# Hadoop Developer Training – Lab Hand Book

**Have the following jar files from $HBASE_INSTALL folder plus all jars in the lib folder in the build path:**

**[First create a table in hbase before running this code.** create 'sampleHBaseTable', 'sampleFamily'
**]**

```
package com.evenkat;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.Get;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.util.Bytes;

public class HBaseFirst {

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

// You need a configuration object to tell the client where to connect.
// When you create a HBaseConfiguration, it reads in whatever you've set
// into your hbase-site.xml and in hbase-default.xml, as long as these
// can be found on the CLASSPATH

Configuration config = HBaseConfiguration.create();

// This instantiates an HTable object that connects you to the
// "sampleHBaseTable" table.

HTable table = new HTable(config, "sampleHBaseTable");

// To add to a row, use Put. A Put constructor takes the name of the row
// you want to insert into as a byte array. In HBase, the Bytes class
// has utility for converting all kinds of java types to byte arrays. In
// the below, we are converting the String "row" into a byte
// array to use as a row key for our update. Once you have a Put
// instance, you can adorn it by setting the names of columns you want
// to update on the row, the timestamp to use in your update, etc.
// If no timestamp, the server applies current time to the edits.
```

```
Put p = new Put(Bytes.toBytes("row"));

// To set the value you'd like to update in the row 'row',
// specify the column family, column qualifier, and value of the table
// cell you'd like to update. The column family must already exist
// in your table schema. The qualifier can be anything.
// All must be specified as byte arrays as hbase is all about byte
// arrays. Lets pretend the table 'sampleHBaseTable' was created
// with a family 'sampleFamily.

p.add(Bytes.toBytes("sampleFamily"), Bytes.toBytes("someQualifier"),
Bytes.toBytes("Some Value"));

// Once you've adorned your Put instance with all the updates you want
// to make, to commit it do the following
// (The HTable#put method takes the Put instance you've been building
// and pushes the changes you made into hbase)

table.put(p);

//Now, to retrieve the data we just wrote. The values that come back
//are Result instances. Generally, a Result is an object that will
//package up the hbase return into the form you find most palatable.

Get g = new Get(Bytes.toBytes("row"));
Result r = table.get(g);
byte[] value = r.getValue(Bytes.toBytes("sampleFamily"), Bytes.toBytes("someQualifier"));

//If we convert the value bytes, we should get back 'Some Value', the
//value we inserted at this location.

String valueStr = Bytes.toString(value);
System.out.println("GET: " + valueStr);

//Sometimes, you won't know the row you're looking for. In this case,
//you use a Scanner. This will give you cursor-like interface to the
//contents of the table. To set up a Scanner, do like you did above
//making a Put and a Get, create a Scan. Adorn it with column names,
//etc.

Scan s = new Scan();
s.addColumn(Bytes.toBytes("sampleFamily"), Bytes.toBytes("someQualifier"));
ResultScanner scanner = table.getScanner(s);
try {

//Scanners return Result instances.
//Now, for the actual iteration. One way is to use a while loop
//like so:
```

```
for (Result rr = scanner.next(); rr != null; rr = scanner.next()) {
//print out the row we found and the columns we were looking
//for
        System.out.println("Found row: " + rr);
        }

        // The other approach is to use a foreach loop. Scanners are
        // iterable!
        // for (Result rr : scanner) {
        // System.out.println("Found row: " + rr);
        // }

        } finally {

        // Make sure you close your scanners when you are done!
        // Thats why we have it inside a try/finally clause
        scanner.close();

        }}}
```

**Second Code: In this example even the Table is created programmatically. Focus on the HbaseAdmin class.**

```
package com.evenkat;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.KeyValue;
import org.apache.hadoop.hbase.MasterNotRunningException;
import org.apache.hadoop.hbase.ZooKeeperConnectionException;
import org.apache.hadoop.hbase.client.Delete;
import org.apache.hadoop.hbase.client.Get;
import org.apache.hadoop.hbase.client.HBaseAdmin;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.util.Bytes;
public class HBaseTest {
private static Configuration conf = null;
```

```
/**
* Initialization
*/
static {
conf = HBaseConfiguration.create();
}
/**
* Create a table
*/
public static void createTable(String tableName, String[] familys)
throws Exception {
HBaseAdmin admin = new HBaseAdmin(conf);
if (admin.tableExists(tableName)) {
System.out.println("table already exists!");
} else {
HTableDescriptor tableDesc = new HTableDescriptor(tableName);
for (int i = 0; i < familys.length; i++) {
tableDesc.addFamily(new HColumnDescriptor(familys[i]));
}
admin.createTable(tableDesc);
System.out.println("create table " + tableName + " ok.");
}
}

public static void deleteTable(String tableName) throws Exception {
try {
HBaseAdmin admin = new HBaseAdmin(conf);
admin.disableTable(tableName);
admin.deleteTable(tableName);
System.out.println("delete table " + tableName + " ok.");
} catch (MasterNotRunningException e) {
e.printStackTrace();
} catch (ZooKeeperConnectionException e) {
e.printStackTrace();
}
}
/**
* Put (or insert) a row
*/
public static void addRecord(String tableName, String rowKey,
String family, String qualifier, String value) throws Exception {
try {
HTable table = new HTable(conf, tableName);
Put put = new Put(Bytes.toBytes(rowKey));
put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes
.toBytes(value));
table.put(put);
System.out.println("insert recored " + rowKey + " to table "
```

```
+ tableName + " ok.");
} catch (IOException e) {
e.printStackTrace();
}
}

public static void delRecord(String tableName, String rowKey) throws IOException {
        HTable table = new HTable(conf, tableName);
        List<Delete> list = new ArrayList<Delete>();
        Delete del = new Delete(rowKey.getBytes());
        list.add(del);
        table.delete(list);
        System.out.println("del recored " + rowKey + " ok.");
        }

public static void getOneRecord (String tableName, String rowKey) throws Exception
{
         HTable table = new HTable(conf, tableName);
         Get get = new Get(rowKey.getBytes());
         Result rs = table.get(get);
         for(KeyValue kv : rs.raw()){
         System.out.print(new String(kv.getRow()) + " " );
         System.out.print(new String(kv.getFamily()) + ":" );
         System.out.print(new String(kv.getQualifier()) + " " );
         System.out.print(kv.getTimestamp() + " " );
         System.out.println(new String(kv.getValue()));
}
}

public static void getAllRecord (String tableName) {
         try{
         HTable table = new HTable(conf, tableName);
         Scan s = new Scan();
         ResultScanner ss = table.getScanner(s);
         for(Result r:ss){
         for(KeyValue kv : r.raw()){
         System.out.print(new String(kv.getRow()) + " ");
         System.out.print(new String(kv.getFamily()) + ":");
         System.out.print(new String(kv.getQualifier()) + " ");
         System.out.print(kv.getTimestamp() + " ");
         System.out.println(new String(kv.getValue()));
         }
         }
         } catch (IOException e){
         e.printStackTrace();
         }
}
```

```
public static void main(String[] agrs) {
        try {
        String tablename = "scores";
        String[] familys = { "grade", "course" };
        HBaseTest.createTable(tablename, familys);
        // add record venkat
        HBaseTest.addRecord(tablename, "venkat", "grade", "", "5");
        HBaseTest.addRecord(tablename, "venkat", "course", "", "90");
        HBaseTest.addRecord(tablename, "venkat", "course", "math", "97");
        HBaseTest.addRecord(tablename, "venkat", "course", "art", "87");
        // add record rakesh
        HBaseTest.addRecord(tablename, "rakesh", "grade", "", "4");
        HBaseTest.addRecord(tablename, "rakesh", "course", "math", "89");
        System.out.println("===========get one record========");
        HBaseTest.getOneRecord(tablename, "venkat");
        System.out.println("===========show all record========");
        HBaseTest.getAllRecord(tablename);
        System.out.println("===========del one record========");
        HBaseTest.delRecord(tablename, "rakesh");
        HBaseTest.getAllRecord(tablename);
        System.out.println("===========show all record========");
        HBaseTest.getAllRecord(tablename);
        } catch (Exception e) {
        e.printStackTrace();
        } }}
```

---

```
scan 'sample', {VERSIONS => 3}

get ' sample ', 'row3', {COLUMN => 'cf:c', VERSIONS => 3}
```

for getting the value of a specific time you can use TIMESTAMP as well.

```
get ' sample', 'row3', {COLUMN => 'cf:c', TIMESTAMP =>
1317945301466}
```

if you need to get values "between" 2 timestamps you should use TimestampsFilter.

```
        HTable tbl = new HTable(tableName);
        Get q= new Get(Bytes.toBytes(key));
        q.setMaxVersions(numberOfVersionsYouWant);
         Result row= tbl.get(q);

 NavigableMap<byte[],NavigableMap<byte[],NavigableMap<Long
 ,byte[]>>> allVersions=row.getMap();
```

```
get 'table1', 'rowid', {COLUMN => 'cf:info1', VERSIONS =>
3}
```