目录

使用 API

Maven依赖

HBase 1.x版本客户端对应1.x版本的HBase集群。
HBase 2.x版本客户端对应2.x版本的HBase集群。

HBase 1.x版本

```
<dependency>
  <groupId>org.apache.hbase</groupId>
  <artifactId>hbase-client</artifactId>
  <version>1.4.12</version>
</dependency>
```

HBase 2.x 版本

```
<dependency>
  <groupId>org.apache.hbase</groupId>
  <artifactId>hbase-client</artifactId>
  <version>2.2.3</version>
</dependency>
```

获取配置

获取集群的 ZK 连接地址

进入 HBase 控制台，在连接信息部分查看ZK连接地址，可以看到类似如下的ZooKeeper的连接地址：

连接信息

ZK链接地址: khbase-f942b70b-gn-7bd03637-core-1:2181,khbase-f942b70b-gn-7bd03637-core-2:2181,khbase-f942b70b-gn-7bd03637-core-3:2181

配置ZK地址，连接集群。
将集群的zk地址替换代码中的zk地址，即可完成配置。

```
1. private static final String TABLE_NAME = "mytable";
2. private static final String CF_DEFAULT = "cf";
3. public static final byte[] QUALIFIER = "col1".getBytes();
4. private static final byte[] ROWKEY = "rowkey1".getBytes();
5.
6. public static void main(String[] args) {
7.     Configuration config = HBaseConfiguration.create();
       String zkAddress = "khbase-f942b70b-gn-7bd03637-core-1:2181, khbase-f942b70b-gn-7bd03637-core-2:2181, khbase-f942b70b-gn-7bd0
3637-core-3:2181";
8.     config.set(HConstants.ZOOKEEPER_QUORUM, zkAddress);
9.     Connection connection = null;
```

创建连接

通过配置conf创建Connection

```
connection = ConnectionFactory.createConnection(conf);
```

建立完连接后，即可使用Java API访问HBase集群。
注意：

1. 创建 HBase连接，在程序生命周期内只需创建一次，该连接线程安全，可以共享给所有线程使用。
2. 在程序结束后，需要将Connection对象关闭，否则会造成连接泄露。也可以采用try finally方式防止泄露。
   下面提供一些简单的Java示例。

DDL操作

```
1.try (Admin admin = connection.getAdmin()){
```

3. // 建表

```
4.  HTableDescriptor htd = new HTableDescriptor(TableName.valueOf("tablename"));
5.  htd.addFamily(new HColumnDescriptor(Bytes.toBytes("family")));
6.  // 创建一个只有一个分区的表
7.  // 在生产上建表时建议根据数据特点预先分区
8.  admin.createTable(htd);
9.
10. // disable 表
11. admin.disableTable(TableName.valueOf("tablename"));
12.
13. // truncate 表
14. admin.truncateTable(TableName.valueOf("tablename"), true);
15.
16. // 删除表
17. admin.deleteTable(TableName.valueOf("tablename")); 17.}
```

DML操作

1.//Table 为非线程安全对象，每个线程在对Table操作时，都必须从Connection中获取相应的Table对象 2.try (Table table = connection.getTable(TableName.valueOf("tablename"))) {

```
3.  // 插入数据
4.  Put put = new Put(Bytes.toBytes("row"));
5.  put.addColumn(Bytes.toBytes("family"), Bytes.toBytes("qualifier"), Bytes.toBytes("value"));
6.  table.put(put);
7.
8.  // 单行读取
9.  Get get = new Get(Bytes.toBytes("row"));
10. Result res = table.get(get);
11.
12. // 删除一行数据
13. Delete delete = new Delete(Bytes.toBytes("row"));
14. table.delete(delete);
15.
16. // scan 范围数据
17. Scan scan = new Scan(Bytes.toBytes("startRow"), Bytes.toBytes("endRow"));
18. ResultScanner scanner = table.getScanner(scan);
19. for (Result result : scanner) {
20. // 处理查询结果result
21. // ...
22. }
23. scanner.close(); 24.}
```