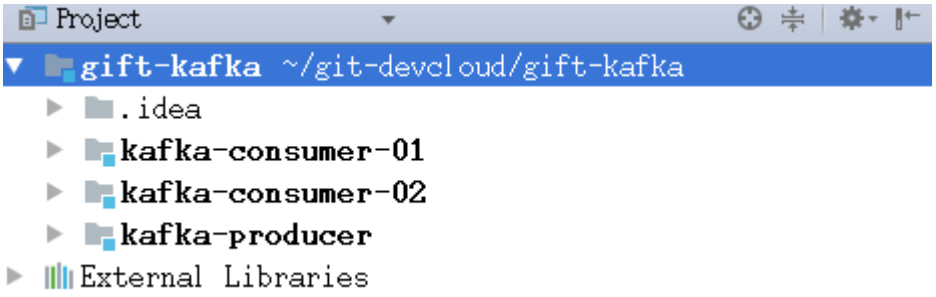
Kafka 实践作业

1. 实践概述

依照 spring-cloud 模式，在 IDEA 中创建 3 个 maven 工程：一个生产者，2 个消费者。



2. 依赖包引入

在每个工程的 pom 文件中，引入 kafka 的依赖包：

```
<dependencies>
    <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-stream</artifactId>
        </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-stream-kafka</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
    </dependency>
</dependencies>
```

3. 配置生产者

配置文件：

```yaml
server:
  port: 8001

spring:
  application:
    name: kafka-producer
  cloud:
    stream:
      instance-count: 1
      instance-index: 0
      kafka:
        binder:
          brokers: localhost:9092
          zk-nodes: localhost:2181
          minPartitionCount: 1
          autoCreateTopics: true
          autoAddPartitions: true
      bindings:
        output:
          destination: ll_topic
          content-type: text/plain
          group: ll_group_1,ll_group_2
          producer:
            partitionCount: 1
            partitioned: false
```

zookeeper与kafka的服务地址

topic名称

group名称

发送消息的 restAPI：

```java
package com.service.demo;

import ...

@SpringBootApplication
@RestController
public class KafkaProducerApplication {

    public static void main(String[] args) { SpringApplication.run(KafkaProducerApplication.class, args); }

    @Autowired
    private SendService service;

    @RequestMapping(value = "/send/{msg}", method = RequestMethod.GET)
    public void send(@PathVariable("msg") String msg){
        service.sendMessage(msg);
    }
}
```

发送消息的实现：

```java
package com.service.demo;

import ...

@EnableBinding(Source.class)
public class SendService {

    @Autowired
    private Source source;

    public void sendMessage(String msg) {

        try {

            source.output().send(MessageBuilder.withPayload(msg).build());

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

4. 配置消费者 1

配置文件：

```yaml
server:
  port: 8002

spring:
  application:
    name: kafka-consumer-01
  cloud:
    stream:
      instance-count: 1
      instance-index: 0
      kafka:
        binder:
          brokers: localhost:9092
          zk-nodes: localhost:2181
          minPartitionCount: 1
          autoCreateTopics: true
          autoAddPartitions: true
      bindings:
        input:
          destination: ll_topic
          content-type: text/plain
          group: ll_group_1
          consumer:
            concurrency: 1
            partitioned: false
```

zookeeper与kakfa的服务地址

topic名称

group名称

接受消息：

```java
package com.service.demo;

import ...

@EnableBinding(Sink.class)
public class MsgSink {
    @StreamListener(Sink.INPUT)
    public void handler(String message) {
        System.out.println("kafka-consumer-01:" + message);
    }
}
```

5. 配置消费者 2

配置文件：

application.yml

```yaml
server:
  port: 8003

spring:
  application:
    name: kafka-c-02
  cloud:
    stream:
      instance-count: 1
      instance-index: 0
      kafka:
        binder:
          brokers: localhost:9092
          zk-nodes: localhost:2181
          minPartitionCount: 1
          autoCreateTopics: true
          autoAddPartitions: true
      bindings:
        input:
          destination: ll_topic
          content-type: text/plain
          group: ll_group_2
          consumer:
            concurrency: 1
            partitioned: false
```

zookeeper与kafka的服务地址

topic名称

group名称

接受消息：

```java
package com.service.demo;

import ...

@EnableBinding(Sink.class)
public class MsgSink {
    @StreamListener(Sink.INPUT)
    public void handler(String message) {
        System.out.println("kafka-consumer-02:" + message);
    }
}
```

6. 结果验证

分别启动服务，kafka-producer，kafka-consumer-01，kafka-consumer-02，浏览器访问：

http://localhost:8001/send/kafkamessage-test-ll

两个消费服务分别输出。