# RabbitMQ入门案例 - Work模式 - 公平分发（Fair Dispat

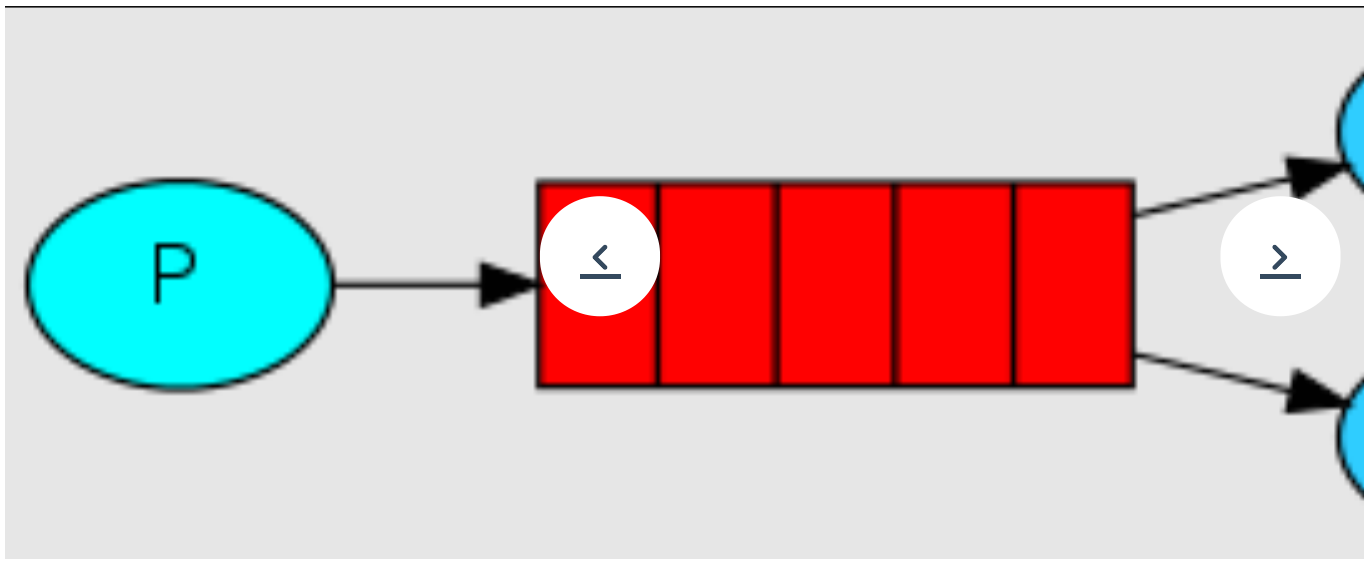
飞哥 VIP 分类：学习笔记 创建时间：2021/03/02 20:05 ☑字体 ☐皮肤

## 01、RabbitMQ支持消息的模式

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

## 02、Work模式 - 公平分发（Fair Dispatch）

### 图解




当有多个消费者时，我们的消息会被哪个消费者消费呢，我们又该如何均衡消费者消费信息的多少呢？
主要有两种模式：
1、轮询模式的分发：一个消费者一条，按均分配；
2、公平分发：根据消费者的消费能力进行公平分发，处理快的处理的多，处理慢的处理的少；按劳分配；

### Work模式 - 公平分发（Fair Dispatch）

- 类型：无
- 特点：由于消息接收者处理消息的能力不同，存在处理快慢的问题，我们就需要能者多劳，处理快的多处理，

### 生产者

```java
package com.xuexiangban.rabbitmq.work.fairr;

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
            //================================end topic模式=====================
            for (int i = 1; i <= 20; i++) {
                //消息的内容
                String msg = "学相伴:" + i;
                // 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("", "queue1", null, msg.getBytes());
            }


            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 - Work1

```java
package com.xuexiangban.rabbitmq.work.fairr;

import com.rabbitmq.client.*;

import java.io.IOException;

/**
 * @author: 学相伴-飞哥
 * @description: Consumer
 * @Date : 2021/3/2
 */
public class Work1 {

    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 = connectionFactory.newConnection("消费者-Work1");
            // 4: 从连接中获取通道 ...nel
            channel = connecti...teChannel();
            // 5: 申明队列queue存...
            /*
             *  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...
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...
             *  @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...
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);

            // 同一时刻，服务器只会推送一条消息给消费者

            // 6： 定义接受消息的回调
            Channel finalChannel = channel;
            finalChannel.basicQos(1);
            finalChannel.basicConsume("queue1", false, new DeliverCallback() {
                @Override
                public void handle(String s, Delivery delivery) throws IOExcepti
                    try{
                        System.out.println("Work1-收到消息是：" + new String(deli
"UTF-8"));
                        Thread.sleep(2000);
                        finalChannel.basicAck(delivery.getEnvelope().getDelivery
                    }catch(Exception ex){
                        ex.printStackTrace();
                    }
                }
            }, new CancelCallback() {
                @Override
                public void handle(String s) throws IOException {

                }
```

```
66.             });
67.
68.             System.out.println("Work1-开始接受消息");
69.             System.in.read();
70.         } catch (Exception ex) {
71.             ex.printStackTrace();
72.             System.out.println("发送消息出现异常...");
73.         } finally {
74.             // 7: 释放连接关闭通道
75.             if (channel != null && channel.isOpen()) {
76.                 try {
77.                     channel.close();
78.                 } catch (Exception ex) {
79.                     ex.printStackTrace();
80.                 }
81.             }
82.             if (connection != null && connection.isOpen()) {
83.                 try {
84.                     connection.close();
85.                 } catch (Exception ex) {
86.                     ex.printStackTrace();
87.                 }
88.             }
89.         }
90.     }
91. }
```

### 消费者 - Work2

```java
package com.xuexiangban.rabbitmq.work.fairr;

import com.rabbitmq.client.*;

import java.io.IOException;

/**
 * @author: 学相伴-飞哥
 * @description: Consumer
 * @Date : 2021/3/2
 */
public class Work2 {

    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 = connectionFactory.newConnection("消费者-Work2");
            // 4: 从连接中获取通道 ...nel
            channel = connecti...teChannel();
            // 5: 申明队列queue存...
            /*
             *  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...
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...
             *  @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...
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, true, false, null);

            // 同一时刻，服务器只会推送一条消息给消费者
            //channel.basicQos(1);

            // 6： 定义接受消息的回调
            Channel finalChannel = channel;
            finalChannel.basicQos(1);
            finalChannel.basicConsume("queue1", false, new DeliverCallback() {
                @Override
                public void handle(String s, Delivery delivery) throws IOExcepti...
                    try{
                        System.out.println("Work2-收到消息是：" + new String(deli...
"UTF-8"));
                        Thread.sleep(200);
                        finalChannel.basicAck(delivery.getEnvelope().getDelivery...
                    }catch(Exception ex){
                        ex.printStackTrace();
                    }
                }
            }, new CancelCallback() {
                @Override
                public void handle(String s) throws IOException {

```

```
                }
            });

            System.out.println("Work2-开始接受消息");
            System.in.read();
        } 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 && connection.isOpen()) {
                try {
                    connection.close();
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
            }
        }
    }
}
```



## 03、小结

从结果可以看到，消费者1在相同时间内，处理了更多的消息；以上代码我们实现了公平分发模式；

- 消费者一次接收一条消息，代码channel.BasicQos(0, 1, false);
- 公平分发需要消费者开启手动应答，关闭自动应答
- 关闭自动应答代码channel.BasicConsume(“queue_test”, false, consumer);
- 消费者开启手动应答代码：channel.BasicAck(ea.DeliveryTag, false);

## 04、总结

（1）当队列里消息较多时，我们通常会开启多个消费者处理消息；公平分发和轮询分发都是我们经常使用的模式
（2）轮询分发的主要思想是“按均分配”，不考虑消费者的处理能力，所有消费者均分；这种情况下，处理能力消息，而处理能力强的服务器，在处理完消息后，处于空闲状态；
(3) 公平分发的主要思想是”能者多劳”，按需分配，能力强的干的多。