

从所有教程的词条中查询...

Java / 6 Kafka生产消费者实战

全部开发者教程

- 3 Kafka使用初体验
- 4 Kafka核心扩展内容
- 5 Kafka核心之存储和容错机制
- 6 Kafka生产消费者实战

- 7 Kafka技巧篇
- 8 Kafka小试牛刀实战篇
- 9 Kafka核心复盘

第15周-极速上手内存数据库Redis

- 1 快速了解Redis
- 2 Redis核心实践
- 3 Redis封装工具类技巧



徐老师 • 更新于 2020-09-25

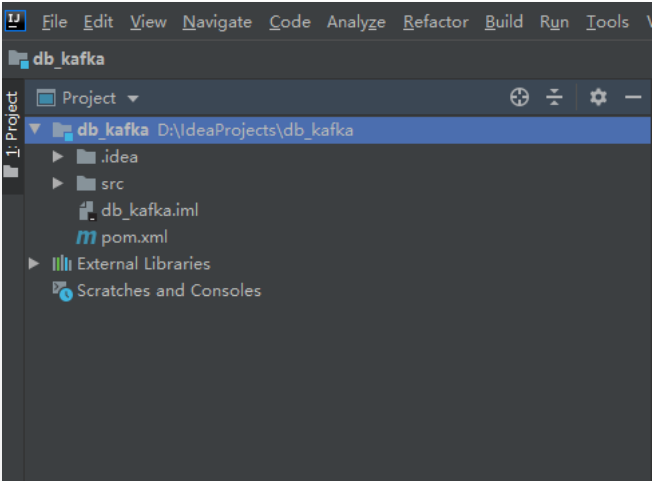
上一节 5 Kafka核心之存... 7 Kafka技巧篇 下一节

前面我们使用基于console的生产者和消费者对topic实现了数据的生产和消费，，这个基于控制台的生产者和消费者主要是让我们做测试用的  
在实际工作中，我们有时候需要将生产者和消费者功能集成到我们已有的系统中，此时就需要写代码实现生产者和消费者的逻辑了。  
在这我们使用java代码来实现生产者和消费者的功能

## Kafka Java代码编程

### Java代码实现生产者代码

先创建maven项目， db\_kafka



添加kafka的maven依赖

<> 代码块

```
1 <dependency>
2   <groupId>org.apache.kafka</groupId>
3   <artifactId>kafka-clients</artifactId>
4   <version>2.4.1</version>
5 </dependency>
```

开发生产者代码

<> 代码块

```
1 package com.imooc.kafka;
2
3 import java.util.Properties;
4
5 import org.apache.kafka.clients.producer.KafkaProducer;
6 import org.apache.kafka.clients.producer.ProducerRecord;
7 import org.apache.kafka.common.serialization.StringSerializer;
8
9 /**
10  * 需求：Java代码实现生产者代码
```

意见反馈

收藏教程

标记书签

```
12 public class ProducerDemo {
13
14     public static void main(String[] args) {
15
16         Properties prop = new Properties();
17         //指定kafka的broker地址
18         prop.put("bootstrap.servers", "bigdata01:9092,bigdata02:9092,bigdata03:9092");
19         //指定key-value数据的序列化格式
20         prop.put("key.serializer", StringSerializer.class.getName());
21         prop.put("value.serializer", StringSerializer.class.getName());
22
23         //指定topic
24         String topic = "hello";
25
26         //创建kafka生产者
27         KafkaProducer<String, String> producer = new KafkaProducer<String, String>(prop);
28
29         //向topic中生产数据
30         producer.send(new ProducerRecord<String, String>(topic, "hello kafka"));
31
32         //关闭链接
33         producer.close();
34
35     }
36
37 }
```

等一会我们把消费者代码实现好了以后一起验证

## Java代码实现消费者代码

开发消费者代码

<> 代码块

```
1 package com.imooc.kafka;
2
3 import java.time.Duration;
4 import java.util.ArrayList;
5 import java.util.Collection;
6 import java.util.Properties;
7
8 import org.apache.kafka.clients.consumer.ConsumerRecord;
9 import org.apache.kafka.clients.consumer.ConsumerRecords;
10 import org.apache.kafka.clients.consumer.KafkaConsumer;
11 import org.apache.kafka.common.serialization.StringDeserializer;
12 /**
13  * 需求: Java代码实现消费者代码
14  */
15 public class ConsumerDemo {
16
17     public static void main(String[] args) {
18
19         Properties prop = new Properties();
20         //指定kafka的broker地址
21         prop.put("bootstrap.servers", "bigdata01:9092,bigdata02:9092,bigdata03:9092");
22         //指定key-value的反序列化类型
23         prop.put("key.deserializer", StringDeserializer.class.getName());
24         prop.put("value.deserializer", StringDeserializer.class.getName());
25         //指定消费者组
26         prop.put("group.id", "con-1");
27
28         //创建消费者
29         KafkaConsumer<String, String> consumer = new KafkaConsumer<String, String>(prop);
```

意见反馈

收藏教程

标记书签

ing,Strin

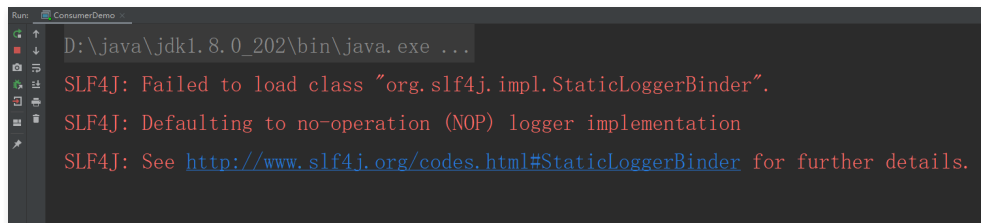
```
30     Collection<String> topics = new ArrayList<String>();
31     topics.add("hello");
32     //订阅指定的topic
33     consumer.subscribe(topics);
34
35     while(true) {
36         //消费数据【注意：需要修改jdk编译级别为1.8，否则Duration.ofSeconds(1)会报错】
37         ConsumerRecords<String, String> poll = consumer.poll(Duration.ofSeconds(1));
38         for (ConsumerRecord<String, String> consumerRecord : poll) {
39             System.out.println(consumerRecord);
40         }
41     }
42
43 }
44
45 }
```

注意：

1. 关闭kafka服务器的防火墙
2. 配置windows的hosts文件 添加kafka节点的hostname和ip的映射关系。[如果我们的hosts文件中没有对kafka节点的 hostnam和ip的映射关系做配置，在这经过多次尝试连接不上就会报错]

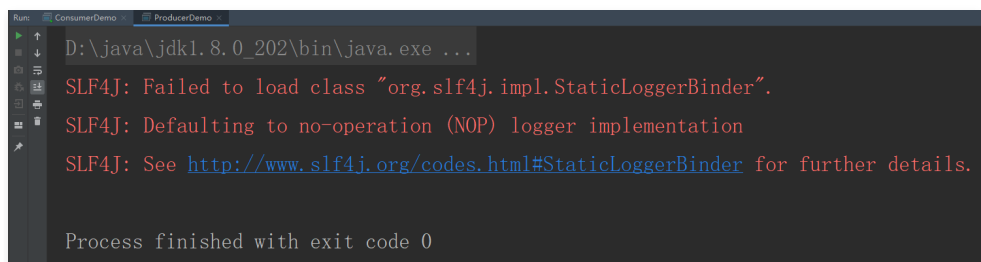
先开启消费者。

发现没有消费到数据，这个topic中是有数据的，为什么之前的数据没有消费出来呢？不要着急，先带着这个问题往下看



```
Run: ConsumerDemo
D:\java\jdk1.8.0_202\bin\java.exe ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
```

再开启生产者，生产者会生产一条数据，然后就结束



```
Run: ConsumerDemo, ProducerDemo
D:\java\jdk1.8.0_202\bin\java.exe ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

Process finished with exit code 0
```

此时回到kafka的消费者端就可以看到消费出来的数据了

<> 代码块

```
1     ConsumerRecord(topic = hello, partition = 3, leaderEpoch = 3, offset = 0, Cre
```

所以这个时候我们发现，新产生的数据我们是可以消费到的，但是之前的数据我们就无法消费了，那下面我们来分析一下这个问题

### 消费者代码扩展

&lt;&gt; 代码块

```
1 //=====
2 //开启自动提交offset功能，默认就是开启的
3 prop.put("enable.auto.commit","true");
4 //自动提交offset的时间间隔，单位是毫秒
5 prop.put("auto.commit.interval.ms","5000");
6 /*
7 注意：正常情况下，kafka消费数据的流程是这样的
8 先根据group.id指定的消费者组到kafka中查找之前保存的offset信息
9 如果查找到了，说明之前使用这个消费者组消费过数据，则根据之前保存的offset继续进行消费
10 如果没查找找到(说明第一次消费)，或者查找到了，但是查找到的那个offset对应的数据已经不存在
11 这个时候消费者该如何消费数据？
12 (因为kafka默认只会保存7天的数据，超过时间数据会被删除)
13
14 此时会根据auto.offset.reset的值执行不同的消费逻辑
15
16 这个参数的值有三种:[earliest,latest,none]
17 earliest：表示从最早的数据开始消费(从头消费)
18 latest【默认】：表示从最新的数据开始消费
19 none：如果根据指定的group.id没有找到之前消费的offset信息，就会抛异常
20
21 解释：【查找到了，但是查找到的那个offset对应的数据已经不存在了】
22 假设你第一天使用一个消费者去消费了一条数据，然后就把消费者停掉了，
23 等了7天之后，你又使用这个消费者去消费数据
24 这个时候，这个消费者启动的时候会到kafka里面查询它之前保存的offset信息
25 但是那个offset对应的数据已经被删了，所以此时再根据这个offset去消费是消费不到数据的
26
27
28 总结，一般在实时计算的场景下，这个参数的值建议设置为latest，消费最新的数据
29
30 这个参数只有在消费者第一次消费数据，或者之前保存的offset信息已过期的情况下才会生效
31 */
32 prop.put("auto.offset.reset","latest");
33 //=====
```

此时我们来验证一下，

先启动一次生产者

再启动一次消费者，看看消费者能不能消费到这条数据，如果能消费到，就说明此时是根据上次保存的offset信息进行消费了。

结果发现是可以消费到的。

注意：消费者消费到数据之后，不要立刻关闭程序，要至少等5秒，因为自动提交offset的时机是5秒提交一次

&lt;&gt; 代码块

```
1 ConsumerRecord(topic = hello, partition = 4, leaderEpoch = 5, offset = 0
```

将 auto.offset.reset 置为 earliest，修改一下 group.id 的值，相当于使用一个新的消费者，验证一下，看是否能把这个topic中的所有数据都取出来，因为新的消费者第一次肯定是获取不到offset信息的，所以就会根据 auto.offset.reset 的值来消费数据

&lt;&gt; 代码块

```
1 prop.put("group.id", "con-2");
2
3 prop.put("auto.offset.reset","earliest");
```

[意见反馈](#)[收藏教程](#)[标记书签](#)

<> 代码块

```
1 ConsumerRecord(topic = hello, partition = 2, leaderEpoch = 0, offset = 0, Cre
2 ConsumerRecord(topic = hello, partition = 3, leaderEpoch = 3, offset = 0, Cre
3 ConsumerRecord(topic = hello, partition = 4, leaderEpoch = 5, offset = 0, Cre
```

此时，关闭消费者(需要等待5秒，这样才会提交offset)，再重新启动，发现没有消费到数据，说明此时就根据上次保存的offset来消费数据了，因为没有新数据产生，所以就消费不到了。

最后来处理一下程序输出的日志警告信息，这里其实是因为缺少依赖日志依赖在pom文件中添加log4j的依赖，然后将 log4j.properties 添加到 resources 目录中

<> 代码块

```
1 <dependency>
2   <groupId>org.slf4j</groupId>
3   <artifactId>slf4j-api</artifactId>
4   <version>1.7.10</version>
5 </dependency>
6 <dependency>
7   <groupId>org.slf4j</groupId>
8   <artifactId>slf4j-log4j12</artifactId>
9   <version>1.7.10</version>
10 </dependency>
```

## Consumer消费offset查询

kafka0.9 版本以前，消费者的 offset 信息保存在 zookeeper 中  
从 kafka0.9 开始，使用了新的消费API，消费者的信息会保存在kafka里面的 \_\_consumer\_offsets 这个topic中  
因为频繁操作zookeeper性能不高，所以kafka在自己的topic中负责维护消费者的offset信息。

如何查询保存在kafka中的Consumer的offset信息呢？

使用 kafka-consumer-groups.sh 这个脚本可以查看

查看目前所有的consumer group

<> 代码块

```
1 [root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-consumer-groups.sh --list --boo
2 con-1
3 con-2
```

具体查看某一个consumer group的信息  
GROUP：当前消费者组，通过group.id指定的值  
TOPIC：当前消费的topic  
PARTITION：消费的分区  
CURRENT-OFFSET：消费者消费到这个分区的offset  
LOG-END-OFFSET：当前分区中数据的最大offset  
LAG：当前分区未消费数据量

<> 代码块

```
1 [root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-consumer-groups.sh --describe -
2 GROUP          TOPIC          PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LA
3 con-1          hello          4          1               1               0
4 con-1          hello          3          1               1               0
5 con-1          hello          2          1               1               0
```

6	con-1	hello	0	0	0	0
7	con-1	hello	1	0	0	0

此时再执行一次生产者代码，生产一条数据，重新查看一下这个消费者的offset情况

<> 代码块

```
1 [root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-consumer-groups.sh --describe -
2 GROUP          TOPIC          PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LA
3 con-1          hello          4          1               2               1
4 con-1          hello          2          1               1               0
5 con-1          hello          3          1               1               0
6 con-1          hello          0          0               0               0
7 con-1          hello          1          0               0               0
```

### Consumer消费顺序

当一个消费者消费一个partition时候，消费的数据顺序和此partition数据的生产顺序是一致的  
当一个消费者消费多个partition时候，消费者按照partition的顺序，首先消费一个partition，当消费完一个partition最新的数据后再消费其它partition中的数据

总之：如果一个消费者消费多个partiton，只能保证消费的数据顺序在一个partition内是有序的  
也就是说消费kafka中的数据只能保证消费partition内的数据是有序的，多个partition之间是无序的。

### Kafka的三种语义

kafka可以实现以下三种语义，这三种语义是针对消费者而言的：

- 至少一次：at-least-once

这种语义有可能会对数据重复处理

实现至少一次消费语义的消费者也很简单。

1: 设置enable.auto.commit为false，禁用自动提交offset

2: 消息处理完之后手动调用consumer.commitSync()提交offset

这种方式是在消费数据之后，手动调用函数consumer.commitSync()异步提交offset，有可能处理多次的场景是消费者的消息处理完并输出到结果库，但是offset还没提交，这个时候消费者挂掉了，再重启的时候会重新消费并处理消息，所以至少会处理一次

- 至多一次：at-most-once

这种语义有可能会丢失数据

至多一次消费语义是kafka消费者的默认实现。配置这种消费者最简单的方式是

1: enable.auto.commit设置为true。

2: auto.commit.interval.ms设置为一个较低的时间范围。

由于上面的配置，此时kafka会有一个独立的线程负责按照指定间隔提交offset。

消费者的offset已经提交，但是消息还在处理中(还没有处理完)，这个时候程序挂了，导致数据没有被成功处理，再重启的时候会从上次提交的offset处消费，导致上次没有被成功处理的消息就丢失了。

- 仅一次：exactly-once

这种语义可以保证数据只被消费处理一次。

- 1: 将enable.auto.commit设置为false，禁用自动提交offset
  - 2: 使用consumer.seek(topicPartition, offset)来指定offset
  - 3: 在处理消息的时候，要同时保存住每个消息的offset。以原子事务的方式保存offset和处理的消  
息结果，这个时候相当于自己保存offset信息了，把offset和具体的数据绑定到一块，数据真正处理成功的时候才会保存offset信息
- 这样就可以保证数据仅被处理一次了。

5 Kafka核心之存储和容错机制 ◀ 上一节      下一节 ▶ 7 Kafka技巧篇

 我要提出意见反馈

