目前我们已经完成了商品详情和搜索系统的开发。我们思考一下，是否存在问题？

商品的原始数据保存在数据库中，增删改查都在数据库中完成。

搜索服务数据来源是索引库，如果数据库商品发生变化，索引库数据不能及时更新。

商品详情做了页面静态化，静态页面数据也不会随着数据库商品发生变化。

如果我们在后台修改了商品的价格，搜索页面和商品详情页显示的依然是旧的价格，这样显然不对。该如何解决？

这里有两种解决方案：

方案1：每当后台对商品做增删改操作，同时要修改索引库数据及静态页面

方案2：搜索服务和商品页面服务对外提供操作接口，后台在商品增删改后，调用接口

以上两种方式都有同一个严重问题：就是代码耦合，后台服务中需要嵌入搜索和商品页面服务，违背了微服务的开闭原则。

所以，我们会通过另外一种方式来解决这个问题：消息队列

**1.2.2.AMQP和JMS**

MQ是消息通信的模型，并不是具体实现。现在实现MQ的有两种主流方式：AMQP、JMS。

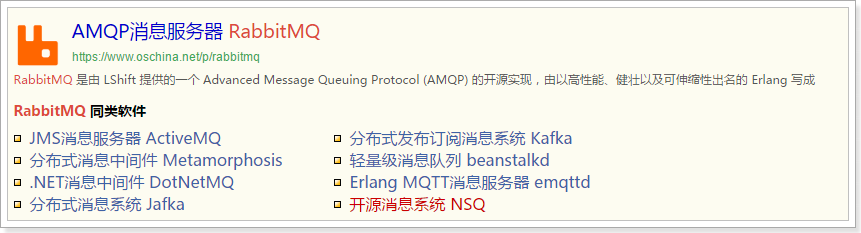
两者间的区别和联系：

JMS是定义了统一的接口，来对消息操作进行统一；AMQP是通过规定协议来统一数据交互的格式

JMS限定了必须使用Java语言；AMQP只是协议，不规定实现方式，因此是跨语言的。

JMS规定了两种消息模型；而AMQP的消息模型更加丰富

**1.2.3.常见MQ产品**



ActiveMQ：基于JMS

RabbitMQ：基于AMQP协议，erlang语言开发，稳定性好

RocketMQ：基于JMS，阿里巴巴产品，目前交由Apache基金会

Kafka：分布式消息系统，高吞吐量

**2.RabbitMQ**

RabbitMQ是基于AMQP的一款消息管理系统

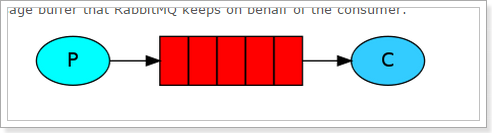
官网： http://www.rabbitmq.com/

官方教程：<http://www.rabbitmq.com/getstarted.html>

**2.1.基本消息模型**

RabbitMQ是一个消息代理：它接受和转发消息。 你可以把它想象成一个邮局：当你把邮件放在邮箱里时，你可以确定邮差先生最终会把邮件发送给你的收件人。 在这个比喻中，RabbitMQ是邮政信箱，邮局和邮递员。

RabbitMQ与邮局的主要区别是它不处理纸张，而是接受，存储和转发数据消息的二进制数据块。



P（producer/ publisher）：生产者，一个发送消息的用户应用程序。

C（consumer）：消费者，消费和接收有类似的意思，消费者是一个主要用来等待接收消息的用户应用程序

队列（红色区域）：rabbitmq内部类似于邮箱的一个概念。虽然消息流经rabbitmq和你的应用程序，但是它们只能存储在队列中。队列只受主机的内存和磁盘限制，实质上是一个大的消息缓冲区。许多生产者可以发送消息到一个队列，许多消费者可以尝试从一个队列接收数据。

总之：

生产者将消息发送到队列，消费者从队列中获取消息，队列是存储消息的缓冲区。

我们将用Java编写两个程序;发送单个消息的生产者，以及接收消息并将其打印出来的消费者。我们将详细介绍Java API中的一些细节，这是一个消息传递的“Hello World”。

我们将调用我们的消息发布者（发送者）Send和我们的消息消费者（接收者）Recv。发布者将连接到RabbitMQ，发送一条消息，然后退出。

 每个virtual host本质上都是一个RabbitMQ Server，拥有它自己的queue，exchagne，和bings rule等等。这保证了你可以在多个不同的application中使用RabbitMQ。

**dock 安装 rabbitMQ**

1. 拉取镜像

docker pull rabbitmq:[3.8.7-management](https://github.com/docker-library/rabbitmq/blob/888638927482f86af6e88bebb67423926cb1112f/3.8/ubuntu/management/Dockerfile)

2.启动

docker run -d --name rabbitmq -p 5672:5672 -p 15672:15672 -v `pwd`/rabbitmq:/var/lib/rabbitmq --hostname myRabbit -e RABBITMQ\_DEFAULT\_VHOST=my\_vhost -e RABBITMQ\_DEFAULT\_USER=admin -e RABBITMQ\_DEFAULT\_PASS=admin rabbitmq:3.8.7-management

说明：

-d 后台运行容器；

--name 指定容器名；

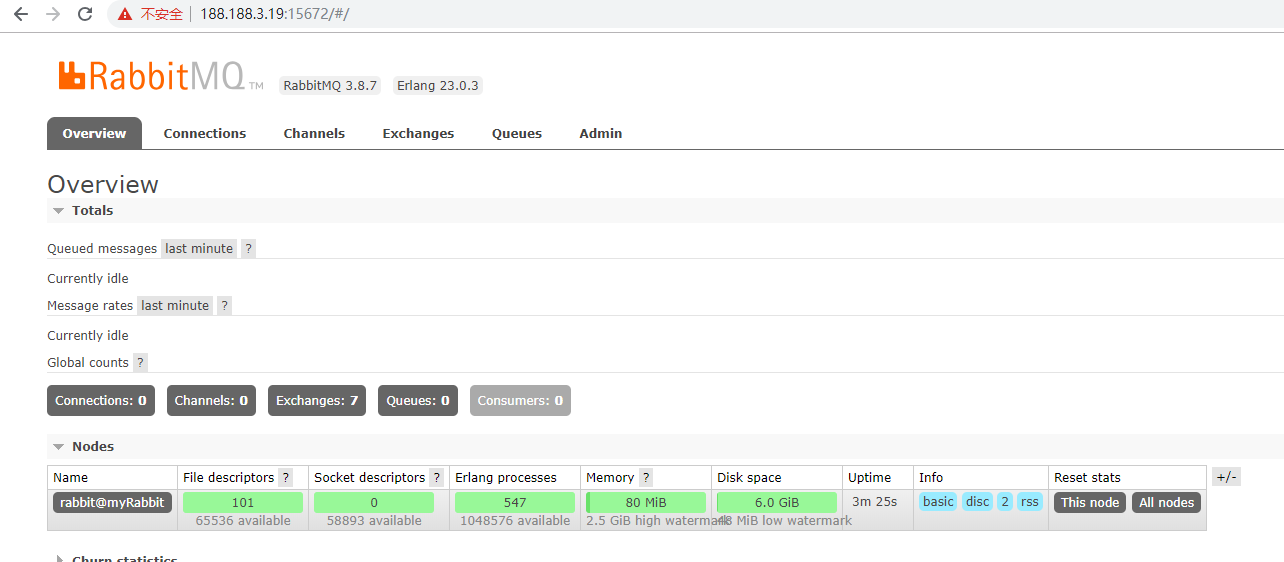
-p 指定服务运行的端口（5672：应用访问端口；15672：控制台Web端口号）；

-v 映射目录或文件，`pwd`当前工作的路径

--hostname  主机名（RabbitMQ的一个重要注意事项是它根据所谓的 “节点名称” 存储数据，默认为主机名）；

-e 指定环境变量；（RABBITMQ\_DEFAULT\_VHOST：默认虚拟机名；RABBITMQ\_DEFAULT\_USER：默认的用户名；RABBITMQ\_DEFAULT\_PASS：默认用户名的密码）

3. 访问，可以使用浏览器打开web管理端：http://Server-IP:15672



**解决消息丢失问题：**

消费者异常：

解决方案：ACK

如果消费者领取消息后，还没执行操作就挂掉了呢？或者抛出了异常？消息消费失败，但是RabbitMQ无从得知，这样消息就丢失了！

因此，RabbitMQ有一个ACK机制。当消费者获取消息后，会向RabbitMQ发送回执ACK，告知消息已经被接收。不过这种回执ACK分两种情况：

自动ACK：消息一旦被接收，消费者自动发送ACK

手动ACK：消息接收后，不会发送ACK，需要手动调用

public static void main(String[] argv) throws Exception {
// 获取到连接
Connection connection = ConnectionUtil.*getConnection*();
// 创建通道
final Channel channel = connection.createChannel();
// 声明队列
channel.queueDeclare(*QUEUE\_NAME*, false, false, false, null);
// 定义队列的消费者
DefaultConsumer consumer = new DefaultConsumer(channel) {
// 获取消息，并且处理，这个方法类似事件监听，如果有消息的时候，会被自动调用
@Override
public void handleDelivery(String consumerTag, Envelope envelope, BasicProperties properties,
byte[] body) throws IOException {
// body 即消息体
String msg = new String(body);
System.*out*.println(" [x] received : " + msg + "!");
// 手动进行ACK
channel.basicAck(envelope.getDeliveryTag(), false);
}
};
// 监听队列，第二个参数false，手动进行ACK
channel.basicConsume(*QUEUE\_NAME*, false, consumer);
}

RabbitMQ服务器宕机：

解决方案，持久化

交换机持久化：

// 声明exchange，指定类型为topic,第三个参数durable
channel.exchangeDeclare(*EXCHANGE\_NAME*, "topic", true);

消息通道持久化

// 声明队列.第二个参数durable
channel.queueDeclare(*QUEUE\_NAME*, true, false, false, null);

发送消息持久化

//发送消息， MessageProperties.PERSISTENT\_TEXT\_PLAIN，持久化消息
channel.basicPublish(*EXCHANGE\_NAME*, "item.update", MessageProperties.*PERSISTENT\_TEXT\_PLAIN*, message.getBytes());

生产者未成功发送消息到MQ中：

生产者确认：

//开启生产者确认
channel.confirmSelect();

发送消息前，将消息持久化到数据库，并记录消息状态（可靠消息服务）

幂等性（同一接口被重复执行，其结果一致）