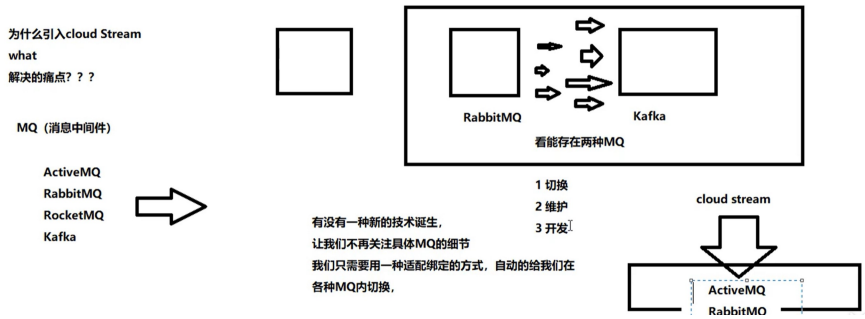
消息驱动

2020年8月6日 14:49

为什么引入cloud stream?

类似jdbc，管理众多sql语言

这是管理众多消息中间件，屏蔽底层消息中间件的差异，降低切换成本，统一消息的编程模型



什么是spring cloud stream?

官方定义 Spring Cloud Stream 是一个构建消息驱动微服务的框架。

应用程序通过 inputs 或者 outputs 来与 Spring Cloud Stream中binder对象交互。

通过我们配置来binding(绑定)，而 Spring Cloud Stream 的 binder对象负责与消息中间件交互。

所以，我们只需要搞清楚如何与 Spring Cloud Stream 交互就可以方便使用消息驱动的方式。

通过使用Spring Integration来连接消息代理中间件以实现消息事件驱动。

Spring Cloud Stream 为一些供应商的消息中间件产品提供了个性化的自动化配置实现，引用了发布-订阅、消费组、分区的三个核心概念。

目前仅支持RabbitMQ、Kafka。

绑定对象：↑↑↑

标准MQ思想：消息走通道，通道里面的订阅者获得

为什么用cloud stream

比方说我们用到了RabbitMQ和Kafka，由于这两个消息中间件的架构上的不同，

像RabbitMQ有exchange，kafka有Topic和Partitions分区，

这些中间件的差异性导致我们实际项目开发给我们造成了一定的困扰，我们如果用了两个消息队列的其中一种，后面的业务需求，我想往另外一种消息队列进行迁移，这时候无疑就是一个灾难性的，一大堆东西都要重新推倒重新做，因为它跟我们的系统耦合了，这时候springcloud Stream给我们提供了一种解耦合的方式。

cloud凭什么可以统一底层差异-------input消费者--------output生产者（往外输出，卖东西就是生产者）

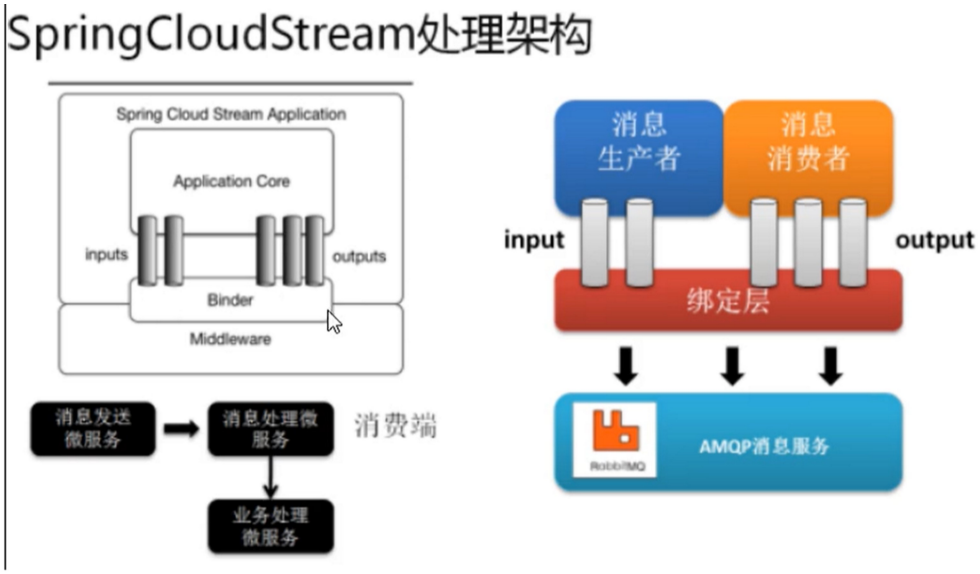
在没有绑定器这个概念的情况下，我们的SpringBoot应用要直接与消息中间件进行信息交互的时候，

由于各消息中间件构建的初衷不同，它们的实现细节上会有较大的差异性

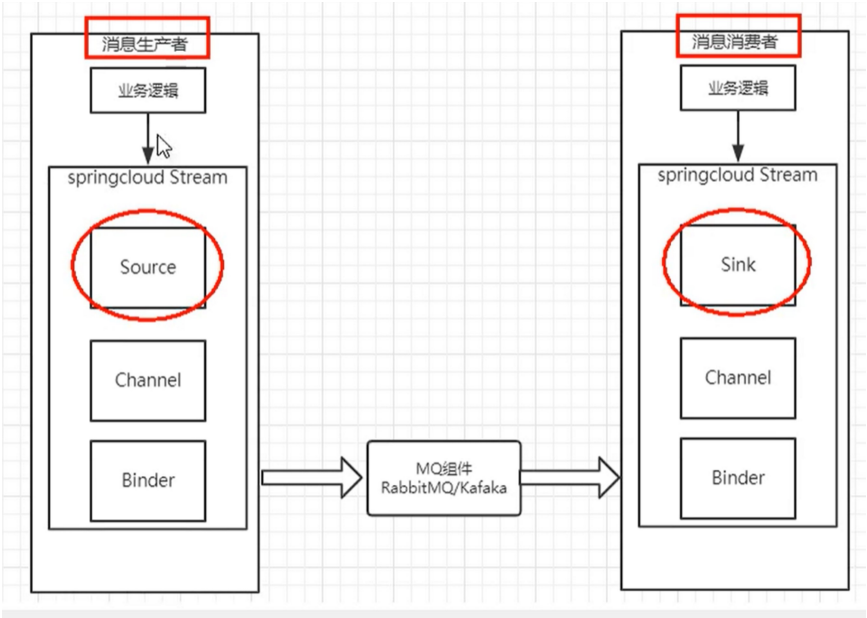
通过定义绑定器作为中间层，完美地实现了应用程序与消息中间件细节之间的隔离。

通过向应用程序暴露统一的Channel通道，使得应用程序不需要再考虑各种不同的消息中间件实现。

通过定义绑定器Binder作为中间层，实现了应用程序与消息中间件细节之间的隔离。



标准流程



常用注解

组成	说明
Middleware	中间件，目前只支持RabbitMQ和Kafka
Binder	Binder是应用与消息中间件之间的封装，目前实行了Kafka和RabbitMQ的Binder，通过Binder可以很方便的连接中间件，可以动态的改变消息类型(对应于Kafka的topic，RabbitMQ的exchange)，这些都可以通过配置文件来实现
@Input	注解标识输入通道，通过该输入通道接收到的消息进入应用程序
@Output	注解标识输出通道，发布的消息将通过该通道离开应用程序
@StreamListener	监听队列，用于消费者的队列的消息接收
@EnableBinding	指信道channel和exchange绑定在一起