

# 三、HBase 程式設計

3.1 程式編譯方法

3.2 Coding & API

3.2.1 常用的 API 說明

3.2.2 實做 I/O 操作

3.2.3 搭配Map Reduce 運算

3.2.4 進階補充

3.3 案例演練

# 三、HBase 程式設計

## 3.1 HBase 程式編譯方法

此篇介紹兩種編譯與執行HBase程式的方法：

Method A – 使用Java JDK 1.6

Method B – 使用Eclipse 套件

# A. Java 之編譯與執行

1. 將hbase\_home目錄內的.jar檔全部拷貝至hadoop\_home/lib/ 資料夾內

2. 編譯

◆ javac  $\Delta$  -classpath  $\Delta$  **hadoop-\*-core.jar:hbase-\*.jar**  $\Delta$  -d  $\Delta$   
MyJava  $\Delta$  MyCode.java

3. 封裝

◆ jar  $\Delta$  -cvf  $\Delta$  MyJar.jar  $\Delta$  -C  $\Delta$  MyJava  $\Delta$  .

4. 執行

◆ bin/hadoop  $\Delta$  jar  $\Delta$  MyJar.jar  $\Delta$  MyCode  $\Delta$  {Input/  $\Delta$  Output/ }

---

• 所在的執行目錄為Hadoop\_Home

• ./MyJava = 編譯後程式碼目錄

• My jar. jar = 封裝後的編譯檔

• 先放些文件檔到HDFS上的input目錄

• ./input; ./ouput 不一定為 hdfs的輸入、輸出目錄

# B Eclipse 之編譯與執行

- HBase 已可以於Hadoop上正常運作
- 請先設定好Eclipse 上得 Hadoop 開發環境
  - ◆ 可參考附錄
  - ◆ Hadoop更詳細說明請參考另一篇 Hadoop 0.20 程式設計
- 建立一個hadoop的專案

# B.1 設定專案的細部屬性

1

在建立好的專案上點選右鍵，並選擇properties

```
package tsmc;

import java.io.BufferedReader;

// b.txt 的檔案格式如：
// T01;1;1;1;1
// T02;2;2
// T03;3;3;3

import java.io.*;

public class LoadFromFile {

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

        File file_in = new File("b.txt");
        Reader fi = new BufferedReader(
            new FileReader(new File(file_in)));

        String line;
        while ((line = fi.readLine()) != null) {

            String str = line.split(";");
            int length = str.length - 1;

            for (int i = 0; i < length; i++) {
                PutData.putData(table_name, str[0], family, "P" +
                    str[(i + 1)]);
            }

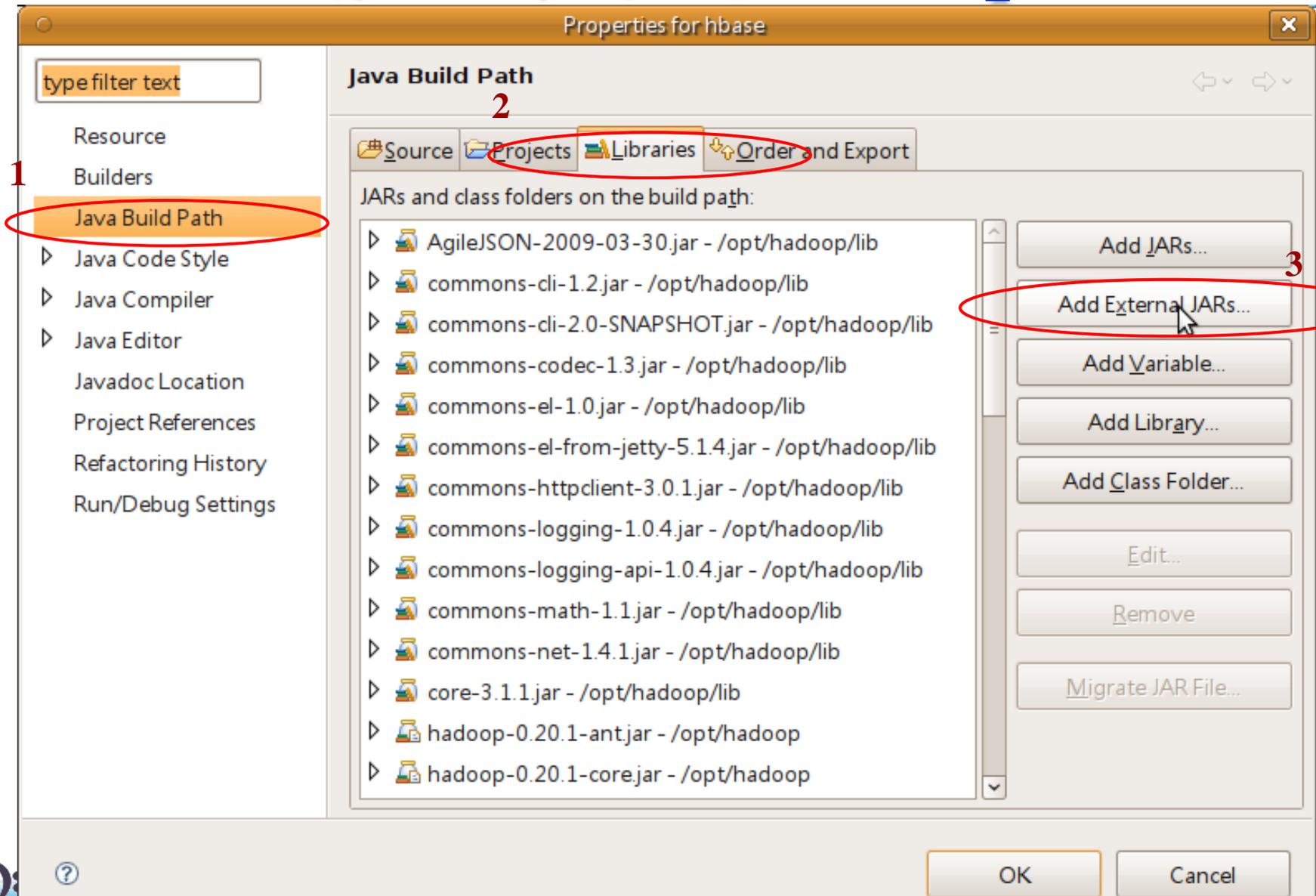
            System.out.println();
        }

        fi.close();
    }

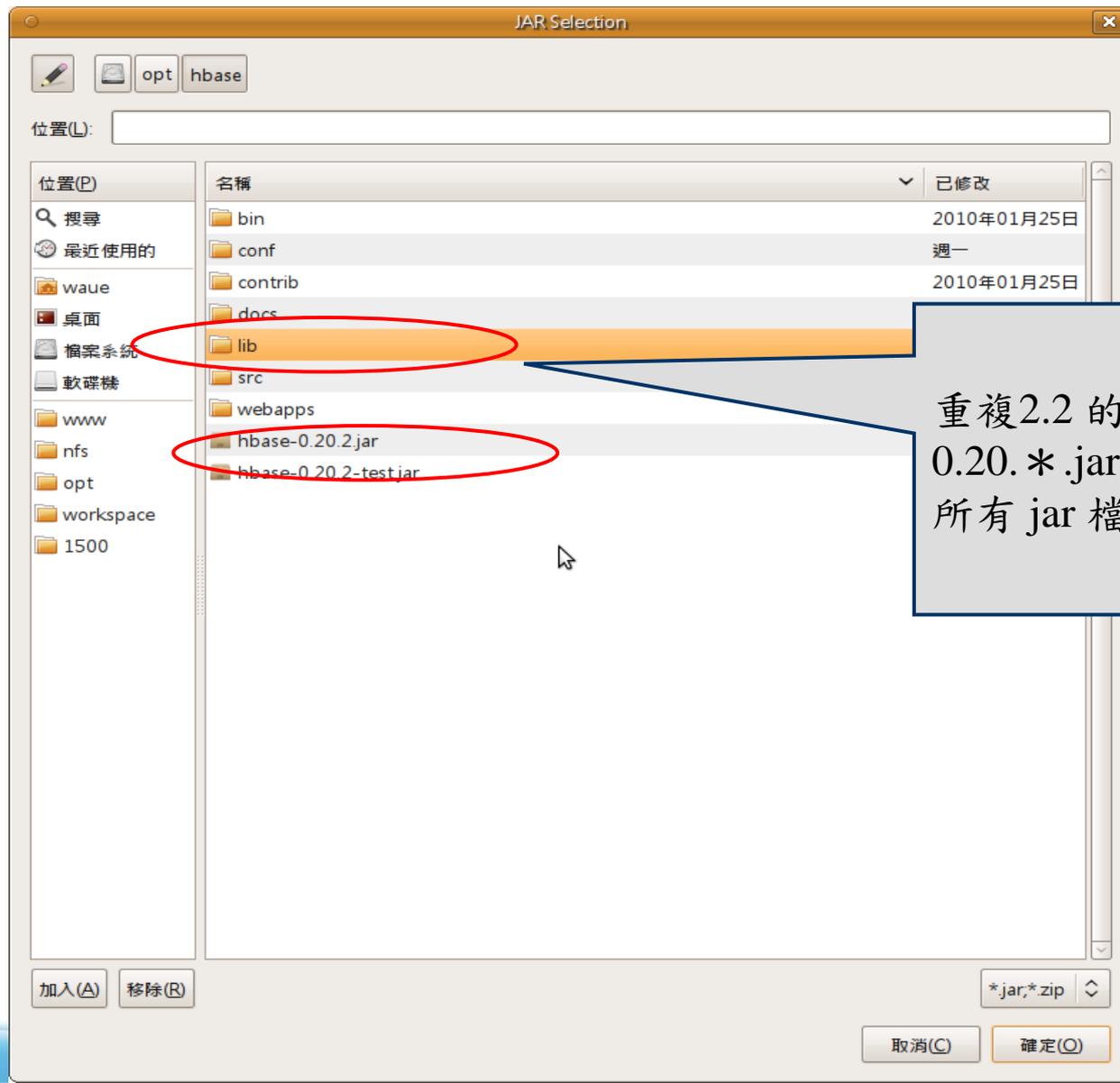
    private void main(String args[]) throws IOException {
        // ...
    }
}
```

10/02/01 19:57:14 INFO zookeeper.ZooKeeper: Client environment:host.name=vpro  
10/02/01 19:57:14 INFO zookeeper.ZooKeeper: Client environment:java.version=1.6.0\_16  
10/02/01 19:57:14 INFO zookeeper.ZooKeeper: Client environment:java.vendor=Sun Microsystems Inc.

# B.2 增加專案的 Classpath

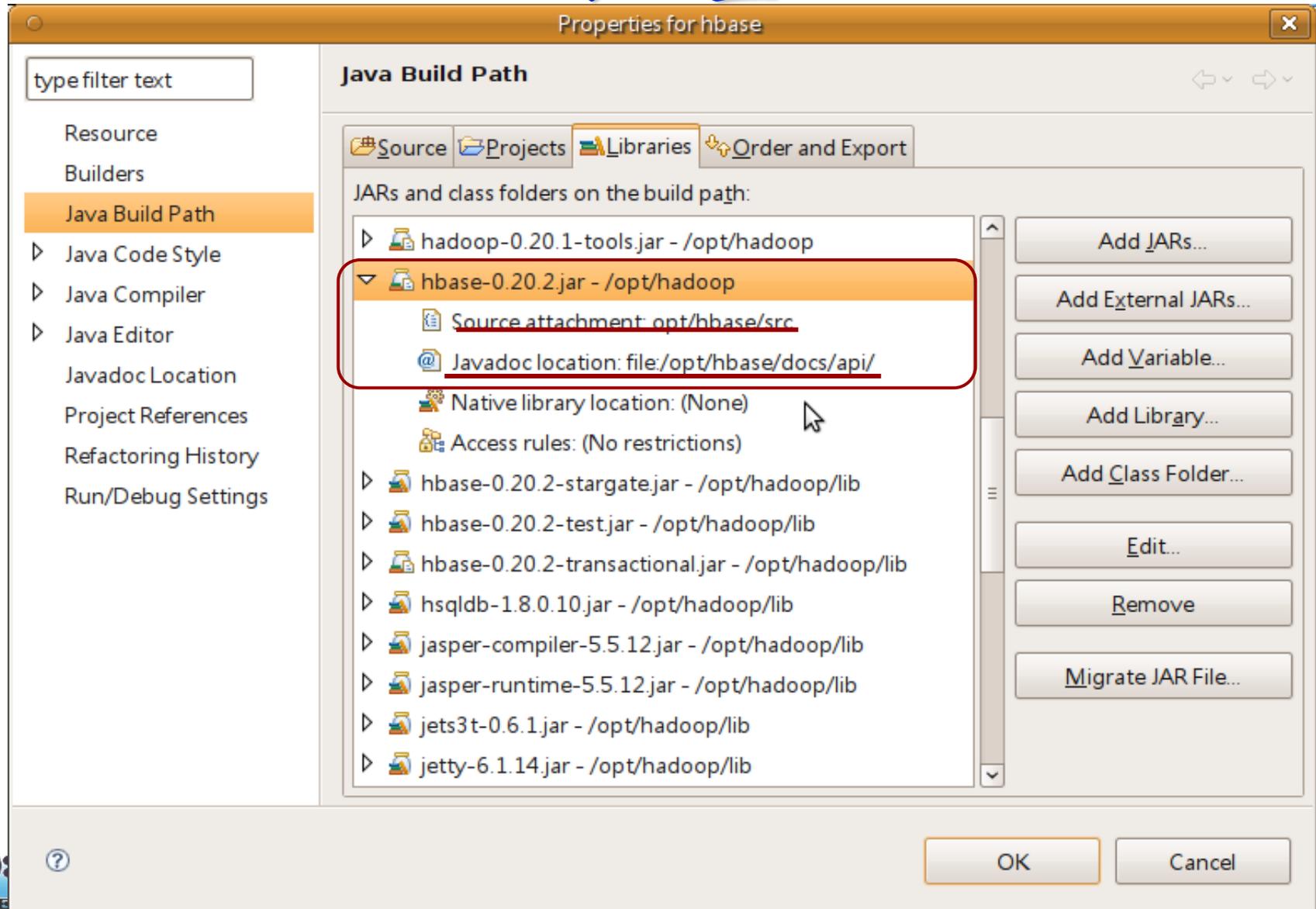


# B.3 選擇classpath 的library



重複2.2 的步驟來選取 hbase-0.20.\*.jar 與 lib/資料夾內的所有 jar 檔

# B.4 為函式庫增加原始碼、說明檔 的配置



## 三、HBase 程式設計

### 3.2.1 常用的 HBase API 說明

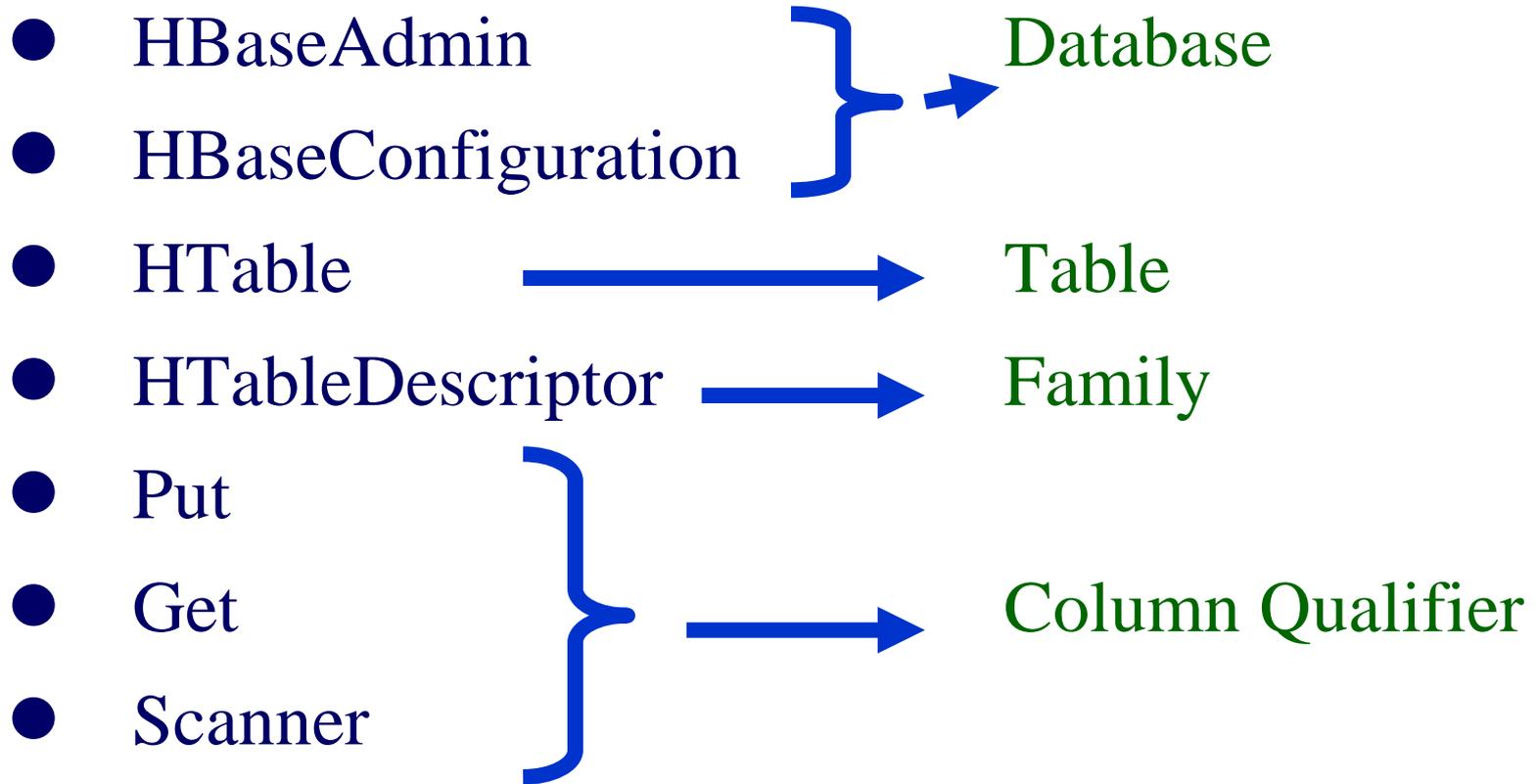
此篇列出用Java寫HBase 程式，常用到的API與說明

# HTable 成員

Table, Family, Column, Qualifier , Row, TimeStamp

		Contents	Department		
			news	bid	sport
t1	com.yahoo.news.tw	“我研發水下6千公尺機器人”	“tech”		
t2		“蚊子怎麼搜尋人肉”	“tech”		
t3		“用腦波「發聲」”	“tech”		
t1	com.yahoo.bid.tw	“... ipad ...”		“3C”	
t1	com.yahoo.sport.tw	“... Wang 40...”			“MBA”

# HBase 常用函式



# HBaseConfiguration

- Adds HBase configuration files to a Configuration
  - ◆ = new HBaseConfiguration ( )
  - ◆ = new HBaseConfiguration (Configuration c)
- 繼承自  
org.apache.hadoop.conf.Configuration

```
<property>
  <name> name
</name>
  <value> value
</value>
</property>
```

回傳值	函數	參數
void	addResource	(Path file)
void	clear	()
String	get	(String name)
String	getBoolean	(String name, boolean defaultValue )
void	set	(String name, String value)
void	setBoolean	(String name, boolean value)

# HBaseAdmin

- HBase的管理介面
  - ◆ = new HBaseAdmin( HBaseConfiguration conf )
- Ex:

```
HBaseAdmin admin = new HBaseAdmin(config);  
admin.disableTable (“tablename”);
```

回傳值	函數	參數
void	addColumn	(String tableName, HColumnDescriptor column)
	checkHBaseAvailable	(HBaseConfiguration conf)
	createTable	(HTableDescriptor desc)
	deleteTable	(byte[] tableName)
	deleteColumn	(String tableName, String columnName)
	enableTable	(byte[] tableName)
	disableTable	(String tableName)
HTableDescriptor[]	listTables	()
void	modifyTable	(byte[] tableName, HTableDescriptor htd)
boolean	tableExists	(String tableName)

# HTableDescriptor

- HTableDescriptor contains the name of an HTable, and its column families.
  - ◆ = new HTableDescriptor()
  - ◆ = new HTableDescriptor(String name)
- Constant-values
  - ◆ org.apache.hadoop.hbase.HTableDescriptor.TABLE\_DESCRIPTOR\_VERSION
- Ex:

```
HTableDescriptor htd = new HTableDescriptor(tablename);  
htd.addFamily ( new HColumnDescriptor (“Family”));
```

回傳值	函數	參數
void	addFamily	(HColumnDescriptor family)
HColumnDescriptor	removeFamily	(byte[] column)
byte[]	getName	( ) = Table name
byte[]	getValue	(byte[] key) = 對應key的value
void	setValue	(String key, String value)

# HColumnDescriptor

- An HColumnDescriptor contains information about a column family
  - ◆ = new HColumnDescriptor(String familyname)
- Constant-values
  - ◆ org.apache.hadoop.hbase.HTableDescriptor.TABLE\_DESCRIPTOR\_VERSION
- Ex:

```
HTableDescriptor htd = new HTableDescriptor(tablename);  
HColumnDescriptor col = new HColumnDescriptor("content:");  
htd.addFamily(col);
```

回傳值	函數	參數
byte[]	getName	( ) = Family name
byte[]	getValue	(byte[] key) = 對應key的value
void	setValue	(String key, String value)

# HTable

- Used to communicate with a single HBase table.
- ◆ = new HTable(HBaseConfiguration conf, String tableName)
- Ex:

```
HTable table = new HTable (conf, Bytes.toBytes ( tablename ));  
ResultScanner scanner = table.getScanner ( family );
```

回傳值	函數	參數
void	checkAndPut	(byte[] row, byte[] family, byte[] qualifier, byte[] value, Put put)
void	close	()
boolean	exists	(Get get)
Result	get	(Get get)
byte[][]	getEndKeys	()
ResultScanner	getScanner	(byte[] family)
HTableDescriptor	getTableDescriptor	()
byte[]	getTableName	()
static boolean	isTableEnabled	(HBaseConfiguration conf, String tableName)
void	put	(Put put)

# Put

- Used to perform Put operations for a single row.
  - ◆ = new Put(byte[] row)
  - ◆ = new Put(byte[] row, RowLock rowLock)
- Ex:

```
HTable table = new HTable (conf, Bytes.toBytes ( tablename ));  
Put p = new Put ( brow );  
p.add (family, qualifier, value);  
table.put ( p );
```

Put	add	(byte[] family, byte[] qualifier, byte[] value)
Put	add	(byte[] column, long ts, byte[] value)
byte[]	getRow	()
RowLock	getRowLock	()
long	getTimeStamp	()
boolean	isEmpty	()
Put	setTimeStamp	(long timestamp)

# Get

- Used to perform Get operations on a single row.
  - ◆ = new Get (byte[] row)
  - ◆ = new Get (byte[] row, RowLock rowLock)
- Ex:

```
HTable table = new HTable(conf, Bytes.toBytes(tablename));  
Get g = new Get(Bytes.toBytes(row));
```

Get	addColumn	(byte[] column)
Get	addColumn	(byte[] family, byte[] qualifier)
Get	addColumnns	(byte[][] columns)
Get	addFamily	(byte[] family)
TimeRange	getTimeRange	()
Get	setTimeRange	(long minStamp, long maxStamp)
Get	setFilter	(Filter filter)

# Result

- Single row result of a Get or Scan query.

- ◆ = new Result()

- Ex:

```
HTable table = new HTable(conf, Bytes.toBytes(tablename));
Get g = new Get(Bytes.toBytes(row));
Result rowResult = table.get(g);
Bytes[] ret = rowResult.getValue( (family + ":" + column) );
```

boolean	containsColumn	(byte[] family, byte[] qualifier)
NavigableMap <byte[],byte[]>	getFamilyMap	(byte[] family)
byte[]	getValue	(byte[] column)
byte[]	getValue	(byte[] family, byte[] qualifier)
int	Size	()

# Scanner

- All operations are identical to **Get**
  - ◆ Rather than specifying a single row, an optional startRow and stopRow may be defined.
- If rows are not specified, the Scanner will iterate over all rows.
  - ◆ = new Scan ()
  - ◆ = new Scan (byte[] startRow, byte[] stopRow)
  - ◆ = new Scan (byte[] startRow, Filter filter)

Get	addColumn	(byte[] column)
Get	addColumn	(byte[] family, byte[] qualifier)
Get	addColumnns	(byte[][] columns)
Get	addFamily	(byte[] family)
TimeRange	getTimeRange	()
Get	setTimeRange	(long minStamp, long maxStamp)
Get	setFilter	(Filter filter)

# Interface ResultScanner

- Interface for client-side scanning. Go to HTable to obtain instances.
- ◆ `HTable.getScanner (Bytes.toBytes(family));`
- Ex:

```
ResultScanner scanner = table.getScanner (Bytes.toBytes(family));  
for (Result rowResult : scanner) {  
    Bytes[] str = rowResult.getValue ( family , column );  
}
```

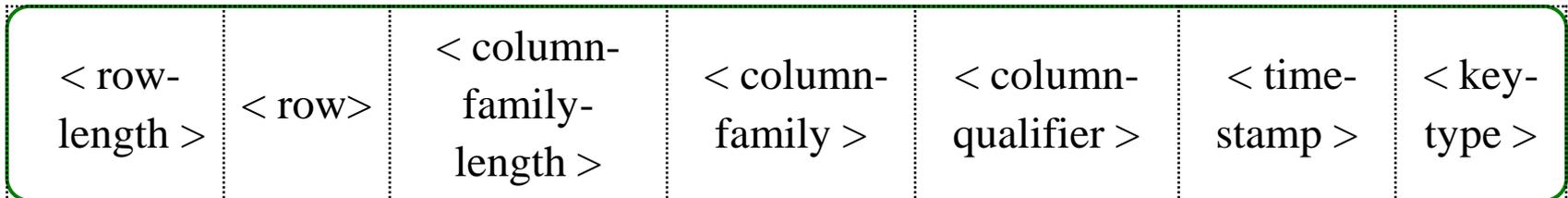
void	close	()
Result	next	()

# HBase Key/Value 的格式

- org.apache.hadoop.hbase.KeyValue
- getRow(), getFamily(), getQualifier(), getTimestamp(), and getValue().
- The KeyValue blob format inside the byte array is:

**<keylength> <valuelength> <key> <value>**

◆ Key 的格式:



- ◆ Rowlength 最大值為 Short.MAX\_SIZE,
- ◆ column family length 最大值為 Byte.MAX\_SIZE,
- ◆ column qualifier + key length 必須小於 Integer.MAX\_SIZE.

## 三、HBase 程式設計

### 3.2.2 實做I/O操作

HBase 的 I/O 操作 範例程式碼

# 範例一：新增Table

〈指令〉

```
create <表名>, {<family>, ....}
```

```
$ hbase shell  
> create 'tablename', 'family1', 'family2', 'family3'  
0 row(s) in 4.0810 seconds  
> List  
tablename  
1 row(s) in 0.0190 seconds
```

# 範例一：新增Table

<程式碼>

```
public static void createHBaseTable ( String tablename, String
    familyname ) throws IOException
{
    HBaseConfiguration config = new HBaseConfiguration();
    HBaseAdmin admin = new HBaseAdmin(config);
    HTableDescriptor htd = new HTableDescriptor( tablename );
    HColumnDescriptor col = new HColumnDescriptor( familyname );
    htd.addFamily ( col );
    if( admin.tableExists(tablename))
    {
        return () }
    admin.createTable(htd);
}
```

# 範例二：Put資料進Column

〈指令〉

```
put '表名', '列', 'column', '值' , ['時間']
```

```
> put 'tablename', 'row1', 'family1:qual1', 'value'
```

```
0 row(s) in 0.0030 seconds
```

## 範例二：Put資料進Column <程式碼>

```
static public void putData(String tablename, String row, String family,
    String column, String value) throws IOException {
    HBaseConfiguration config = new HBaseConfiguration();
    HTable table = new HTable(config, tablename);
    byte[] brow = Bytes.toBytes(row);
    byte[] bfamily = Bytes.toBytes(family);
    byte[] bcolumn = Bytes.toBytes(column);
    byte[] bvalue = Bytes.toBytes(value);
    Put p = new Put(brow);
    p.add(bfamily, bcolumn, bvalue);
    table.put(p);
    table.close();
}
```

# 範例三： Get Column Value

<指令>

```
get '表名', '列'
```

```
> get 'tablename', 'row1'
```

```
COLUMN
```

```
CELL
```

```
family1:column1    timestamp=1265169495385, value=value
```

```
1 row(s) in 0.0100 seconds
```

## 範例三：Get Column Value 〈程式碼〉

```
String getColumn ( String tablename, String row, String
family, String column ) throws IOException {
HBaseConfiguration conf = new HBaseConfiguration();
HTable table;
table = new HTable( conf, Bytes.toBytes( tablename));
Get g = new Get(Bytes.toBytes(row));
Result rowResult = table.get(g);
return Bytes.toString( rowResult.getValue (
        Bytes.toBytes (family + “:” + column)));
}
```

# 範例四：Scan all Column

<指令>

scan '表名'

```
> scan 'tablename'
```

```
ROW COLUMN+CELL
```

```
row1 column=family1:column1, timestamp=1265169415385, value=value1
```

```
row2 column=family1:column1, timestamp=1263534411333, value=value2
```

```
row3 column=family1:column1, timestamp=1263645465388, value=value3
```

```
row4 column=family1:column1, timestamp=1264654615301, value=value4
```

```
row5 column=family1:column1, timestamp=1265146569567, value=value5
```

```
5 row(s) in 0.0100 seconds
```

## 範例四：Scan all Column <程式碼>

```
static void ScanColumn(String tablename, String family, String
column) throws IOException {
    HBaseConfiguration conf = new HBaseConfiguration();
    HTable table = new HTable ( conf, Bytes.toBytes(tablename));
    ResultScanner scanner = table.getScanner(
        Bytes.toBytes(family));
    int i = 1;
    for (Result rowResult : scanner) {
        byte[] by = rowResult.getValue(
            Bytes.toBytes(family), Bytes.toBytes(column) );
        String str = Bytes.toString ( by );
        System.out.println("row " + i + " is \"" + str + "\"");
        i++;
    }
}
```

# 範例五：刪除資料表

<指令>

```
disable '表名'  
drop '表名'
```

```
> disable 'tablename'  
0 row(s) in 6.0890 seconds  
> drop 'tablename'  
0 row(s) in 0.0090 seconds  
0 row(s) in 0.0090 seconds  
0 row(s) in 0.0710 seconds
```

# 範例五：刪除資料表

<程式碼>

```
static void drop ( String tablename ) throws IOExceptions {  
    HBaseConfiguration conf = new HBaseConfiguration();  
    HBaseAdmin admin = new HBaseAdmin (conf);  
    if (admin.tableExists(tablename))  
    {  
        admin.disableTable(tablename);  
        admin.deleteTable(tablename);  
    }else{  
        System.out.println(" [" + tablename+ "] not found!");  
    }  
}
```

## 三、HBase 程式設計

# 3.2.3 MapReduce 與 HBase 的搭配

HBase 的 MapReduce 範例程式碼

# 範例六：WordCountHBase

說明：

此程式碼將輸入路徑的檔案內的字串取出做字數統計  
再將結果塞回HTable內

運算方法：

將此程式運作在hadoop 0.20 平台上，用(參考2)的方法加入hbase參數後，將此程式碼打包成XX.jar

結果：

```
> scan 'wordcount'
```

```
ROW      COLUMN+CELL
```

```
am       column=content:count, timestamp=1264406245488, value=1
```

```
chen     column=content:count, timestamp=1264406245488, value=1
```

```
hi,      column=content:count, timestamp=1264406245488, value=2
```

注意：

1. 在hdfs 上來源檔案的路徑為 "/user/\$YOUR\_NAME/input"

請注意必須先放資料到此hdfs上的資料夾內，且此資料夾內只能放檔案，不可再放資料夾

運算完後，程式將執行結果放在hbase的wordcount資料表內

# 範例六：WordCountHBase

<1>

```
public class WordCountHBase
{
    public static class Map extends
        Mapper<LongWritable,Text,Text,
        IntWritable>
    {
        private IntWritable i = new
            IntWritable(1);
        public void map(LongWritable
            key,Text value,Context context)
            throws IOException,
            InterruptedException
        {
            String s[] =
                value.toString().trim().split(" ");
            for( String m : s)
            {
                context.write(new Text(m), i);
            }
        }
    }
}
```

```
public static class Reduce extends
    TableReducer<Text, IntWritable,
    NullWritable> {
    public void reduce(Text key,
        Iterable<IntWritable> values, Context
        context) throws IOException,
        InterruptedException {
        int sum = 0;
        for(IntWritable i : values) {
            sum += i.get(); }
        Put put = new
            Put(Bytes.toBytes(key.toString()));
        put.add(Bytes.toBytes("content"),
            Bytes.toBytes("count"),
            Bytes.toBytes(String.valueOf(sum)));
        context.write(NullWritable.get(),
            put);
    }
}
```

# 範例六：WordCountHBase

<2>

```
public static void createHBaseTable(String
    tablename)throws IOException
{
    HTableDescriptor htd = new
        HTableDescriptor(tablename);
    HColumnDescriptor col = new
        HColumnDescriptor("content:");
    htd.addFamily(col);
    HBaseConfiguration config = new
        HBaseConfiguration();
    HBaseAdmin admin = new
        HBaseAdmin(config);
    if(admin.tableExists(tablename))
    {
        admin.disableTable(tablename);
        admin.deleteTable(tablename);
    }
    System.out.println("create new table: " +
        tablename);
    admin.createTable(htd);
}
```

```
public static void main(String args[]) throws Exception
{
    String tablename = "wordcount";
    Configuration conf = new Configuration();
    conf.set(TableOutputFormat.OUTPUT_TABLE,
        tablename);
    createHBaseTable(tablename);
    String input = args[0];
    Job job = new Job(conf, "WordCount " + input);
    job.setJarByClass(WordCountHBase.class);
    job.setNumReduceTasks(3);
    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TableOutputFormat.class);
    FileInputFormat.addInputPath(job, new Path(input));
    System.exit(job.waitForCompletion(true)?0:1);
}
```

# 範例七：LoadHBaseMapper

說明：

此程式碼將HBase的資料取出來，再將結果塞回hdfs上

運算方法：

將此程式運作在hadoop 0.20 平台上，用(參考2)的方法加入hbase參數後，將此程式碼打包成XX.jar

結果：

```
$ hadoop fs -cat <hdfs_output>/part-r-00000
```

```
-----  
54 30 31   GunLong  
54 30 32   Esing  
54 30 33   SunDon  
54 30 34   StarBucks  
-----
```

注意：

1. 請注意hbase上必須要有table, 並且已經有資料
2. 運算完後，程式將執行結果放在你指定hdfs的<hdfs\_output>內  
請注意沒有<hdfs\_output>資料夾

# 範例七：LoadHBaseMapper <1>

```
public class LoadHBaseMapper {
public static class HtMap extends
    TableMapper<Text, Text> {
public void
    map(ImmutableBytesWritable key,
        Result value,
            Context context) throws
                IOException,
                InterruptedException {
    String res =
        Bytes.toString(value.getValue(Bytes
            .toBytes("Detail"),
                Bytes.toBytes("Name")));
    context.write(new
        Text(key.toString()), new
        Text(res));
    }}
}
```

```
public static class HtReduce extends
    Reducer<Text, Text, Text, Text> {
public void reduce(Text key, Iterable<Text>
    values, Context context)
        throws IOException,
            InterruptedException {
    String str = new String("");
    Text final_key = new Text(key);
    Text final_value = new Text();
    for (Text tmp : values) {
        str += tmp.toString();    }
    final_value.set(str);
    context.write(final_key, final_value);
    }}
}
```

# 範例七：LoadHBaseMapper <2>

```
public static void main(String args[])
    throws Exception {
String input = args[0];
String tablename = "tcr";
Configuration conf = new
    Configuration();
Job job = new Job (conf, tablename + "
    hbase data to hdfs");
job.setJarByClass
    (LoadHBaseMapper.class);
TableMapReduceUtil.
    initTableMapperJob
(tablename, myScan,
    HtMap.class, Text.class,
    Text.class, job);
job.setMapperClass (HtMap.class);
job.setReducerClass (HtReduce.class);
job.setMapOutputKeyClass (Text.class);
job.setMapOutputValueClass
    (Text.class);
job.setInputFormatClass (
    TableInputFormat.class);
job.setOutputFormatClass (
    TextOutputFormat.class);
job.setOutputKeyClass( Text.class);
job.setOutputValueClass( Text.class);
FileOutputFormat.setOutputPath ( job,
    new Path(input));
System.exit (job.waitForCompletion
    (true) ? 0 : 1);
}}
```

## 三、HBase 程式設計

# 3.2.4 進階補充

HBase內contrib的項目，如

A. Trancational

B. Thrift

# A. Transactional HBase

- Indexed Table = Secondary Index = Transactional HBase
- 內容與原本table相似的另一張table，但key不同，利於排列內容

Primary Table

	name	price	description
1	apple	10	xx
2	orig	5	ooo
3	banana	15	vvvv
4	tomato	8	uu



Indexed Table

	name	price	description
2	orig	5	ooo
4	tomato	8	uu
1	apple	10	xx
3	banana	15	vvvv

# A.1 Transactional HBase

## 環境設定

需在 \$HBASE\_INSTALL\_DIR/conf/hbase-site.xml 檔內  
增加兩項內容

```
<property>
  <name> hbase.regionserver.class </name>
  <value> org.apache.hadoop.hbase.ipc.IndexedRegionInterface
  </value>
</property>
<property>
  <name> hbase.regionserver.impl </name>
  <value>
org.apache.hadoop.hbase.regionserver.tableindexed.IndexedRegionServer
  </value>
</property>
```

# A.a Ex : 從一個原有的Table 增加 IndexedTable

```
public void addSecondaryIndexToExistingTable  
(String TableName, String IndexID, String  
IndexColumn) throws IOException {  
    HBaseConfiguration conf = new HBaseConfiguration();  
    IndexedTableAdmin admin = null;  
    admin = new IndexedTableAdmin(conf);  
    admin.addIndex(Bytes.toBytes(TableName), new  
    IndexSpecification(  
        IndexID, Bytes.toBytes(IndexColumn)));  
}
```

# A.b Ex : 建立一個新的Table 附帶 IndexedTable

```
public void createTableWithSecondaryIndexes(String TableName,
      String IndexColumn) throws IOException {
    HBaseConfiguration conf = new HBaseConfiguration();
    conf.addResource(new Path("/opt/hbase/conf/hbase-site.xml"));
    HTableDescriptor desc = new HTableDescriptor(TableName);
    desc.addFamily(new HColumnDescriptor("Family1"));
    IndexedTableDescriptor Idxdesc = new
    IndexedTableDescriptor(desc);
    Idxdesc.addIndex(new IndexSpecification(IndexColumn, Bytes
      .toBytes(" Family1 :" + IndexColumn)));
    IndexedTableAdmin admin = new IndexedTableAdmin(conf);
    admin.createIndexedTable(Idxdesc);
}
```

# B. Thrift

- 由 Facebook 所開發
- 提供跨語言做資料交換的平台
- 你可以用任何 Thrift 有支援的語言來存取 HBase
  - ◆ PHP
  - ◆ Perl
  - ◆ C++
  - ◆ Python
  - ◆ .....

# B.1 安裝方法

- Thrift ( 0.2 ~ 0.4 ) + PHP 5 + HBase 0.20
- <http://trac.nchc.org.tw/cloud/wiki/waue/2010/HbaseThrift>
  - ◆ 請先安裝HBase之後，並用套件安裝工具安裝PHP 相關套件，再用此連結內的教材安裝Thrift

## B.2 Thrift 的PHP程式碼

- <http://trac.nchc.org.tw/cloud/wiki/waue/2010/HbaseThrift>
  - ◆ 連線
  - ◆ 列出所有table
  - ◆ 刪除table
  - ◆ 寫入資料
  - ◆ 讀取資料
  - ◆ Scan 一整張table

## 三、HBase 程式設計

# 3.3 案例演練

利用一個虛擬的案例來運用之前的程式碼

# 中科餐廳開張囉！

- 故事背景：
  - ◆ 中科餐城即將開張，預估有大批人潮
  - ◆ 管理局想統計餐城內所有餐廳的營運狀況
- 用傳統資料庫可能：
  - ◆ 大規模資料、同時讀寫，資料分析運算、…（自行發揮）
- 因此員工餐廳將導入
  - ◆ HBase資料庫存放資料
  - ◆ 透過 Hadoop進行Map Reduce分析運算

## 3.3.1. 建立商店資料

假設：目前有四間商店進駐中科餐廳，分別為位在  
第1區的GunLong，品項4項單價為<20, 40, 30, 50>  
第2區的ESing, 品項1項單價為<50>  
第3區的SunDon, 品項2項單價為<40, 30>  
第4區的StarBucks, 品項3項單價為<50, 50, 20>

	Detail		Products				Turnover			
	Name	Locate	P1	P2	P3	P4				
T01	GunLong	01	20	40	30	50				
T02	ESing	02	50							
T03	SunDon	03	40	30						
T04	StarBuck s	04	50	50	20					

## 3.3.1.a 建立初始HTable

<程式碼>

```
public void createHBaseTable(String tablename, String[] family)
    throws IOException {
    HTableDescriptor htd = new HTableDescriptor(tablename);
    for (String fa : family) {
        htd.addFamily(new HColumnDescriptor(fa));
    }
    HBaseConfiguration config = new HBaseConfiguration();
    HBaseAdmin admin = new HBaseAdmin(config);
    if (admin.tableExists(tablename)) {
        System.out.println("Table: " + tablename + "Existed.");
    } else {
        System.out.println("create new table: " + tablename);

        admin.createTable(htd);
    }
}
```

## 3.3.1.a 執行結果

Table: TCRC

Family	Detail	Products	Turnover
Qualifier	...	...	...
Row1	value		
Row2			
Row3			
...			

## 3.3.1.b 用讀檔方式把資料匯入HTable

<程式碼>

```
void loadFile2HBase(String file_in, String table_name) throws IOException {
    BufferedReader fi = new BufferedReader(
        new FileReader(new File(file_in)));
    String line;
    while ((line = fi.readLine()) != null) {
        String[] str = line.split(";");
        int length = str.length;
        PutData.putData(table_name, str[0].trim(), "Detail", "Name", str[1]
            .trim());
        PutData.putData(table_name, str[0].trim(), "Detail", "Locate",
            str[2].trim());
        for (int i = 3; i < length; i++) {
            PutData.putData(table_name, str[0], "Products", "P" + (i - 2),
                str[i]);
        }
        System.out.println();
    }
    fi.close();
}
```

## 3.3.1.b 執行結果

	Detail		Products				Turnover
	Name	Locat e	P1	P2	P3	P4	
T01	GunLong	01	20	40	30	50	
T02	ESing	02	50				
T03	SunDon	03	40	30			
T04	StarBuck s	04	50	50	20		

# 3.3.1. 螢幕輸出結果

```
create new table: tcrc
Put data : "GunLong" to Table: tcrc's Detail:Name
Put data : "01" to Table: tcrc's Detail:Locate
Put data : "20" to Table: tcrc's Products:P1
Put data : "40" to Table: tcrc's Products:P2
Put data : "30" to Table: tcrc's Products:P3
Put data : "50" to Table: tcrc's Products:P4

Put data : "Esing" to Table: tcrc's Detail:Name
Put data : "02" to Table: tcrc's Detail:Locate
Put data : "50" to Table: tcrc's Products:P1

Put data : "SunDon" to Table: tcrc's Detail:Name
Put data : "03" to Table: tcrc's Detail:Locate
Put data : "40" to Table: tcrc's Products:P1
Put data : "30" to Table: tcrc's Products:P2

Put data : "StarBucks" to Table: tcrc's Detail:Name
Put data : "04" to Table: tcrc's Detail:Locate
Put data : "50" to Table: tcrc's Products:P1
Put data : "50" to Table: tcrc's Products:P2
Put data : "20" to Table: tcrc's Products:P3
```

## 3.3.2 計算單月每個品項的購買次數

- 刷卡購餐的系統將每人每次購餐紀錄成檔案，格式如右
- 讀紀錄檔並統計每天每個品項的消費次數
  - ◆ 將檔案上傳至hdfs
  - ◆ 使用Hadoop運算
- 計算完後寫入HBase
  - ◆ Turnover:P1,P2,P3,P4

```
waue:T01:P1:xx  
jazz:T01:P2:xxx  
lia:T01:P3:xxxx  
hung:T02:P1:xx  
lia:T04:P1:xxxx  
lia:T04:P1:xxxx  
hung:T04:P3:xx  
hung:T04:P2:xx
```

.....

## 3.3.2. 用Hadoop的Map Reduce運算並 把結果匯入HTable

<map 程式碼>

```
public class tcrc2Count {
public static class HtMap extends
    Mapper<LongWritable, Text, Text,
    IntWritable> {
private IntWritable one = new
    IntWritable(1);
public void map(LongWritable key, Text
    value, Context context)
    throws IOException,
    InterruptedException {
    String s[] =
    value.toString().trim().split(":");
    // xxx:T01:P4:oooo => T01@P4
    String str = s[1] + "@" + s[2];
    context.write(new Text(str), one);
}
}
```

<reduce程式碼>

```
public static class HtReduce extends
    TableReducer<Text, IntWritable,
    LongWritable> {
public void reduce(Text key, Iterable<IntWritable>
    values,
    Context context) throws IOException,
    InterruptedException {
    int sum = 0;
    for (IntWritable i : values) sum += i.get();
    String[] str = (key.toString()).split("@");
    byte[] row = (str[0]).getBytes();
    byte[] family = Bytes.toBytes("Turnover");
    byte[] qualifier = (str[1]).getBytes();
    byte[] summary =
    Bytes.toBytes(String.valueOf(sum));
    Put put = new Put(row);
    put.add(family, qualifier, summary );
    context.write(new LongWritable(), put);
}}
```

## 3.3.2. 用Hadoop的Map Reduce運算並把結果匯入HTable

### < Main 程式碼 >

```
public static void main(String args[]) throws Exception {
    String input = "income";
    String tablename = "tcrc";
    Configuration conf = new Configuration();
    conf.set(TableOutputFormat.OUTPUT_TABLE, tablename);
    Job job = new Job(conf, "Count to tcrc");
    job.setJarByClass(tcrc2Count.class);
    job.setMapperClass(HtMap.class);
    job.setReducerClass(HtReduce.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TableOutputFormat.class);
    FileInputFormat.addInputPath(job, new Path(input));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

## 3.3.2 執行結果

	Detail		Products				Turnover			
	Name	Locate	P1	P2	P3	P4	P1	P2	P3	P4
T01	GunLong	01	20	40	30	50	1	1	1	1
T02	ESing	02	50				2			
T03	SunDon	03	40	30			3			
T04	StarBucks	04	50	50	20		2	1	1	

> scan 'terc'

ROW

COLUMN+CELL

T01	column=Detail:Locate, timestamp=1265184360616, value=01
T01	column=Detail:Name, timestamp=1265184360548, value=GunLong
T01	column=Products:P1, timestamp=1265184360694, value=20
T01	column=Products:P2, timestamp=1265184360758, value=40
T01	column=Products:P3, timestamp=1265184360815, value=30
T01	column=Products:P4, timestamp=1265184360866, value=50
T01	column=Turnover:P1, timestamp=1265187021528, value=1
T01	column=Turnover:P2, timestamp=1265187021528, value=1
T01	column=Turnover:P3, timestamp=1265187021528, value=1
T01	column=Turnover:P4, timestamp=1265187021528, value=1
T02	column=Detail:Locate, timestamp=1265184360951, value=02
T02	column=Detail:Name, timestamp=1265184360910, value=Esing
T02	column=Products:P1, timestamp=1265184361051, value=50
T02	column=Turnover:P1, timestamp=1265187021528, value=2
T03	column=Detail:Locate, timestamp=1265184361124, value=03
T03	column=Detail:Name, timestamp=1265184361098, value=SunDon
T03	column=Products:P1, timestamp=1265184361189, value=40
T03	column=Products:P2, timestamp=1265184361259, value=30
T03	column=Turnover:P1, timestamp=1265187021529, value=3
T04	column=Detail:Locate, timestamp=1265184361311, value=04
T04	column=Detail:Name, timestamp=1265184361287, value=StarBucks
T04	column=Products:P1, timestamp=1265184361343, value=50
T04	column=Products:P2, timestamp=1265184361386, value=50
T04	column=Products:P3, timestamp=1265184361422, value=20
T04	column=Turnover:P1, timestamp=1265187021529, value=2
T04	column=Turnover:P2, timestamp=1265187021529, value=1
T04	column=Turnover:P3, timestamp=1265187021529, value=1

4 row(s) in 0.0310 seconds

## 3.3.3. 計算當天營業額

- 計算每間商店的營業額
  - ◆  $\Sigma$  (<該項商品單價> X <被購買的次數>)
  - ◆ 透過 Hadoop 的 Map () 從 HBase 內的 Products:{P1,P2,P3,P4} 與 Turnover:{P1,P2,P3,P4} 調出來
  - ◆ 經過計算後由 Hadoop 的 Reduce () 寫回 HBase 內 Turnover:Sum 的 Column 內
    - 需考慮到表格內每家的商品數量皆不同、有的品項沒有被購買

# 3.3.3. Hadoop 來源與輸出皆為 HBase

<map 程式碼>

<reduce 程式碼>

```
public class tcr3CalculateMR {
public static class HtMap extends TableMapper<Text, Text> {
public void map(ImmutableBytesWritable key, Result value,
    Context context) throws IOException, InterruptedException {
String row = Bytes.toString(value.getValue(Bytes.toBytes("Detail"),
    Bytes.toBytes("Locate"))));
int sum = 0;
for (int i = 0; i < 4; i++) {
String v = Bytes.toString(value.getValue(Bytes
    .toBytes("Products"), Bytes.toBytes("P" + (i +
    1))));
String c = Bytes.toString(value.getValue(Bytes
    .toBytes("Turnover"), Bytes.toBytes("P" + (i +
    1))));
if (v != null) {
if(c == null) c="0";
System.err.println("p=" + v);
System.err.println("c=" + c);
sum += Integer.parseInt(v) * Integer.parseInt(c);
System.err.println("T" + row + ":" + "p[" + i + "]" * " + "c["
    + i + "] => " + v + "*" + c + "+="
    + (sum)); }}
context.write(new Text("T" + row), new Text(String.valueOf(sum))); }}
}
```

```
public static class HtReduce extends
    TableReducer<Text, Text,
    Text> {
public void reduce(Text key,
    Iterable<Text> values,
    Context context)
    throws IOException,
    InterruptedException {
String sum = "";
for (Text i : values) {
sum += i.toString();
}
Put put = new
    Put(Bytes.toBytes(key.toStri
    ng()));
put.add(Bytes.toBytes("Turnover"),
    Bytes.toBytes("Sum"), Bytes
    .toBytes(sum));
context.write(new Text(), put);
}}
```

# 3.3.3. Hadoop 來源與輸出皆為 HBase

## < Main 程式碼 >

```
public static void main(String args[]) throws  
    Exception {  
String tablename = "tcr3";  
Scan myScan = new Scan();  
myScan.addColumn("Detail:Locate".getBytes());  
myScan.addColumn("Products:P1".getBytes());  
myScan.addColumn("Products:P2".getBytes());  
myScan.addColumn("Products:P3".getBytes());  
myScan.addColumn("Products:P4".getBytes());  
myScan.addColumn("Turnover:P1".getBytes());  
myScan.addColumn("Turnover:P2".getBytes());  
myScan.addColumn("Turnover:P3".getBytes());  
myScan.addColumn("Turnover:P4".getBytes());  
Configuration conf = new Configuration();
```

```
Job job = new Job(conf, "Calculating ");  
job.setJarByClass(tcr3CalculateMR.class);  
job.setMapperClass(HtMap.class);  
job.setReducerClass(HtReduce.class);  
job.setMapOutputKeyClass(Text.class);  
job.setMapOutputValueClass(Text.class);  
job.setInputFormatClass(TableInputFormat.class);  
job.setOutputFormatClass(TableOutputFormat.class  
    );  
TableMapReduceUtil.initTableMapperJob(tablename,  
    myScan, HtMap.class,  
    Text.class, Text.class, job);  
TableMapReduceUtil.initTableReducerJob(tablename,  
    HtReduce.class, job);  
System.exit(job.waitForCompletion(true) ? 0 : 1);  
}  
}
```

> scan 'trc'

ROW	COLUMN+CELL
T01	column=Detail:Locate, timestamp=1265184360616, value=01
T01	column=Detail:Name, timestamp=1265184360548, value=GunLong
T01	column=Products:P1, timestamp=1265184360694, value=20
T01	column=Products:P2, timestamp=1265184360758, value=40
T01	column=Products:P3, timestamp=1265184360815, value=30
T01	column=Products:P4, timestamp=1265184360866, value=50
T01	column=Turnover:P1, timestamp=1265187021528, value=1
T01	column=Turnover:P2, timestamp=1265187021528, value=1
T01	column=Turnover:P3, timestamp=1265187021528, value=1
T01	column=Turnover:P4, timestamp=1265187021528, value=1
T01	column=Turnover:sum, timestamp=1265190421993, value=140
T02	column=Detail:Locate, timestamp=1265184360951, value=02
T02	column=Detail:Name, timestamp=1265184360910, value=Esing
T02	column=Products:P1, timestamp=1265184361051, value=50
T02	column=Turnover:P1, timestamp=1265187021528, value=2
T02	column=Turnover:sum, timestamp=1265190421993, value=100
T03	column=Detail:Locate, timestamp=1265184361124, value=03
T03	column=Detail:Name, timestamp=1265184361098, value=SunDon
T03	column=Products:P1, timestamp=1265184361189, value=40
T03	column=Products:P2, timestamp=1265184361259, value=30
T03	column=Turnover:P1, timestamp=1265187021529, value=3
T03	column=Turnover:sum, timestamp=1265190421993, value=120
T04	column=Detail:Locate, timestamp=1265184361311, value=04
T04	column=Detail:Name, timestamp=1265184361287, value=StarBucks
T04	column=Products:P1, timestamp=1265184361343, value=50
T04	column=Products:P2, timestamp=1265184361386, value=50
T04	column=Products:P3, timestamp=1265184361422, value=20
T04	column=Turnover:P1, timestamp=1265187021529, value=2
T04	column=Turnover:P2, timestamp=1265187021529, value=1
T04	column=Turnover:P3, timestamp=1265187021529, value=1
T04	column=Turnover:sum, timestamp=1265190421993, value=170

4 row(s) in 0.0460 seconds

### 3.3.3. 執行結果

	Detail		Products				Turnover				Sum
	Name	Locate	P1	P2	P3	P4	P1	P2	P3	P4	
T01	GunLong	01	20	40	30	50	1	1	1	1	140
T02	ESing	02	50				2				100
T03	SunDon	03	40	30			3	3			210
T04	StarBucks	04	50	50	20		4	4	4		480

## 3.3.4. 狀況來了

- 管理局想知道餐廳的營運狀況，因此需要產生出最後的報表
  - ◆ 資料由小到大排序
  - ◆ 過濾掉營業額  $< 130$  的資料

# 3.3.4. 回顧 Transactional HBase

- Indexed Table = Secondary Index = Transactional HBase
- 內容與原本table相似的另一張table，但key不同，利於排列內容

Primary Table

	name	price	description
1	apple	10	xx
2	orig	5	ooo
3	banana	15	vvvv
4	tomato	8	uu



Indexed Table

	name	price	description
2	orig	5	ooo
4	tomato	8	uu
1	apple	10	xx
3	banana	15	vvvv

# 3.3.4.回顧：Transactional HBase 環境設定

需在 \$HBASE\_INSTALL\_DIR/conf/hbase-site.xml 檔內  
增加兩項內容

```
<property>  
  <name> hbase.regionserver.class </name>  
  <value> org.apache.hadoop.hbase.ipc.IndexedRegionInterface  
  </value>  
</property>  
<property>  
  <name> hbase.regionserver.impl </name>  
  <value>  
org.apache.hadoop.hbase.regionserver.tableindexed.IndexedRegionServer  
  </value>  
</property>
```

## 3.3.4.1 建立Indexed Table

```
public class tcrc4SortTurnover {
public void addIndexToTurnover(String OriTable, String IndexID,
    String OriColumn) throws IOException {
HBaseConfiguration conf = new HBaseConfiguration();
conf.addResource(new Path("/opt/hbase/conf/hbase-site.xml"));
IndexedTableAdmin admin = new IndexedTableAdmin(conf);
admin.addIndex(Bytes.toBytes(OriTable), new IndexSpecification(IndexID,
    Bytes.toBytes(OriColumn)));
}
public static void main(String[] args) throws IOException {
tcrc4SortTurnover tt = new tcrc4SortTurnover();
tt.addIndexToTurnover("tcrc", "Sum", "Turnover:Sum");
tt.readSortedValGreater("130");
}}
```

# 3.3.4.1 Indexed Table 輸出結果

```
> scan 'tcrs-Sum'
```

ROW	COLUMN+CELL
100T02	column=Turnover:Sum, timestamp=1265190782127, value=100
100T02	column=__INDEX__:ROW, timestamp=1265190782127, value=T02
120T03	column=Turnover:Sum, timestamp=1265190782128, value=120
120T03	column=__INDEX__:ROW, timestamp=1265190782128, value=T03
140T01	column=Turnover:Sum, timestamp=1265190782126, value=140
140T01	column=__INDEX__:ROW, timestamp=1265190782126, value=T01
170T04	column=Turnover:Sum, timestamp=1265190782129, value=170
170T04	column=__INDEX__:ROW, timestamp=1265190782129, value=T04

```
4 row(s) in 0.0140 seconds
```

## 3.3.4.2 產生排序且篩選過的資料

```
public void readSortedValGreater(String filter_val)
    throws IOException {
    HBaseConfiguration conf = new
        HBaseConfiguration();
    conf.addResource(new
        Path("/opt/hbase/conf/hbase-site.xml"));
    // the id of the index to use
    String tablename = "tcrc";
    String indexId = "Sum";
    byte[] column_1 = Bytes.toBytes("Turnover:Sum");
    byte[] column_2 = Bytes.toBytes("Detail:Name");
    byte[] indexStartRow =
        HConstants.EMPTY_START_ROW;
    byte[] indexStopRow = null;
    byte[][] indexColumns = null;
    SingleColumnValueFilter indexFilter = new
        SingleColumnValueFilter(Bytes
            .toBytes("Turnover"),
            Bytes.toBytes("Sum"),
            CompareFilter.CompareOp.GREATER_OR
            _EQUAL, Bytes.toBytes(filter_val));
```

```
byte[][] baseColumns = new byte[][] { column_1,
    column_2 };
    IndexedTable table = new IndexedTable(conf,
        Bytes.toBytes(tablename));
    ResultScanner scanner =
        table.getIndexedScanner(indexId,
            indexStartRow,
            indexStopRow, indexColumns, indexFilter,
            baseColumns);
    for (Result rowResult : scanner) {
        String sum =
            Bytes.toString(rowResult.getValue(column_1)
            );
        String name =
            Bytes.toString(rowResult.getValue(column_2)
            );
        System.out.println(name + " 's turnover is " +
            sum + " $.");
    }
    table.close();
}
```

## 3.3.4.2 列出最後結果

- 營業額大於130元者

**GunLong 's turnover is 140 \$.**  
**StarBucks 's turnover is 170 \$.**

## 3.3.5. 狀況來了

- 高層嫌報表輸出
  - ◆ 只能用cmd
  - ◆ 無法直接看
  - ◆ 介面....

## 3.3.5.回顧：還記得 Thrift ？

你可以用任何 Thrift 有支援的語言來直接存取、操作 HBase

- 由 Facebook 所開發
- 提供跨語言做資料交換的平台
- 支援的程式語言
  - ◆ PHP
  - ◆ Perl
  - ◆ C++
  - ◆ Python
  - ◆ …..

http://secuse.nchc.org.t... x

← → ↻ 🏠 ☆ http://secuse.nchc.org.tw/hbase/scantablehtml.php

For quick access, place your bookmarks here on the bookmarks bar. [Import bo](#)

row	Fam:Col	value
T01	Detail:Locate	01
T01	Detail:Name	GunLong
T01	Products:P1	20
T01	Products:P2	40
T01	Products:P3	30
T01	Products:P4	50
T01	Turnover:P1	1
T01	Turnover:P2	1
T01	Turnover:P3	1
T01	Turnover:Sum	90
T02	Detail:Locate	02
T02	Detail:Name	Esing
T02	Products:P1	50
T02	Turnover:P1	2
T02	Turnover:Sum	100

<http://localhost/hbase/scantablehtml.php>

# Conclusions

- HBase 適用於巨量資料的儲存與鬆散框架應用
- 是Hadoop 的開放原始碼子項目
- 叢集模式需Zookeeper 介入
- 提供 shell 的交互操作介面
- 擁有豐富的Java API 讓你能直接寫code 存取
- 透過Thrift，你可以用你熟悉的程式語言對HBase 作操控