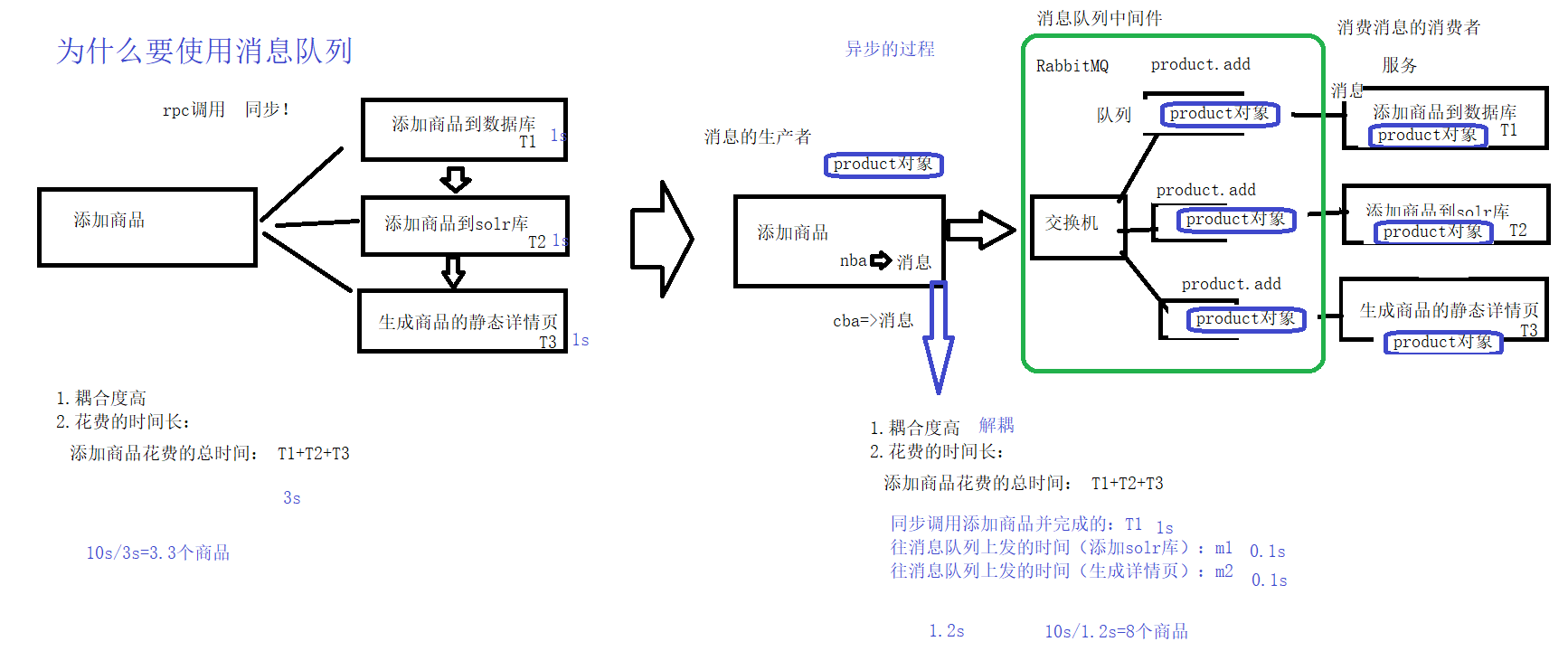
# 一、消息队列使用场景的引入



# 二、RabbitMQ的安装

参考文档

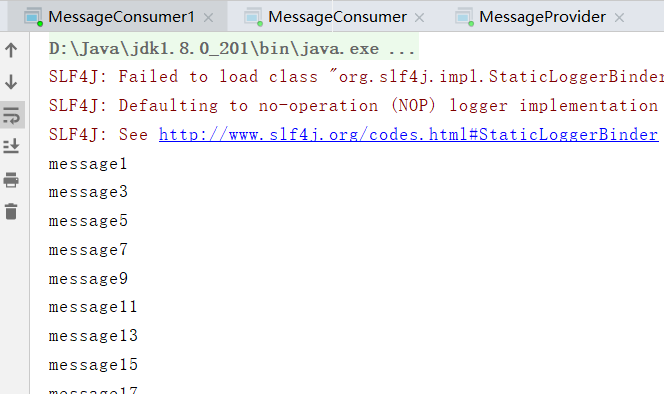
# 三、基于maven使用rabbitmq的简单队列模式

创建生产者

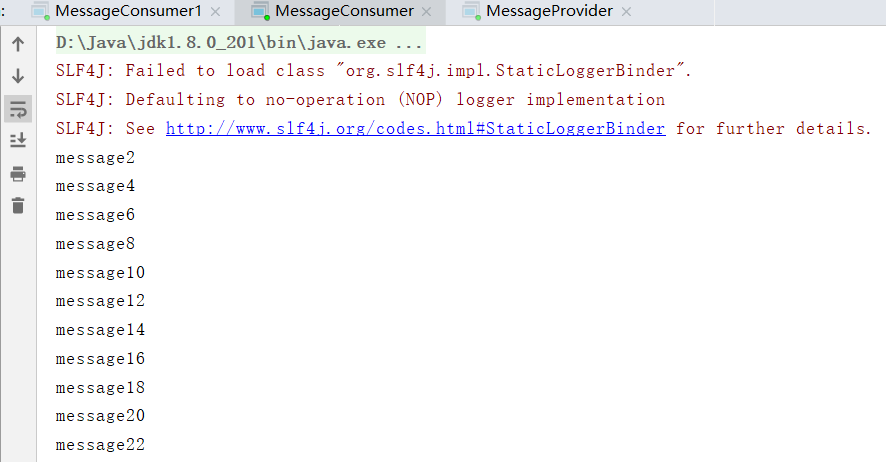
创建消费者

简单队列模式，多个消费者公平消费消息队列中的消息。

consumer1:



consumer2:



# 四、work模式——能者多劳

|  |
| --- |
| *//1.创建连接工厂* ConnectionFactory connectionFactory = **new** ConnectionFactory();  *//2.设置参数 // connectionFactory.setVirtualHost("java1904");* connectionFactory.setPassword(**"123456"**);  connectionFactory.setUsername(**"rabbit"**);  connectionFactory.setHost(**"192.168.2.141"**);  connectionFactory.setPort(5672);  *//3.获得连接对象* Connection connection = connectionFactory.newConnection();   Channel channel = connection.createChannel();   channel.basicQos(1);*//声明一次只处理一个消息* channel.queueDeclare(**"my\_simple\_queue4"**,**true**,**false**,**false**,**null**);   *// DeliverCallback deliverCallback = (consumerTag, delivery) -> { // String message = new String(delivery.getBody(), "UTF-8"); // System.out.println(" [x] Received '" + message + "'"); // };* Consumer consumer = **new** DefaultConsumer(channel){   **public void** handleDelivery(String consumerTag,  Envelope envelope,  AMQP.BasicProperties properties,  **byte**[] body)  **throws** IOException  {  **try** {  Thread.*sleep*(100);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  *// no work to do* String message = **new** String(body,**"utf-8"**);  System.***out***.println(message);  *//手动回复 ：处理完成* channel.basicAck(envelope.getDeliveryTag(),**false**);   }   };    channel.basicConsume(**"my\_simple\_queue4"**, **false**, consumer);*//不自动回复* |

# 五、发布订阅模式

消息提供者往交互机上发消息，交换机上绑定的所有的队列都能收到相同的消息。

消息提供者：

需要声明交互机，再往交互机上发消息，而非往队列上发消息。

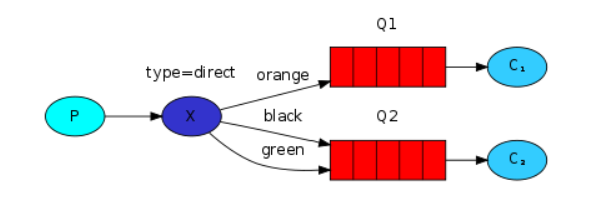
|  |
| --- |
| *//往交换机上发* channel.basicPublish(**"my\_ps\_exchange"**,**""**,**null**,message.getBytes()); |

消息消费者：

需要声明队列，声明交换机，将队列绑在交换机上，然后再监听队列。

|  |
| --- |
| *//声明队列* channel.queueDeclare(***QUEUE\_NAME***,**true**,**false**,**false**,**null**); *//声明交换机* channel.exchangeDeclare(***EXCHAGNE\_NAME***,**"fanout"**); *//将队列绑在交换机上* channel.queueBind(***QUEUE\_NAME***,***EXCHAGNE\_NAME***,**""**); |

# 六、路由模式



交换机通过routingKey将消息发送给指定的队列

消息的提供者：

*//往交换机上发，mq1是routingKey*channel.basicPublish(**"my\_routing\_exchange"**, **"mq1"**, **null**, message.getBytes());

消息的消费者：

*//将队列绑在交换机上,并指明了routingKey*channel.queueBind(***QUEUE\_NAME***,***EXCHAGNE\_NAME***,**"mq1"**);

一个消费者也可以指明多个routingKey

*//将队列绑在交换机上*channel.queueBind(***QUEUE\_NAME***,***EXCHAGNE\_NAME***,**"mq1"**);*\*channel.queueBind(***QUEUE\_NAME***,***EXCHAGNE\_NAME***,**"mq2"**);

# 七、通配符topic模式

通过通配符\* # 来实现路由key的多样性

*//往交换机上发*channel.basicPublish(**"my\_topic\_exchange1"**, **"product.add"**, **null**,**"product.add"**.getBytes());  
channel.basicPublish(**"my\_topic\_exchange1"**, **"product.del"**, **null**,**"product.del"**.getBytes());  
channel.basicPublish(**"my\_topic\_exchange1"**, **"product.del.other"**, **null**,**"product.del.other"**.getBytes());

product.\* ==> product.add product.del

product.# ==> product.add product.del product.del.other