SpringBoot - Kafka的集成与使用详解1（安装配置、基本用法）

发布：hangge 阅读：3376 2020-06-12

Kafka 是一个由 **LinkedIn** 开发的分布式消息系统，详细介绍可以查看我之前的文章（点击跳转）。本文演示如何在 **Spring Boot** 项目中集成并使用 **Kafka**。

一、安装配置

1，环境准备

关于 **Kafka** 和 **ZooKeeper** 的安装，可以参考我之前写的文章：

- 消息中间件 Kafka 介绍与安装教程（使用CentOS环境）
- Docker - 通过容器部署Kafka环境教程（以及ZooKeeper）

2，项目配置

（1）首先编辑项目的 **pom.xml** 文件，添加 **spring-kafka** 依赖：

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

（2）然后在 **application.properties** 中添加 **Kafka** 相关配置：

> **提示：**这里我们直接使用 **Kafka** 提供的 **StringSerializer** 和 **StringDeserializer** 进行数据的序列化和反序列化，然后使用 **json** 作为标准的数据传输格式。
>
> - 虽然我们也可以自定义序列化和反序列化器进行实体类的序列化和反序列化，但实现起来十分麻烦，而且还有很多类型不支持，非常脆弱。复杂类型的支持更是一件痛苦的事情，不同版本之间的兼容性问题更是一个极大的挑战。由于 **Serializer** 和 **Deserializer** 影响到上下游系统，导致牵一发而动全身。所以建议直接使用 **Kafka** 提供的序列化和反序列化类。

```
###########【Kafka集群】###########
spring.kafka.bootstrap-servers=192.168.60.133:9092

###########【初始化生产者配置】###########
# 重试次数
spring.kafka.producer.retries=0
# 应答级别:多少个分区副本备份完成时向生产者发送ack确认(可选0、1、all/-1)
spring.kafka.producer.acks=1
# 批量大小
spring.kafka.producer.batch-size=16384
# 提交延时
spring.kafka.producer.properties.linger.ms=0
# 当生产端积累的消息达到batch-size或接收到消息linger.ms后,生产者就会将消息提交给kafka
# linger.ms为0表示每接收到一条消息就提交给kafka,这时候batch-size其实就没用了

# 生产端缓冲区大小
spring.kafka.producer.buffer-memory = 33554432
# Kafka提供的序列化和反序列化类
spring.kafka.producer.key-serializer=org.apache.kafka.common.serialization.StringSerializer
spring.kafka.producer.value-serializer=org.apache.kafka.common.serialization.StringSerializer
# 自定义分区器
# spring.kafka.producer.properties.partitioner.class=com.felix.kafka.producer.CustomizePartitioner

###########【初始化消费者配置】###########
# 默认的消费组ID
spring.kafka.consumer.properties.group.id=defaultConsumerGroup
# 是否自动提交offset
spring.kafka.consumer.enable-auto-commit=true
# 提交offset延时(接收到消息后多久提交offset)
spring.kafka.consumer.auto.commit.interval.ms=1000
# 当kafka中没有初始offset或offset超出范围时将自动重置offset
# earliest:重置为分区中最小的offset;
# latest:重置为分区中最新的offset(消费分区中新产生的数据);
# none:只要有一个分区不存在已提交的offset,就抛出异常;
spring.kafka.consumer.auto-offset-reset=latest
# 消费会话超时时间(超过这个时间consumer没有发送心跳,就会触发rebalance操作)
spring.kafka.consumer.properties.session.timeout.ms=120000
# 消费请求超时时间
spring.kafka.consumer.properties.request.timeout.ms=180000
# Kafka提供的序列化和反序列化类
spring.kafka.consumer.key-deserializer=org.apache.kafka.common.serialization.StringDeserializer
spring.kafka.consumer.value-deserializer=org.apache.kafka.common.serialization.StringDeserializer
# 消费端监听的topic不存在时，项目启动会报错(关掉)
spring.kafka.listener.missing-topics-fatal=false
# 设置批量消费
# spring.kafka.listener.type=batch
# 批量消费每次最多消费多少条消息
# spring.kafka.consumer.max-poll-records=50
```

附：基本用法

1，创建生产者

消息发送主要是使用 **KafkaTemplate**，它具有多个方法可以发送消息，这里我们使用最简单的：

```
@RestController
public class KafkaProducer {
    @Autowired
    private KafkaTemplate<String, Object> kafkaTemplate;

    // 发送消息
    @GetMapping("/test/{message}")
    public void sendMessage1(@PathVariable("message") String normalMessage) {
        kafkaTemplate.send("topic1", normalMessage);
    }
}
```

2，创建消费者

监听器主要是使用 **@KafkaListenter** 注解即可，可以监听多个 **topic** 也可以监听单个（多个的话用逗号隔开）：

```
@Component
public class KafkaConsumer {
    // 消费监听
    @KafkaListener(topics = {"topic1"})
    public void listen1(String data) {
        System.out.println(data);
    }
}
```

3，开始测试

（1）启动项目后，我们访问 **/test** 接口发送一条消息：

文章类别

编程社区

Swift Flex / Flash
HTML5 / CSS3 Cordova
SpringBoot / Cloud Docker / K8s
React / RN 其他

奇文共赏

本类相关

SpringBoot - 使用ApplicationPower快速生成初始项目…
SpringBoot - Lombok使用详解3（@NoArgsConstruct…
SpringBoot - @Configuration、@Bean注解的使用详…
分布式配置中心Spring Cloud Config使用详解7（使用…
分布式配置中心Spring Cloud Config使用详解12（请…
SpringBoot - MyBatis-Plus使用详解12（使用ActiveRe…
分布式服务跟踪Spring Cloud Sleuth使用详解3（设置…
SpringBoot - 实现静态资源的访问（附：修改过滤规则…
消息驱动微服务框架Spring Cloud Stream使用详解1（…
SpringBoot - MyBatis-Plus使用详解10（Service的CR…

（2）控制台这边打印如下信息，说明消息已经成功被消费：

上一篇 SpringBoot - MyBatis-Plus使用详解21（动态表名）
下一篇 SpringBoot - Kafka的集成与使用详解2（手动创建、修改、查询Topic）

评论 (0)

在这里留下你想说的话。

昵称 邮箱 提交