Simplehbase v1.0说明文档

Simplehbase代码地址

<https://github.com/zhang-xzhi/simplehbase>

simplehbaseviewer地址

<https://github.com/zhang-xzhi/simplehbaseviewer>

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本 | 日期 | 作者 | 备注 |
| 0.1 | 2014-12-13 | 信之 | 初稿 |
|  |  |  |  |

# Simplehbase 3分钟简介

## simplehbase简介

simplehbase是java和hbase之间的轻量级中间件。 主要包含以下功能。

* 数据类型映射：java类型和hbase的bytes之间的数据转换。
* 简单操作封装：封装了hbase的put,get,scan等操作为简单的java操作方式。
* hbase query封装：封装了hbase的filter，可以使用sql-like的方式操作hbase。
* 动态query封装：类似于myibatis，可以使用xml配置动态语句查询hbase。
* insert,update支持: 建立在hbase的checkAndPut之上。
* hbase多版本支持：提供接口可以对hbase多版本数据进行查询,映射。
* hbase批量操作支持。
* hbase原生接口支持。
* HTablePool管理。
* HTable count和sum功能。

## simplehbase示例(DemoMain.java)

### 使用simplehbaseclient操作hbase

|  |
| --- |
| SimpleHbaseClient simpleHbaseClient = getSimpleHbaseClient();  //insert one record.  Person one = new Person();  one.setId(1);  one.setName("allen");  one.setAge(30);  one.setGender(Gender.MALE);  simpleHbaseClient.putObject(new PersonRowKey(1), one);  //insert another record.  Person two = new Person();  two.setId(2);  two.setName("dan");  two.setAge(31);  two.setGender(Gender.FEMALE);  simpleHbaseClient.putObject(new PersonRowKey(2), two);  //search by row key.  Person result = simpleHbaseClient.findObject(new PersonRowKey(1),  Person.class);  log.info(result);  //search by range.  List<Person> resultList = simpleHbaseClient.findObjectList(  new PersonRowKey(1), new PersonRowKey(3), Person.class);  log.info(resultList);  //HQL query.  Map<String, Object> para = new HashMap<String, Object>();  para.put("id", 1);  resultList = simpleHbaseClient.findObjectList(new PersonRowKey(1),  new PersonRowKey(3), Person.class, "queryById", para);  log.info(resultList);  //dynamic HQL.  para.put("name", "allen");  para.put("age", 0);  resultList = simpleHbaseClient.findObjectList(new PersonRowKey(1),  new PersonRowKey(3), Person.class, "queryByNameAndAge", para);  log.info(resultList);  //batch delete.  simpleHbaseClient.deleteObjectList(new PersonRowKey(0),  new PersonRowKey(100), Person.class); |

### 初始化simplehbase

|  |
| --- |
| HBaseDataSource hbaseDataSource = new HBaseDataSource();  List<Resource> hbaseConfigResources = new ArrayList<Resource>();  //If run on hbase cluster, modify the following config files.  //If run on hbase stand alone mode, comment out the following config files.  hbaseConfigResources.add(new CachedFileSystemResource(  "demo\\hbase\_site"));  hbaseConfigResources.add(new CachedFileSystemResource("demo\\zk\_conf"));  hbaseDataSource.setHbaseConfigResources(hbaseConfigResources);  hbaseDataSource.init();  HBaseTableConfig hbaseTableConfig = new HBaseTableConfig();  //simplehbase config file.  hbaseTableConfig.setConfigResource(new CachedFileSystemResource(  "demo\\myRecord.xml"));  hbaseTableConfig.init();  SimpleHbaseClient tClient = new SimpleHbaseClientImpl();  tClient.setHbaseDataSource(hbaseDataSource);  tClient.setHbaseTableConfig(hbaseTableConfig);  return tClient; |

### simplehbase配置xml

包含htable的配置和2个动态查询的配置

|  |
| --- |
| <SimpleHbase>  <HBaseTableSchema tableName="MyRecordV\_Allen3" defaultFamily="MyRecordFamily">  <HBaseColumnSchema qualifier="id" typeName="int" />  <HBaseColumnSchema qualifier="name" typeName="string" />  <HBaseColumnSchema qualifier="date" typeName="date" />  <HBaseColumnSchema qualifier="gender" typeName="allen.demo.Gender" />  <HBaseColumnSchema qualifier="age" typeName="int" />  </HBaseTableSchema>      <statements>    <statement id="queryByNameAndAge">  select where id greaterequal #id#  <isPropertyAvailable prepend="and" property="name">  name equal #name#  </isPropertyAvailable>  <isPropertyAvailable prepend="and" property="age">  age greater #age#  </isPropertyAvailable>  </statement>    <statement id="queryById">  select where id equal #id#  </statement>    </statements>  </SimpleHbase> |

### 定义DO对象

|  |
| --- |
| public class Person {  private int id;  private String name;  private Date date;  private Gender gender;  private int age;  } |

### 定义该DO对象对应的rowkey

|  |
| --- |
| public class PersonRowKey implements RowKey {    private int row;    public PersonRowKey(int row) {  this.row = row;  }    @Override  public byte[] toBytes() {  return Bytes.toBytes(row);  }  } |

# 核心概念

## simplehbase的设计

* 贴近hbase的设计，提供hbase之上的封装，简化hbase之上java app的开发，需要使用者对hbase有一定了解。
* 做SQL-Like，不做SQL兼容。

## simplehbase的核心概念

* RowKey：对应于htable的rowkey。
* DO: java app data object。
* ResultMapping: 解析hbase查询结果为DO的过程。
* HQL: hbase query language,一套简单的hbase查询语言,SQL-Like。
* Dynamic HQL: 类似于myibatis，可以提供动态拼接HQL的能力。
* SimpleHbaseClient: simplehbase的client入口。提供Layered Service。
* Layered Service: simpleHbase提供多个层次的服务，使用者可以通过simplehbase使用hbase原生服务，也可以通过simplehbase使用结合了DHQL,hbase多版本,ResultMapping的高级服务。

## simplehbase的主要功能

* 数据类型映射：java类型和hbase的bytes之间的数据转换。
* 简单操作封装：封装了hbase的put,get,scan等操作为简单的java操作方式。
* hbase query封装：封装了hbase的filter，可以使用sql-like的方式操作hbase。
* 动态query封装：类似于myibatis，可以使用xml配置动态语句查询hbase。
* insert,update支持: 建立在hbase的checkAndPut之上。
* hbase多版本支持：提供接口可以对hbase多版本数据进行查询,映射。
* hbase批量操作支持。
* hbase原生接口支持。
* HTablePool管理。
* HTable count和sum功能。

## simplehbase的核心技术依赖

* antlr4:定义HQL。
* hbase filter:实现HQL。
* spring resource:配置simplehbase。
* java反射，java annotation。
* hbase coprocessor:实现countAndSum。

# simplehbaseclient 配置

使用simplehbaseclient,需要3个配置。

\* HBaseDataSource配置hbase数据源。

\* HBaseTableConfig配置htable的映射关系，HQL。

\* SimpleHbaseRuntimeSetting配置simplehbase运行时的一些选项和plug-in。

## HBaseDataSource

可以使用java或spring方式配置。

### hbaseConfigResources

其中,Resource为spring标准的resource接口，支持各种类型的配置方式。

|  |
| --- |
| //hbase原生配置资源。如hbase,zk配置文件。  private List<Resource> hbaseConfigResources; |

### htablePoolType(可选配置)

默认使用hbase自带的HTablePool。

值为SimpleHbaseHTablePool时，使用simplehbase自带的SimpleHbaseHTablePool。

SimpleHbaseHTablePool支持定期flush HTable。

### poolMaxSize(可选配置)

默认值为10。

Pool的maxSize，适用于HTablePool和SimpleHbaseHTablePool。

### tableFactory(可选配置)

默认值为Hbase自带的HTableFactory。

可以由client自定义该tableFactory，simplehbase提供SimpleHbaseHTableFactory，可以配置HTable的autoflush和executorService。

适用于HTablePool和SimpleHbaseHTablePool。

### flushInterval(可选配置)

默认不起作用。

当使用SimpleHbaseHTablePool时，设置HTable的flushCommits时间间隔，单位ms。

### htablePoolService(可选配置)

用户可以自定义htablePoolService。

## HbaseTableConfig

Resource configResource 为对应于htable的配置资源，每个htable对应一个HBaseTableConfig。

其中,Resource为spring标准的resource接口，支持各种类型的配置方式。一个简单的配置文件如下:

|  |
| --- |
| <SimpleHbase>  <HBaseTableSchema tableName="MyRecordV\_Allen3" defaultFamily="MyRecordFamily">  <HBaseColumnSchema qualifier="id" typeName="int" />  <HBaseColumnSchema qualifier="name" typeName="string" />  <HBaseColumnSchema qualifier="date" typeName="date" />  <HBaseColumnSchema qualifier="gender" typeName="allen.demo.Gender" />  <HBaseColumnSchema qualifier="age" typeName="int" />  </HBaseTableSchema>    <statements>  <statement id="queryByNameAndAge">  select where id greaterequal #id#  <isPropertyAvailable prepend="and" property="name">  name equal #name#  </isPropertyAvailable>  <isPropertyAvailable prepend="and" property="age">  age greater #age#  </isPropertyAvailable>  </statement>    <statement id="queryById">  select where id equal #id#  </statement>    </statements>  </SimpleHbase> |

### HbaseTableSchema

定义htable的schema。

tableName 必填 表名。

defaultFamily 选填 默认family名。

rowKeyHandlerName 选填 hbase rowkey由bytes转为RowKey Java object的handler类名。默认值为BytesRowKeyHandler。

HBaseColumnSchema中

family

如果指定了defaultFamily，则可以不填该列的family，如果未指定defaultFamily，则必填。

qualifier

列名 必填

typeName

java类型 必填

handler

java类型和bytes之间的数据转换handler，选填，不填则使用默认的handler。

typeName对于普通java类型需要使用全类名。

对于以下类型可以使用缩写。

byte short int long char float double boolean

Byte Short Int Long Char Float Double Boolean

string String date Date

### statements

simplehbase的查询语句。

statement的id，为statement的语句id，必填。

## SimpleHbaseRuntimeSetting(可选配置)

可以配置一些simplehbase运行时配置，如果未配置，则使用系统默认配置。

### scanCachingSize

默认值为20

### deleteBatchSize

默认值为50

### intelligentScanSize

可以配置intelligentScanSize，当intelligentScanSize为true时，框架自动根据QueryExtInfo中startIndex和length来计算scanCachingSize。比如，startIndex为5，length为10，则自动计算出scancachingsize为15。

该选项默认为false。

### literalInterpreterList(可选配置)

字面值解释器列表。

### rowKeyTextFuncList(可选配置)

rowkey生成函数列表。

# 配置相关的一些概念

## TypeHandler

Handler负责转换java对象和hbase中的bytes。

### 默认TypeHandler

当不指定handler时，使用如下默认handler。

|  |
| --- |
| boolean Boolean BooleanHandler 1 byte 保存。  byte Byte ByteHandler 1 byte 保存。  char Character CharacterHandler 2 bytes 保存。  double Double DoubleHandler 8 bytes 保存。  float Float FloatHandler 4 bytes 保存。  int Integer IntegerHandler 4 bytes 保存。  long Long LongHandler 8 bytes 保存。  short Short ShortHandler 2 bytes 保存。  String StringHandler utf-8编码后保存。  Date DateHandler 8 bytes 保存。保存getTime值。  Enum EnumHandler name utf-8编码后保存。 |

### 自定义TypeHandler

可以通过实现TypeHandler来自定义TypeHandler。

推荐直接extends AbstractTypeHandler实现。

TypeHandler定义如下

|  |
| --- |
| /\*\*  \* TypeHandler特定于java的class type,处理和hbase相关的格式转换等工作。  \*  \* <pre>  \* 一般而言，一个TypeHandler处理一个java class。  \* 处理装箱类型可以放到同一个TypeHandler里面。  \* </pre>  \*  \* @author xinzhi  \* @version $Id: TypeHandler.java 2013-09-11 上午11:27:31 xinzhi $  \* \*/  public interface TypeHandler {  /\*\*  \* 转换value为hbase的列值。  \*  \* @param type java的type。  \* @param value java的value。  \* @return hbase的列值。  \* \*/  public byte[] toBytes(Class<?> type, @Nullable Object value);  /\*\*  \* 转换hbase的列值为java的object。  \*  \* @param type java的type。  \* @param bytes hbase的列值。  \* @return java的value。  \* \*/  public Object toObject(Class<?> type, @Nullable byte[] bytes);  } |

## Rowkey生成函数 RowKeyTextFunc

Rowkey生成函数解释一个字符串为bytes，做为rowkey使用。

Rowkey生成函数目前主要为simplehbaseviewer使用。

使用rowkey生成函数，可以方便simplehbaseviewer中语句的编写。

例子： 月账单rowkey设计如下： md5(userId)前4位+userId反转+账单周期反转，难以在查询语句中直接表达复杂的rowkey生成逻辑。

自定义一个rowkey生成函数billCategoryRowKey，配置在simplehbaseviewer中，可以使用如下方式查询。

select \* from billCategory rowkey is billCategoryRowKey ("2088123456781234\_20140401")

目前simplehbase提供以下内置rowkey生成函数。

|  |
| --- |
| intkey( String s ) : 解释s为int后，编码为4字节bytes。  如intkey( "1" ) 将生成 0 0 0 1 4个字节的bytes。  stringkey( String s ) : 使用s的utf-8编码bytes。  如stringkey( "a" ) 将生成 97 1个字节的bytes。  hexkey( String s ) : 解释s为16进制的bytes。  如hexkey ( "00 0A" ) 将生成 0 10 2个字节的bytes。 |

## 字面值解释器 LiteralInterpreter

simplehbase的字面值解释器：

在HQL/DHQL中，需要将形如"abc"之类的字面值解释为java object，用来做各种运算。

如 列a equal "abc"

需要将"abc"解释为java object。

对于以下类型，simplehbase可以使用内置的解释器。

* boolean,byte,char,short,int,long,float,double,
* String,Date,Enum
* HexBytes(bytes的一个封装类)

对于用户自定义的类型，需要定制LiteralInterpreter，配置在simplehbase中。

# hbase query language

目前simplehbase支持以下查询语句。

## SELECT

select where 条件语句

## COUNT

count where 条件语句

## COUNTSUM

countsum (列名列表 逗号分隔) [ where 条件语句 ]

## 条件语句

比较值

可以是常量或者变量形式

二元比较符

列名 二元比较符 比较值

lessequal less greaterequal greater notequal equal

String类型特有的二元比较符

notmatch match

包含操作符

in notin

in ( 常量 , 常量 , 常量 )

in 变量 变量必须为列表

notin ( 常量 , 常量 , 常量 )

notin 变量 变量必须为列表

between操作符

between 常量 and 常量

between 变量 and 变量

notbetween 常量 and 常量

notbetween 变量 and 变量

单目操作符

列名 isnull

列名 isnotnull

列名 ismissing

列名 isnotmissing

逻辑连接符

and or

优先级符

小括号()

## 常量

常量使用字符串外包双引号的形式。如"1","2011-10-10"。

目前支持的常量类型。

String,Boolean,Byte,Short,Character,Integer,Long,Float,Double。

Date 支持格式

yyyy-MM-dd\_HH:mm:ss:SSS

yyyy-MM-dd\_HH:mm:ss

yyyy-MM-dd\_HH:mm

yyyy-MM-dd\_HH

yyyy-MM-dd

Enum支持name的字符串表示。

另外，支持用户自定义常量解释器。

具体请参考SimpleHbaseRuntimeSetting和LiteralInterpreter。

## 变量

变量使用字符串外包#的形式。如#name#,#age#。

由查询时的Map对象传入。#key#自动解析参数map中的value。

## 列名

[列簇名:]列名 列簇名可选，当列名可以唯一确定一个列时，不用指定列簇名。

## 例子

|  |
| --- |
| <statement id="queryById">  select where id equal #id#  </statement> |

# 动态查询

simplehbase支持动态查询，动态查询可以避免应用代码大量if-else之类的判断，拼接查询条件，提高代码质量。

## 单目动态语句

格式

<单目操作符 prepend="prependValue" property="propertyName">

content

</单目操作符>

prependValue为可选项,当条件成立时附加在content之前。

propertyName为操作符比较对象，从查询条件传入的map中获取。

单目操作符目前支持

isNull

isNotNull

isEmpty

isNotEmpty

isPropertyAvailable

isNotPropertyAvailable

例如

<isNotNull prepend="and" property="age">

age less #age#

</isNotNull>

## 双目动态语句

格式

<双目操作符 prepend="prependValue" property="propertyName" compareValue="compareValueConstant">

content

</双目操作符>

prependValue为可选项,当条件成立时附加在content之前。

propertyName为操作符比较对象，从查询条件传入的map中获取。

compareValueConstant为常量。

双目操作符目前支持

isEqual

isNotEqual

isGreaterThan

isGreaterEqual

isLessThan

isLessEqual

例如

<isEqual prepend="and" property="age" compareValue="10">

age less #age#

</isEqual>

## dynamic

格式

<dynamic prepend="prependValue">

content

</dynamic>

prepend为可选项。

当prepend不为空时，会覆盖其子动态条件语句中第一个条件成立的prepend值。

# 配置Data Object

Data Object需要被simpleHbase识别，目前有3种方式来配置JOPO方式。

simplehbase在v0.9（包含v0.9）之前对于POJO的配置，只支持annotation的形式。

## annotation的方式

HBaseTable标记DO类型。

|  |
| --- |
| @Target(ElementType.TYPE)  @Retention(RetentionPolicy.RUNTIME)  public @interface HBaseTable {  /\*\*  \* Default family name.  \*  \* @return default family.  \*/  public String defaultFamily() default "";  } |

HBaseColumn标记DO类型中的属性。

|  |
| --- |
| @Target(ElementType.FIELD)  @Retention(RetentionPolicy.RUNTIME)  public @interface HBaseColumn {  /\*\*  \* Family name.  \*  \* @return family.  \*/  public String family() default "";  /\*\*  \* Qualifier name.  \*  \* @return qualifier.  \*/  public String qualifier();  } |

## xml配置方式

xml配置方式如下：在ConfigResource增加如下xml配置。

|  |
| --- |
| <MappingType className="com.alipay.simplehbase.myrecord.MyRecordV3" defaultFamily="MyRecordFamily">  <field name="id" qualifier="id" family="MyRecordFamily"  isVersioned="false" />  <field name="name" qualifier="name" />  <field name="age" />  <field name="date" />  <field name="gender" />  </MappingType> |

其中

MappingType中

className必填，对应JOPO的classname。

defaultFamily选填。

field中

name必填，对应于JOPO的field name。

qualifier选填，无值时，qualifier为name。

family选填，无值时，使用defaultFamily，defaultFamily为空时，使用qualifier查找对应的family，需要在配置的HTable中，该列名只能在唯一的列簇下。

isVersioned选填，true时表示isVersioned，其他情况为false。

## 无配置方式

有如下2个注意点。

* JOPO中的static field不解析为htable对应的列信息。
* 对于每一个field，对应HTable的列名，在配置的HTable中，该列名只能在唯一的列簇下。

## 3种解析方式的优先级

* 1 xml config方式。
* 2 annotation方式。
* 3 无配置方式

# SimpleHbaseClient的使用

SimpleHbaseClient是simplehbase的client接口,该接口设计为分层服务。

* BasicService：简单的put,delete,scan封装，支持批量操作，支持DHQL，支持数据结果mapping。
* AggregateService:简单的Aggregate功能，支持DHQL，目前支持count,countandsum。
* SimpleHbaseVersionedService:insert/update服务，建立在checkAndPut之上。
* HbaseMultipleVersionService:带hbase version的put，多版本查询,支持DHQL。
* HbaseService：hbase原生服务。

## RowKey

htable的设计中，rowkey设计是一个重要方面，对于rowkey有一个单独的接口。

|  |
| --- |
| /\*\*  \* RowKey对应于hbase的rowkey。  \*  \* @author xinzhi  \* @version $Id: RowKey.java 2013-09-11 上午11:27:31 xinzhi $  \* \*/  public interface RowKey {  /\*\*  \* 转换对象为byte[]的形式。  \*  \* @return rowKey的byte[]形式。  \* \*/  public byte[] toBytes();  } |

## BasicService

* findObject(RowKey, Class<? extends T>)
* findObject(RowKey, Class<? extends T>, QueryExtInfo)
* findObject(RowKey, Class<? extends T>, String, Map)
* findObject(RowKey, Class<? extends T>, String, Map, QueryExtInfo)
* findObjectList(RowKey, RowKey, Class<? extends T>)
* findObjectList(RowKey, RowKey, Class<? extends T>, QueryExtInfo)
* findObjectList(RowKey, RowKey, Class<? extends T>, String, Map)
* findObjectList(RowKey, RowKey, Class<? extends T>, String, Map, QueryExtInfo)
* findObjectAndKey(RowKey, Class<? extends T>)
* findObjectAndKey(RowKey, Class<? extends T>, QueryExtInfo)
* findObjectAndKey(RowKey, Class<? extends T>, String, Map)
* findObjectAndKey(RowKey, Class<? extends T>, String, Map, QueryExtInfo)
* findObjectAndKeyList(RowKey, RowKey, Class<? extends T>)
* findObjectAndKeyList(RowKey, RowKey, Class<? extends T>, QueryExtInfo)
* findObjectAndKeyList(RowKey, RowKey, Class<? extends T>, String, Map)
* findObjectAndKeyList(RowKey, RowKey, Class<? extends T>, String, Map, QueryExtInfo)
* putObject(RowKey, T)
* deleteObject(RowKey, Class<?>)
* deleteObjectList(RowKey, RowKey, Class<?>)

批量操作

* putObjectList(List>)
* findObjectBatch(List, Class<? extends T>)
* findObjectAndKeyBatch(List, Class<? extends T>)
* deleteObjectList(List, Class<?>)

使用SimpleHbaseDOWithKeyResult返回DO和对应的key。

## AggregateService

* count(RowKey, RowKey)
* count(RowKey, RowKey, AggregateExtInfo)
* count(RowKey, RowKey, String, Map)
* count(RowKey, RowKey, String, Map, AggregateExtInfo)
* countAndSum(RowKey, RowKey)
* countAndSum(RowKey, RowKey, AggregateExtInfo)
* countAndSum(RowKey, RowKey, String, Map)
* countAndSum(RowKey, RowKey, String, Map, AggregateExtInfo)

注意：使用countAndSum需要使用simplehbase的coprocessor。

CommonEndpointImpl2

## SimpleHbaseVersionedService

使用@HBaseVersion来标记DO中哪个属性为版本对象。

* insertObject(RowKey, T)
* updateObject(RowKey, T, T)
* updateObjectWithVersion(RowKey, T, Object)

## HbaseMultipleVersionService

* putObjectMV(RowKey, T, long)
* putObjectMV(RowKey, T, Date)
* findObjectMV(RowKey, Class<? extends T>, QueryExtInfo)
* findObjectMV(RowKey, Class<? extends T>, String, Map, QueryExtInfo)
* findObjectListMV(RowKey, RowKey, Class<? extends T>, QueryExtInfo)
* findObjectListMV(RowKey, RowKey, Class<? extends T>, String, Map, QueryExtInfo)
* deleteObjectMV(RowKey, Class<?>, long)
* deleteObjectMV(RowKey, Class<?>, Date)

批量操作

* putObjectListMV(List>, long)
* putObjectListMV(List>, Date)
* putObjectListMV(List>)
* deleteObjectListMV(List, Class<?>, long)
* deleteObjectListMV(List, Class<?>, Date)
* deleteObjectListMV(List, Class<?>)

使用SimpleHbaseDOResult对象封装带版本信息的DO对象。

## HbaseService

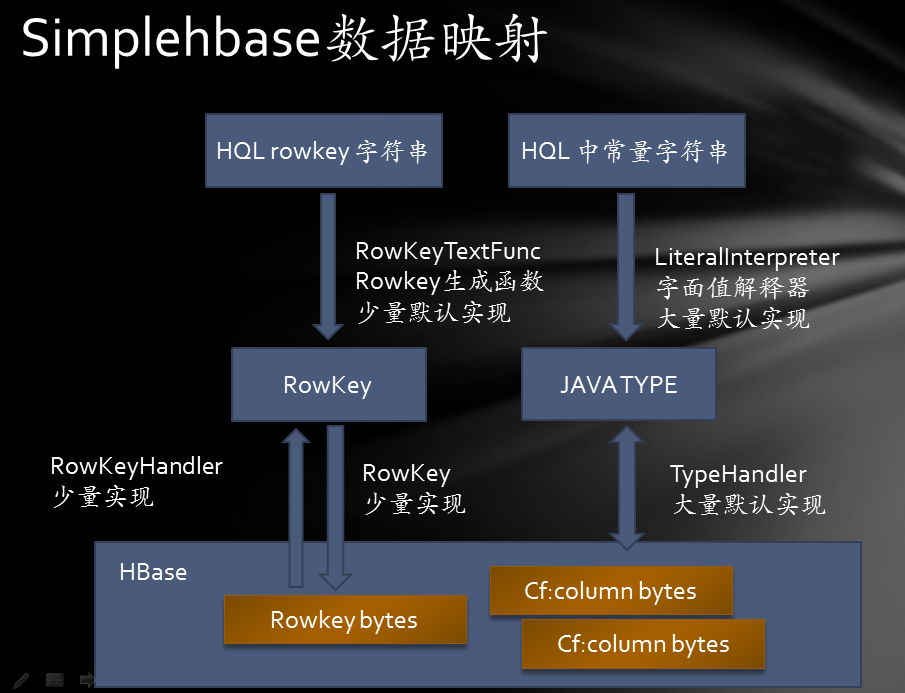
|  |
| --- |
| /\*\*  \* Get a reference to the specified table.  \*  \* @param tableName table name  \* @return a reference to the specified table  \*/  public HTableInterface getTable(String tableName);  /\*\*  \* Get HBaseAdmin.  \*  \* @return HBaseAdmin  \* \*/  public HBaseAdmin getHBaseAdmin(); |

## QueryExtInfo

QueryExtInfo为查询时的附带信息参数。 目前支持版本数,时间戳,scan的startIndex和length。

|  |
| --- |
| public class QueryExtInfo {  private boolean isMaxVersionSet;  private int maxVersions;  private boolean isTimeRangeSet;  private long minStamp;  private long maxStamp;  private boolean isLimitSet;  private long startIndex;  private long length;  } |

# simplehbase数据mapping图



# Test

## test做为样例

大量的test可以做为simplehbase使用样例参考。

## 质量保证

simplehbase编写了大量的test来确保一定的质量。 目前各类测试250+。

## test运行方式

搭建hbase集群。

修改Config文件中的配置(hbase,zk配置)。

运行CreateTestTable建立hbase测试htable。

所有test都是junit测试代码，可以mvn test或者eclipse中运行。

运行DeleteTestTable删除hbase测试htable。

## 文件说明

allen.demo demo

allen.perf some perf test

allen.studyhbase some basic test for hbase

allen.test test config and create test table program

com.alipay.simplehbase.\* test for simplehbase

# simplehbaseviewer简介

simplehbaseviewer是基于simplehbase的web操作界面

<https://github.com/zhang-xzhi/simplehbaseviewer>

# simphbase simplehbaseviewer使用说明

## simplehbase/simplehbaseviewer版本

v1.0

## jdk

jdk/jre 1.6

## maven

maven2

## 代码下载

最新simplehbase代码下载。 <https://github.com/zhang-xzhi/simplehbase> 右下角的download zip。

带版本的simplehbase版本下载。 <https://github.com/zhang-xzhi/simplehbase/releases>

最新simplehbaseviewer代码下载。 <https://github.com/zhang-xzhi/simplehbaseviewer> 右下角的download zip。

带版本的simplehbaseviewer版本下载。 <https://github.com/zhang-xzhi/simplehbaseviewer/releases>

## Simplehbase本地验证

代码下载后，mvn eclipse:eclipse后导入eclipse。 运行simplehbase的test。(Junit)

目前simplehbase默认的测试环境为

|  |
| --- |
| hbase.zookeeper.quorum=10.209.192.33,10.209.192.34,10.209.192.35,10.209.192.36,10.209.192.37  hbase.zookeeper.property.clientPort=2181  zookeeper.znode.parent=/hbase-94 |

* 1 若无法连接，修改test/zk\_conf文件，集成测试使用，修改demo/zk\_conf文件(demo程序使用)。
* 2 建测试表。运行CreateTestTable, 新建测试表。
* 3 建表成功后，重新运行test，后续运行，不需要重新建立测试表。
* 4 新增htable的配置，请参考既有表的配置。

## Simplehbaseviewer本地验证

安装simplehbase到本地仓库。 在simplehbase项目中，mvn install。

simplehbaseviewer代码下载后，mvn eclipse:eclipse后导入eclipse。

运行Main。

访问http://localhost:4040/hbaseviewer/ 看simplehbaseview是否启动正常。

目前simplehbaseviewer默认的测试环境为

|  |
| --- |
| hbase.zookeeper.quorum=10.209.192.33,10.209.192.34,10.209.192.35,10.209.192.36,10.209.192.37  hbase.zookeeper.property.clientPort=2181  zookeeper.znode.parent=/hbase-94 |

* 1 若无法连接，修改config/zk\_conf文件。
* 2 建测试表。运行CreateTestTable, 新建测试表。
* 3 建表成功后，重新运行Main。

## Simplehbase jar包依赖

pom依赖，请参考simplehbase的pom文件。

其中,hadoop和hbase可以修改为目前集团hbase使用的版本。

Simplehbase开发测试过程中使用的hbase版本为0.94.0。

## Simplehbaseclient配置

SimpleHbaseClient为simplehbase的核心接口。

Java方式配置可以参考simplehbase的DemoMain。

Spring方式配置可以参考simplehbaseviewer的 \hbaseviewer\WEB-INF\spring\simplehbase.xml

如何配置可以参考或直接查看代码： <https://github.com/zhang-xzhi/simplehbase/wiki/C02-simplehbaseclient-%E9%85%8D%E7%BD%AE>

## doc

* <https://github.com/zhang-xzhi/simplehbase/wiki>
* <https://github.com/zhang-xzhi/simplehbaseviewer/wiki>
* simplehbase下载后simplehbase\doc

# FAQ

## Simplehbase的兼容性如何？

Simplehbase基于hbase0.94.0开发，尽量使用hbase稳定的接口，但是由于hbase本身各个版本接口不一定兼容，建议使用simplehase时，使用方可以验证simplehbase自带的测试代码，来保证兼容性。

## Simplehbase目前有线上系统使用吗？

Simplehbase从v0.6以来，已经有16+个系统使用，在线上运行。

## 有些功能点simplehbase没有提供，怎么办？

1 可以在github上面提个issue。

2 或者联系开发。联系旺旺:汪信之 Mail:zhang\_xzhi@163.com

3 可以认为simplehbase提供了一套基础设施，自己进行开发。

## 想设置HTable的autoFlush为false，怎么办？

HTable的autoFlush默认为true，想设置为false，请参考HBaseDataSource中的SimpleHbaseHTableFactory。

## 想自定义HTableInterface的实现，怎么办?

请参考HBaseDataSource中的HTablePoolService。

## Scan的时候只想要前几条，有没有什么优化措施？

请参考SimpleHbaseRuntimeSetting的intelligentScanSize。

## 听说批量的put,get性能比较好，simplehbase有没有支持？

支持，请参考BasicService。

## 使用的是定制版的hbase，有没有办法修改运行时的scan请求？

可以，所有涉及到scan的地方都会调用postConstructScan，可以覆盖该方法来修改scan。

## 要做前缀为xyz的scan，如何得到endKey?

参考RowKeyUtil的getEndRowKeyOfPrefix。

## Simplehbase有没有性能测试代码？

参考allen.perf。

# 相关文章

hbase的基本操作

<http://zhang-xzhi-xjtu.iteye.com/blog/1926729>

HBase Client使用注意点

<http://zhang-xzhi-xjtu.iteye.com/blog/2050682>

hbase的CoprocessorProtocol及一个简单的通用扩展实现V2

<http://zhang-xzhi-xjtu.iteye.com/admin/blogs/2163321>

hbase 0.94.0 0.94.9 0.94.24 功能不兼容初步分析

<http://zhang-xzhi-xjtu.iteye.com/admin/blogs/2163258>

Phoenix和simplehbase功能简单比较

<http://zhang-xzhi-xjtu.iteye.com/blog/2040072>