> 先知道是什么，再去了解为什么

MapReduce之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。下面我们介绍几种常见的 OutputFormat实现类。

- 文本输出**TextoutputFormat**默认的输出格式是 TextOutputFormat，它把每条记录写为文本行。它的键和值可以是任意类型，因为textOutputFormat调用 toString方法把它们转换为字符串
- **SequenceFileoutputFormat**将 SequenceFileOutput Format输出作为后续 MapReduce任务的输入，这便是一种好的输出格式，因为它的**格式紧凑，很容易被压缩。**
- 自定义 OutputFormat根据用户需求，自定义实现输出。

自定义OutputFormat

1.使用场景

> 为了实现控制最终文件的输出路径和输出格式，可以自定义 OutputFormat。
> 例如：要在一个 MapReduce程序中根据数据的不同输出两类结果到不同目录，这类灵活的输出需求可以通过自定义 OutputFormat来实现。

2.自定义 OutputFormat步骤

（1）自定义一个类继承 FileOutputFormat
（2）改写Record writer，具体改写输出数据的方法write。

3.实际需求

> 过滤日志文件中，包含shuaiqi的字符串然后输出到output.log文件中，不包含shuaiqi的输出到other.log中

OuputFormat代码

```
import java.io.IOException;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.RecordWriter;
import org.apache.hadoop.mapreduce.TaskAttemptContext;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

// 继承FileOutputFormat，规定输出泛型
public class FilterOutputFormat extends FileOutputFormat<Text, NullWritable>{

    @Override
    public RecordWriter<Text, NullWritable> getRecordWriter(TaskAttemptContext
job)         throws IOException, InterruptedException {

        // 创建一个RecordWriter,重写里面的write方法
        return new FilterRecordWriter(job);
    }
```

```
}
```

RecordWrite代码

```
import java.io.IOException;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.RecordWriter;
import org.apache.hadoop.mapreduce.TaskAttemptContext;

public class FilterRecordWriter extends RecordWriter<Text, NullWritable> {

    // 定义两个输出流
    FSDataOutputStream out = null;
    FSDataOutputStream otherOut = null;

    public FilterRecordWriter(TaskAttemptContext job) {

        // 1 获取文件系统
        FileSystem fs;

        try {
            fs = FileSystem.get(job.getConfiguration());

            // 2 创建输出文件路径
            Path outPath = new Path("e:/output.log");
            Path otherPath = new Path("e:/other.log");

            // 3 创建输出流
            out = fs.create(outPath);
            otherOut = fs.create(otherPath);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void write(Text key, NullWritable value) throws IOException,
InterruptedException {

        // 判断是否包含“atguigu”输出到不同文件
        if (key.toString().contains("shuaiqi")) {
            out.write(key.toString().getBytes());
        } else {
            otherOut.write(key.toString().getBytes());
        }
    }

    @Override
    public void close(TaskAttemptContext context) throws IOException,
InterruptedException {

        // 关闭资源
        if (out != null) {
```

```
            out.close();
        }

        if (otherOut != null) {
            otherOut.close();
        }
    }
}
```

FilterMapper代码

```
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class FilterMapper extends Mapper<LongWritable, Text, Text, NullWritable>
{

    @Override
    protected void map(LongWritable key, Text value, Context context)   throws
IOException, InterruptedException {

        // 写出
        context.write(value, NullWritable.get());
    }
}
```

FilterReducer代码

```
import java.io.IOException;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class FilterReducer extends Reducer<Text, NullWritable, Text,
NullWritable> {

Text k = new Text();

    @Override
    protected void reduce(Text key, Iterable<NullWritable> values, Context
context)     throws IOException, InterruptedException {

        String line = key.toString();
        line = line + "\r\n";

       k.set(line);

        context.write(k, NullWritable.get());
    }
```

FilterDriver代码

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class FilterDriver {

    public static void main(String[] args) throws Exception {
// 输入输出路径需要根据自己电脑上实际的输入输出路径设置
args = new String[] { "e:/input/inputoutputformat", "e:/output2" };

        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf);

        job.setJarByClass(FilterDriver.class);
        job.setMapperClass(FilterMapper.class);
        job.setReducerClass(FilterReducer.class);

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

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(NullWritable.class);

        // 要将自定义的输出格式组件设置到job中
        job.setOutputFormatClass(FilterOutputFormat.class);

        FileInputFormat.setInputPaths(job, new Path(args[0]));

        // 虽然我们自定义了outputformat，但是因为我们的outputformat继承自
fileoutputformat
        // 而fileoutputformat要输出一个_SUCCESS文件，所以，在这还得指定一个输出目录
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        boolean result = job.waitForCompletion(true);
        System.exit(result ? 0 : 1);
    }
}
```