MapReduce4 OutputFormat数据输出

OutputFormat接口实现类

> OutputFormat是MapReduce输出的基类，所有实现Mapreduce输出都实现了OutputFormat接口，下面是几个常见的接口

- TextOutputFormat
- SequenceFileOutputFormat
- 自定义输出

TextOutputFormat

> 文本输出TextoutputFormat默认的输出格式是TextOutputFormat，它把每一条记录写为文本行，它的键和值可以是任意类型，因为TextOutputFormat调用toString方法把它们转化为字符串

```
Configuration conf = new Configuration();
conf.set("mapred.textoutputformat.separator", ": ");
//默认输出格式，并不用进行设置setOutputFormatClass
```

SequenceFileOutputFormat

> 将SequenceFileOutputFormat输出作为后续MapReduce任务的输入，这便是一个很好的输出格式，因为它格式紧凑，很容易被压缩

```
job.setOutputFormatClass(SequenceFileOutputFormat.class);
//使用SequenceFileOutFoemat合并输出
SequenceFileOutputFormat.setOutputPath(job,new Path(args[1]));
```

自定义OutputFormat输出

- 自定义一个类继承FileOutputFormat
- 改写RecordWrite，具体改写输出数据的方法write

自定义OutputFormat实例

需求

> 现有一个订单的评论数据：
>
> 将订单的好评与差评区分开来，将最终的数据分开到不同的文件夹下面去，其中数据第九字段表示好评0，中评1，差评2

分析

第一个想到的是自定义分区，但是自定义分区后的程序运行的结果是数据保存在了同一个目录下的不同文件中。而本题的关键点是要在一个mapReduce程序中根据数据的不同输出两类结果到不同的目录，这需要我们通过自定义OutputFormat来实现。

上手代码

自定义一个FileOutputFormat

```
public class Custom_OutputFormat extends FileOutputFormat<Text, NullWritable>{

    @Override
    public RecordWriter<Text, NullWritable> getRecordWriter(TaskAttemptContext
context) throws IOException, InterruptedException{
        Configuration conf = new Configuration();
        FileSystem fileSystem = FileSystem.get(conf);

        //两个输出路径
        FSDataOutputStream fsDataOutputStream1 = fileSystem.create(new
Path("E:\\hadoop\\outputformat1\\1.txt"));
        FSDataOutputStream fsDataOutputStream2 = fileSystem.create(new
Path("E:\\hadoop\\outputformat1\\2.txt"));

        Custom_RecordWriter custom_recordWriter = new
Custom_RecordWriter(fsDataOutputStream1, fsDataOutputStream2);

        return custom_recordWriter;
    }
}
```

自定义一个RecordWriter

```
public class Custom_RecordWriter extends RecordWriter<Text, NullWritable>{

    FSDataOutputStream out1 = null;
    FSDataOutputStream out2 = null;

    @Override
    public String toString(){
        return "Custom_RecordWriter{"
            + "out1=" + out1 +
            ", out2=" + out2 +
            "}";
    }

    public FSDataOutputStream getOut1(){
        return out1;
    }

    public void setOut1(FSDataOutputStream out1){
        this.out1 = out1;
    }

    public FSDataOutputStream getOut2(){
        return out2;
    }

    public void setOut1(FSDataOutputStream out2){
        this.out2 = out2;
    }
```

```
    public Custom_RecordWriter(){

    }

    public Custom_RecordWriter(FSDataOutputStream out1, FSDataOutputStream out2)
{
        this.out1 = out1;
        this.out2 = out2;
    }

    /*
     * 写入数据的方法
     * @param key 要写入的key值
     * @param value 要写入的value值
     *
     */
    @Override
    public void write(Text key, NullWritable value) throws IOException,
InterruptedException{

        if(key.toString().split("\t")[9].equals("0")){
            //好评
            out1.write(key.toString().getBytes());
            out1.write("\r\n".getBytes());
        }else{
            //中评和差评
            out2.write(key.toString().getBytes());
            out2.write("\r\n".getBytes());
        }
    }

    //关闭
    @Override
    public void close(TaskAttemptContext taskAttemptContext) throws IOException,
InterruptedException{
        if(out1 != null){
            out1.close();
        }
        if(out2 != null){
            out2.close();
        }
    }
}
```

自定义Mapper

```
public class Custom_Mapper extends Mapper<LongWritable, Text, Text,
NullWritable>{
    @Override
    protect void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        context.write(value, NullWritable.get());
}
```

开发mapreduce处理流程

```
public class Custom_Driver{
    public static void main(String[] args) throws Exception{
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
        job.setJarByClass(Custom_Drive.class);

        //设置输入
        job.setInputFormatClass(TextInputFormat.class);

        TextInputFormat.addInputPath(job, new
Path("E:\\hadoop\\test\\ordercomment.csv"));

        job.setMapperClass(Custom_Mapper.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(NullWritable.class);

        job.setOutputFormatClass(Custom_OutputFormat.class);

        // 这里path的路径可以任意设置,因为我们在自定义outPutFormat中已经将输出路径确定
        Custom_OutputFormat.setOutputPath(job,new Path("E:\\hadoop\\DeBug\\测试结
果\\outputformat1"));

        boolean b = job.waitForCompletion(true);

        System.out.println(b?0:1);

    }
}
```

在现实实现过程中，经常会自定义输入和输出一起使用，所以需要熟练掌握自定义输入输出的代码。