# RabbitMQ 工作模式介绍

* 发表于 *2018-01-19* | * 分类于 后端 | * 阅读数: *693* | * 评论数: 2

## 一、前言

之前，笔者写过[《 CentOS 7.2 安装 RabbitMQ》 这篇文章，今天整理一下 RabbitMQ 相关的笔记便于以后复习。

## 二、模式介绍

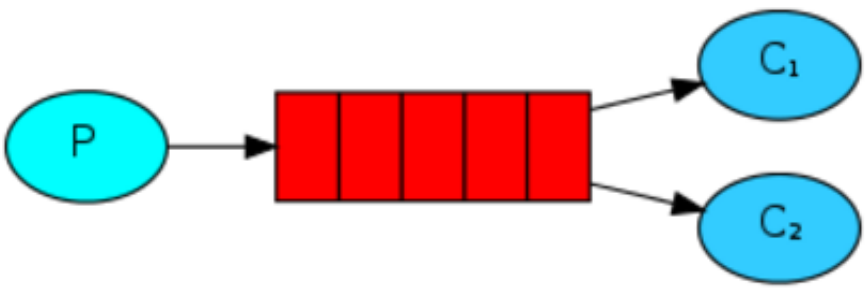
在 RabbitMQ 官网上提供了 6 中工作模式：简单模式、工作队列模式、发布/订阅模式、路由模式、主题模式 和 RPC 模式。

本篇只对前 5 种工作方式进行介绍。

### 2.1 简单模式与工作队列模式

之所以将这两种模式合并在一起介绍，是因为它们工作原理非常简单，由 3 个对象组成：生产者、队列、消费者。


生产者负责生产消息，将消息发送到队列中，消费者监听队列，队列有消息就进行消费。



当有多个消费者时，消费者平均消费队列中的消息。

代码演示：

生产者：

```
//1.获取连接
Connection connection = ConnectionUtil.getConnection();
//2.创建通道
Channel channel = connection.createChannel();
//3.申明队列
channel.queueDeclare(QUEUE_NAME, false, false, false, null);
//4.发送消息
channel.basicPublish("", QUEUE_NAME, null, "hello simple".getBytes());

System.out.println("发送成功");
//5.释放连接
channel.close();
connection.close();
```

消费者：

```
// 1.获取连接
Connection connection = ConnectionUtil.getConnection();
// 2.创建通道
Channel channel = connection.createChannel();
// 3.申明队列
channel.queueDeclare(QUEUE_NAME, false, false, false, null);
// 4.监听消息
channel.basicConsume(QUEUE_NAME, true, new DefaultConsumer(channel) {
    @Override
    public void handleDelivery(String consumerTag, Envelope envelope,
BasicProperties properties,
            byte[] body) throws IOException {
        String message = new String(body, "UTF-8");
        System.out.println("接收:" + message);
    }
});
```

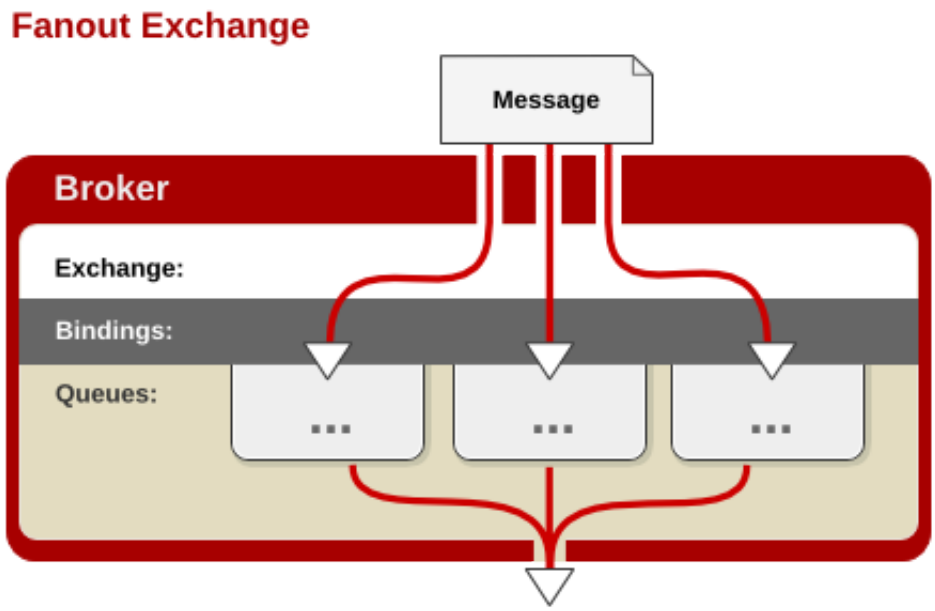
## 2.2 发布/订阅、路由与主题模式

这 3 种模式都使用到交换机。

生产者不直接与队列交互，而是将消息发送到交换机中，再由交换机将消息放入到已绑定该交换机的队列中给消费者消费。

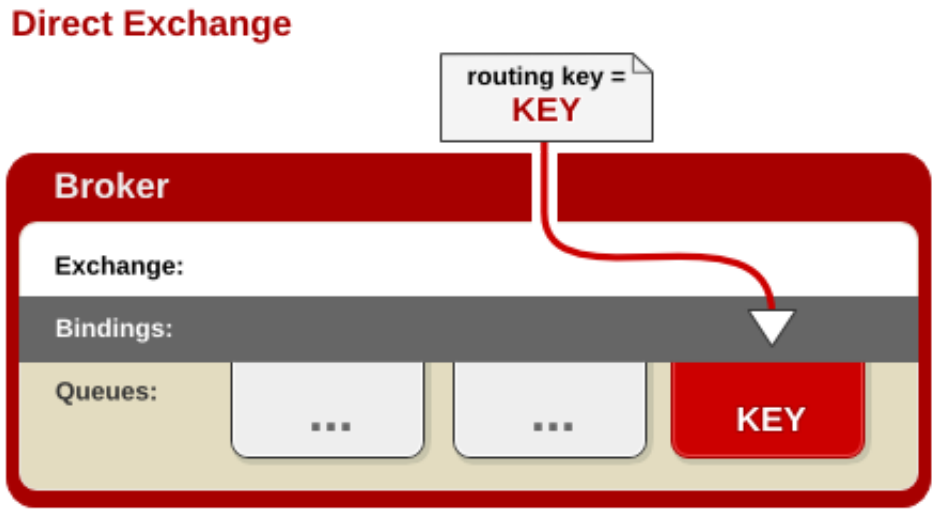
常用的交换机类型有 3 种：fanout、direct、topic。

工作原理图如下：



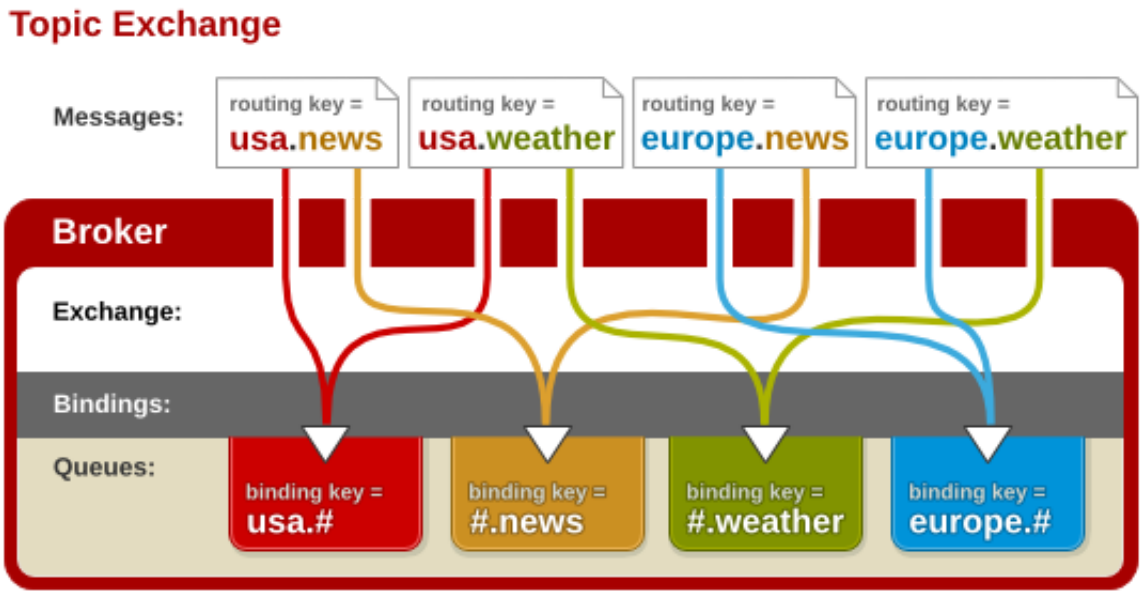

**fanout**：不处理路由键。只需要简单的将队列绑定到交换机上。一个发送到交换机的消息都会被转发到与该交换机绑定的所有队列上。很像子网广播，每台子网内的主机都获得了一份复制的消息。fanout 类型交换机转发消息是最快的。

其中，发布/订阅模式使用的是 **fanout** 类型的交换机。



**direct**：处理路由键。需要将一个队列绑定到交换机上，要求该消息与一个特定的路由键完全匹配。如果一个队列绑定到该交换机上要求路由键 “dog”，则只有被标记为 “dog” 的消息才被转发，不会转发 dog.puppy，也不会转发 dog.guard，只会转发dog。

其中，路由模式使用的是 **direct** 类型的交换机。




**topic**：将路由键和某模式进行匹配。此时队列需要绑定要一个模式上。符号 “#” 匹配一个或多个词，符号“”匹配不多不少一个词。因此*“audit.#”* 能够匹配到*“audit.irs.corporate”*，但是*“audit.”* 只会匹配到 “audit.irs”。

其中，主题模式使用的是 **topic** 类型的交换机。

代码演示：

生产者：

```
//1.获取连接
Connection connection = ConnectionUtil.getConnection();
//2.创建通道
Channel channel = connection.createChannel();
//3.申明交换机
channel.exchangeDeclare(EXCHANGE_NAME, "fanout");
//4.发送消息
for (int i = 0; i < 100; i++) {
    channel.basicPublish(EXCHANGE_NAME, "", null, ("hello ps" + i +
"").getBytes());
}

System.out.println("发送成功");
//5.释放连接
channel.close();
connection.close();
```

多个消费者：

```
// 1.获取连接
Connection connection = ConnectionUtil.getConnection();
// 2.创建通道
Channel channel = connection.createChannel();
// 3.申明交换机
channel.exchangeDeclare(EXCHANGE_NAME, "fanout");
// 4.队列绑定交换机
channel.queueDeclare(QUEUE_NAME, false, false, false, null);
channel.queueBind(QUEUE_NAME, EXCHANGE_NAME, "");
// 5.消费消息
channel.basicQos(1);
channel.basicConsume(QUEUE_NAME, false, new DefaultConsumer(channel) {
    @Override
    public void handleDelivery(String consumerTag, Envelope envelope,
AMQP.BasicProperties properties, byte[] body) throws IOException {
        String message = new String(body, "UTF-8");
        System.out.println("recv1:" + message);

        try {
            Thread.sleep(500);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        channel.basicAck(envelope.getDeliveryTag(), false);
    }
});
```

由于代码量问题，在此就不粘贴全部代码，感兴趣的读者可以下载查看源码。

# 三、源码下载

- [RabbitMQ 案例源码]

# 四、参考资料

- 官方文档