用Spark往Kafka里面写对象设计与实现

Spark和Kafka都是比较常用的两个大数据框架，Spark里面提供了对Kafka读写的支持。默认情况下我们Kafka只能写Byte数组到Topic里面，如果我们想往Topic里面读写String类型的消息，可以分别使用Kafka里面内置的StringEncoder编码类和StringDecoder解码类。那如果我们想往Kafka里面写对象怎么办？

别担心，Kafka中的kafka.serializer里面有Decoder和Encoder两个trait，这两个trait就是Kafka Topic消息相关的解码类和编码类，内置的StringDecoder和StringEncoder类分别都是继承那两个trait的。直接将String对象用给定的编码转换成Byte数组。来看下Decoder和Encoder两个trait的实现：

```
/**
 * A decoder is a method of turning byte arrays into objects.
 * An implementation is required to provide a constructor that
 * takes a VerifiableProperties instance.
 */
trait Decoder[T] {
  def fromBytes(bytes: Array[Byte]): T
}

/**
 * An encoder is a method of turning objects into byte arrays.
 * An implementation is required to provide a constructor that
 * takes a VerifiableProperties instance.
 */
trait Encoder[T] {
  def toBytes(t: T): Array[Byte]
}
```

也就是说，我们自定义的编码和解码类只需要分别实现toBytes和fromBytes函数即可。那我们如何将对象转换成Byte数组，并且如何将Byte数组转换回对象呢？记得Java中写对象的类没？我们可以用ByteArrayOutputStream并结合ObjectOutputStream类将对象转换成Byte数组；并用ByteArrayInputStream结合ObjectInputStream类将Byte数组转换回对象。这不就实现了吗？？废话不多说，来看看怎么实现：

```
class IteblogDecoder[T](props: VerifiableProperties = null) extends Decoder[T] {

   def fromBytes(bytes: Array[Byte]): T = {
     var t: T = null.asInstanceOf[T]
     var bi: ByteArrayInputStream = null
```

```
    var oi: ObjectInputStream = null
    try {
      bi = new ByteArrayInputStream(bytes)
      oi = new ObjectInputStream(bi)
      t = oi.readObject().asInstanceOf[T]
    }
    catch {
      case e: Exception => {
        e.printStackTrace(); null
      }
    } finally {
      bi.close()
      oi.close()
    }
    t
  }
}

class IteblogEncoder[T](props: VerifiableProperties = null) extends Encoder[T] {

  override def toBytes(t: T): Array[Byte] = {
    if (t == null)
      null
    else {
      var bo: ByteArrayOutputStream = null
      var oo: ObjectOutputStream = null
      var byte: Array[Byte] = null
      try {
        bo = new ByteArrayOutputStream()
        oo = new ObjectOutputStream(bo)
        oo.writeObject(t)
        byte = bo.toByteArray
      } catch {
        case ex: Exception => return byte
      } finally {
        bo.close()
        oo.close()
      }
      byte
    }
  }
}
```

这样我们就定义了自己的编码和解码器。那如何使用呢？？假设我们有一个Person类。如下：

```
case class Person(var name: String, var age: Int)
```

我们可以在发送数据这么使用：

```
def getProducerConfig(brokerAddr: String): Properties = {
   val props = new Properties()
   props.put("metadata.broker.list", brokerAddr)
   props.put("serializer.class", classOf[IteblogEncoder[Person]].getName)
   props.put("key.serializer.class", classOf[StringEncoder].getName)
   props
 }

def sendMessages(topic: String, messages: List[Person], brokerAddr: String) {
   val producer = new Producer[String, Person](
            new ProducerConfig(getProducerConfig(brokerAddr)))
   producer.send(messages.map {
     new KeyedMessage[String, Person](topic, "Iteblog", _)
   }: _*)
   producer.close()
}

def main(args: Array[String]) {
   val sparkConf = new S parkConf().setAppName(this.getClass.getSimpleName)
   val ssc = new StreamingContext(sparkConf, Milliseconds(500))
   val topic = args(0)
   val brokerAddr = ":9092"

   val data = List(Person("wyp", 23), Person("spark", 34), Person("kafka", 23),
       Person("iteblog", 23))
   sendMessages(topic, data, brokerAddr)
}
```

在接收端可以这么使用

```
val sparkConf = new S parkConf().setAppName(this.getClass.getSimpleName)
val ssc = new StreamingContext(sparkConf, Milliseconds(500))
val (topic, groupId) = (args(0), args(1))
```

```
val kafkaParams = Map("zookeeper.connect" -> ":2181",
    "group.id" -> groupId,
    "auto.offset.reset" -> "smallest")

val stream = KafkaUtils.createStream[String, Person, StringDecoder,
      IteblogDecoder[Person]](ssc, kafkaParams, Map(topic -> 1),
      StorageLevel.MEMORY_ONLY)
  stream.foreachRDD(rdd => {
    if (rdd.count() != 0) {
      rdd.foreach(item => if (item != null) println(item))
    } else {
      println("Empty rdd!!")
    }
  })
  ssc.start()
  ssc.awaitTermination()
  ssc.stop()
```

这样可以发送任意可序列化的对象了。下面是效果：

```
Empty rdd!!
(Iteblog,Person(wyp,23))
(Iteblog,Person(spark,34))
(Iteblog,Person(kafka,23))
(Iteblog,Person(iteblog,23))
Empty rdd!!
Empty rdd!!
Empty rdd!!
Empty rdd!!
```

在例子中我们只是简单的将接收到的消息打印到控制台。如果没有接收到消息，则打印Empty rdd!!。