

01、RabbitMQ支持消息的模式

02、Work模式轮询模式 (Round-Robin...

图解

Work模式 - 轮询模式 (Round-Robin)

消费者 - Work1

消费者 - Work2

03、小结

# RabbitMQ入门案例 - Work模式 - 轮询模式 (Round-Robin)

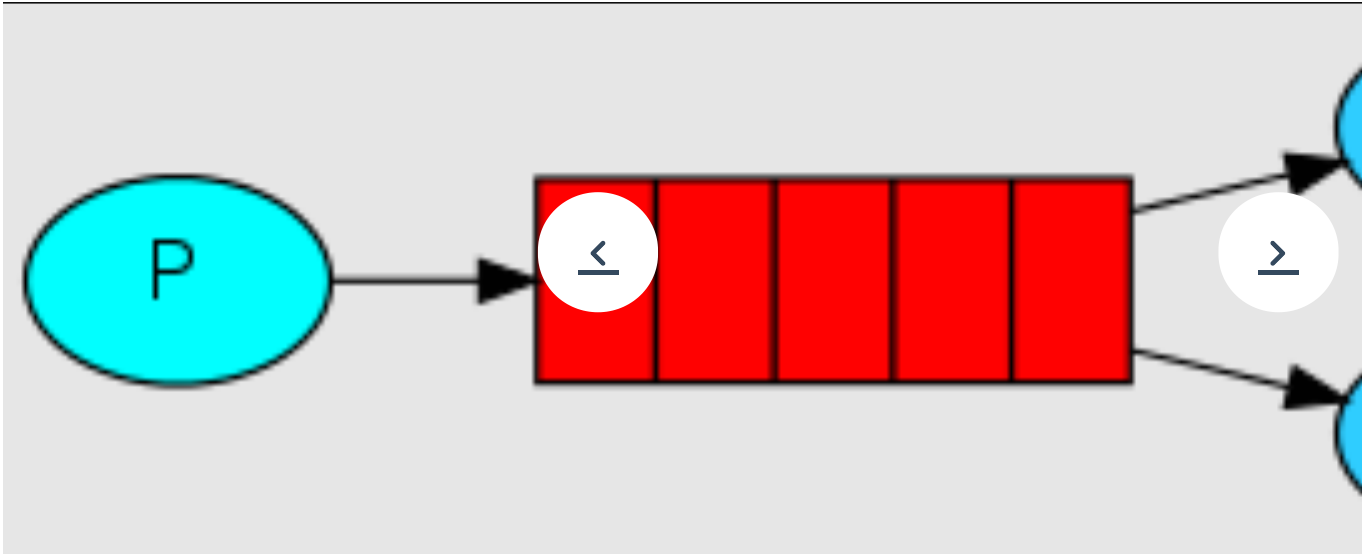
飞哥 VIP 分类: 学习笔记 创建时间: 2021/03/02 19:19 字体 皮肤

## 01、RabbitMQ支持消息的模式

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

## 02、Work模式轮询模式 (Round-Robin)

### 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 - 轮询模式 (Round-Robin)

- 类型：无
- 特点：该模式接收消息是当有多个消费者接入时，消息的分配模式是一个消费者分配一条，直至消息消费完成

### 生产者



```

1. package com.xuexiangban.rabbitmq.work.lunxun;
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模式轮询模式 \(Round-Robin...](#)[图解](#)[Work模式 - 轮询模式 \(Round-Robin\)](#)[消费者 - Work1](#)[消费者 - Work2](#)[03、小结](#)

消费者 - Work1

[01、RabbitMQ支持消息的模式](#)

[02、Work模式轮询模式（Round-Robin...](#)

[图解](#)



[Work模式 - 轮询模式（Round-Robin）](#)

[消费者 - Work1](#)

[消费者 - Work2](#)

[03、小结](#)





```

1. package com.xuexiangban.rabbitmq.work.lunxun;
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", true, 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.                     }catch(Exception ex){
58.                         ex.printStackTrace();
59.                     }
60.                 }, new CancelCallback() {
61.                     @Override
62.                     public void handle(String s) throws IOException {
63.
64.                     }
65.                 });

```

[01、RabbitMQ支持消息的模式](#)[02、Work模式轮询模式 \(Round-Robin...](#)[图解](#)[Work模式 - 轮询模式 \(Round-Robin\)](#)[消费者 - Work1](#)[消费者 - Work2](#)[03、小结](#)

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

[01、RabbitMQ支持消息的模式](#)

[02、Work模式轮询模式 \(Round-Robin...](#)

[图解](#)



[Work模式 - 轮询模式 \(Round-Robin\)](#)

[消费者 - Work1](#)

[消费者 - Work2](#)

[03、小结](#)

消费者 - Work2





```

1. package com.xuexiangban.rabbitmq.work.lunxun;
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
42.              * 有效期等等。
43.              */
44.             // 这里如果queue已经被创建过一次了，可以不需要定义
45.             //channel.queueDeclare("queue1", false, true, false, null);
46.
47.             // 同一时刻，服务器只会推送一条消息给消费者
48.             //channel.basicQos(1);
49.
50.             // 6: 定义接受消息的回调
51.             Channel finalChannel = channel;
52.             finalChannel.basicQos(1);
53.             finalChannel.basicConsume("queue1", true, new DeliverCallback() {
54.                 @Override
55.                 public void handle(String s, Delivery delivery) throws IOException {
56.                     try{
57.                         System.out.println("Work2-收到消息是：" + new String(delivery.getBody("UTF-8")));
58.                         Thread.sleep(200);
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模式轮询模式 \(Round-Robin...](#)[图解](#)[Work模式 - 轮询模式 \(Round-Robin\)](#)[消费者 - Work1](#)[消费者 - Work2](#)[03、小结](#)

```
66.         });
67.
68.         System.out.println("Work2-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模式轮询模式（Round-Robin...](#)

[图解](#)



[Work模式 - 轮询模式（Round-Robin）](#)

[消费者 - Work1](#)

[消费者 - Work2](#)

[03、小结](#)

### 03、小结



work1和work2的消息处理能力不同，但是最后处理的消息条数相同，是“按均分配”。



[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](#)