消息驱动 | SpringCloud Stream

Spring Cloud Stream 是一个构建消息驱动微服务的框架。

在SpringBoot之上，提供了Kafka，RabbitMQ等消息中间件的个性化配置，引入了发布订阅、消费组和分区的语义概念，有效的简化了上层研发人员对MQ使用的复杂度，让开发人员更多的精力投入到核心业务的处理。

使用Spring Cloud Stream来整合我们的消息中间件，可以降低系统和中间件的耦合性。

整合 RabbitMQ

stream通过定义发送、接收通道来进行消息的传递。

消息发送

引入依赖

XML

```xml
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream</artifactId>
</dependency>
<!--生产者-->
<dependency>
    <groupId>org.sshengcpringframework.cloud</groupId>
    <artifactId>spring-cloud-starter-stream-rabbit</artifactId>
</dependency>
```

配置

YAML

```yaml
spring:
  application:
    name: rabbitmq-service #指定服务名
  rabbitmq:
    addresses: 127.0.0.1
    username: itcast
    password: itcast
    virtual-host: myhost
  cloud:
    stream:
      bindings:
        output: #发送通道
          destination: itcast-default #指定消息发送目的地
        input: #消费通道
          destination: itcast-default #指定消息来源
          #contentType: text/plain #消息类型
      binders:  #配置绑定器
        defaultRabbit:
          type: rabbit
```

发送消息

定义发送通道：output

Java

```java
public interface Source {
```

```
2
3     @Output("output") // 发送通道接口
4     MessageChannel myoutput();
5
6   }
```

消息发送：

```Java
1   @Component
2   @EnableBinding(Source.class) //绑定通道
3   public class MessageSender {
4
5     @Autowired
6     private MessageChannel messageChannel ;
7
8     //发送消息
9     public void send(Object o) {
10      messageChannel.send(MessageBuilder.withPayload(o).build());
11    }
12
13  }
```

消息接收

引入依赖

```XML
1   <dependency>
2       <groupId>org.springframework.cloud</groupId>
3       <artifactId>spring-cloud-stream</artifactId>
4   </dependency>
5   <!--消费者-->
6   <dependency>
7       <groupId>org.springframework.cloud</groupId>
8       <artifactId>spring-cloud-stream-binder-rabbit</artifactId>
9   </dependency>
```

配置

```YAML
1   spring:
2     application:
3       name: rabbitmq-service #指定服务名
4     rabbitmq:
5       addresses: 127.0.0.1
6       username: itcast
7       password: itcast
8       virtual-host: myhost
9     cloud:
10      stream:
11        bindings:
12          output: #发送通道
13            destination: itcast-default #指定消息发送目的地
14          input: #消费通道
15            destination: itcast-default #指定消息来源
16            #contentType: text/plain #消息类型
17        binders:  #配置绑定器
18          defaultRabbit:
19            type: rabbit
```

接收消息

定义接收通道：input

```Java
1   public interface Sink {
2
3     @Input("input") // 接收通道接口
4     SubscribableChannel input();
5
6   }
```

消息接收：

```java
@Component
@EnableBinding(Sink.class) //绑定通道
public class GetMessageSender {

    @StreamListener("input") // 监听通道
    private MessageChannel messageChannel ;

    //接收消息
    public void getMessage(String message) {
       System.out.println( message);
    }
}
```

自定义通道

依赖：

```xml
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream</artifactId>
</dependency>
<!--生产者-->
<dependency>
    <groupId>org.sshengcpringframework.cloud</groupId>
    <artifactId>spring-cloud-starter-stream-rabbit</artifactId>
</dependency>
<!--消费者-->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream-binder-rabbit</artifactId>
</dependency>
```

配置：

```yaml
spring:
  cloud:
    stream:
      defaultBinder: defaultRabbit
    bindings:
      inputOrder:        # 消费者
        destination: mqTestOrder
      outputOrder:       # 生产者
        destination: mqTestOrder
```

通道接口：

```java
/*自定义的消息通道*/
public interface MyProcessor {

  String MYOUTPUT = "myoutput";
  String MYINPUT = "myinput";

  @Output("myoutput")  //发送通道
  MessageChannel myoutput();

  @Input("myinput")  //接收通道
  SubscribableChannel myinput();

}
```

发送、接收：

```java
@Component
@EnableBinding(MyProcessor.class)
public class Message{

  //消息发送-----------------------------------------
  @Autowired
  private MessageChannel myoutput;
  public void send(Object msg){
    myoutput.send((MessageBuilder.withPayload(msg).build());
```

```
  }

  //消息接收-----------------------------------------
  @StreamListener(MyProcessor.MYINPUT)
  public void input(String msg) {
    System.out.println("获取到消息: "+msg);
  }
}
```

消息分组

默认情况下，在服务启动多个实例时，实例都会绑定到同一个消息通道的目标主题上，每个实例都可以接收，如果不想这样，就需要消息分组。

消息分组配置：

YAML

```yaml
spring:
 cloud:
  stream:
   bindings:
    input:
      destination: itcast-default
    myinput:
      destination: testChannel
      group: group-2  #分组
```

消息分区

多个相同特征的消息被一个实例接收。

生产者配置：

YAML

```yaml
spring:
 cloud:
   stream:
     bindings:
       output:
         destination: itcast-default
       outputOrder:
         destination: testChannel
         producer:
           partition-key-expression: payload #分区关键字
           partition-count: 2  #分区大小
     binders:
       defaultRabbit:
         type: rabbit
```

消费者配置：

YAML

```yaml
spring:
 cloud:
   stream:
     instanceCount: 2 #消费者总数
     instanceIndex: 1 #当前索引，0开始
     bindings:
       input:
         destination: itcast-default
       inputOrder:
         destination: testChannel
         group: group-2
         consumer:
           partitioned: true #开启分区支持
     binders:
       defaultRabbit:
         type: rabbit
```