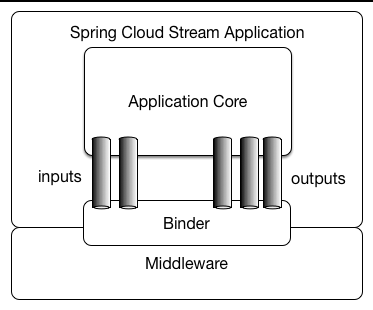
# 

# **Spring Cloud Stream**

https://blog.csdn.net/jack281706/article/details/73743148

Spring Cloud Stream 是一个用来为微服务应用构建消息驱动能力的框架。它可以基于Spring Boot 来创建独立的，可用于生产的Spring 应用程序。他通过使用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.其实说白了就是解耦



**二 核心概念**

## 2.1、应用模型

Spring Cloud Stream由一个中间件中立的核组成。应用通过Spring Cloud Stream插入的input和output通道与外界交流。通道通过指定中间件的Binder实现与外部代理连接。业务开发者不再关注具体消息中间件，只需关注Binder对应用程序提供的抽象概念来使用消息中间件实现业务即可。

## 2.2、绑定器

通过定义绑定器作为中间层，实现了应用程序与消息中间件细节之间的隔离。通过向应用程序暴露统一的Channel通过，是的应用程序不需要再考虑各种不同的消息中间件的实现。当需要升级消息中间件，或者是更换其他消息中间件产品时，我们需要做的就是更换对应的Binder绑定器而不需要修改任何应用逻辑 。

目前只提供了RabbitMQ和Kafka的Binder实现

2.3 分组:项目中有多个实例的时候，只让一个实例接收

实战部分

### **Sink介绍**

##### **Sink是stream中接受消息的接口，主要用来绑定管道接受消息。代码如下：**

public interface Sink {

String INPUT = "input";

@Input(Sink.INPUT)

SubscribableChannel input();

}

* 1
* 2
* 3
* 4
* 5

从字面的介绍也可以看出@Input注解表示绑定订阅输入管道，通过管道接受来自发布者的消息。

### **Source介绍**

##### **Source是stream中的发布消息的接口，主要用来绑定管道发布消息。代码如下：**

public interface Source {

String OUTPUT = "output";

@Output(Source.OUTPUT)

MessageChannel output();

}

@output注解表示绑定输出管道，通过管道发布消息。

### **Processer介绍**

Processor是stream中绑定输入输出的接口，主要用来讲发布者和订阅者绑定到一起。代码如下：

public interface Processor extends Source, Sink {

}

* 1
* 2

可以用@SendTo注解返回消费完了返回的管道

Processer继承了 Source和 Sink 具有输入和输出管道。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*坑\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Input output同名，不能创建相同实例的时候，