前言

上一篇文章写了 Flink 消费 Kafka 的数据后写入到 ElasticSearch 中，但是在我们身边我们可能还会有其他的小伙伴们有写入 HBase 的需求，这里我将详细地讲解一下 Flink HBase Connector，讲解如何读取 HBase 中的数据和将数据写入到 HBase 中。下面先在本地安装一下 HBase！

HBase 安装

安装命令

终端中输入以下命令：

```
brew install hbase
```

HBase 最终会安装在路径 `/usr/local/Cellar/hbase/` 下面，安装版本不同，文件名也不同。

配置 HBase

打开 `libexec/conf/hbase-env.sh` 修改里面的 JAVA_HOME：

```
# The java implementation to use.  Java 1.7+ required.
export JAVA_HOME="/Library/Java/JavaVirtualMachines/jdk1.8.0_152.jdk/Contents/Home"
```

根据你自己的 JAVA_HOME 来配置这个变量。

打开 `libexec/conf/hbase-site.xml` 配置 HBase 文件存储目录:

```
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <!-- 配置HBase存储文件的目录 -->
    <value>file:///usr/local/var/hbase</value>
  </property>
  <property>
    <name>hbase.zookeeper.property.clientPort</name>
    <value>2181</value>
  </property>
  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <!-- 配置HBase存储内建zookeeper文件的目录 -->
    <value>/usr/local/var/zookeeper</value>
```

```
  </property>
  <property>
    <name>hbase.zookeeper.dns.interface</name>
    <value>lo0</value>
  </property>
  <property>
    <name>hbase.regionserver.dns.interface</name>
    <value>lo0</value>
  </property>
  <property>
    <name>hbase.master.dns.interface</name>
    <value>lo0</value>
  </property>

</configuration>
```

运行 HBase

执行启动的命令：

```
./bin/start-hbase.sh
```

执行后打印出来的日志如：

```
starting master, logging to /usr/local/var/log/hbase/hbase-zhisheng-master-zhisheng.out
```

验证是否安装成功

使用 jps 命令：

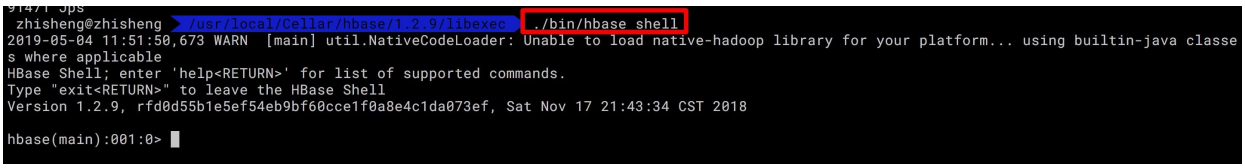
```
zhisheng@zhisheng  /usr/local/Cellar/hbase/1.2.9/libexec  jps
91302 HMaster
62535 RemoteMavenServer
1100
91471 Jps
```

出现 HMaster 说明安装运行成功。

启动 HBase Shell

执行下面命令：

```
./bin/hbase shell
```



停止 HBase

执行下面的命令：

```
./bin/stop-hbase.sh
```

```
hbase(main):001:0> exit
 zhisheng@zhisheng  /usr/local/Cellar/hbase/1.2.9/libexec  ./bin/stop-hbase.sh
stopping hbase...................
 zhisheng@zhisheng  /usr/local/Cellar/hbase/1.2.9/libexec
```

HBase 常用命令

HBase 中常用的命令有：list（列出已存在的表）、create（创建表）、put（写数据）、get（读数据）、scan（读数据，读全表）、describe（显示表详情）

命令	描述	范例
list	显示存在的表	list
create	创建表	create 'zhisheng', 'info'
put	写数据	put 'zhisheng', 'first', 'info:bar', 'hello'
get	读数据	get 'zhisheng', 'first'
scan	读数据(读全表)	scan 'zhisheng'
describe	显示表详情	describe 'zhisheng'

```
hbase(main):003:0> create 'zhisheng', 'info'
0 row(s) in 1.3830 seconds

=> Hbase::Table - zhisheng
hbase(main):004:0> list
TABLE
zhisheng
1 row(s) in 0.0110 seconds

=> ["zhisheng"]
hbase(main):005:0> put 'zhisheng', 'first', 'info:bar', 'hello'
0 row(s) in 0.0930 seconds

hbase(main):006:0> get 'zhisheng', 'first'
COLUMN                                 CELL
 info:bar                              timestamp=1556942396838, value=hello
1 row(s) in 0.0320 seconds

hbase(main):007:0> scan 'zhisheng'
ROW                                    COLUMN+CELL
 first                                 column=info:bar, timestamp=1556942396838, value=hello
1 row(s) in 0.0370 seconds

hbase(main):008:0> describe 'zhisheng'
Table zhisheng is ENABLED
zhisheng
COLUMN FAMILIES DESCRIPTION
{NAME => 'info', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL
 => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
1 row(s) in 0.0500 seconds

hbase(main):009:0>
```

添加依赖

在 pom.xml 中添加 HBase 相关的依赖：

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-hbase_${scala.binary.version}</artifactId>
    <version>${flink.version}</version>
</dependency>
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-common</artifactId>
    <version>2.7.4</version>
</dependency>
```

Flink HBase Connector 中，HBase 不仅可以作为数据源，也还可以写入数据到 HBase 中去，我们先来看看如何从 HBase 中读取数据。

读取 HBase 数据

准备数据

先往 HBase 中插入五条数据如下：

```
put 'zhisheng', 'first', 'info:bar', 'hello'
put 'zhisheng', 'second', 'info:bar', 'zhisheng001'
put 'zhisheng', 'third', 'info:bar', 'zhisheng002'
put 'zhisheng', 'four', 'info:bar', 'zhisheng003'
put 'zhisheng', 'five', 'info:bar', 'zhisheng004'
```

scan 整个 `zhisheng` 表的话，有五条数据：

```
hbase(main):012:0> scan 'zhisheng'
ROW                                    COLUMN+CELL
 first                                 column=info:bar, timestamp=1556942396838, value=hello
 five                                  column=info:bar, timestamp=1556955086302, value=zhisheng004
 four                                  column=info:bar, timestamp=1556955071851, value=zhisheng003
 second                                column=info:bar, timestamp=1556955038391, value=zhisheng001
 third                                 column=info:bar, timestamp=1556955058527, value=zhisheng002
5 row(s) in 0.0210 seconds

hbase(main):013:0>
```

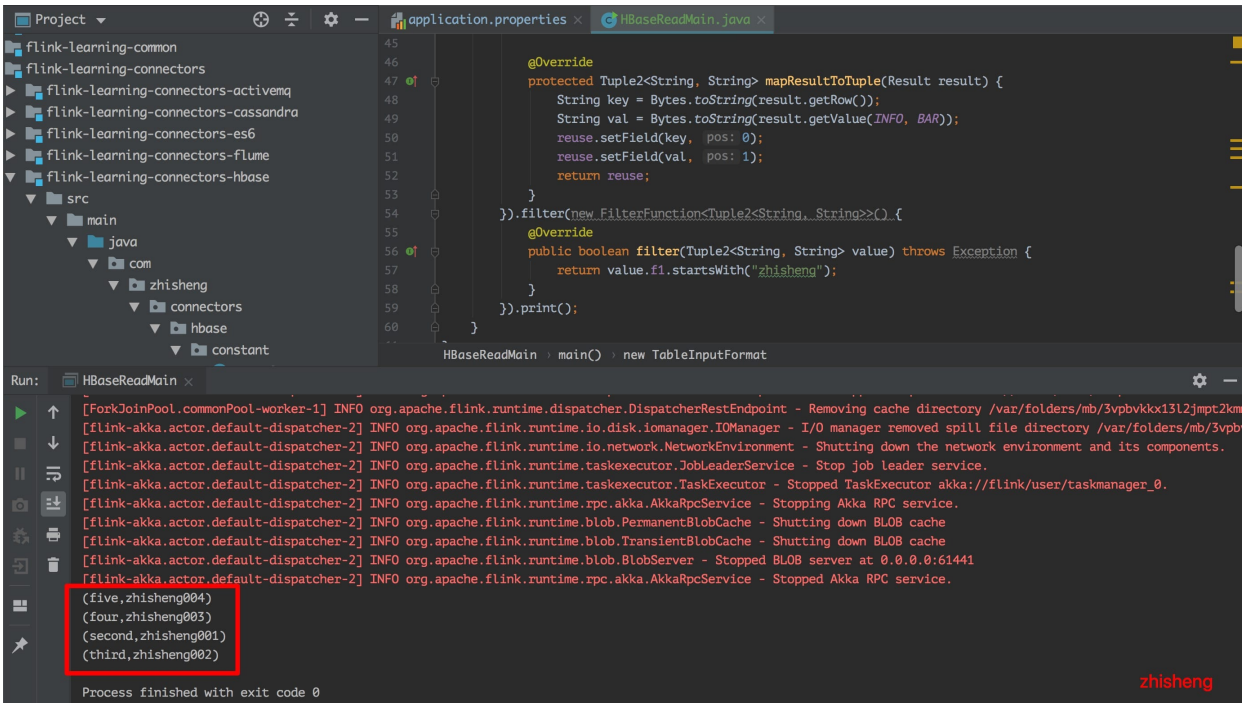
Flink Job 代码

```
import org.apache.flink.addons.hbase.TableInputFormat;
import org.apache.flink.api.common.functions.FilterFunction;
import org.apache.flink.api.java.ExecutionEnvironment;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.configuration.ConfigConstants;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.util.Bytes;

/**
 * Desc: 读取 HBase 数据
```

```
 * blog: http://www.54tianzhisheng.cn/
 * 微信公众号: zhisheng
 */
public class HBaseReadMain {
    //表名
    public static final String HBASE_TABLE_NAME = "zhisheng";
    // 列族
    static final byte[] INFO = "info".getBytes(ConfigConstants.DEFAULT_CHARSET);
    //列名
    static final byte[] BAR = "bar".getBytes(ConfigConstants.DEFAULT_CHARSET);


    public static void main(String[] args) throws Exception {
        ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
        env.createInput(new TableInputFormat<Tuple2<String, String>>() {

            private Tuple2<String, String> reuse = new Tuple2<String, String>();

            @Override
            protected Scan getScanner() {
                Scan scan = new Scan();
                scan.addColumn(INFO, BAR);
                return scan;
            }

            @Override
            protected String getTableName() {
                return HBASE_TABLE_NAME;
            }

            @Override
            protected Tuple2<String, String> mapResultToTuple(Result result) {
                String key = Bytes.toString(result.getRow());
                String val = Bytes.toString(result.getValue(INFO, BAR));
                reuse.setField(key, 0);
                reuse.setField(val, 1);
                return reuse;
            }
        }).filter(new FilterFunction<Tuple2<String, String>>() {
            @Override
            public boolean filter(Tuple2<String, String> value) throws Exception {
                return value.f1.startsWith("zhisheng");
            }
        }).print();
    }
}
```

上面代码中将 HBase 中的读取全部读取出来后然后过滤以 `zhisheng` 开头的 value 数据。

读取结果：



可以看到输出的结果中已经将以 `zhisheng` 开头的四条数据都打印出来了。

写入数据到 HBase

添加依赖

在 pom.xml 中添加依赖：

```
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-mapreduce-client-core</artifactId>
    <version>2.6.0</version>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-hadoop-compatibility_${scala.binary.version}
</artifactId>
    <version>${flink.version}</version>
</dependency>
```

要在 HBase 中提交创建 `zhisheng_sink` 表，并且 Column 为 `info_sink` （如果先运行程序的话是会报错说该表不存在的）：

```
create 'zhisheng_sink', 'info_sink'
```

```
hbase(main):028:0> create 'zhisheng_sink', 'info_sink'
0 row(s) in 1.2540 seconds

=> Hbase::Table - zhisheng_sink
hbase(main):029:0> descibe 'zhisheng_sink'
NoMethodError: undefined method `descibe' for #<Object:0x6c0e13b7>

hbase(main):030:0> describe 'zhisheng_sink'
Table zhisheng_sink is ENABLED

zhisheng_sink

COLUMN FAMILIES DESCRIPTION

{NAME => 'info_sink', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE'
, TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_V
ERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

1 row(s) in 0.0130 seconds

hbase(main):031:0>
```

创建表

zhisheng

Flink Job 代码

接着写 Flink Job 的代码，这里我们将 WordCount 的结果 KV 数据写入到 HBase 中去，代码如下：

```java
import org.apache.flink.api.common.functions.FlatMapFunction;
import org.apache.flink.api.common.functions.RichMapFunction;
import org.apache.flink.api.java.ExecutionEnvironment;
import org.apache.flink.api.java.hadoop.mapreduce.HadoopOutputFormat;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.configuration.ConfigConstants;
import org.apache.flink.configuration.Configuration;
import org.apache.flink.util.Collector;
import org.apache.hadoop.hbase.client.Mutation;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.mapreduce.TableOutputFormat;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;

/**
 * Desc: 写入数据到 HBase
 * blog: http://www.54tianzhisheng.cn/
 * 微信公众号: zhisheng
 */
public class HBaseWriteMain {

    //表名
    public static final String HBASE_TABLE_NAME = "zhisheng_sink";
    // 列族
    static final byte[] INFO = "info_sink".getBytes(ConfigConstants.DEFAULT_CHARSET);
    //列名
    static final byte[] BAR = "bar_sink".getBytes(ConfigConstants.DEFAULT_CHARSET);


    public static void main(String[] args) throws Exception {
        ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();

        Job job = Job.getInstance();
        job.getConfiguration().set(TableOutputFormat.OUTPUT_TABLE, HBASE_TABLE_NAME);

```

```
        env.fromElements(WORDS)
                .flatMap(new FlatMapFunction<String, Tuple2<String, Integer>>() {
                    @Override
                    public void flatMap(String value, Collector<Tuple2<String, Integer>> out) throws Exception {
                        String[] splits = value.toLowerCase().split("\\W+");

                        for (String split : splits) {
                            if (split.length() > 0) {
                                out.collect(new Tuple2<>(split, 1));
                            }
                        }
                    }
                })
                .groupBy(0)
                .sum(1)
                .map(new RichMapFunction<Tuple2<String, Integer>, Tuple2<Text, Mutation>>() {
                    private transient Tuple2<Text, Mutation> reuse;

                    @Override
                    public void open(Configuration parameters) throws Exception {
                        super.open(parameters);
                        reuse = new Tuple2<Text, Mutation>();
                    }

                    @Override
                    public Tuple2<Text, Mutation> map(Tuple2<String, Integer> value) throws Exception {
                        reuse.f0 = new Text(value.f0);
                        Put put = new Put(value.f0.getBytes(ConfigConstants.DEFAULT_CHARSET));
                        put.addColumn(INFO, BAR, Bytes.toBytes(value.f1.toString()));
                        reuse.f1 = put;
                        return reuse;
                    }
                }).output(new HadoopOutputFormat<Text, Mutation>(new TableOutputFormat<Text>(), job));


        env.execute("Flink Connector HBase sink Example");
    }


    private static final String[] WORDS = new String[]{
            "To be, or not to be,--that is the question:--",
            "The fair is be in that orisons"
    };
}
```

运行该 Job 的话，然后再用 HBase shell 命令去验证数据是否插入成功了：

```
hbase(main):032:0> scan 'zhisheng_sink'
ROW                                                    COLUMN+CELL
 be                                                    column=info_sink:bar_sink, timestamp=1556965142316, value=3
 fair                                                  column=info_sink:bar_sink, timestamp=1556965369504, value=1
 in                                                    column=info_sink:bar_sink, timestamp=1556965369512, value=1
 is                                                    column=info_sink:bar_sink, timestamp=1556965142316, value=2
 not                                                   column=info_sink:bar_sink, timestamp=1556965369480, value=1
 or                                                    column=info_sink:bar_sink, timestamp=1556965369512, value=1
 orisons                                               column=info_sink:bar_sink, timestamp=1556965369512, value=1
 question                                              column=info_sink:bar_sink, timestamp=1556965369504, value=1
 that                                                  column=info_sink:bar_sink, timestamp=1556965369504, value=2
 the                                                   column=info_sink:bar_sink, timestamp=1556965369480, value=2
 to                                                    column=info_sink:bar_sink, timestamp=1556965369504, value=2
11 row(s) in 0.0440 seconds
```

可以看见数据已经成功写入了 11 条，然后我们验证一下数据的条数是不是一样的呢?

我们在上面的代码中将 map 和 output 算子给注释掉，然后用上 print 打印出来的话，结果如下：

```
(be,3)
(is,2)
(in,1)
(or,1)
(orisons,1)
(not,1)
(the,2)
(fair,1)
(question,1)
(that,2)
(to,2)
```

统计的结果刚好也是 11 条数据，说明我们的写入过程中没有丢失数据。

但是运行 Job 的话你会看到日志中报了一条这样的错误：

```
java.lang.IllegalArgumentException: Can not create a Path from a null
string
```

```
Run:  HBaseWriteMain
[flink-akka.actor.default-dispatcher-6] INFO org.apache.flink.runtime.executiongraph.ExecutionGraph - DataSink (org.apache.flink.api.java.hadoop.mapreduce.HadoopOutputFormat@4b5d6a01) (1/4) (e64db2e26b0a3402a5f3c5b7420e060
[flink-akka.actor.default-dispatcher-2] INFO org.apache.flink.runtime.executiongraph.ExecutionGraph - Try to restart or fail the job Flink Connector HBase sink Example (23cd6cbc9d157c28f87dd4017f492a04) if no longer possib
[flink-akka.actor.default-dispatcher-2] INFO org.apache.flink.runtime.executiongraph.ExecutionGraph - Job Flink Connector HBase sink Example (23cd6cbc9d157c28f87dd4017f492a04) switched from state FAILING to FAILED.
java.lang.IllegalArgumentException: Can not create a Path from a null string
    at org.apache.hadoop.fs.Path.checkPathArg(Path.java:122)
    at org.apache.hadoop.fs.Path.<init>(Path.java:134)
    at org.apache.flink.api.java.hadoop.mapreduce.HadoopOutputFormatBase.close(HadoopOutputFormatBase.java:175)
    at org.apache.flink.runtime.operators.DataSinkTask.invoke(DataSinkTask.java:224)
    at org.apache.flink.runtime.taskmanager.Task.run(Task.java:711)
    at java.lang.Thread.run(Thread.java:748)
[flink-akka.actor.default-dispatcher-2] INFO org.apache.flink.runtime.executiongraph.ExecutionGraph - Could not restart the job Flink Connector HBase sink Example (23cd6cbc9d157c28f87dd4017f492a04) because the restart stra
java.lang.IllegalArgumentException: Can not create a Path from a null string
    at org.apache.hadoop.fs.Path.checkPathArg(Path.java:122)
    at org.apache.hadoop.fs.Path.<init>(Path.java:134)
    at org.apache.flink.api.java.hadoop.mapreduce.HadoopOutputFormatBase.close(HadoopOutputFormatBase.java:175)
    at org.apache.flink.runtime.operators.DataSinkTask.invoke(DataSinkTask.java:224)
    at org.apache.flink.runtime.taskmanager.Task.run(Task.java:711)
    at java.lang.Thread.run(Thread.java:748)
[flink-akka.actor.default-dispatcher-6] INFO org.apache.flink.runtime.dispatcher.StandaloneDispatcher - Job 23cd6cbc9d157c28f87dd4017f492a04 reached globally terminal state FAILED.
[main] INFO org.apache.flink.runtime.minicluster.MiniCluster - Shutting down Flink Mini Cluster
```

这个问题是因为：

```
Path partitionsPath = new Path(conf.get("mapred.output.dir"), "partitions_"
+ UUID.randomUUID());
```

当配置项 mapred.output.dir 不存在时，conf.get() 将返回 null，从而导致上述异常。

那么该如何解决这个问题呢?

> 需要在代码中或配置文件中添加配置项 mapred.output.dir。

比如在代码里加上这行代码：

```
job.getConfiguration().set("mapred.output.dir", "/tmp");
```

再次运行这个 Job 你就不会发现报错了。

流程序

从上面两个程序中你可以发现两个都是批程序（从 HBase 读取批量的数据、写入批量的数据进 HBase），下面跟着我来演示一个流程序。

读取数据

本来是打算演示从 Kafka 读取 String 类型的数据，但是为了好演示，我这里直接在代码里面造一些数据：

```
DataStream<String> dataStream = env.addSource(new SourceFunction<String>() {
    private static final long serialVersionUID = 1L;

    private volatile boolean isRunning = true;

    @Override
    public void run(SourceContext<String> out) throws Exception {
        while (isRunning) {
            out.collect(String.valueOf(Math.floor(Math.random() * 100)));
        }
    }

    @Override
    public void cancel() {
        isRunning = false;
    }
});
```

如果是读取 Kafka 数据请对应替换成：

```
env.addSource(new FlinkKafkaConsumer011<>(
    parameterTool.get(METRICS_TOPIC),   //这个 kafka topic 需要和上面的工具类的 topic 一致
    new SimpleStringSchema(),
    props));
```

写入数据

然后获取到数据后就需要将数据写入到 HBase，这里使用的实现 HBaseOutputFormat 接口，然后重写里面的 configure、open、writeRecord、close 方法，代码如下：

```
private static class HBaseOutputFormat implements OutputFormat<String> {

    private org.apache.hadoop.conf.Configuration configuration;
```

```java
    private Connection connection = null;
    private String taskNumber = null;
    private Table table = null;
    private int rowNumber = 0;

    @Override
    public void configure(Configuration parameters) {
        configuration = HBaseConfiguration.create();
        configuration.set(HBASE_ZOOKEEPER_QUORUM, ExecutionEnvUtil.PARAMETER_TOOL.get(HBASE_ZOOKEEPER_QUORUM));
        configuration.set(HBASE_ZOOKEEPER_PROPERTY_CLIENTPORT, ExecutionEnvUtil.PARAMETER_TOOL.get(HBASE_ZOOKEEPER_PROPERTY_CLIENTPORT));
        configuration.set(HBASE_RPC_TIMEOUT, ExecutionEnvUtil.PARAMETER_TOOL.get(HBASE_RPC_TIMEOUT));
        configuration.set(HBASE_CLIENT_OPERATION_TIMEOUT, ExecutionEnvUtil.PARAMETER_TOOL.get(HBASE_CLIENT_OPERATION_TIMEOUT));
        configuration.set(HBASE_CLIENT_SCANNER_TIMEOUT_PERIOD, ExecutionEnvUtil.PARAMETER_TOOL.get(HBASE_CLIENT_SCANNER_TIMEOUT_PERIOD));
    }

    @Override
    public void open(int taskNumber, int numTasks) throws IOException {
        connection = ConnectionFactory.createConnection(configuration);
        TableName tableName = TableName.valueOf(ExecutionEnvUtil.PARAMETER_TOOL.get(HBASE_TABLE_NAME));
        Admin admin = connection.getAdmin();
        if (!admin.tableExists(tableName)) { //检查是否有该表，如果没有，创建
            log.info("==============不存在表 = {}", tableName);
                admin.createTable(new HTableDescriptor(TableName.valueOf(ExecutionEnvUtil.PARAMETER_TOOL.get(HBASE_TABLE_NAME)))
                        .addFamily(new HColumnDescriptor(ExecutionEnvUtil.PARAMETER_TOOL.get(HBASE_COLUMN_NAME)));
        }
        table = connection.getTable(tableName);

        this.taskNumber = String.valueOf(taskNumber);
    }

    @Override
    public void writeRecord(String record) throws IOException {
        Put put = new Put(Bytes.toBytes(taskNumber + rowNumber));

 put.addColumn(Bytes.toBytes(ExecutionEnvUtil.PARAMETER_TOOL.get(HBASE_COLUMN_NAME)), Bytes.toBytes("zhisheng"),
                Bytes.toBytes(String.valueOf(rowNumber)));
        rowNumber++;
        table.put(put);
    }

    @Override
    public void close() throws IOException {
        table.close();
        connection.close();
    }
}
```

配置文件

配置文件中的一些配置如下：

```
kafka.brokers=localhost:9092
kafka.group.id=zhisheng
kafka.zookeeper.connect=localhost:2181
metrics.topic=zhisheng
stream.parallelism=4
stream.sink.parallelism=4
stream.default.parallelism=4
stream.checkpoint.interval=1000
stream.checkpoint.enable=false

# HBase
hbase.zookeeper.quorum=localhost:2181
hbase.client.retries.number=1
hbase.master.info.port=-1
hbase.zookeeper.property.clientPort=2081
hbase.rpc.timeout=30000
hbase.client.operation.timeout=30000
hbase.client.scanner.timeout.period=30000

# HBase table name
hbase.table.name=zhisheng_stream
hbase.column.name=info_stream
```

运行项目

运行项目后然后你再去用 HBase shell 命令查看你会发现该 `zhisheng_stream` 表之前没有建立，现在建立了，再通过 scan 命令查看的话，你会发现数据一直在更新，不断增加数据条数。

```
hbase(main):001:0> list
TABLE
hbase.table.name
zhisheng
zhisheng_sink
zhisheng_sink001
zhisheng_stream
zhisheng_stream-1111
6 row(s) in 0.3230 seconds

=> ["hbase.table.name", "zhisheng", "zhisheng_sink", "zhisheng_sink001", "zhisheng_stream", "zhisheng_stream-1111"]
hbase(main):002:0> scan 'zhisheng_stream'
ROW                                                    COLUMN+CELL
 00                                                    column=info_stream:zhisheng, timestamp=1556979677254, value=0
 01                                                    column=info_stream:zhisheng, timestamp=1556979677276, value=1
 010                                                   column=info_stream:zhisheng, timestamp=1556979677339, value=10
 0100                                                  column=info_stream:zhisheng, timestamp=1556979677789, value=100
 01000                                                 column=info_stream:zhisheng, timestamp=1556979679934, value=1000
 010000                                                column=info_stream:zhisheng, timestamp=1556979690097, value=10000
 010001                                                column=info_stream:zhisheng, timestamp=1556979690098, value=10001
 010002                                                column=info_stream:zhisheng, timestamp=1556979690099, value=10002
 010003                                                column=info_stream:zhisheng, timestamp=1556979690099, value=10003
 010004                                                column=info_stream:zhisheng, timestamp=1556979690099, value=10004
 010005                                                column=info_stream:zhisheng, timestamp=1556979690100, value=10005
 010006                                                column=info_stream:zhisheng, timestamp=1556979690101, value=10006
 010007                                                column=info_stream:zhisheng, timestamp=1556979690101, value=10007
 010008                                                column=info_stream:zhisheng, timestamp=1556979690102, value=10008
 010009                                                column=info_stream:zhisheng, timestamp=1556979690103, value=10009
 01001                                                 column=info_stream:zhisheng, timestamp=1556979679937, value=1001
 010010                                                column=info_stream:zhisheng, timestamp=1556979690103, value=10010
 010011                                                column=info_stream:zhisheng, timestamp=1556979690104, value=10011
 010012                                                column=info_stream:zhisheng, timestamp=1556979690104, value=10012
 010013                                                column=info_stream:zhisheng, timestamp=1556979690105, value=10013
 010014                                                column=info_stream:zhisheng, timestamp=1556979690105, value=10014
```

zhisheng

总结

本文一开始讲解了如何在 Mac 上安装 HBase，还简单地介绍了常用的几个 HBase shell 命令，然后通过一个案例从 HBase 中读取数据，又通过一个案例将数据写入到 HBase，这两个案例都是使用 DataSet API，再接着我通过一个流程序讲解了该怎样将数据持续写入 HBase 中，如果你有将数据写入 HBase 相关的需求，我期望本节可以帮助到你。

Github 代码仓库

https://github.com/zhisheng17/flink-learning/tree/master/flink-learning-connectors/flink-learning-connectors-hbase