hbase基础

实验目的

安装hbase并熟悉常用操作

实验过程

一、安装hbase

- 建立shell脚本，以普通用户身份执行

```
bash hbase_conf.sh
```

- shell脚本

```
#!/bin/bash
# install jdk
cp ~/big_data_tools/hbase1.4.10.tar.gz /apps/
tar -xzvf /apps/hbase1.4.10.tar.gz -C /apps/
rm -rf /apps//hbase1.4.10.tar.gz

# add hbase to path
echo 'HBASE_HOME=/apps/hbase' >> ~/.bashrc
echo 'export PATH=$HBASE_HOME/bin:$HBASE_HOME/sbin:$PATH' >> ~/.bashrc

# config hbase
cd /apps/hbase/conf
echo 'export JAVA_HOME=/apps/java' >> hbase-env.sh
echo 'export HBASE_MANAGES_ZK=true' >> hbase-env.sh
echo 'export HBASE_CLASSPATH=/apps/hbase/conf' >> hbase-env.sh
cp -f ~/big_data_tools/conf_hbase/hbase-site.xml /apps/hbase/conf/
mkdir -p /data/tmp/zookeeper-hbase

sudo reboot
```

- ~/big_data_tools/conf_hbase/hbase-site.xml

```
 <configuration>

 <property>
     <name>hbase.master</name>
     <value>localhost</value>
 </property>
  <property>
      <name>hbase.rootdir</name>
      <value>hdfs://localhost:9000/hbase</value>
  </property>
  <property>
      <name>hbase.cluster.distributed</name>
      <value>true</value>
```

```
  </property>
  <property>
      <name>hbase.zookeeper.quorum</name>
      <value>localhost</value>
  </property>
  <property>
      <name>hbase.zookeeper.property.dataDir</name>
      <value>/data/tmp/zookeeper-hbase</value>
  </property>
</configuration>
```

- hbase.master：HBase 主节点地址。
- hbase.rootdir：HBase 文件在 HDFS 上的存储位置。
- hbase.cluster.distributed：HBase 是否为分布式模式。
- hbase.zookeeper.quorum：配置 ZooKeeper 服务器地址。
- hbase.zookeeper.property.dataDir：HBase 在 ZooKeeper 上存储数据的位置。

- 启动hbase

```
start-all.sh
start-hbase.sh
```

```
chen@ubuntu:~$ jps
1713 NameNode
3427 HMaster
3734 Jps
1911 DataNode
2777 NodeManager
3369 HQuorumPeer
3579 HRegionServer
2156 SecondaryNameNode
2431 ResourceManager
```

二、熟悉hbase常用操作

- 测试hbase,进入hbase shell接口

```
chen@ubuntu:~$ hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/apps/hbase/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/apps/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 1.4.10, r76ab087819fe82ccf6f531096e18ad1bed079651, Wed Jun  5 16:48:11 PDT 2019

hbase(main):001:0>
```

- list查看表

```
hbase(main):001:0> list
TABLE
0 row(s) in 0.1690 seconds

=> []
hbase(main):002:0>
```

- 创建一张表 tb，表中含有一个列簇 mycf。

```
hbase(main):002:0> create 'tb','mycf'
0 row(s) in 1.4090 seconds

=> Hbase::Table - tb
hbase(main):003:0> list
TABLE
tb
1 row(s) in 0.0080 seconds

=> ["tb"]
hbase(main):004:0>
```

- HBase 在 HDFS 上的存储位置是在 hbase-site.xml 设置的，可以使用 HDFS shell 命令进 行查看

```
chen@ubuntu:~$ hadoop fs -ls /
Found 1 items
drwxr-xr-x   - chen supergroup          0 2020-10-25 20:54 /hbase
chen@ubuntu:~$
```

- help

```
hbase(main):002:0> help 'list'
List all user tables in hbase. Optional regular expression parameter could
be used to filter the output. Examples:

  hbase> list
  hbase> list 'abc.*'
  hbase> list 'ns:abc.*'
  hbase> list 'ns:.*'
hbase(main):003:0>
```

general命令

- 查看服务状态

```
hbase(main):003:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 3.0000 average load

hbase(main):004:0>
```

- 查看版本号

```
hbase(main):004:0> version
1.4.10, r76ab087819fe82ccf6f531096e18ad1bed079651, Wed Jun  5 16:48:11 PDT 2019

hbase(main):005:0>
```

DDL操作（data defination language）定义、修改、查询

- 创建一个表，包含两个列族

```
hbase(main):006:0> create 'students','info','address'
0 row(s) in 1.3790 seconds

=> Hbase::Table - students
hbase(main):007:0>
```

- 列出所有表

```
hbase(main):007:0> list
TABLE
students
tb
2 row(s) in 0.0090 seconds

=> ["students", "tb"]
hbase(main):008:0>
```

- 获取表的描述

```
hbase(main):008:0> describe 'students'
Table students is ENABLED
students
COLUMN FAMILIES DESCRIPTION
{NAME => 'address', 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'}
{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'}
2 row(s) in 0.0620 seconds
```

- 删除一个列族

```
hbase(main):001:0> alter 'students',{NAME=>'info',METHOD=>'delete'}
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 2.4380 seconds

hbase(main):002:0> describe 'students'
Table students is ENABLED
students
COLUMN FAMILIES DESCRIPTION
{NAME => 'address', 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 0.0250 seconds

hbase(main):003:0>
```

- 删除一个表，先disable,再drop

```
hbase(main):003:0> disable 'students'
0 row(s) in 2.3250 seconds

hbase(main):004:0> drop 'students'
0 row(s) in 1.2750 seconds

hbase(main):005:0> exists 'students'
Table students does not exist
0 row(s) in 0.0150 seconds

hbase(main):006:0>
```

- 查询表是否可用

```
hbase(main):006:0> is_enabled 'tb'
true
0 row(s) in 0.0140 seconds

hbase(main):007:0>
```

DML操作(data manupulation language)增、删、改、查

- 创建一个具有三个列族(name,address,info)的表students

```
hbase(main):008:0> create 'students','name','address','info'
0 row(s) in 1.2830 seconds

=> Hbase::Table - students
hbase(main):009:0> list
TABLE
students
tb
2 row(s) in 0.0120 seconds

=> ["students", "tb"]
hbase(main):010:0> describe 'students'
Table students is ENABLED
students
COLUMN FAMILIES DESCRIPTION
{NAME => 'address', 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'}
{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'}
{NAME => 'name', 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'}
3 row(s) in 0.0340 seconds
```

- 插入数据（put）

```
hbase(main):011:0> put 'students','xiaoming','address:province','zhejiang'
0 row(s) in 0.1140 seconds

hbase(main):012:0> put 'students','xiaoming','address:city','hangzhou'
0 row(s) in 0.0110 seconds

hbase(main):013:0> put 'students','xiaoming','info:age','20'
0 row(s) in 0.0060 seconds
```

- 获取数据（get）
  - 获取students表的xiaoming行的所有数据

```
hbase(main):015:0> get 'students','xiaoming'
COLUMN                                CELL
 address:city                         timestamp=1603689610393, value=hangzhou
 address:province                     timestamp=1603689579989, value=zhejiang
 info:age                             timestamp=1603689625749, value=20
1 row(s) in 0.0550 seconds
```

  - 获取students表的xiaoming行的address列族的所有数据

```
hbase(main):016:0> get 'students','xiaoming','address'
COLUMN                                CELL
 address:city                         timestamp=1603689610393, value=hangzhou
 address:province                     timestamp=1603689579989, value=zhejiang
1 row(s) in 0.0210 seconds
```

  - 获取students表的xiaoming行的address列族中city列的数据

```
hbase(main):017:0> get 'students','xiaoming','address:city'
COLUMN                                CELL
 address:city                         timestamp=1603689610393, value=hangzhou
1 row(s) in 0.0080 seconds
```

- 更新一条记录（put）
  - 更新students表的xiaoming 行、address列族中province列的值

```
hbase(main):018:0> put 'students','xiaowang','address:city','shanghai'
0 row(s) in 0.0090 seconds
```

  - 查看更新的结果

```
hbase(main):019:0> get 'students','xiaowang','address:city'
COLUMN                                CELL
 address:city                         timestamp=1603690010101, value=shanghai
1 row(s) in 0.0090 seconds
```

- 全表扫描（scan）

```
hbase(main):020:0> scan 'students'
ROW                                   COLUMN+CELL
 xiaoming                             column=address:city, timestamp=1603689610393, value=hangzhou
 xiaoming                             column=address:province, timestamp=1603689579989, value=zhejiang
 xiaoming                             column=info:age, timestamp=1603689625749, value=20
 xiaowang                             column=address:city, timestamp=1603690010101, value=shanghai
2 row(s) in 0.0220 seconds
```

- 删除一列（delete）
  - 删除students表xiaoming行的address列族的列city

```
hbase(main):021:0> delete 'students','xiaoming','address:city'
0 row(s) in 0.0370 seconds
```

  - 查看操作结果

```
hbase(main):024:0> get 'students','xiaoming'
COLUMN                                  CELL
 address:province                       timestamp=1603689579989, value=zhejiang
 info:age                               timestamp=1603689625749, value=20
1 row(s) in 0.0090 seconds
```

- 删除行的所有单元格（deleteall）
    - 使用deleteall命令删除students表xiaoming行的所有列

```
hbase(main):025:0> deleteall 'students','xiaoming'
0 row(s) in 0.0220 seconds
```

- 统计表中的行数

```
hbase(main):026:0> count 'students'
1 row(s) in 0.0340 seconds

=> 1
```

- 清空整张表

```
hbase(main):027:0> truncate 'students'
Truncating 'students' table (it may take a while):
 - Disabling table...
 - Truncating table...
0 row(s) in 3.6970 seconds
```

- 存储多各版本的数据 创建的表，默认列族的 VERSIONS=1，也就是只会存取一个版本的列数据，当再次插入的 时候，后面的值会覆盖前面的值

```
hbase(main):028:0> describe 'students'
Table students is ENABLED
students
COLUMN FAMILIES DESCRIPTION
{NAME => 'address', 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'}
{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'}
{NAME => 'name', 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'}
3 row(s) in 0.0530 seconds
```

- 修改表结构，让表支持存储 3 个本版

```
hbase(main):029:0> alter 'students',{NAME=>'address',VERSIONS=>3}
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 2.1790 seconds
```

```
hbase(main):030:0> describe 'students'
Table students is ENABLED
students
COLUMN FAMILIES DESCRIPTION
{NAME => 'address', BLOOMFILTER => 'ROW', VERSIONS => '3', 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'}
{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'}
{NAME => 'name', 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'}
3 row(s) in 0.0640 seconds
```

- 插入三行数据

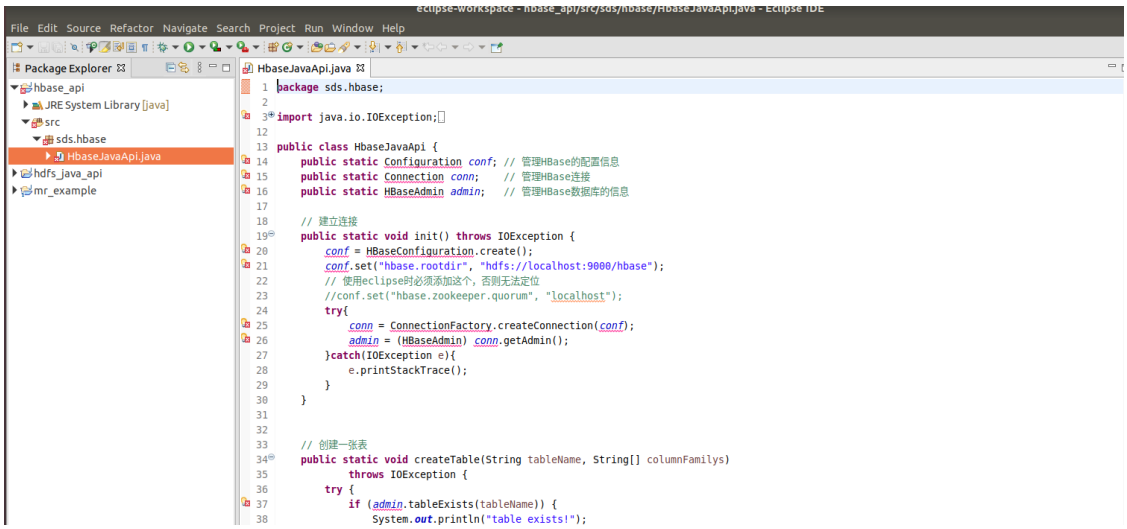
```
hbase(main):031:0> put 'students', 'xiaoming', 'address:city', 'hangzhou'
0 row(s) in 0.1520 seconds

hbase(main):032:0> put 'students', 'xiaoming', 'address:city', 'suzhou'
0 row(s) in 0.0060 seconds

hbase(main):033:0> put 'students', 'xiaoming', 'address:city', 'shanghai'
0 row(s) in 0.0130 seconds
```

- 一次去除三个数据

```
hbase(main):034:0> get 'students','xiaoming', {COLUMN=>'address:city', VERSIONS=>3}
COLUMN                                  CELL
 address:city                           timestamp=1603691437678, value=shanghai
 address:city                           timestamp=1603691431130, value=suzhou
 address:city                           timestamp=1603691426364, value=hangzhou
1 row(s) in 0.0120 seconds
```

三、hbase的API

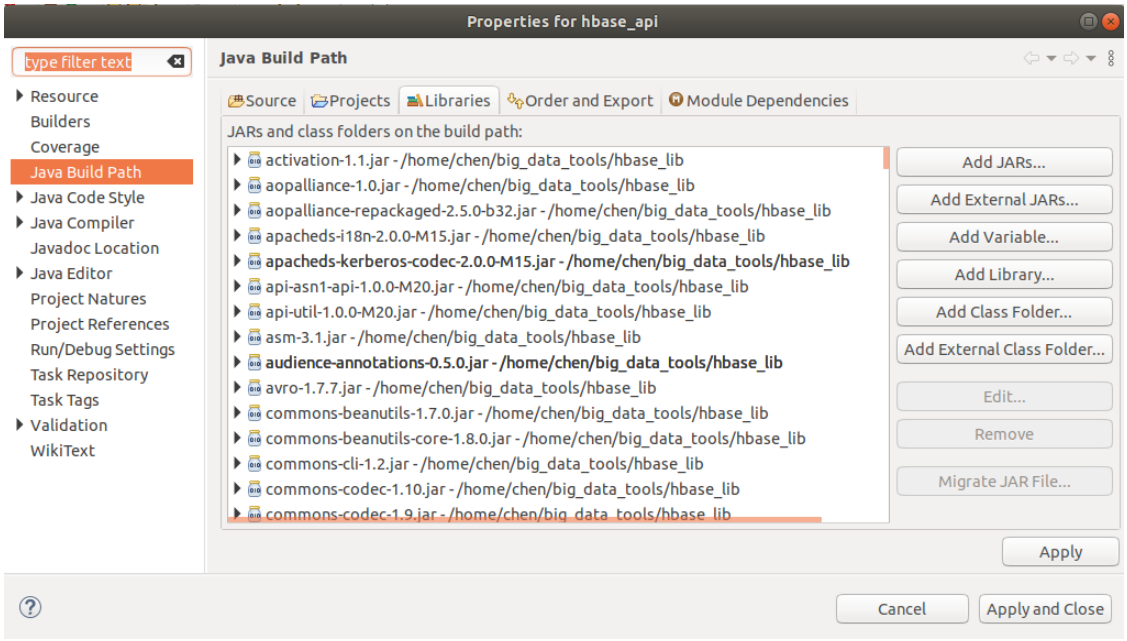
- 新建项目hbase_api



- 将 HBase 目录/apps/hbase/lib 中所 有的 jar 包和/apps/hbase/lib/client-facing-thirdparty 下的所有 jar 包导入到 hbase 的 工程当中。
  - 将以上jar包先导入到~/big_data_tools/hbase_lib/

```
chen@ubuntu:/apps/hbase/lib$ cd client-facing-thirdparty/
chen@ubuntu:/apps/hbase/lib/client-facing-thirdparty$ ls
audience-annotations-0.5.0.jar  findbugs-annotations-1.3.9-1.jar  log4j-1.2.17.jar      slf4j-log4j12-1.7.25.jar
commons-logging-1.2.jar         htrace-core4-4.2.0-incubating.jar  slf4j-api-1.7.25.jar
chen@ubuntu:/apps/hbase/lib/client-facing-thirdparty$ cp *.jar ~/big_data_tools/hbase_lib/
```

  - 导入jar包



- 运行as java

```
<terminated> HbaseJavaApi [Java Application] /apps/java/bin/java (Oct 27, 2020 10:02:38 PM – 10:02:49 PM)
2020-10-27 22:02:42,879 INFO  [main] zookeeper.ZooKeeper (ZooKeeper.java:<init>(438)) - Initiating client connection, connectString=localhost:2181 sessionTimeout=900
2020-10-27 22:02:42,994 INFO  [main-SendThread(localhost:2181)] zookeeper.ClientCnxn (ClientCnxn.java:logStartConnect(1032)) - Opening socket connection to server lo
2020-10-27 22:02:43,031 INFO  [main-SendThread(localhost:2181)] zookeeper.ClientCnxn (ClientCnxn.java:primeConnection(876)) - Socket connection established to localh
2020-10-27 22:02:43,061 INFO  [main-SendThread(localhost:2181)] zookeeper.ClientCnxn (ClientCnxn.java:onConnected(1299)) - Session establishment complete on server l
table exists!
columnfamilys:
address
info
name
put'xiaoming',address:province','zhejiang'
put'xiaoming',address:city','jinhua'
put'xiaoming',info:age','20'
put'xiaowang',address:city','hangzhou'
Get: 20
----------------
rowKey: xiaoming
address:city jinhua
----------------
rowKey: xiaowang
address:city hangzhou
2020-10-27 22:02:46,132 INFO  [main] client.HBaseAdmin (HBaseAdmin.java:call(1380)) - Started disable of students
2020-10-27 22:02:48,451 INFO  [main] client.HBaseAdmin (HBaseAdmin.java:postOperationResult(1409)) - Disabled students
2020-10-27 22:02:49,698 INFO  [main] client.HBaseAdmin (HBaseAdmin.java:postOperationResult(965)) - Deleted students
2020-10-27 22:02:49,698 INFO  [main] client.ConnectionManager$HConnectionImplementation (ConnectionManager.java:closeMasterService(2251)) - Closing master protocol: M
2020-10-27 22:02:49,700 INFO  [main] client.ConnectionManager$HConnectionImplementation (ConnectionManager.java:closeZooKeeperWatcher(1774)) - Closing zookeeper sessi
2020-10-27 22:02:49,704 INFO  [main-EventThread] zookeeper.ClientCnxn (ClientCnxn.java:run(519)) - EventThread shut down for session: 0x1756d84d69b0006
2020-10-27 22:02:49,705 INFO  [main] zookeeper.ZooKeeper (ZooKeeper.java:close(684)) - Session: 0x1756d84d69b0006 closed
```

四、将tsv文件存入hbase

```
hadoop fs -mkdir /input/music/
hadoop fs -put ~/music.txt /input/music/
```

- 调用 HBase 提供的 importtsv 工具在 HBase 上创建表 music, 并指定列族和列。

```
hadoop jar /apps/hbase/lib/hbase-server-1.4.10.jar importtsv -
Dimporttsv.bulk.output=/user/li -
Dimporttsv.columns=HBASE_ROW_KEY,info:name,info:singer,info:gender,info:rygh
me,info:terminal music /input/music/
```

-
```
hbase(main):001:0> list
TABLE
music
tb
2 row(s) in 0.1840 seconds

=> ["music", "tb"]
hbase(main):002:0> scan 'music'
ROW                   COLUMN+CELL
0 row(s) in 0.1190 seconds
```

此时数据还没有存入 HBase，数据暂存在 HDFS 上的/user/chen/tmp

-
```
chen@ubuntu:~$ hadoop fs -ls -R /user
drwxr-xr-x   - chen supergroup          0 2020-10-27 22:56 /user/li
-rw-r--r--   1 chen supergroup          0 2020-10-27 22:56 /user/li/_SUCCESS
drwxr-xr-x   - chen supergroup          0 2020-10-27 22:56 /user/li/info
-rw-r--r--   1 chen supergroup       8799 2020-10-27 22:56 /user/li/info/e52796bff5d94d2b9f6b8d9d53bf4f42
```

- 要把数据存入 HBase，还需要调用 HBase 提供的 completebulkload 工具

```
chen@ubuntu:/data/tmp$ hadoop jar /apps/hbase/lib/hbase-server-1.4.10.j
ar completebulkload /user/li music
```

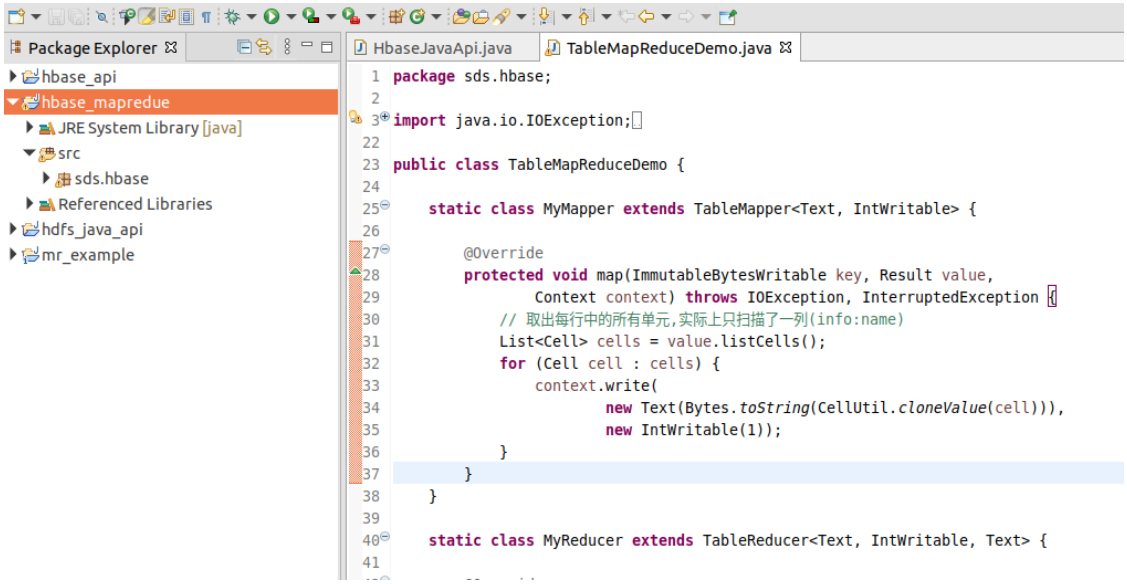
- 查看表中数据

```
hbase(main):002:0> scan 'music'
ROW                                   COLUMN+CELL
 10_song4_2016-1-11                   column=info:gender, timestamp=1603864538591, value=woman
 10_song4_2016-1-11                   column=info:name, timestamp=1603864538591, value=song4
 10_song4_2016-1-11                   column=info:ryghme, timestamp=1603864538591, value=slow
 10_song4_2016-1-11                   column=info:singer, timestamp=1603864538591, value=singer4
```

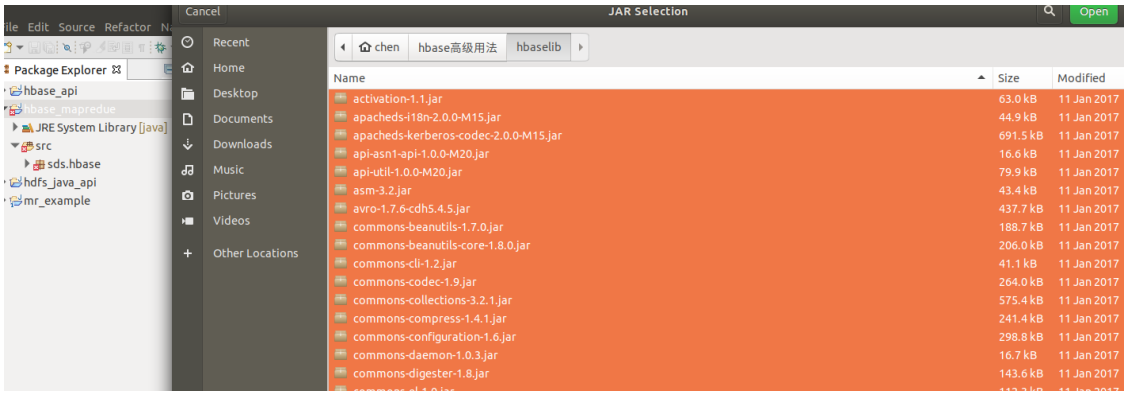
- 创建namelist用于存储mapreduce的结果

```
hbase(main):001:0> create 'namelist','details'
0 row(s) in 1.5530 seconds
```

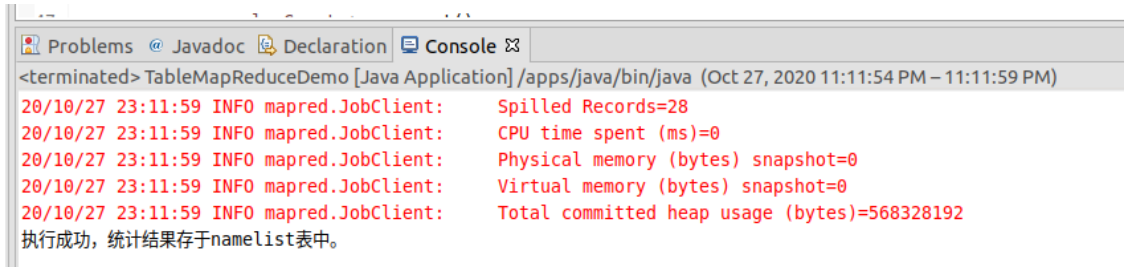
- 新建项目hbase_mapreduce



- 导入新的jar包



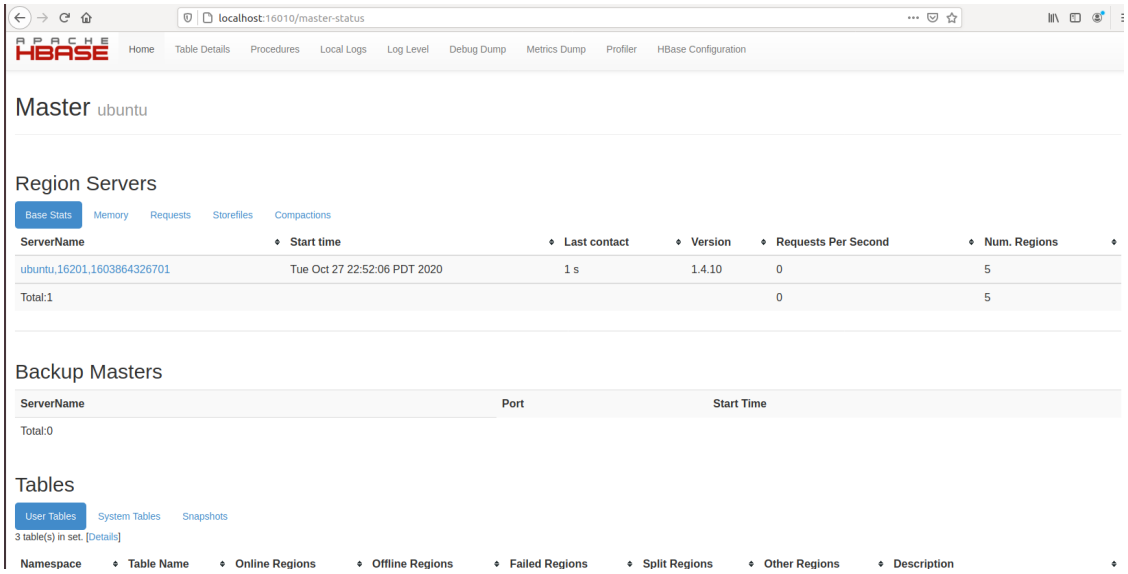
- run as hadoop



- 查看结果

```
hbase(main):001:0> lsit
NameError: undefined local variable or method `lsit' for #<Object:0x5c6a5192>

hbase(main):002:0> list
TABLE
music
namelist
tb
3 row(s) in 0.2500 seconds

=> ["music", "namelist", "tb"]
hbase(main):003:0> scan 'namelist'
ROW                   COLUMN+CELL
 song1                column=details:rank, timestamp=1603865519068, value=\x00\x
                      00\x00\x01
 song2                column=details:rank, timestamp=1603865519068, value=\x00\x
                      00\x00\x02
 song3                column=details:rank, timestamp=1603865519068, value=\x00\x
                      00\x00\x02
 song4                column=details:rank, timestamp=1603865519068, value=\x00\x
                      00\x00\x02
 song5                column=details:rank, timestamp=1603865519068, value=\x00\x
                      00\x00\x01
 song6                column=details:rank, timestamp=1603865519068, value=\x00\x
                      00\x00\x03
 song7                column=details:rank, timestamp=1603865519068, value=\x00\x
                      00\x00\x01
 song8                column=details:rank, timestamp=1603865519068, value=\x00\x
                      00\x00\x01
 song9                column=details:rank, timestamp=1603865519068, value=\x00\x
                      00\x00\x01
9 row(s) in 0.1370 seconds
```

- 浏览器查看localhost:16010


localhost:16010/master-status
APACHE HBASE
Home
Table Details
Procedures
Local Logs
Log Level
Debug Dump
Metrics Dump
Profiler
HBase Configuration
Master ubuntu
Region Servers
Base Stats
Memory
Requests
Storefiles
Compactions
ServerName
Start time
Last contact
Version
Requests Per Second
Num. Regions
ubuntu,16201,1603864326701
Tue Oct 27 22:52:06 PDT 2020
1 s
1.4.10
0
5
Total:1
0
5
Backup Masters
ServerName
Port
Start Time
Total:0
Tables
User Tables
System Tables
Snapshots
3 table(s) in set. [Details]
Namespace
Table Name
Online Regions
Offline Regions
Failed Regions
Split Regions
Other Regions
Description