Spring Cloud Stream结合Kafka

1）Spring Cloud Stream优势

> 常用中间件RabbitMQ和Kafka存在架构上的不同，通过Spring Cloud Stream消息驱动可以对这两种方式进行解耦合，在以后切换中间件时更简单

2）Spring Cloud Stream概念

- Binder

> 作为应用和消息中间件的粘合剂，目前Spring Cloud Stream支持上述两种中间件

- Publish-Subscribe

> Spring Cloud Stream也是通过发布与订阅的方式

- Consumner Group

> Spring Cloud Stream沿用Kafka的消费者组，以确保消息不会被重复消费

- Bindings

> Spring Cloud Stream的核心配置，通过修改其配置来修改topic、type等

3）Spring Cloud Stream配置文件

- 依赖

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream-binder-kafka</artifactId>
</dependency>
```

- application.properties

```
# Kafka配置
spring.cloud.stream.kafka.binder.brokers=Master:9092,Worker1:9092,Worker2:9092
spring.cloud.stream.kafka.binder.zkNodes=Master:2181,Worker1:2181,Worker2:9092
spring.cloud.stream.kafka.binder.auto-create-topics=true
spring.cloud.stream.kafka.binder.requiredAcks=1

# 输入通道、输出通道配置
spring.cloud.stream.bindings.msg_output.destination=output_topic
spring.cloud.stream.bindings.msg_output.binder=kafka
spring.cloud.stream.bindings.msg_input.destination=input_topic
spring.cloud.stream.bindings.msg_input.binder=kafka
spring.cloud.stream.bindings.msg_input.group=consumer_group
```

4）Spring Cloud Stream API

- 除了提供Sink(输入通道)、Source(输出通道)、Processor(集成输入输出通道)，Spring Cloud Stream还允许通过@Input和@Output自定义信息通道

```
// 通道接口类
public interface MyMsgChannel {
    // 输入通道名称
    String input_channel = "msg_input";
    // 输出通道名称
    String output_channel = "msg_output";

    @Output(output_channel)
    MessageChannel sendMessage();

    @Input(input_channel)
    MessageChannel receiveMessage();
}
// 消息发送器
@EnableBinding(MsMsgChannel.class)
public class MyKafkaMessageSender {
    @Autowired
    private MyMsgChannel myMsgChannel;

    public void sendToChannel(String message){
        myMsgChannel.sendMessage().send(MessageBuilder.withPayload(message).build());
    }
}
```

```
// 通道消息接收器
@EnableBinding(MyMsgChannel.class)
public class MyStreamListener {

    @StreamListener(MyMsgChannel.input_channel)
    public void receive(Message<String> message){
        // 处理收到的数据
    }
}
// 消息中转
@EnableBinding(MyMsgChannel.class)
public class TransFromService{

    @ServiceActivar(inputChannel = MyMsgChannel.msg_input,
                    outputChannel = MyMsgChannel.msg_output)
    public Object transform(Object payload){
        // 处理并返回payload
    }
}
```