toc: true title: 《从0到1学习Flink》—— 如何自定义 Data Sink？ date: 2018-10-31 tags:

- Flink
- 大数据
- 流式计算

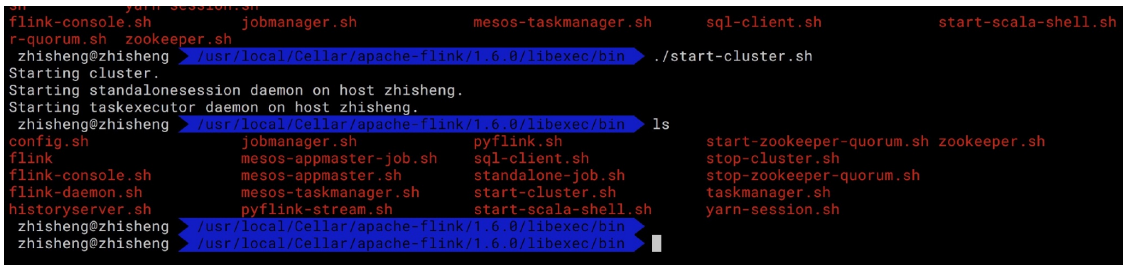
前言

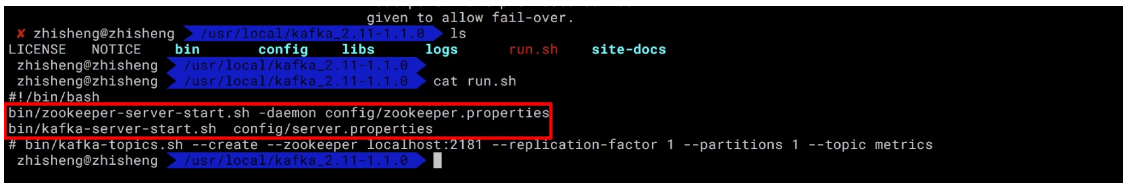
前篇文章 《从0到1学习Flink》—— Data Sink 介绍 介绍了 Flink Data Sink，也介绍了 Flink 自带的 Sink，那么如何自定义自己的 Sink 呢？这篇文章将写一个 demo 教大家将从 Kafka Source 的数据 Sink 到 MySQL 中去。

准备工作

我们先来看下 Flink 从 Kafka topic 中获取数据的 demo，首先你需要安装好了 FLink 和 Kafka 。

运行启动 Flink、Zookepeer、Kafka，





好了，都启动了！

数据库建表

```
DROP TABLE IF EXISTS `student`;
CREATE TABLE `student` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(25) COLLATE utf8_bin DEFAULT NULL,
  `password` varchar(25) COLLATE utf8_bin DEFAULT NULL,
  `age` int(10) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8
COLLATE=utf8_bin;
```

实体类

Student.java

```
package com.zhisheng.flink.model;

/**
 * Desc:
 * weixin: zhisheng_tian
```

```
 * blog: http://www.54tianzhisheng.cn/
 */
public class Student {
    public int id;
    public String name;
    public String password;
    public int age;

    public Student() {
    }

    public Student(int id, String name, String password, int age) {
        this.id = id;
        this.name = name;
        this.password = password;
        this.age = age;
    }

    @Override
    public String toString() {
        return "Student{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", password='" + password + '\'' +
                ", age=" + age +
                '}';
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPassword() {
        return password;
    }
```

```
    public void setPassword(String password) {
        this.password = password;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

工具类

工具类往 kafka topic student 发送数据

```
import com.alibaba.fastjson.JSON;
import com.zhisheng.flink.model.Metric;
import com.zhisheng.flink.model.Student;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;

import java.util.HashMap;
import java.util.Map;
import java.util.Properties;

/**
 * 往kafka中写数据
 * 可以使用这个main函数进行测试一下
 * weixin: zhisheng_tian
 * blog: http://www.54tianzhisheng.cn/
 */
public class KafkaUtils2 {
    public static final String broker_list = "localhost:9092";
    public static final String topic = "student";  //kafka topic 需
要和 flink 程序用同一个 topic

    public static void writeToKafka() throws InterruptedException {
        Properties props = new Properties();
        props.put("bootstrap.servers", broker_list);
        props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        KafkaProducer producer = new KafkaProducer<String, String>
```

```
        KafkaProducer producer = new KafkaProducer<String, String>
(props);

        for (int i = 1; i <= 100; i++) {
            Student student = new Student(i, "zhisheng" + i,
"password" + i, 18 + i);
            ProducerRecord record = new ProducerRecord<String,
String>(topic, null, null, JSON.toJSONString(student));
            producer.send(record);
            System.out.println("发送数据: " +
JSON.toJSONString(student));
        }
        producer.flush();
    }

    public static void main(String[] args) throws
InterruptedException {
        writeToKafka();
    }
}
```

SinkToMySQL

该类就是 Sink Function，继承了 RichSinkFunction ，然后重写了里面的方法。在 invoke 方法中将数据插入到 MySQL 中。

```
package com.zhisheng.flink.sink;

import com.zhisheng.flink.model.Student;
import org.apache.flink.configuration.Configuration;
import
org.apache.flink.streaming.api.functions.sink.RichSinkFunction;

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

/**
 * Desc:
 * weixin: zhisheng_tian
 * blog: http://www.54tianzhisheng.cn/
 */
public class SinkToMySQL extends RichSinkFunction<Student> {
    PreparedStatement ps;
    private Connection connection;
```

```
    private Connection connection;

    /**
     * open() 方法中建立连接，这样不用每次 invoke 的时候都要建立连接和释放连
接
     *
     * @param parameters
     * @throws Exception
     */
    @Override
    public void open(Configuration parameters) throws Exception {
        super.open(parameters);
        connection = getConnection();
        String sql = "insert into Student(id, name, password, age)
values(?, ?, ?, ?);";
        ps = this.connection.prepareStatement(sql);
    }

    @Override
    public void close() throws Exception {
        super.close();
        //关闭连接和释放资源
        if (connection != null) {
            connection.close();
        }
        if (ps != null) {
            ps.close();
        }
    }

    /**
     * 每条数据的插入都要调用一次 invoke() 方法
     *
     * @param value
     * @param context
     * @throws Exception
     */
    @Override
    public void invoke(Student value, Context context) throws
Exception {
        //组装数据，执行插入操作
        ps.setInt(1, value.getId());
        ps.setString(2, value.getName());
        ps.setString(3, value.getPassword());
        ps.setInt(4, value.getAge());
        ps.executeUpdate();
    }

    private static Connection getConnection() {
```

```
        Connection con = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/test?
useUnicode=true&characterEncoding=UTF-8", "root", "root123456");
        } catch (Exception e) {
            System.out.println("-----------mysql get connection has
exception , msg = "+ e.getMessage());
        }
        return con;
    }
}
```

Flink 程序

这里的 source 是从 kafka 读取数据的，然后 Flink 从 Kafka 读取到数据（JSON）后用阿里 fastjson 来解析成 student 对象，然后在 addSink 中使用我们创建的 SinkToMySQL，这样就可以把数据存储到 MySQL 了。

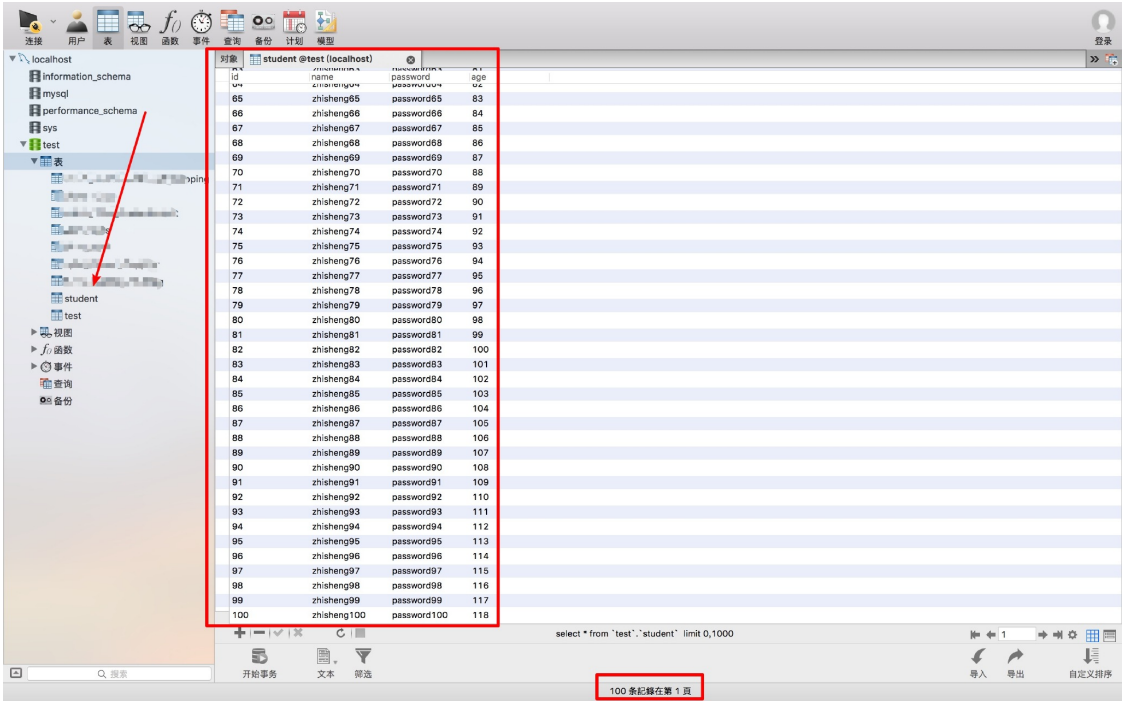
```
package com.zhisheng.flink;

import com.alibaba.fastjson.JSON;
import com.zhisheng.flink.model.Student;
import com.zhisheng.flink.sink.SinkToMySQL;
import
org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.streaming.api.datastream.DataStreamSource;
import
org.apache.flink.streaming.api.datastream.SingleOutputStreamOperato
r;
import
org.apache.flink.streaming.api.environment.StreamExecutionEnvironme
nt;
import
org.apache.flink.streaming.api.functions.sink.PrintSinkFunction;
import
org.apache.flink.streaming.connectors.kafka.FlinkKafkaConsumer011;
import
org.apache.flink.streaming.connectors.kafka.FlinkKafkaProducer011;

import java.util.Properties;
```

```
/**
 * Desc:
 * weixin: zhisheng_tian
 * blog: http://www.54tianzhisheng.cn/
 */
public class Main3 {
    public static void main(String[] args) throws Exception {
        final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("zookeeper.connect", "localhost:2181");
        props.put("group.id", "metric-group");
        props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        props.put("auto.offset.reset", "latest");

        SingleOutputStreamOperator<Student> student =
env.addSource(new FlinkKafkaConsumer011<>(
                "student",   //这个 kafka topic 需要和上面的工具类的
topic 一致
                new SimpleStringSchema(),
                props)).setParallelism(1)
                .map(string -> JSON.parseObject(string,
Student.class)); //Fastjson 解析字符串成 student 对象

        student.addSink(new SinkToMySQL()); //数据 sink 到 mysql

        env.execute("Flink add sink");
    }
}
```

结果

运行 Flink 程序，然后再运行 KafkaUtils2.java 工具类，这样就可以了。

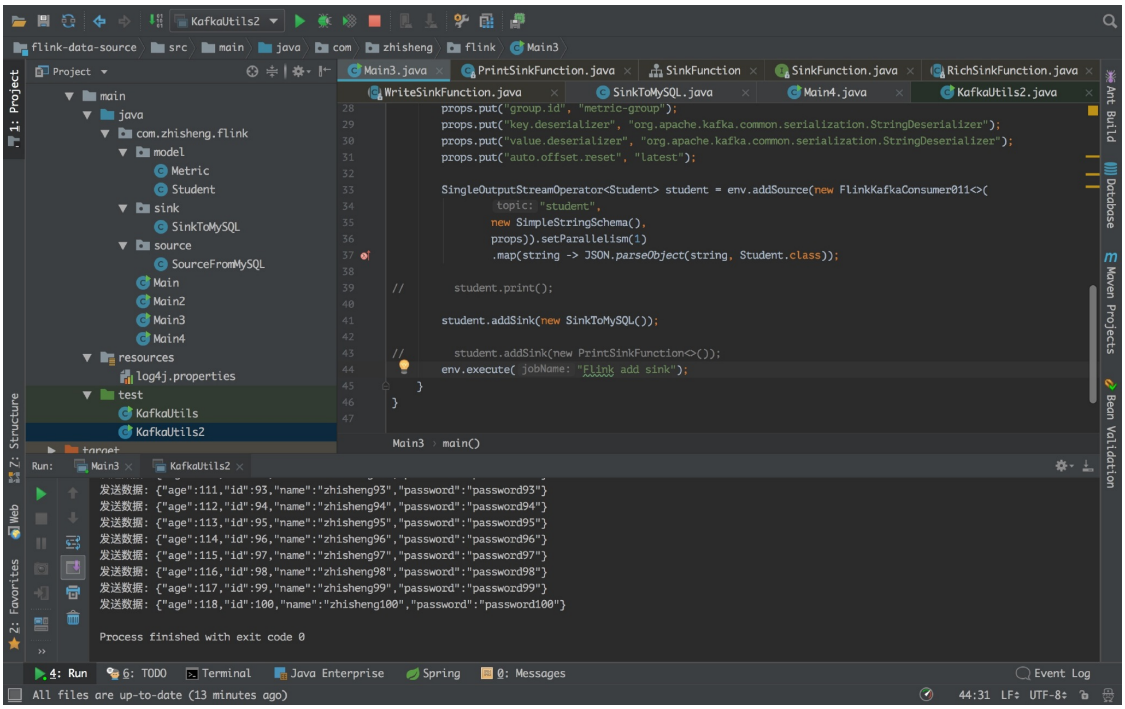
如果数据插入成功了，那么我们查看下我们的数据库：



数据库中已经插入了 100 条我们从 Kafka 发送的数据了。证明我们的 SinkToMySQL 起作用了。是不是很简单?

项目结构

怕大家不知道我的项目结构，这里发个截图看下：



最后

本文主要利用一个 demo，告诉大家如何自定义 Sink Function，将从 Kafka 的数据 Sink 到 MySQL 中，如果你项目中有其他的数据来源，你也可以换成对应的 Source，也有可能你的 Sink 是到其他的地方或者其他不同的方式，那么依旧是这个套路：继承 RichSinkFunction 抽象类，重写 invoke 方法。

关注我

转载请务必注明原创地址为：http://www.54tianzhisheng.cn/2018/10/31/flink-create-sink/

另外我自己整理了些 Flink 的学习资料，目前已经全部放到微信公众号了。你可以加我的微信：zhisheng_tian，然后回复关键字：Flink 即可无条件获取到。

Github 代码仓库

https://github.com/zhisheng17/flink-learning/

以后这个项目的所有代码都将放在这个仓库里，包含了自己学习 flink 的一些 demo 和博客

相关文章

1、《从0到1学习Flink》—— Apache Flink 介绍

2、《从0到1学习Flink》—— Mac 上搭建 Flink 1.6.0 环境并构建运行简单程序入门

3、《从0到1学习Flink》—— Flink 配置文件详解

4、《从0到1学习Flink》—— Data Source 介绍

5、《从0到1学习Flink》—— 如何自定义 Data Source ？

6、《从0到1学习Flink》—— Data Sink 介绍

7、《从0到1学习Flink》—— 如何自定义 Data Sink ？

8、《从0到1学习Flink》—— Flink Data transformation(转换)

9、《从0到1学习Flink》—— 介绍Flink中的Stream Windows

10、《从0到1学习Flink》—— Flink 中的几种 Time 详解

11、《从0到1学习Flink》—— Flink 写入数据到 ElasticSearch