

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

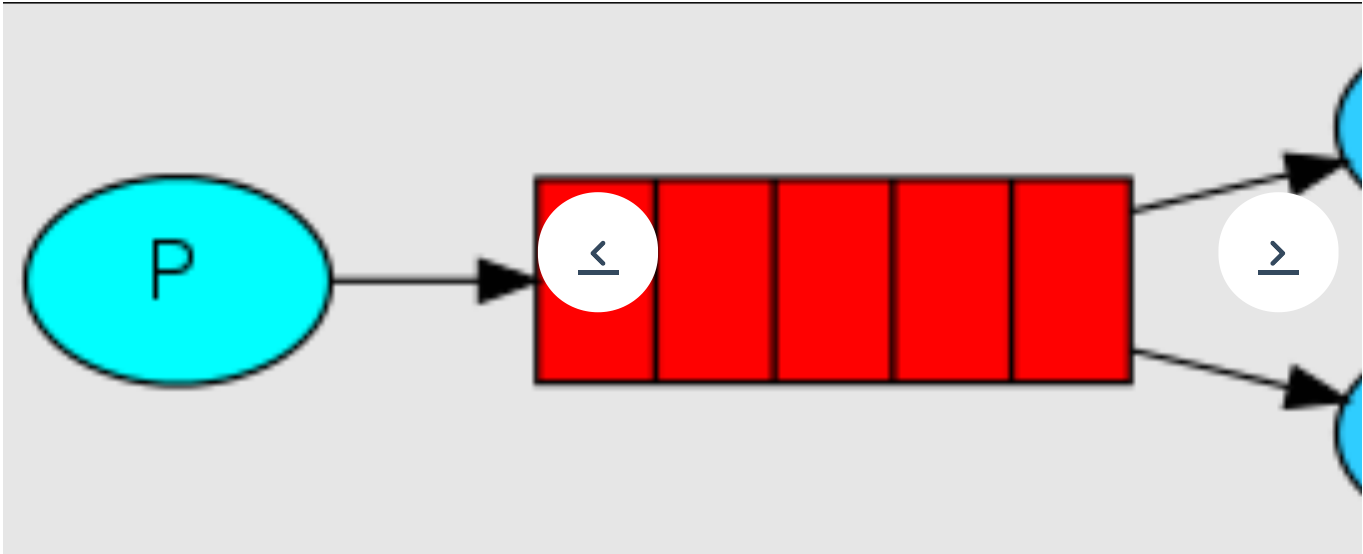
飞哥 VIP 分类: 学习笔记 创建时间: 2021/03/02 20:05 字体 皮肤

## 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

- 01、RabbitMQ支持消息的模式
- 02、Work模式 - 公平分发 (Fair Dispatch)..
  - 图解
  - Work模式 - 公平分发 (Fair Dispatch) 1/7
  - 生产者
  - 消费者 - Work1
  - 消费者 - Work2
- 03、小结
- 04、总结



```
1. package com.xuexiangban.rabbitmq.work.fairr;
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
31.             channel = connection.createChannel();
32.             // 6: 准备发送消息的内容
33.             //=====end topic模式=====
34.             for (int i = 1; i <= 20; i++) {
35.                 //消息的内容
36.                 String msg = "学相伴:" + i;
37.                 // 7: 发送消息给中间件rabbitmq-server
38.                 // @params1: 交换机exchange
39.                 // @params2: 队列名称/routingkey
40.                 // @params3: 属性配置
41.                 // @params4: 发送消息的内容
42.                 channel.basicPublish("", "queue1", null, msg.getBytes());
43.             }
44.
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. }
```

[01、RabbitMQ支持消息的模式](#)[02、Work模式 - 公平分发 \(Fair Dispatch..](#)[图解](#)[Work模式 - 公平分发 \(Fair Dispatch\) >](#)[生产者](#)[消费者 - Work1](#)[消费者 - Work2](#)[03、小结](#)[04、总结](#)

消费者 - Work1

[01、RabbitMQ支持消息的模式](#)

[02、Work模式 - 公平分发 \(Fair Dispatch..](#)

[图解](#)



[Work模式 - 公平分发 \(Fair Dispatch\)](#) ▾

[生产者](#)

[消费者 - Work1](#)

[消费者 - Work2](#)

[03、小结](#)

[04、总结](#)





```
1. package com.xuexiangban.rabbitmq.work.fairr;
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 Work1 {
13.
14.     public static void main(String[] args) {
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.         Connection connection = null;
25.         Channel channel = null;
26.         try {
27.             // 3: 从连接工厂中获取连接
28.             connection = connectionFactory.newConnection("消费者-Work1");
29.             // 4: 从连接中获取通道
30.             channel = connection.createChannel();
31.             // 5: 申明队列queue
32.             /*
33.              * 如果队列不存在，则会创建
34.              * Rabbitmq不允许创建两个相同的队列名称，否则会报错。
35.              *
36.              * @params1: queue 队列的名称
37.              * @params2: durable 队列是否持久化
38.              * @params3: exclusive 是否排他，即是否私有的，如果为true,会对当前队列加
39.              * 锁，并且连接自动关闭
40.              * @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
41.              * @params5: arguments 可以设置队列附加参数，设置队列的有效期，消息的最大有效期等等。
42.              */
43.             // 这里如果queue已经被创建过一次了，可以不需要定义
44.             channel.queueDeclare("queue1", false, false, false, null);
45.
46.             // 同一时刻，服务器只会推送一条消息给消费者
47.
48.             // 6: 定义接受消息的回调
49.             Channel finalChannel = channel;
50.             finalChannel.basicQos(1);
51.             finalChannel.basicConsume("queue1", false, new DeliverCallback() {
52.                 @Override
53.                 public void handle(String s, Delivery delivery) throws IOException {
54.                     try{
55.                         System.out.println("Work1-收到消息是：" + new String(delivery.getBody("UTF-8")));
56.                         Thread.sleep(2000);
57.                         finalChannel.basicAck(delivery.getEnvelope().getDeliveryTag(), false);
58.                     }catch(Exception ex){
59.                         ex.printStackTrace();
60.                     }
61.                 }, new CancelCallback() {
62.                     @Override
63.                     public void handle(String s) throws IOException {
64.
65.                     }
```

[01、RabbitMQ支持消息的模式](#)[02、Work模式 - 公平分发 \(Fair Dispatch..](#)[图解](#)[Work模式 - 公平分发 \(Fair Dispatch\) >](#)[生产者](#)[消费者 - Work1](#)[消费者 - Work2](#)[03、小结](#)[04、总结](#)

```
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. }
```

01、RabbitMQ支持消息的模式

02、Work模式 - 公平分发 (Fair Dispatch..

图解

> Work模式 - 公平分发 (Fair Dispatch) >

生产者

消费者 - Work1

消费者 - Work2

03、小结

04、总结

消费者 - Work2





```

1. package com.xuexiangban.rabbitmq.work.fairr;
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 Work2 {
13.
14.     public static void main(String[] args) {
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.         Connection connection = null;
25.         Channel channel = null;
26.         try {
27.             // 3: 从连接工厂中获取连接
28.             connection = connectionFactory.newConnection("消费者-Work2");
29.             // 4: 从连接中获取通道
30.             channel = connection.createChannel();
31.             // 5: 申明队列queue
32.             /*
33.              * 如果队列不存在，则会创建
34.              * Rabbitmq不允许创建两个相同的队列名称，否则会报错。
35.              *
36.              * @params1: queue 队列的名称
37.              * @params2: durable 队列是否持久化
38.              * @params3: exclusive 是否排他，即是否私有的，如果为true,会对当前队列加
39.              * 锁，并且连接自动关闭
40.              * @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
41.              * @params5: arguments 可以设置队列附加参数，设置队列的有效期，消息的最大有效期等等。
42.              */
43.             // 这里如果queue已经被创建过一次了，可以不需要定义
44.             //channel.queueDeclare("queue1", false, true, false, null);
45.
46.             // 同一时刻，服务器只会推送一条消息给消费者
47.             //channel.basicQos(1);
48.
49.             // 6: 定义接受消息的回调
50.             Channel finalChannel = channel;
51.             finalChannel.basicQos(1);
52.             finalChannel.basicConsume("queue1", false, new DeliverCallback() {
53.                 @Override
54.                 public void handle(String s, Delivery delivery) throws IOException {
55.                     try{
56.                         System.out.println("Work2-收到消息是：" + new String(delivery.getBody("UTF-8")));
57.                         Thread.sleep(200);
58.                         finalChannel.basicAck(delivery.getEnvelope().getDeliveryTag(), false);
59.                     }catch(Exception ex){
60.                         ex.printStackTrace();
61.                     }
62.                 }, new CancelCallback() {
63.                     @Override
64.                     public void handle(String s) throws IOException {
65.

```

[01、RabbitMQ支持消息的模式](#)[02、Work模式 - 公平分发 \(Fair Dispatch\)...](#)[图解](#)[Work模式 - 公平分发 \(Fair Dispatch\) >](#)[生产者](#)[消费者 - Work1](#)[消费者 - Work2](#)[03、小结](#)[04、总结](#)

```
66.         }
67.     });
68.
69.     System.out.println("Work2-开始接受消息");
70.     System.in.read();
71. } catch (Exception ex) {
72.     ex.printStackTrace();
73.     System.out.println("发送消息出现异常...");
74. } finally {
75.     // 7: 释放连接关闭通道
76.     if (channel != null && channel.isOpen()) {
77.         try {
78.             channel.close();
79.         } catch (Exception ex) {
80.             ex.printStackTrace();
81.         }
82.     }
83.     if (connection != null && connection.isOpen()) {
84.         try {
85.             connection.close();
86.         } catch (Exception ex) {
87.             ex.printStackTrace();
88.         }
89.     }
90. }
91. }
92. }
```

01、RabbitMQ支持消息的模式

02、Work模式 - 公平分发 (Fair Dispatch..

图解

> Work模式 - 公平分发 (Fair Dispatch) ▾

生产者

消费者 - Work1

消费者 - Work2

03、小结

04、总结

### 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。

