Spark RDD API扩展开发(2):自定义RDD

《Spark RDD API扩展开发(1)》、《Spark RDD API扩展开发(2):自定义RDD》

在本博客的《Spark RDD API扩展开发(1)》文章中我介绍了如何在现有的RDD中添加自定义的函数。本文将介绍如何自定义一个RDD类，假如我们想对没见商品进行打折，我们想用Action操作来实现这个操作，下面我将定义IteblogDiscountRDD类来计算商品的打折，步骤如下：

一、创建IteblogDiscountRDD类

自定义RDD类需要继承Spark中的RDD类，并实现其中的方法：

```
/**
 * User: 过往记忆
 * Date: 15-04-01
 * Time: 上午00:59
 * bolg: https://www.iteblog.com
 * 本文地址：https://www.iteblog.com/archives/1299
 * 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 * 过往记忆博客微信公共帐号：iteblog_hadoop
 */
class IteblogDiscountRDD(prev:RDD[SalesRecord],xxxxx:Double)
   extends RDD[SalesRecord](prev){

//继承compute方法
override def compute(split: Partition, context: TaskContext): Iterator[SalesRecord] =  {
  firstParent[SalesRecord].iterator(split, context).map(salesRecord => {
     val discount = salesRecord.itemValue*discountPercentage
     new SalesRecord(salesRecord.id,
     salesRecord.customerId,salesRecord.itemId,discount)
})}

//继承getPartitions方法
override protected def getPartitions: Array[Partition] =
     firstParent[SalesRecord].partitions
}
```

上面代码中，我创建了一个IteblogDiscountRDD类，这个RDD只操纵销售数据，当我们继承RDD类时，我们必须重载两个方法：

compute

这个函数是用来计算RDD中每个的分区的数据，在我代码中，我们输入了销售数据，并对其中的数据计算打折计算。

getPartitions

getPartitions函数允许开发者为RDD定义新的分区，在我们的代码中，并没有改变RDD的分区，重用了父RDD的分区。

定义IteblogDiscountRDD的时候将类型写死了(SalesRecord)，它只能用来处理SalesRecord数据。如果我们想定义一个通用的RDD，只需要类似下面写即可

```
class IteblogRDD(prev:RDD[T],XXXX:C)
   extends RDD[T](prev){

//继承compute方法
override def compute(split: Partition, context: TaskContext): Iterator[T] =  {
  ..............................
}

//继承getPartitions方法
override protected def getPartitions: Array[Partition] =
     ..............................
}
```

二、自定义discount函数

我们自定义discount函数，该函数可以创建一个IteblogDiscountRDD：

```
def discount(discountPercentage:Double) = new IteblogDiscountRDD(rdd,discountPercentage)
```

三、使用IteblogDiscountRDD

使用IteblogDiscountRDD也是非常简单的，我们可以像使用内置的RDD一样来使用：

```
import IteblogCustomFunctions._

val discountRDD = salesRecordRDD.discount(0.1)
```

```
println(discountRDD.collect().toList)
```

自此，我们已经学会了如何在现有的RDD中定义方法和自定义自己的RDD。