链滴

Flink(2)——定制输入输出源

作者：flowaters

原文链接：https://ld246.com/article/1512401421055

来源网站：链滴

概要

本文先从上一篇中的本地输入输出出发，先制作从Kafka输入，再制作输出到MySQL

本地输入输出

代码

```
package com.abeffect.blink;

import org.apache.flink.api.java.DataSet;
import org.apache.flink.api.java.ExecutionEnvironment;
import org.apache.flink.api.common.functions.FlatMapFunction;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.util.Collector;

public class WordCount {

    public static void main(String[] args) throws Exception {

        // set up the execution environment
        final ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();

        // get input data
        DataSet<String> text = env.fromElements(
                "To be, or not to be,--that is the question:--",
                "Whether 'tis nobler in the mind to suffer",
                "The slings and arrows of outrageous fortune",
                "Or to take arms against a sea of troubles,"
                );

        DataSet<Tuple2<String, Integer>> counts =
                // split up the lines in pairs (2-tuples) containing: (word,1)
                text.flatMap(new LineSplitter())
                // group by the tuple field "0" and sum up tuple field "1"
                .groupBy(0)
                .sum(1);

        // execute and print result
        counts.print();

    }

    public static final class LineSplitter implements FlatMapFunction<String, Tuple2<String, Int
ger>> {

        @Override
        public void flatMap(String value, Collector<Tuple2<String, Integer>> out) {
            // normalize and split the line
            String[] tokens = value.toLowerCase().split("\\W+");

            // emit the pairs
```

```
            for (String token : tokens) {
                if (token.length() > 0) {
                    out.collect(new Tuple2<String, Integer>(token, 1));
                }
            }
        }
    }
}
```

kafka 输入, stdout输出

代码

输出类 StdoutSink.java

```
package com.abeffect.blink;

import org.apache.flink.api.java.tuple.Tuple1;
import org.apache.flink.streaming.api.functions.sink.RichSinkFunction;

public class StdoutSink extends
        RichSinkFunction<Tuple1<String>> {

    @Override
    public void invoke(Tuple1<String> value) throws Exception {
        System.out.println(value.f0);
    }
}
```

执行类 KafkaCount.java

```
package com.abeffect.blink;

import org.apache.commons.lang.StringUtils;
import org.apache.flink.api.common.functions.FilterFunction;
import org.apache.flink.api.common.functions.MapFunction;
import org.apache.flink.api.java.tuple.Tuple1;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.connectors.kafka.FlinkKafkaConsumer010;
import org.apache.flink.streaming.util.serialization.SimpleStringSchema;

import java.util.Properties;

public class KafkaCount {

    public static void main(String[] args) throws Exception {

        final StreamExecutionEnvironment env = StreamExecutionEnvironment
                .getExecutionEnvironment();

        Properties properties = new Properties();
        properties.setProperty("bootstrap.servers", "localhost:9092");
```

```
        DataStream<String> sourceStream = env
                .addSource(new FlinkKafkaConsumer010<>("fw-blink-test", new SimpleStringSche
a(), properties));

        DataStream<Tuple1<String>> sourceStreamTra = sourceStream.filter(new FilterFunctio
<String>() {
            @Override
            public boolean filter(String value) throws Exception {
                return StringUtils.isNotBlank(value);
            }
        }).map(new MapFunction<String, Tuple1<String>>() {
            private static final long serialVersionUID = 1L;
            @Override
            public Tuple1<String> map(String value)
                    throws Exception {
                String[] args = value.split(":");
                return new Tuple1<String>(args[0]);
            }
        });

        sourceStreamTra.addSink(new StdoutSink());
        env.execute("data to stdout start");
    }
}
```

kafka输入测试

```
bin/kafka-console-producer.sh --broker-list localhost:9092 --topic fw-blink-test
```

结果查看

```
$ tailf flink-abeffect-jobmanager-0-fox.local.out
3
1
2
3
11
12
13
```

kafka 输入, mysql输出

代码

输出类 MySQLSink.java

```
package com.abeffect.blink;

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
```

```
import org.apache.flink.api.java.tuple.Tuple1;
import org.apache.flink.streaming.api.functions.sink.RichSinkFunction;

public class MySQLSink extends
        RichSinkFunction<Tuple1<String>> {

    private static final long serialVersionUID = 1L;
    private Connection connection;
    private PreparedStatement preparedStatement;
    String username = "root";
    String password = "toor";
    String drivername = "com.mysql.jdbc.Driver";
    String dburl = "jdbc:mysql://localhost:3306/blink_test";

    @Override
    public void invoke(Tuple1<String> value) throws Exception {
        Class.forName(drivername);
        connection = DriverManager.getConnection(dburl, username, password);
        String sql = "insert into sink0 (`key`) values (?)";
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString(1, value.f0);
        preparedStatement.executeUpdate();
        if (preparedStatement != null) {
            preparedStatement.close();
        }
        if (connection != null) {
            connection.close();
        }
    }
}
```

执行类 KafkaCount.java

```
package com.abeffect.blink;

import org.apache.commons.lang.StringUtils;
import org.apache.flink.api.common.functions.FilterFunction;
import org.apache.flink.api.common.functions.MapFunction;
import org.apache.flink.api.java.tuple.Tuple1;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.connectors.kafka.FlinkKafkaConsumer010;
import org.apache.flink.streaming.util.serialization.SimpleStringSchema;

import java.util.Properties;

public class KafkaCount {

    public static void main(String[] args) throws Exception {

        final StreamExecutionEnvironment env = StreamExecutionEnvironment
                .getExecutionEnvironment();
```

```
        Properties properties = new Properties();
        properties.setProperty("bootstrap.servers", "localhost:9092");
        DataStream<String> sourceStream = env
                .addSource(new FlinkKafkaConsumer010<>("fw-blink-test", new SimpleStringSche
a(), properties));

        DataStream<Tuple1<String>> sourceStreamTra = sourceStream.filter(new FilterFunctio
<String>() {
            @Override
            public boolean filter(String value) throws Exception {
                return StringUtils.isNotBlank(value);
            }
        }).map(new MapFunction<String, Tuple1<String>>() {
            private static final long serialVersionUID = 1L;
            @Override
            public Tuple1<String> map(String value)
                    throws Exception {
                String[] args = value.split(":");
                return new Tuple1<String>(args[0]);
            }
        });

        sourceStreamTra.addSink(new MySQLSink());
        env.execute("data to mysql start");
    }
}
```

kafka输入测试

```
bin/kafka-console-producer.sh --broker-list localhost:9092 --topic fw-blink-test
```

结果查看

```
mysql> select * from sink0;
+----+------+
| id | key  |
+----+------+
|  1 | 000  |
|  2 | a2   |
|  3 | a3   |
|  4 | b1   |
|  5 | b2   |
|  6 | b3   |
+----+------+
```