

实验三

林恺越 181098163

1、下载并安装HBase

安装的是hbase-1.4.13，安装时遇到如下报错：

```
root@lky-0:/# hbase version
Invalid maximum heap size: -Xmx4G
The specified size exceeds the maximum representable size.
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
```

搜索得知：

原因：jvm设置的内存过大，减小配置文件hbase-env.sh内的设置即可。例如：

```
export HBASE_HEAPSIZE=1024
```

完成安装：

```
root@lky-0:/# hbase version
HBase 1.4.13
Source code repository git://Sakthis-MacBook-Pro-2.local/Users/sakthi/dev/hbase
revision=38bf65a22b7e9320f07aeb27677e4533b9a77ef4
Compiled by sakthi on Sun Feb 23 02:06:36 PST 2020
From source with checksum cfb98e5fbeecca2068278ea88175d751b
```

2、单机Standalone模式

基本没遇到问题，运行截图：

```
root@lky-0:~# jps
3666 NameNode
6451 QuorumPeerMain
3811 DataNode
9590 ZooKeeperMain
9782 HMaster
4262 ResourceManager
4025 SecondaryNameNode
4426 NodeManager
9933 HRegionServer
10062 Jps
```

网页：

APACHE

HBASE

[Home](#)[Table Details](#)[Procedures](#)[Local Logs](#)[Log Level](#)[Debug Dump](#)[Metrics Dump](#)[Profiler](#)[HBase Configuration](#)

Master localhost

Region Servers

Base StatsMemoryRequestsStorefilesCompactions

| ServerName | Start time | Last contact | Version | Requests Per Second | Num. Regions |
|-----------------------------|------------------------------|--------------|---------|---------------------|--------------|
| localhost162011605694319728 | Wed Nov 18 18:11:59 CST 2020 | 0 s | 1.4.13 | 0 | 2 |
| Total:1 | | | | 0 | 2 |

Backup Masters

| ServerName | Port | Start Time |
|------------|------|------------|
| Total:0 | | |

Tables

User TablesSystem TablesSnapshots

3、伪分布式模式

单机模式运行过后一会儿发现HMaster挂了，于是安装外置zookeeper

下载的是zookeeper-3.4.13版本，但是配置完运行发现HMaster还是消失得很快，搜索过程中发现一张表：

| | HBase-1.2.x, HBase-1.3.x | HBase-1.4.x | HBase-2.0.x | HBase-2.1.x |
|------------------|--------------------------|-------------|-------------|-------------|
| Hadoop-2.4.x | √ | × | × | × |
| Hadoop-2.5.x | √ | × | × | × |
| Hadoop-2.6.0 | × | × | × | × |
| Hadoop-2.6.1+ | √ | × | √ | × |
| Hadoop的2.7.0 | × | × | × | × |
| Hadoop-2.7.1+ | √ | √ | √ | √ |
| Hadoop-2.8.[0-1] | × | × | × | × |
| Hadoop-2.8.2 | ? | ? | ? | ? |
| Hadoop-2.8.3+ | ? | ? | √ | √ |
| Hadoop-2.9.0 | × | × | × | × |
| Hadoop-2.9.1+ | ? | ? | ? | ? |
| Hadoop-3.0.[0-2] | × | × | × | × |
| Hadoop-3.0.3+ | × | × | √ | √ |
| Hadoop-3.1.0 | × | × | × | × |
| Hadoop-3.1.1+ | × | × | √ | √ |

怀疑是版本不匹配造成的问题，于是更换hbase版本，使用hbase-2.2.5，配置如下：

配置hbase-env.sh：

```
export HBASE_OPTS="$HBASE_OPTS -XX:+UseConcMarkSweepGC"
#true为单机版，false为集群版
export HBASE_MANAGES_ZK=false
#jdk目录
export JAVA_HOME=/usr/java/jdk1.8.0_05
#hadoop 目录
export HADOOP_HOME=/usr/local/hadoop
#hbase目录
export HBASE_HOME=/usr/local/hbase
export HBASE_CLASSPATH=/usr/local/hbase/conf
```

配置hbase-site.xml：

```

<property>
  <name>hbase.rootdir</name>
  <value>hdfs://127.0.0.1:9000/hbase</value>
</property>
<!-- 是否分布式, 设置为true会启动regionserver -->
<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
<!-- HBase master地址 -->
<property>
  <name>hbase.master</name>
  <value>master:60010</value>
</property>
<!-- Zookeeper地址 -->
<property>
  <name>hbase.zookeeper.property.dataDir</name>
  <value>/home/lky/zookeeper/temp</value>
</property>
<!-- Zookeeper集群URL配置 -->
<property>
  <name>hbase.zookeeper.quorum</name>
  <value>master:2181</value>
</property>
<!-- Zookeeper 客户端端口 -->
<property>
  <name>hbase.zookeeper.property.clientPort</name>
  <value>2181</value>
</property>

<property>
  <name>hbase.tmp.dir</name>
  <value>./tmp</value>
</property>
<property>
  <name>hbase.unsafe.stream.capability.enforce</name>
  <value>false</value>
</property>
</configuration>

```

配置Zookeeper:

配置 /etc/profile (截图时太卡所以有截图框的橙色) :

```

export JAVA_HOME=/usr/java/jdk1.8.0_05
export PATH=${JAVA_HOME}/bin:${PATH}
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

export ZOOKEEPER_HOME=/home/lky/zookeeper
export PATH=$PATH:$ZOOKEEPER_HOME/bin
export HBASE_HOME=/usr/local/hbase
export PATH=$PATH:$HBASE_HOME/bin

```

修改zoo_sample.cfg为:

```
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# an example sake.
dataDir=/home/lky/zookeeper/data
dataLogDir=/home/lky/zookeeper/logs
# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
maxClientCnxns=60
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_maintenance
#
# The number of snapshots to retain in dataDir
#autopurge.snapRetainCount=3
# Purge task interval in hours
# Set to "0" to disable auto purge feature
#autopurge.purgeInterval=1
server.0=master:2888:3888
~
~
```

复制一份：

```
root@master:/Zookeeper/conf# cp zoo_sample.cfg zoo.cfg
```

在zookeeper的dataDir内新建一个myid文件，内容为zoo.cfg中server.后面的数字，这里为0：

```
root@master:/Zookeeper/data# echo "0">>myid
```

启动时遇到问题（截图是网上找的，遇到问题的时候没截图）：

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/HBase/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/Hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLogger
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
```

搜索到的解决方法：

Class path contains multiple SLF4J bindings警告。

解决办法

上面提示HBase的slf4j-log4j12-1.7.25.jar包和Hadoop的slf4j-log4j12-1.7.10.jar包起冲突了，因此只需删除其中的一个即可，为了防止误删，这里只是将slf4j-log4j12-1.7.25.jar更改了一下名称

```
mv /HBase/lib/slf4j-log4j12-1.7.25.jar /HBase/lib/slf4j-log4j12-1.7.25.jar_cp
```

之后再启动HBase就不会报警告啦。

启动成功：

```

root@lky-0:/usr/local/hadoop# jps
3874 SecondaryNameNode
5397 QuorumPeerMain
32039 HBaseFsck
3672 DataNode
3528 NameNode
1465 Jps
6634 HRegionServer
7163 ZooKeeperMain
749 Main
302 HMaster

```

没过多久HMaster还是挂了...查询日志发现:

```

-----
java.lang.RuntimeException: HMaster Aborted
    at org.apache.hadoop.hbase.master.HMasterCommandLine.startMaster
(HMasterCommandLine.java:244)
    at org.apache.hadoop.hbase.master.HMasterCommandLine.run(HMasterCommandLine.java:140)
    at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
    at org.apache.hadoop.hbase.util.ServerCommandLine.doMain(ServerCommandLine.java:149)
    at org.apache.hadoop.hbase.master.HMaster.main(HMaster.java:2945)
2020-11-25 14:11:37,949 INFO [main-EventThread] zookeeper.ClientCnxn: EventThread shut down for
session: 0x1000010815e0004

```

根据这个报错查找了很多,但是改来改去都没解决,包括使用zkCli.sh清除hbase的数据等等

```

[zk: localhost:2181(CONNECTED) 0] ls /
[zookeeper, hbase]
[zk: localhost:2181(CONNECTED) 1] rmr / hbase
Command failed: java.lang.IllegalArgumentException: Invalid path string "//zook
eeper" caused by empty node name specified @1
[zk: localhost:2181(CONNECTED) 2]

```

后来在日志里发现了这个:

```

2020-11-25 14:11:35,461 ERROR [master/localhost:16000:becomeActiveMaster] master.HMaster: *****
ABORTING master localhost,16000,1606284689696: Unhandled exception. Starting shutdown. *****
org.apache.hadoop.hbase.util.FileSystemVersionException: hbase.version file is missing. Is your
hbase.rootdir valid? You can restore hbase.version file by running 'HBCK2 filesystem -fix'. See
https://github.com/apache/hbase-operator-tools/tree/master/hbase-hbck2
    at org.apache.hadoop.hbase.util.FSUtils.checkVersion(FSUtils.java:446)

```

它提到了hdfs上的文件,于是决定删除hdfs上的hbase文件夹重新来一遍。

再次启动HMaster就没问题了。

(1) hbase shell:

1、创建表并插入数据:

```

hbase(main):001:0> create 'students','ID','Description','Courses','Home'
Created table students
Took 3.7684 seconds
=> Hbase::Table - students
hbase(main):002:0> put 'students','001','Description:Name','Li Lei'
Took 0.6485 seconds
hbase(main):003:0> put 'students','001','Description:Height','176'
Took 0.0180 seconds
hbase(main):004:0> put 'students','001','Courses:Chinese','80'
Took 0.0228 seconds
hbase(main):005:0> put 'students','001','Courses:Math','90'
Took 0.0103 seconds
hbase(main):006:0> put 'students','001','Courses:Physics','95'
Took 0.0121 seconds
hbase(main):007:0> put 'students','001','Home:Province','Zhejiang'
Took 0.0320 seconds
hbase(main):008:0> put 'students','002','Description:Name','Han Meimei'
Took 0.0089 seconds
hbase(main):009:0> put 'students','002','Description:Height','183'
Took 0.0116 seconds
hbase(main):010:0> put 'students','002','Courses:Chinese','88'
Took 0.0098 seconds
hbase(main):011:0> put 'students','002','Courses:Math','77'

```


创建结果:

```
hbase(main):020:0> describe 'students'
Table students is ENABLED
students
COLUMN FAMILIES DESCRIPTION
{NAME => 'Courses', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
{NAME => 'Description', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
{NAME => 'Home', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
{NAME => 'ID', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
```

2、扫描表:

```
hbase(main):022:0> scan 'students'
COLUMN+CELL
001 column=Courses:Chinese, timestamp=1606286184770, value=80
001 column=Courses:Math, timestamp=1606286213455, value=90
001 column=Courses:Physics, timestamp=1606286228777, value=95
001 column=Description:Height, timestamp=1606286161049, value=176
001 column=Description:Name, timestamp=1606286140481, value=Ll Lei
001 column=Home:Province, timestamp=1606286257284, value=Zhejiang
002 column=Courses:Chinese, timestamp=1606286309429, value=88
002 column=Courses:Math, timestamp=1606286321000, value=77
002 column=Courses:Physics, timestamp=1606286333637, value=66
002 column=Description:Height, timestamp=1606286297034, value=183
002 column=Description:Name, timestamp=1606286280690, value=Han Meimei
002 column=Home:Province, timestamp=1606286354190, value=Beijing
003 column=Courses:Chinese, timestamp=1606286459967, value=90
003 column=Courses:Math, timestamp=1606286470673, value=90
003 column=Courses:Physics, timestamp=1606286481351, value=90
003 column=Description:Height, timestamp=1606286446215, value=162
003 column=Description:Name, timestamp=1606286429116, value=Xiao Ming
003 column=Home:Province, timestamp=1606286496625, value=Shanghai
3 row(s)
Took 0.1516 seconds
```

3、查询学生的家乡信息:

方法一:

```
hbase(main):023:0> get 'students','001','Home:Province'
COLUMN CELL
Home:Province timestamp=1606286257284, value=Zhejiang
1 row(s)
Took 0.0615 seconds
hbase(main):024:0> get 'students','002','Home:Province'
COLUMN CELL
Home:Province timestamp=1606286354190, value=Beijing
1 row(s)
Took 0.0103 seconds
hbase(main):025:0> get 'students','003','Home:Province'
COLUMN CELL
Home:Province timestamp=1606286496625, value=Shanghai
1 row(s)
Took 0.0139 seconds
```

方法二:

```
hbase(main):030:0> scan 'students',{COLUMNS=>'Home'}
COLUMN+CELL
001 column=Home:Province, timestamp=1606286257284, value=Zhejiang
002 column=Home:Province, timestamp=1606286354190, value=Beijing
003 column=Home:Province, timestamp=1606286496625, value=Shanghai
3 row(s)
```

4、插入新列Course: English

```
Took 0.0139 seconds
hbase(main):026:0> put 'students','003','Courses:English','95'
Took 0.0195 seconds
hbase(main):027:0> put 'students','002','Courses:English','85'
Took 0.0091 seconds
hbase(main):028:0> put 'students','001','Courses:English','98'
Took 0.0089 seconds
```

5、插入新列族Contact并添加信息:

```
hbase(main):032:0> alter 'students',NAME=>'Contact',VERSIONS=>2
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 5.0249 seconds
```

```
hbase(main):035:0> put 'students','001','Contact:Email','lilei@qq.com'
Took 0.0088 seconds
hbase(main):036:0> put 'students','002','Contact:Email','hanmeimei@qq.com'
Took 0.0153 seconds
hbase(main):037:0> put 'students','003','Contact:Email','xiaoming@qq.com'
Took 0.0116 seconds
```

6、删除表:

```
Took 0.0204 seconds
hbase(main):040:0> disable 'students'
Took 1.4516 seconds
hbase(main):041:0> drop 'students'
Took 0.9547 seconds
```

(2)java操作hbase

1、创建表并添加数据:

```
public static void creatTable(String tableName, String[] family)
    throws Exception {
    HBaseAdmin admin = new HBaseAdmin(conf);
    HTableDescriptor desc = new HTableDescriptor(tableName);
    for (int i = 0; i < family.length; i++) {
        desc.addFamily(new HColumnDescriptor(family[i]));
    }
    if (admin.tableExists(tableName)) {
        System.out.println("table Exists!");
        System.exit(0);
    } else {
        admin.createTable(desc);
        System.out.println("create table Success!");
    }
}
```

```

public static void addData(String rowKey, String tableName,
    String[] column1, String[] value1, String[] column2, String[] value2, String[] column3, String[] value3)
    throws IOException {
    Put put = new Put(Bytes.toBytes(rowKey)); // 设置rowkey
    HTable table = new HTable(conf, Bytes.toBytes(tableName)); // HTable负责跟记录相关的操作如增删改查等//

    HColumnDescriptor[] columnFamilies = table.getTableDescriptor() // 获取所有的列族
        .getColumnFamilies();

    for (int i = 0; i < columnFamilies.length; i++) {
        String familyName = columnFamilies[i].getNameAsString(); // 获取列族
        if (familyName.equals("Description")) { // Description列族put数据
            for (int j = 0; j < column1.length; j++) {
                put.add(Bytes.toBytes(familyName),
                    Bytes.toBytes(column1[j]), Bytes.toBytes(value1[j]));
            }
        }
        if (familyName.equals("Courses")) { // Courses列族put数据
            for (int j = 0; j < column2.length; j++) {
                put.add(Bytes.toBytes(familyName),
                    Bytes.toBytes(column2[j]), Bytes.toBytes(value2[j]));
            }
        }
        if (familyName.equals("Home")) { // Home列族put数据
            for (int j = 0; j < column3.length; j++) {
                put.add(Bytes.toBytes(familyName),
                    Bytes.toBytes(column3[j]), Bytes.toBytes(value3[j]));
            }
        }
    }
    table.put(put);
    System.out.println("add data Success!");
}

```

```

String tableName = "students";
String[] family = { "ID", "Description", "Courses", "Home" };
creatTable(tableName, family);

// 为表添加数据

String[] column1 = { "Name", "Height" };
String[] value11 = { "Li Lei", "176" };
String[] value12 = { "Han Meimei", "183" };
String[] value13 = { "Xiao Ming", "162" };
String[] column2 = { "Chinese", "Math", "Physics" };
String[] value21 = { "80", "90", "95" };
String[] value22 = { "88", "77", "66" };
String[] value23 = { "90", "90", "90" };
String[] column3 = { "Province" };
String[] value31 = { "Zhejiang" };
String[] value32 = { "Beijing" };
String[] value33 = { "Shanghai" };

addData("001", "students", column1, value11, column2, value21, column3, value31);
addData("002", "students", column1, value12, column2, value22, column3, value32);
addData("003", "students", column1, value13, column2, value23, column3, value33);

//扫描表
scanTable("students");

```

运行结果:

```

create table Success!
add data Success!
add data Success!
add data Success!

```

2、扫描表:


```

public static void scanTable(String tableName) throws IOException {
    System.out.println("*****扫描数据*****");

    //获取数据表对象
    Table table = connection.getTable(TableName.valueOf(tableName));

    //获取表中的数据
    ResultScanner scanner = table.getScanner(new Scan());

    //循环输出表中的数据
    for (Result result : scanner) {

        byte[] row = result.getRow();

        for (Cell cell : result.rawCells()) {
            System.out.println("rowkey=" + new String(CellUtil.cloneRow(cell)) +
                " family=" + new String(CellUtil.cloneFamily(cell)) +
                " column=" + new String(CellUtil.cloneQualifier(cell)) +
                " value=" + new String(CellUtil.cloneValue(cell)) +
                " timestamp=" + cell.getTimestamp());
        }
    }
}

```

```

//扫描表
scanTable("students");

```

运行结果:

```

*****扫描数据*****
rowkey=001 family=Courses column=Chinese value=80 timestamp=1606286470507
rowkey=001 family=Courses column=Math value=90 timestamp=1606286473379
rowkey=001 family=Courses column=Physics value=95 timestamp=1606286476252
rowkey=001 family=Description column=Height value=176 timestamp=1606286479124
rowkey=001 family=Description column=Name value=Li Lei timestamp=1606286481996
rowkey=001 family=Home column=Province value=Zhejiang timestamp=1606286484869
rowkey=002 family=Courses column=Chinese value=88 timestamp=1606286487741
rowkey=002 family=Courses column=Math value=77 timestamp=1606286490613
rowkey=002 family=Courses column=Physics value=66 timestamp=1606286493485
rowkey=002 family=Description column=Height value=183 timestamp=1606286496358
rowkey=002 family=Description column=Name value=Han Meimei timestamp=1606286499230
rowkey=002 family=Home column=Province value=Beijing timestamp=1606286502102
rowkey=003 family=Courses column=Chinese value=90 timestamp=1606286504975
rowkey=003 family=Courses column=Math value=90 timestamp=1606286507847
rowkey=003 family=Courses column=Physics value=90 timestamp=1606286510719
rowkey=003 family=Description column=Height value=162 timestamp=1606286513592
rowkey=003 family=Description column=Name value=Xiao Ming timestamp=1606286516464
rowkey=003 family=Home column=Province value=Shanghai timestamp=1606286519336

```

3、查询学生省份:

```

public static void getResultByColumn(String tableName, String rowKey, String family, String column) throws IOException{
    Table table = connection.getTable(TableName.valueOf(tableName));
    Get get = new Get(rowKey.getBytes());
    get.addColumn(family.getBytes(), column.getBytes());
    Result result = table.get(get);
    for (Cell cell : result.rawCells()) {
        System.out.println("rowkey=" + new String(CellUtil.cloneRow(cell)) +
            " family=" + new String(CellUtil.cloneFamily(cell)) +
            " column=" + new String(CellUtil.cloneQualifier(cell)) +
            " value=" + new String(CellUtil.cloneValue(cell)) +
            " timestamp=" + cell.getTimestamp());
    }
}

```

```
// 查询学生省份
getResultByColumn("students", "001", "Home", "Province");
getResultByColumn("students", "002", "Home", "Province");
getResultByColumn("students", "003", "Home", "Province");
```

运行结果:

| | | | | |
|------------|-------------|-----------------|----------------|-------------------------|
| rowkey=001 | family=Home | column=Province | value=Zhejiang | timestamp=1606286484869 |
| rowkey=002 | family=Home | column=Province | value=Beijing | timestamp=1606286502102 |
| rowkey=003 | family=Home | column=Province | value=Shanghai | timestamp=1606286519336 |

4、添加新列Course: English

```
public static void addNewval(String tableName,String rowKey,
    String family, String column, String value)
    throws IOException {
    Put put = new Put(Bytes.toBytes(rowKey));
    HTable table = new HTable(conf, Bytes.toBytes(tableName));

    HColumnDescriptor[] columnFamilies = table.getTableDescriptor()
        .getColumnFamilies();

    for (int i = 0; i < columnFamilies.length; i++) {
        String familyName = columnFamilies[i].getNameAsString();
        if (familyName.equals(family)) {
            put.add(Bytes.toBytes(familyName),
                Bytes.toBytes(column), Bytes.toBytes(value));
        }
    }
    table.put(put);
    System.out.println("add value to new column Success!");
}
```

```
//插入新列
addNewval("students","001","Courses","English","95");
addNewval("students","002","Courses","English","85");
addNewval("students","003","Courses","English","98");
```

运行结果:

```
add value to new column Success!
add value to new column Success!
add value to new column Success!
```

5、添加新列族和信息

```

public static void addColumnFamily(String tableNameString, String columnFamily) throws IOException {
    TableName tableName = TableName.valueOf(tableNameString);

    //创建列簇对象
    HColumnDescriptor columnDescriptor = new HColumnDescriptor(columnFamily);

    admin.addColumn(tableName, columnDescriptor);

    System.out.println("create new family Success!");
}

```

```

//插入新列
addColumnFamily("students", "Contact") ;

//插入新列
addNewval("students","001","Contact","Email","lilei@qq.com");
addNewval("students","002","Contact","Email","hanmeimei@qq.com");
addNewval("students","003","Contact","Email","xiaominmvg@qq.com");

```

运行结果：

```

create new family Success!
add value to new column Success!
add value to new column Success!
add value to new column Success!

```

6、删除表：

```

public static void deleteTable(String tableName) throws IOException {
    HBaseAdmin admin = new HBaseAdmin(conf);
    admin.disableTable(tableName);
    admin.deleteTable(tableName);
    System.out.println(tableName + "is deleted!");
}

```

```

// 删除表
deleteTable("students");

```

运行结果：

```

students is deleted!

```