# Simplehbase v0.1

By 张信之

## 修订历史

2013-10-08 V0.1

初稿。

## Simplehbase 3分钟简介

### Simplehbase介绍

简化基于hbase的java应用开发。

### Step by step

S1 定义rowKey。

|  |
| --- |
| public class MyRecordRowKey implements RowKey {  private String row;  public MyRecordRowKey(String row) {  this.row = row;  }  @Override  public byte[] toBytes() {  return Bytes.toBytes(row);  }  @Override  public String getTableName() {  return MyRecordConstants.TableName;  }  } |

S2 定义要存储在hbase的POJO。

|  |
| --- |
| @HBaseTable(defaultFamily = MyRecordConstants.ColumnFamilyName, defaultConvertor = DefaultColumnConvertor.class)  public class MyRecord {  @HBaseColumn(qualifier = "id")  private int id;  @HBaseColumn(qualifier = "name")  private String name;  @HBaseColumn(qualifier = "date")  private Date date;  @HBaseVersion  @HBaseColumn(qualifier = "version")  private long version;  } |

S3 初始化simplehbase

|  |
| --- |
| // simplehbase config.  Map<String, String> configMap = new HashMap<String, String>();  configMap.put(ConfigConstants.HTABLE\_POOL\_SIZE, "5");  SimpleHBaseConfig simpleHBaseConfig = new SimpleHBaseConfig();  simpleHBaseConfig.setSimpleHbaseConfig(configMap);  // init simplehbase.  simpleHBaseConfig.init(); |

S4 获得simpleHbaseClient

|  |
| --- |
| simpleHbaseClient = SimpleHbaseClientFactory.getSimpleHbaseClient(); |

S5 使用simpleHbaseClient操作hbase

|  |
| --- |
| MyRecord myRecord = new MyRecord();  myRecord.setId(0);  myRecord.setName("allen1");  myRecord.setDate(new Date());  myRecord.setVersion(0L);  MyRecordRowKey myRecordRowKey = new MyRecordRowKey("key\_allen1");  simpleHbaseClient.putObject(myRecordRowKey, myRecord); |

## 背景

海量数据处理中，可以使用Hbase做为数据存储。(本文假设读者对hbase有一定了解)

hbase关注数据存储，面向字节。

业务应用系统关注业务模型，基于面向对象思想进行构建。

由于hbase原生API大都是涉及到字节数组操作，因此，不利于快速开发基于hbase的业务应用系统。

Simplehbase做为一个hbase和业务应用系统之间的java中间层，试图简化业务应用系统对hbase的数据操作，简化基于hbase的java应用编程。

## Simplehbase的功能

Simplehbase主要提供以下功能：

1 POJO和hbase的映射。

2 提供client，简化hbase的数据操作。

3 提供强语义的插入和更新操作。

## Simplehhbase初始化

### SimpleHBaseConfig说明

Simplehbase使用SimpleHBaseConfig进行初始化。

关于该类的说明如下：

|  |
| --- |
| SimpleHBaseConfig主要有2类配置项：    hbase原生配置项：可以通过配置文件路径hbaseConfigFilePaths或者本对象的属性hbaseConfig来配置。  当key相同时，hbaseConfig的优先级高于hbaseConfigFilePaths。    simpleHbase私有配置项：可以通过配置文件路径simpleHbaseConfigFilePath或者本对象的属性simpleHbaseConfig来配置。  当key相同时，simpleHbaseConfig的优先级高于simpleHbaseConfigFilePath。    配置文件的格式: 每一行为key=value。    目前使用SimpleHBaseConfig主要有2种方式:  1 使用spring配置。配置时，应该使用init方法初始化simplehbase框架。  2 应用设置SimpleHBaseConfig的属性，调用init方法进行配置。 |

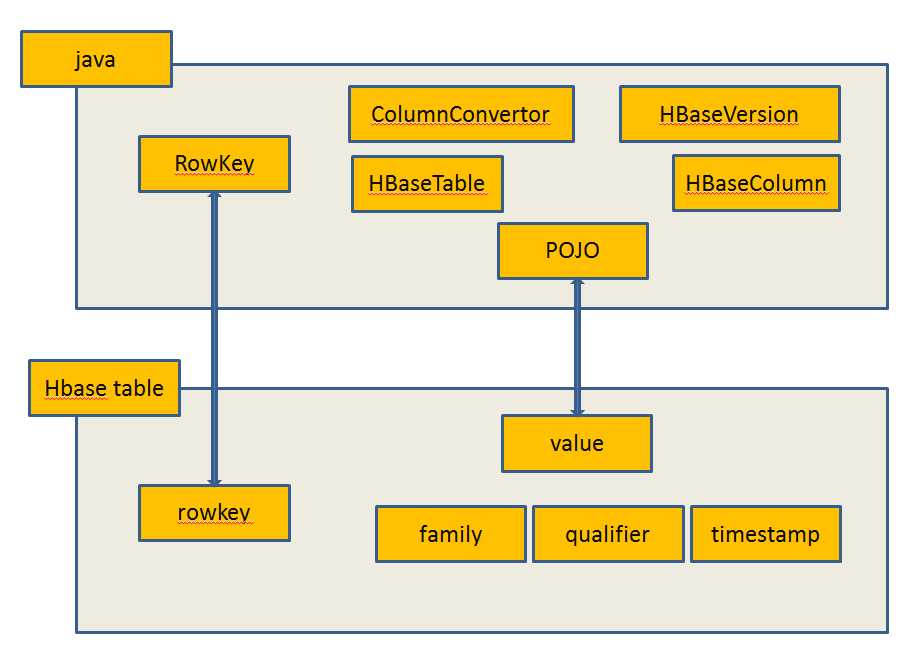
### Simplehbase的私有配置

HTABLE\_POOL\_SIZE

控制HTablePool的大小。大于0。

## Hbase和POJO的映射

### Hbase和POJO的映射



Hbase和POJO的映射，分为2部分。

Simplehbase定义的RowKey对应于Hbase的每一对Key-Value中的Key。

使用simplehbase的注解标记业务系统定义的POJO，对应于Hbase的每一对Key-Value中的Value。

### RowKey的定义

RowKey对应于hbase的rowkey。

|  |
| --- |
| public interface RowKey {  /\*\*  \* 转换对象为byte[]的形式。  \*  \* @return rowKey的byte[]形式。  \* \*/  public byte[] toBytes();  /\*\*  \* 得到该rowKey对应的table。  \*  \* @return rowKey对应的table。  \* \*/  public String getTableName();  } |

### @HBaseTable和@HBaseColumn

@HBaseTable

作用于POJO的type。

可以配置该POJO默认的family和convertor。

@HBaseColumn

作用于POJO的field。

可以配置该POJO属性的family,qualifier,convertor。优先级高于@HBaseTable配置。

### ColumnConvertor

ColumnConvertor定义了Hbase的列和POJO的field值之间的转换器。

|  |
| --- |
| public interface ColumnConvertor {  /\*\*  \* 转换POJO的field为hbase的列值。  \*  \* @param field POJO的Field。  \* @param obj POJO。  \* @return hbase的列值。  \* \*/  public byte[] toBytes(Field field, Object obj);  /\*\*  \* 转换hbase的列值为POJO的field。  \*  \* @param field POJO的Field。  \* @param bytes hbase的列值。  \* @return POJO该field的值。  \* \*/  public Object toObject(Field field, byte[] bytes);  /\*\*  \* 转换value为hbase的列值。  \*  \* @param field POJO的Field。  \* @param value POJO对应field的value。  \* @return hbase的列值。  \* \*/  public byte[] fieldValueToBytes(Field field, Object value);  } |

Simplehbase提供一个默认的转换器DefaultColumnConvertor。

可以处理以下类型：

boolean,Boolean (1 byte 存储)

int,Integer (4 bytes 存储)

long,Long (8 bytes 存储)

String (utf-8 存储)

Date (8 bytes 存储)

定制ColumnConvertor，可以继承AbstractColumnConvertor。

## SimpleHbase的使用

### SimpleHbaseClient

simpleHbase的主要对外接口为SimpleHbaseClient。

|  |
| --- |
| /\*\*  \* 查找rowKey指定的对象。  \*  \* @param rowKey rowKey。  \* @param type POJO type。  \* @return POJO。  \* \*/  public <T> T findObject(RowKey rowKey, Class<? extends T> type);  /\*\*  \* 查找[startRowKey,endRowKey)范围的POJO对象列表。  \*  \* @param startRowKey startRowKey。  \* @param endRowKey endRowKey。  \* @param type POJO type。  \* @return POJO对象列表。  \* \*/  public <T> List<T> findObjectList(RowKey startRowKey, RowKey endRowKey, Class<? extends T> type);  /\*\*  \* Put POJO。  \*  \* @param rowKey rowKey。  \* @param t POJO。  \* \*/  public <T> void putObject(RowKey rowKey, T t);  /\*\*  \* Delete POJO。  \*  \* @param rowKey rowKey。  \* \*/  public void deleteObject(RowKey rowKey); |

### SimpleHbaseClient的获取

方式1：使用SimpleHbaseClientImpl。

方式2：使用SimpleHbaseClientFactory，通过该工厂类的工厂方法，可以给原有的client增加以下功能。

simpleHbase的普通日志。

simpleHbase的性能日志。

simpleHbase的异常转换。

## POJO的版本号

Hbase的数据模型，是典型的key-value模型。

Put操作：无论数据是否存在，都put最新值到hbase。

有时候，业务应用系统有插入和更新的需求。

Insert操作：只有原数据不存在时，才put成功。

Update操作：只有原数据存在时，才put成功。

此时，可以通过封装hbase的checkAndPut来得到类似效果。

由于checkAndPut需要一个value做为检查对象，因此，simplehbase引入一个版本号的概念来解决该问题。

当insert/update一个对象时，应先检查对象是否在hbase中存在，然后决定使用insert还是update。同时，每一次insert/update时，版本号应该递增，避免多线程同时更新同一条数据的情况。

### @HBaseVersion

作用于POJO的field。

用于标示该field为POJO的版本号对象。

标示为HBaseVersion的field必须同时标示为HBaseColumn，否则无效。

1个POJO只能有至多1个field使用该标示。

### SimpleHbaseClient对于版本号的支持

|  |
| --- |
| /\*\*  \* 插入POJO。  \* <pre>  \* rowKey对应的POJO版本为null时(数据不存在)插入成功。  \* </pre>  \*  \* @param rowKey rowKey。  \* @param t POJO  \* @return 是否插入成功。  \* \*/  public <T> boolean insertObject(RowKey rowKey, T t);  /\*\*  \* 更新POJO。  \* <pre>  \* oldT中的版本号为老版本号。  \* newT中的版本号为新版本号。  \* </pre>  \*  \* @param rowKey rowKey。  \* @param oldT oldT。  \* @param newT newT。  \* @return 是否更新成功。  \* \*/  public <T> boolean updateObject(RowKey rowKey, T oldT, T newT);  /\*\*  \* 更新POJO。  \*  \* @param rowKey rowKey。  \* @param t POJO。  \* @param oldVersion 老版本。  \* @return 是否更新成功。  \* \*/  public <T> boolean updateObjectWithVersion(RowKey rowKey, T t, Object oldVersion); |

## 异常

Simplehbase自身抛出的异常都为SimpleHbaseException。

## 日志

Simplehbase使用log4j。

## TODO

1 支持xml配置hbase和POJO的映射关系。

2 支持带条件和limit的查询。

3 支持聚集函数。

4 支持get时，只get所需的列。