笔记 创建时间：2021/03/02 19:18 ☑字体 ☐皮肤

RabbitMQ支持消息的模式

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

01、RabbitMQ的模式之发布订阅模式

图解



01-1、发布订阅模式具体实现

- web操作查看视频
- 类型：fanout
- 特点：Fanout—发布与订阅模式，是一种广播机制，它是没有路由key的模式。

生产者

```java
package com.xuexiangban.rabbitmq.routing;

import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;

/**
 * @author: 学相伴-飞哥
 * @description: Producer 简单队列生产者
 * @Date : 2021/3/2
 */
public class Producer {

    public static void main(String[] args) {

        // 1: 创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        // 2: 设置连接属性
        connectionFactory.setHost("47.104.141.27");
        connectionFactory.setPort(5672);
        connectionFactory.setVirtualHost("/");
        connectionFactory.setUsername("admin");
        connectionFactory.setPassword("admin");

        Connection connection = null;
        Channel channel = null;
        try {
            // 3: 从连接工厂中获取连接
            connection = connect·  actory.newConnection("生产者");
            // 4: 从连接中获取通  ·l
            channel = connectio·  ateChannel();
            // 6： 准备发送消息的内容
            String message = "你好，学相伴！！！";

            String  exchangeName = "fanout-exchange";
            String routingKey = "";
            // 7: 发送消息给中间件rabbitmq-server
            // @params1: 交换机exchange
            // @params2: 队列名称/routingkey
            // @params3: 属性配置
            // @params4: 发送消息的内容
            channel.basicPublish(exchangeName, routingKey, null, message.getByte

            System.out.println("消息发送成功!");
        } catch (Exception ex) {
            ex.printStackTrace();
            System.out.println("发送消息出现异常...");
        } finally {
            // 7: 释放连接关闭通道
            if (channel != null && channel.isOpen()) {
                try {
                    channel.close();
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
            }
            if (connection != null) {
                try {
                    connection.close();
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
            }
        }
    }
}
```

消费者

```java
package com.xuexiangban.rabbitmq.routing;

import com.rabbitmq.client.*;

import java.io.IOException;

/**
 * @author: 学相伴-飞哥
 * @description: Consumer
 * @Date : 2021/3/2
 */
public class Consumer {

    private static Runnable runnable = () -> {
        // 1: 创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        // 2: 设置连接属性
        connectionFactory.setHost("47.104.141.27");
        connectionFactory.setPort(5672);
        connectionFactory.setVirtualHost("/");
        connectionFactory.setUsername("admin");
        connectionFactory.setPassword("admin");

        //获取队列的名称
        final String queueName = Thread.currentThread().getName();

        Connection connection = null;
        Channel channel = null;
        try {
            // 3: 从连接工厂中获取
            connection = connectionFactory.newConnection("生产者");
            // 4: 从连接中获取通道channel
            channel = connection.createChannel();
            // 5: 申明队列queue存储消息
            /*
             *  如果队列不存在，则会创建
             *  Rabbitmq不允许创建两个相同的队列名称，否则会报错。
             *
             *  @params1： queue 队列的名称
             *  @params2： durable 队列是否持久化
             *  @params3： exclusive 是否排他，即是否私有的，如果为true,会对当前队列加锁，其他的通道不能访问，并且连接自动关闭
             *  @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动删除
             *  @params5： arguments 可以设置队列附加参数，设置队列的有效期，消息的最大长度，队列的消息生命周期等等。
             * */
            // 这里如果queue已经被创建过一次了，可以不需要定义
            //channel.queueDeclare("queue1", false, false, false, null);

            // 6： 定义接受消息的回调
            Channel finalChannel = channel;
            finalChannel.basicConsume(queueName, true, new DeliverCallback() {
                @Override
                public void handle(String s, Delivery delivery) throws IOExcepti
                    System.out.println(queueName + "：收到消息是：" + new String(d
"UTF-8"));
                }
            }, new CancelCallback() {
                @Override
                public void handle(String s) throws IOException {

                }
            });

            System.out.println(queueName + "：开始接受消息");
            System.in.read();
        } catch (Exception ex) {
            ex.printStackTrace();
```

```
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. }
```