TSMC教育訓練課程

HBase Programming

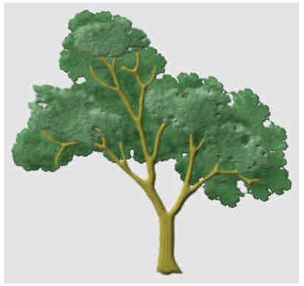
< V 0.20 >

王耀聰 陳威宇

Jazz@nchc.org.tw

waue@nchc.org.tw

財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING



Outline

- HBase 程式編譯方法
- HBase 程式設計
  - 常用的HBase API 說明
  - 實做 I/O 操作
  - 搭配Map Reduce 運算
- 案例演練
- 其他專案

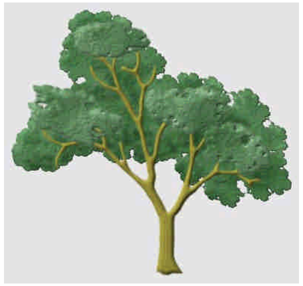
HBase 程式編譯方法

此篇介紹兩種編譯與執行HBase程式的方法：

Method 1 – 使用Java JDK 1.6

Method 2 – 使用Eclipse 套件

財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING



1. 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 △ -classpath △ **hadoop-*-core.jar:hbase-*.jar** △ -d △ MyJava △ MyCode.java
3. 封裝
   - jar △ -cvf △ MyJar.jar △ -C △ MyJava △ .
4. 執行
   - bin/hadoop △ jar △ MyJar.jar △ MyCode △ {Input/ △ Output/ }

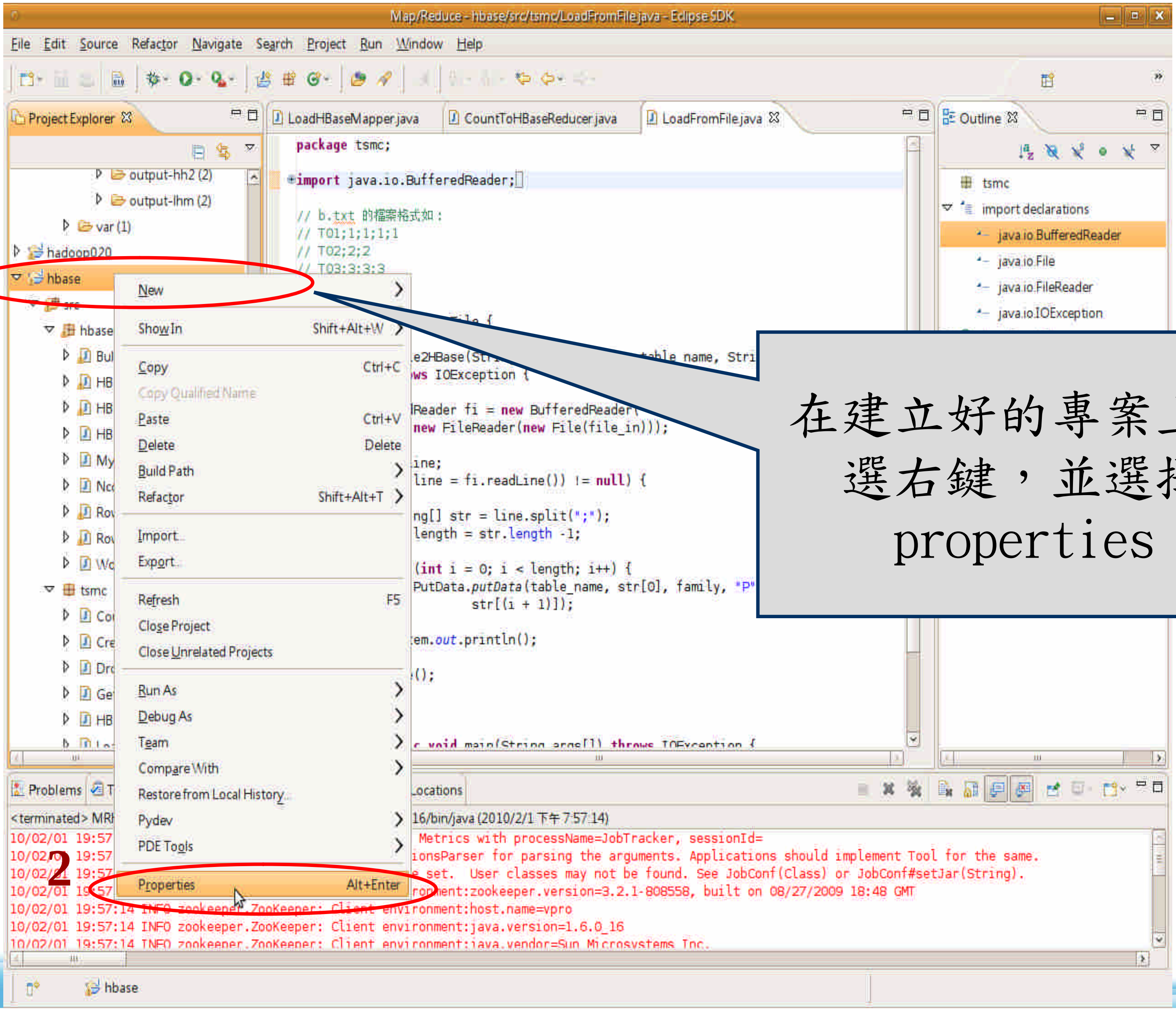
- 所在的執行目錄為Hadoop_Home
- ./MyJava = 編譯後程式碼目錄
- Myjar.jar = 封裝後的編譯檔

- 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

2.0 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

2.1 設定專案的細部屬性



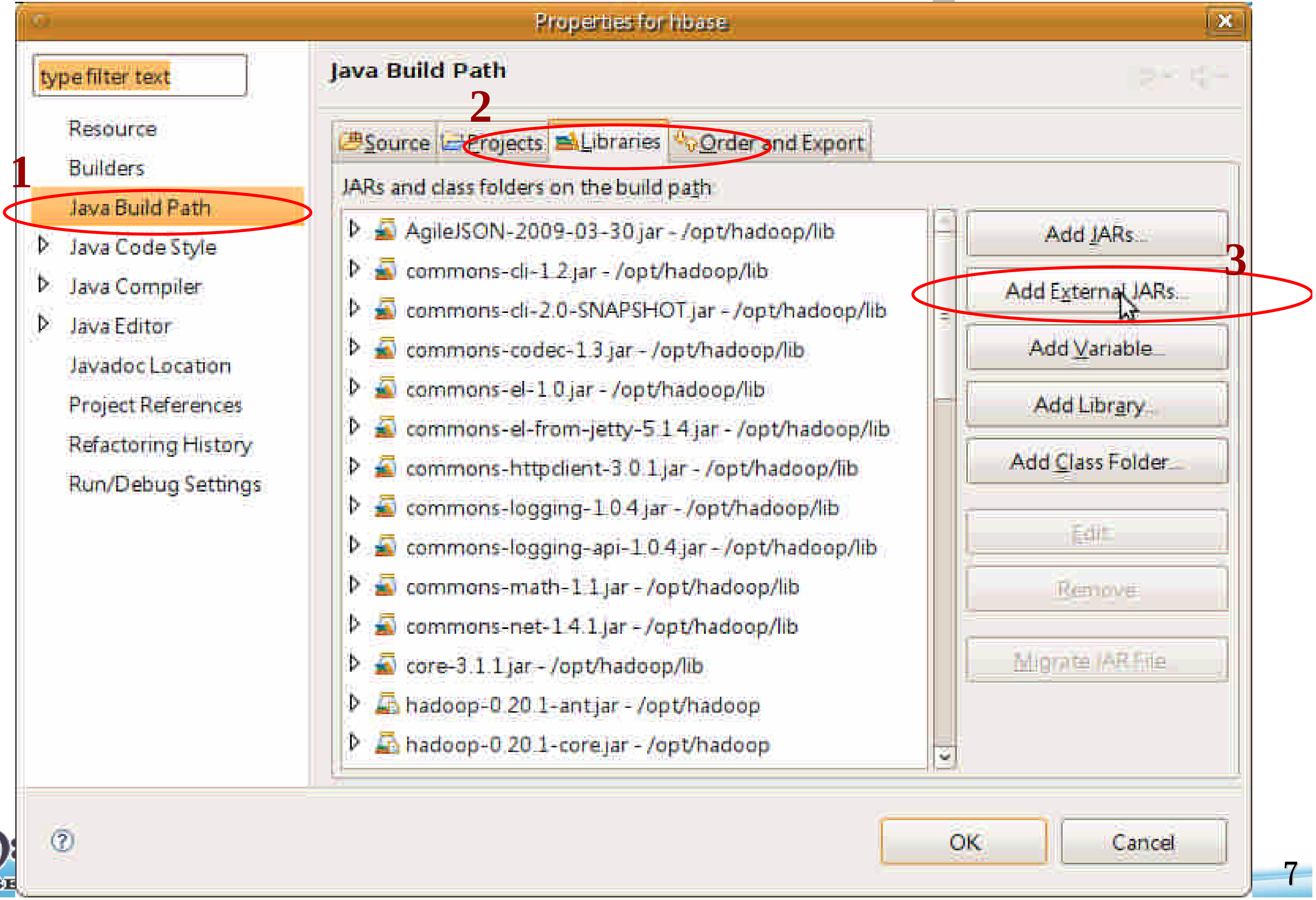
```
package tsmc;

import java.io.BufferedReader;

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

```
e2HBase(St                    table name, Stri
ws IOException {

Reader fi = new BufferedReader(
new FileReader(new File(file_in)));

ine;
line = fi.readLine()) != null) {

ng[] str = line.split(";");
length = str.length -1;

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

em.out.println();

();

c void main(String args[]) throws IOException {
```

```
<terminated> MR                16/bin/java (2010/2/1 下午 7:57:14)
10/02/01 19:57                 Metrics with processName=JobTracker, sessionId=
10/02/01 19:57                ionsParser for parsing the arguments. Applications should implement Tool for the same.
10/02/01 19:57                e set.  User classes may not be found. See JobConf(Class) or JobConf#setJar(String).
10/02/01 19:57                ronment:zookeeper.version=3.2.1-808558, built on 08/27/2009 18:48 GMT
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.
```

在建立好的專案上點選右鍵，並選擇 properties

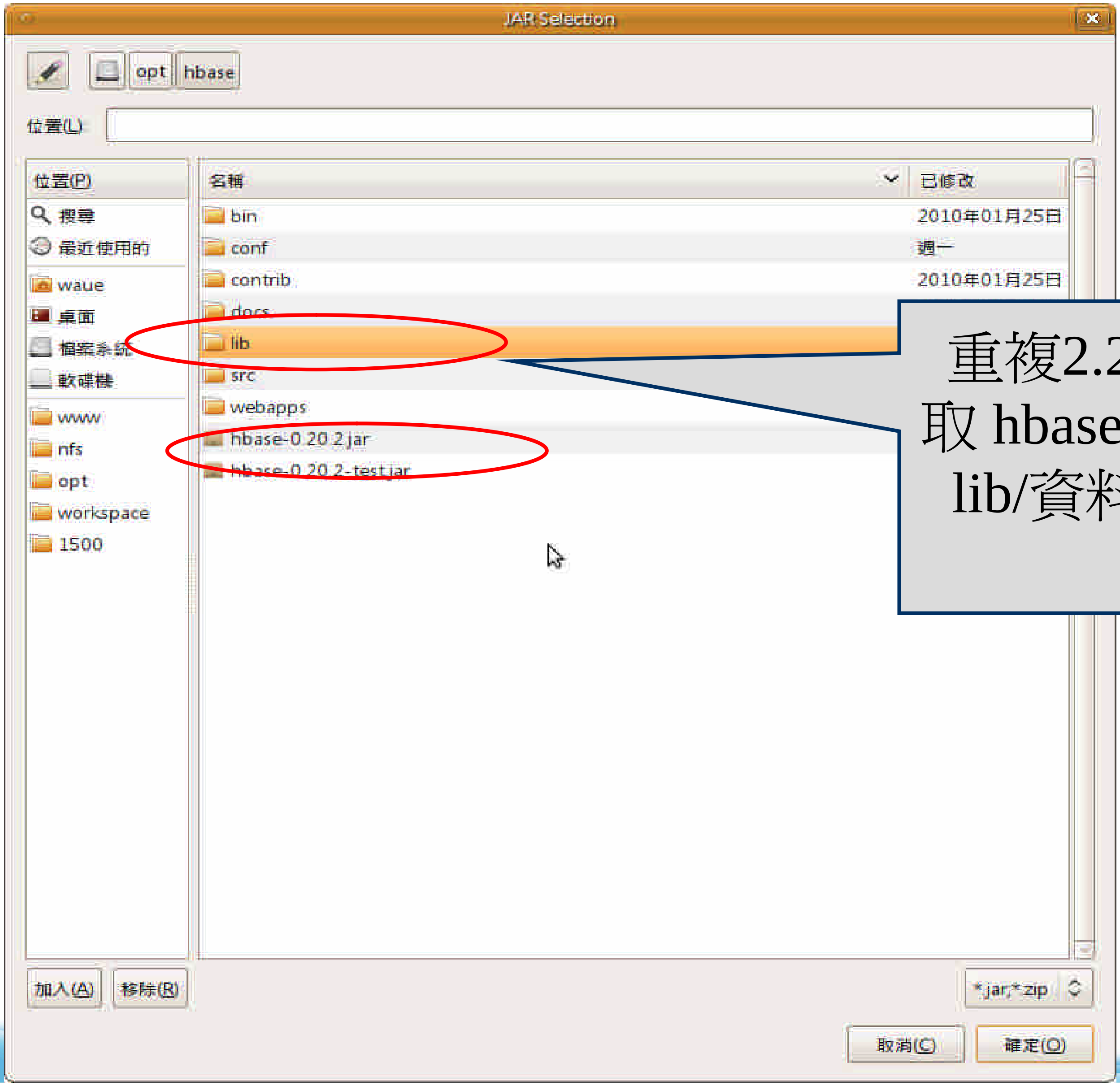
2.2 增加 專案的 Classpath



JARs and class folders on the build path

```
AgileJSON-2009-03-30.jar - /opt/hadoop/lib
commons-cli-1.2.jar - /opt/hadoop/lib
commons-cli-2.0-SNAPSHOT.jar - /opt/hadoop/lib
commons-codec-1.3.jar - /opt/hadoop/lib
commons-el-1.0.jar - /opt/hadoop/lib
commons-el-from-jetty-5.1.4.jar - /opt/hadoop/lib
commons-httpclient-3.0.1.jar - /opt/hadoop/lib
commons-logging-1.0.4.jar - /opt/hadoop/lib
commons-logging-api-1.0.4.jar - /opt/hadoop/lib
commons-math-1.1.jar - /opt/hadoop/lib
commons-net-1.4.1.jar - /opt/hadoop/lib
core-3.1.1.jar - /opt/hadoop/lib
hadoop-0.20.1-ant.jar - /opt/hadoop
hadoop-0.20.1-core.jar - /opt/hadoop
```

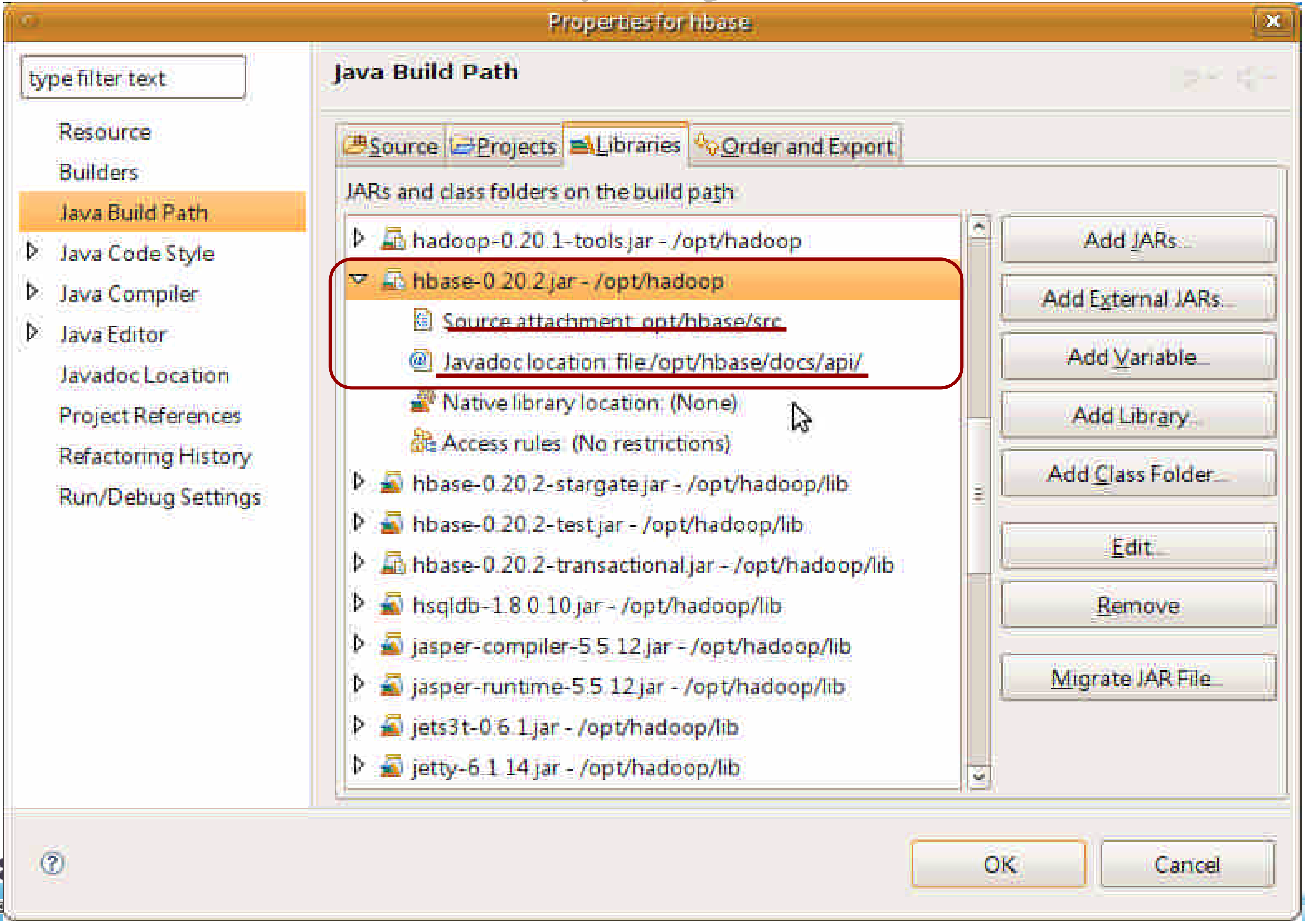
2.3 選擇classpath 的library



名稱	已修改
bin	2010年01月25日
conf	週一
contrib	2010年01月25日
docs	
lib	
src	
webapps	
hbase-0.20.2.jar	
hbase-0.20.2-test.jar	

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

2.4 為函式庫增加原始碼、說明檔的配置


Properties for hbase

type filter text

- Resource
- Builders
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- Project References
- Refactoring History
- Run/Debug Settings

Java Build Path

Source | Projects | Libraries | Order and Export

JARs and class folders on the build path

- hadoop-0.20.1-tools.jar - /opt/hadoop
- hbase-0.20.2.jar - /opt/hadoop
  - Source attachment: opt/hbase/src
  - Javadoc location: file:/opt/hbase/docs/api/
  - Native library location: (None)
  - Access rules: (No restrictions)
- hbase-0.20.2-stargate.jar - /opt/hadoop/lib
- hbase-0.20.2-test.jar - /opt/hadoop/lib
- hbase-0.20.2-transactional.jar - /opt/hadoop/lib
- hsqldb-1.8.0.10.jar - /opt/hadoop/lib
- jasper-compiler-5.5.12.jar - /opt/hadoop/lib
- jasper-runtime-5.5.12.jar - /opt/hadoop/lib
- jets3t-0.6.1.jar - /opt/hadoop/lib
- jetty-6.1.14.jar - /opt/hadoop/lib

HBase 程式設計

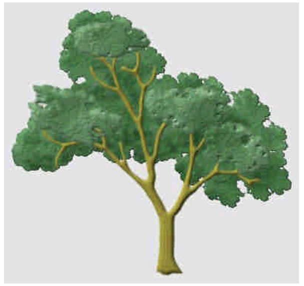
此篇介紹如何撰寫HBase程式

常用的HBase API 說明

實做 I/O 操作

搭配Map Reduce 運算

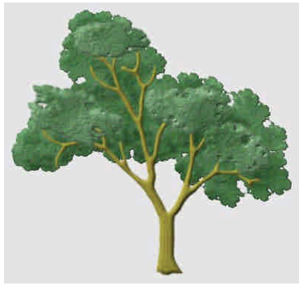
財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING



HBase 程式設計

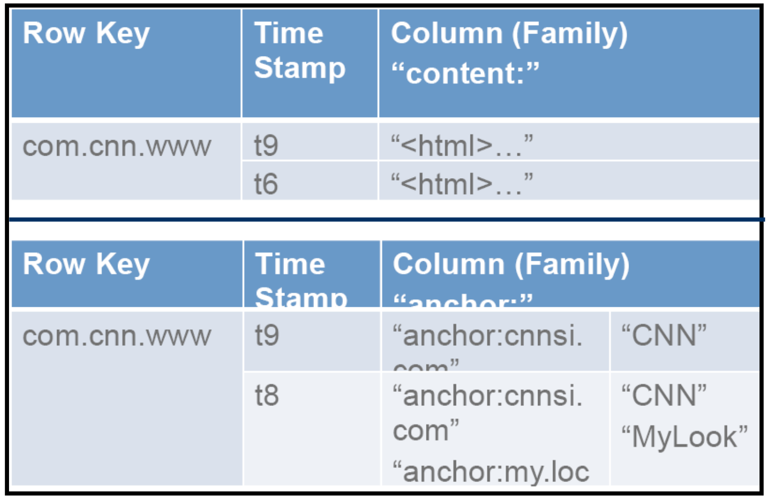
常用的HBase API 說明

財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING



HTable 成員

- Table, Family
- Column, Qualifier
- Row, TimeStamp,
- Cell, Lock



Row Key	Time Stamp	Column (Family) “content:”
com.cnn.www	t9	“<html>…”
	t6	“<html>…”

Row Key	Time Stamp	Column (Family) “anchor:”	
com.cnn.www	t9	“anchor:cnnsi. com”	“CNN”
	t8	“anchor:cnnsi. com” “anchor:my.loc	“CNN” “MyLook”

Row Key	Time Stamp	Column (Family) “content:”	Column (Family) “anchor:”	
com.cnn.www	t9	“<html>…”	“anchor:cnnsi.com”	“CNN”
	t8		“anchor:cnnsi.com”	“CNN”
			“anchor:my.lock.ca”	“MyLook”
	t6	“<html>…”		

HBase 常用函式

- HBaseAdmin
- HBaseConfiguration $\Big\} \rightarrow$ Database
- HTable $\longrightarrow$ Table
- HTableDescriptor $\longrightarrow$ Family
- Put
- Get
- Scanner $\Bigg\} \longrightarrow$ Column Qualifier

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	addColumns	(byte[][] columns)
Get	addFamily	(byte[] family)
TimeRange	getTimeRange	()
Get	setTimeRange	(long minStamp, long maxStamp)
Get	setFilter	(Filter filter)

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	addColumns	(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	()

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:

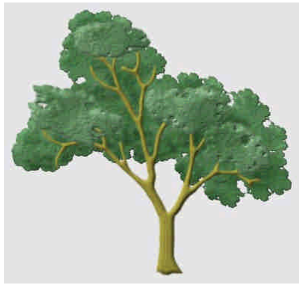
< row-length >	< row>	< column-family-length >	< column-family >	< column-qualifier >	< time-stamp >	< key-type >

- 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.

HBase 程式設計

實做I/O操作

財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING



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
   {
      HTableDescriptor htd = new HTableDescriptor( tablename );
      HColumnDescriptor col = new HColumnDescriptor( familyname );
      htd.addFamily ( col );
      HBaseConfiguration config = new HBaseConfiguration();
      HBaseAdmin admin = new HBaseAdmin(config);
      if(admin.tableExists(tablename))
      {     return ()   }
      admin.createTable(htd);
   }
```

範例二：Put資料進Column

<指令>

```
put ‘表名’, ‘列’ , ‘family:qualifier’, ‘值’ , [‘時間’]
```

```
> put 'tablename','row1', 'family1:qua1', '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>

```
static String getColumn ( String tablename, String row, String family,
          String column ) {
      HBaseConfiguration conf = new HBaseConfiguration();
      String ret = "";
      HTable table;
      try {
          table = new HTable(conf, Bytes.toBytes(tablename));
          Get g = new Get(Bytes.toBytes(row));
          Result rowResult = table.get(g);
          ret = Bytes.toString(rowResult.getValue(Bytes.toBytes(family + “:” + 
      column)));
          table.close();
      } catch (IOException e) {
          e.printStackTrace();  }
      return ret;
}
```

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) {
        HBaseConfiguration conf = new HBaseConfiguration();
        HTable table;
        try {
            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++;
            }
        } catch (IOException e) {
            e.printStackTrace();    }
}
```

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
```

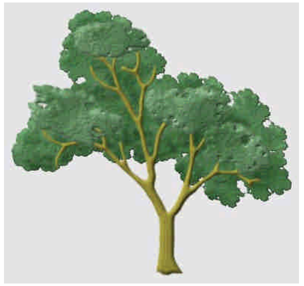
範例五： 刪除資料表 <程式碼>

```
static void drop ( String tablename ) {
        HBaseConfiguration conf = new HBaseConfiguration();
        try {
             HBaseAdmin admin = new HBaseAdmin(conf);
            if (admin.tableExists(tablename))
            {
                        admin.disableTable(tablename);
                        admin.deleteTable(tablename);
            }else{
                        System.out.println("Table [" + tablename+ "] was not
        found!"); }
        } catch (IOException e) {
            e.printStackTrace();    }
}
```

HBase 程式設計

MapReduce與 HBase的搭配

財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING



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：

```
---------------------------
        $ hbase shell
        > 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
 ......(略)
---------------------------
```

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

2.　運算完後，程式將執行結果放在hbase的wordcount資料表內

參考：

1.程式碼改編於： http://blog.ring.idv.tw/comment.ser?i=337

2.hbase 運作 mapreduce 程式的方法參考於：http://wiki.apache.org/hadoop/Hbase/MapReduce

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

執行：

```
---------------------------
hadoop jar XX.jar LoadHBaseMapper <hdfs_output>
---------------------------
```

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(Byt
        es.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 = "tsmc";
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);
}}
```

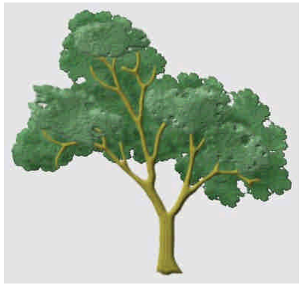
HBase 程式設計

其他用法補充

HBase內contrib的項目，如

Trancational

Thrift



財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

1. 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

1.1 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>
```

1.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();
        conf.addResource(new Path("/opt/hbase/conf/hbase-site.xml"));
        IndexedTableAdmin admin = null;
        admin = new IndexedTableAdmin(conf);
        admin.addIndex(Bytes.toBytes(TableName), new
        IndexSpecification(
                        IndexID, Bytes.toBytes(IndexColumn)));
}}
```

1.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);
}
```

2. 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
  - …..

2.1 Thrift PHP Example

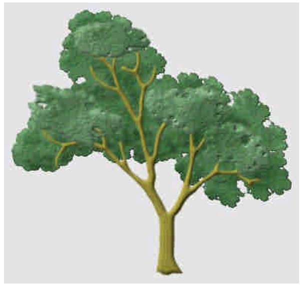
- Insert data into HBase by PHP thrift client

```
$mutations = array(
  new Mutation( array(
    'column' => 'entry:num',
    'value' => array('a','b','c')
  ) ), );
$client->mutateRow( $t, $row, $mutations );
```

H·BASE

案例演練

利用一個虛擬的案例來運用之前的
程式碼



TSMC餐廳開張囉！

- 故事背景：
  - ◆ TSMC的第101廠即將開張，預計此廠員工將有200萬人
- 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

1. 建立商店資料

假設：目前有四間商店進駐TSMC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	StarBucks	04	50	50	20					

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);
}
}
```

1.a 執行結果

Table: TSMC

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

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();
}
```

1.b 執行結果

	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	StarBucks	04	50	50	20		

1. 螢幕輸出結果

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

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

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

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

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

……
```

2. 用Hadoop的Map Reduce運算並把結果匯入HTable

<map 程式碼>

```
public class TSMC2Count {
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);
}}
```

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 = "tsmc";
Configuration conf = new Configuration();
conf.set(TableOutputFormat.OUTPUT_TABLE, tablename);
Job job = new Job(conf, "Count to tsmc");
job.setJarByClass(TSMC2Count.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);
}
}
```

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 'tsmc'
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. 計算當天營業額

- 計算每間商店的營業額
  - Σ（<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. Hadoop 來源與輸出皆為 HBase

<map 程式碼>

```
public class TSMC3CalculateMR {
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))); }}
```

<reduce程式碼>

```
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. Hadoop 來源與輸出皆為 HBase

< Main 程式碼>

```
public static void main(String args[]) throws
          Exception {
String tablename = "tsmc";
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(TSMC3CalculateMR.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(tablena
          me, myScan, HtMap.class,
               Text.class, Text.class, job);
TableMapReduceUtil.initTableReducerJob(tablena
          me, HtReduce.class, job);
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

```
> scan ‘tsmc’
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. 執行結果

	Detail		Products				Turnover				
	Name	Locate	P1	P2	P3	P4	P1	P2	P3	P4	Sum
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

4. 產生最終報表

- TSMC 高層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

4.a 建立Indexed Table

```
public class TSMC4SortTurnover {
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 {
TSMC4SortTurnover tt = new TSMC4SortTurnover();
tt.addIndexToTurnover("tsmc", "Sum", "Turnover:Sum");
tt.readSortedValGreater("130");
}}
```

4.a Indexed Table 輸出結果

```
> scan 'tsmc-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
```

4.b 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 = "tsmc";
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();
}
```

列出最後結果

- 營業額大於130元者

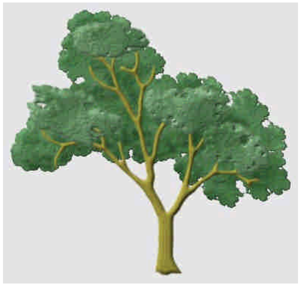
```
GunLong 's turnover is 140 $.
StarBucks 's turnover is 170 $.
```

其他專案

介紹其他與HDFS相關的類資料庫專案

PIG

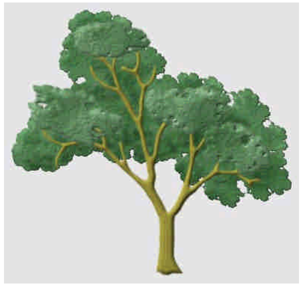
HIVE



其他專案

PIG

- Motivation
- Pig Latin
- Why a new Language ?
- How it works
- Branch mark
- Example
- More Comments
- Conclusions



Motivation

- Map Reduce is very powerful,
- but:
  - – It requires a Java programmer.
  - – User has to re-invent common
  - functionality (join, filter, etc.)

Pig Latin

- Pig provides a higher level language, Pig Latin, that:
- Increases productivity. In one test
  - 10 lines of Pig Latin ≈ 200 lines of Java.
  - What took 4 hours to write in Java took 15 minutes in Pig Latin.
- Opens the system to non-Java programmers.
- Provides common operations like join, group, filter, sort.

Why a new Language ?

- Pig Latin is a data flow language rather than procedural or declarative.
- User code and existing binaries can be included almost anywhere.
- Metadata not required, but used when available.
- Support for nested types.
- Operates on files in HDFS.

How it works

Pig Latin

```
A = LOAD 'myfile'
    AS (x, y, z);
B = FILTER A by x > 0;
C = GROUP B BY x;
D = FOREACH A GENERATE
    x, COUNT(B);
STORE D INTO 'output';
```

pig.jar:

- parses
- checks
- optimizes
- plans execution
- submits jar to Hadoop
- monitors job progress

Execution Plan

Map:
  Filter

Reduce:
  Count

Branch mark

- Release 0.2.0 is at 1.6x MR
- Run date: January 4, 2010, run against 0.6 branch as of that day, Almost be 1.03 x MR

Example

- Let’s count the number of times each user

```
log = LOAD ‘excite-small.log’
AS (user, timestamp, query);
grpd = GROUP log BY user;
cntd = FOREACH grpd GENERATE group, COUNT(log);
STORE cntd INTO ‘output’;
```

- Results:

```
002BB5A52580A8ED 18
005BD9CD3AC6BB38 18
```

More Comments

Pig Command	What it does
load	Read data from file system.
store	Write data to file system.
foreach	Apply expression to each record and output one or more records.
filter	Apply predicate and remove records that do not return true.
group/cogroup	Collect records with the same key from one or more inputs.
join	Join two or more inputs based on a key.
order	Sort records based on a key.
distinct	Remove duplicate records.
union	Merge two data sets.
split	Split data into 2 or more sets, based on filter conditions.
stream	Send all records through a user provided binary.
dump	Write output to stdout.
limit	Limit the number of records.

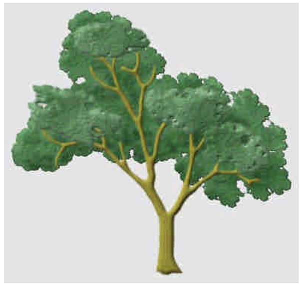
Conclusions

- Opens up the power of Map Reduce.
- Provides common data processing operations.
- Supports rapid iteration of adhoc queries.

其他專案

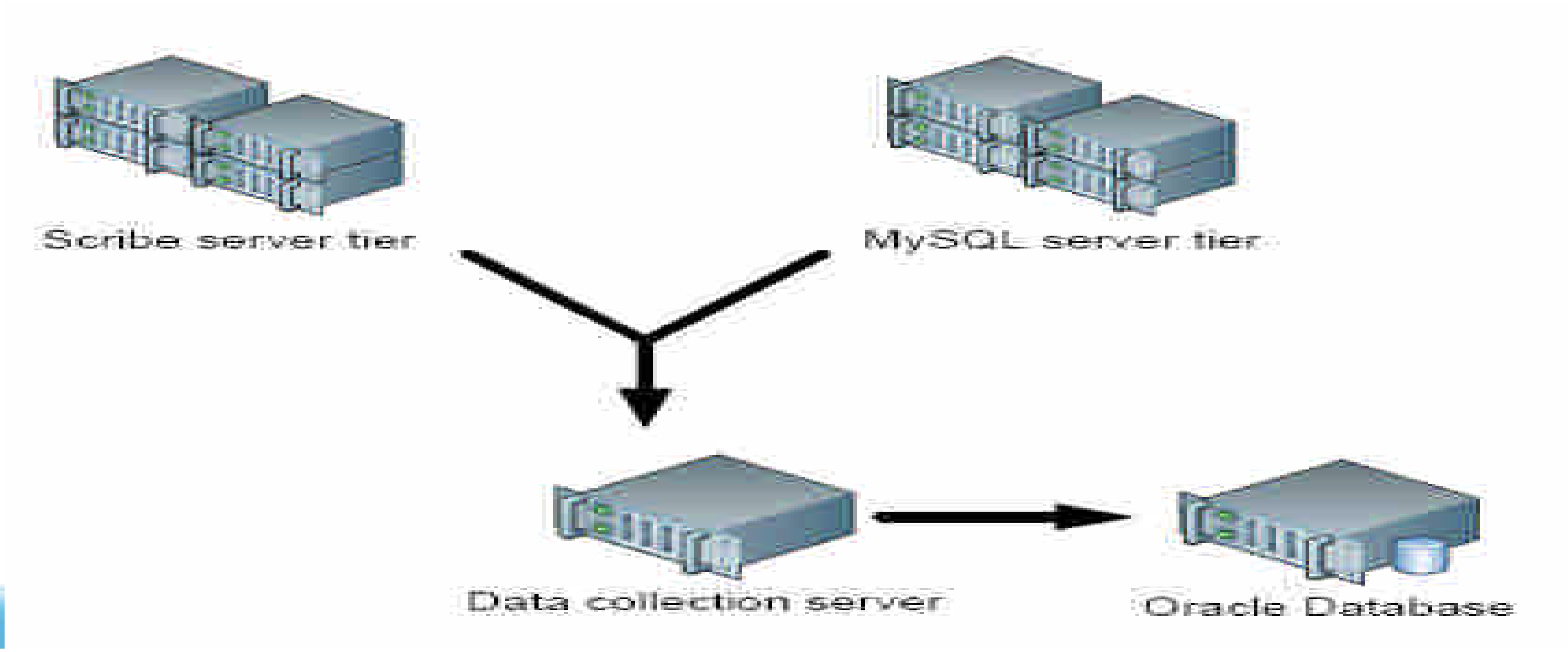
Hive

- Background
- Hive Applications
- Example
- Usages
- Conclusions



Background

- Started at Facebook
- Data was collected by nightly cron jobs into Oracle DB
- "ETL" via hand-coded python
- Grew from 10s of GBs (2006) to 1 TB/day new data (2007), now 10x that.



Hive Applications

- Log processing
- Text mining
- Document indexing
- Customer-facing business intelligence (e.g., Google Analytics)
- Predictive modeling, hypothesis testing

Examples

- load
  - ◆ hive> LOAD DATA INPATH “shakespeare_freq” INTO TABLE shakespeare;
- select
  - ◆ hive> SELECT * FROM shakespeare LIMIT 10;
- join
  - ◆ hive> INSERT OVERWRITE TABLE merged SELECT s.word, s.freq, k.freq FROM shakespeare s JOIN kjv k ON (s.word = k.word) WHERE s.freq >= 1 AND k.freq >= 1;

Usages

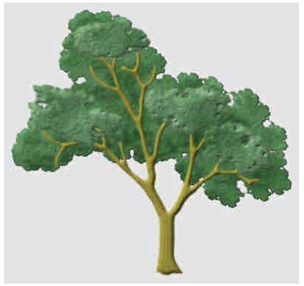
- Creating Tables
- Browsing Tables and Partitions
- Loading Data
- Simple Query
- Partition Based Query
- Joins
- Aggregations
- Multi Table/File Inserts
- Inserting into local files
- Sampling
- Union all
- Array Operations
- Map Operations
- Custom map/reduce scripts
- Co groups
- Altering Tables
- Dropping Tables and Partitions

Conclusions

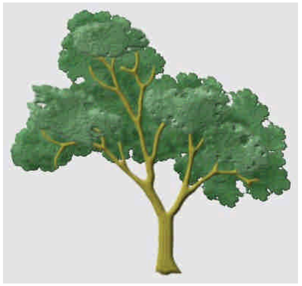
- Supports rapid iteration of ad-hoc queries
- Can perform complex joins with minimal code
- Scales to handle much more data than many similar systems

Questions and Thanks

財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

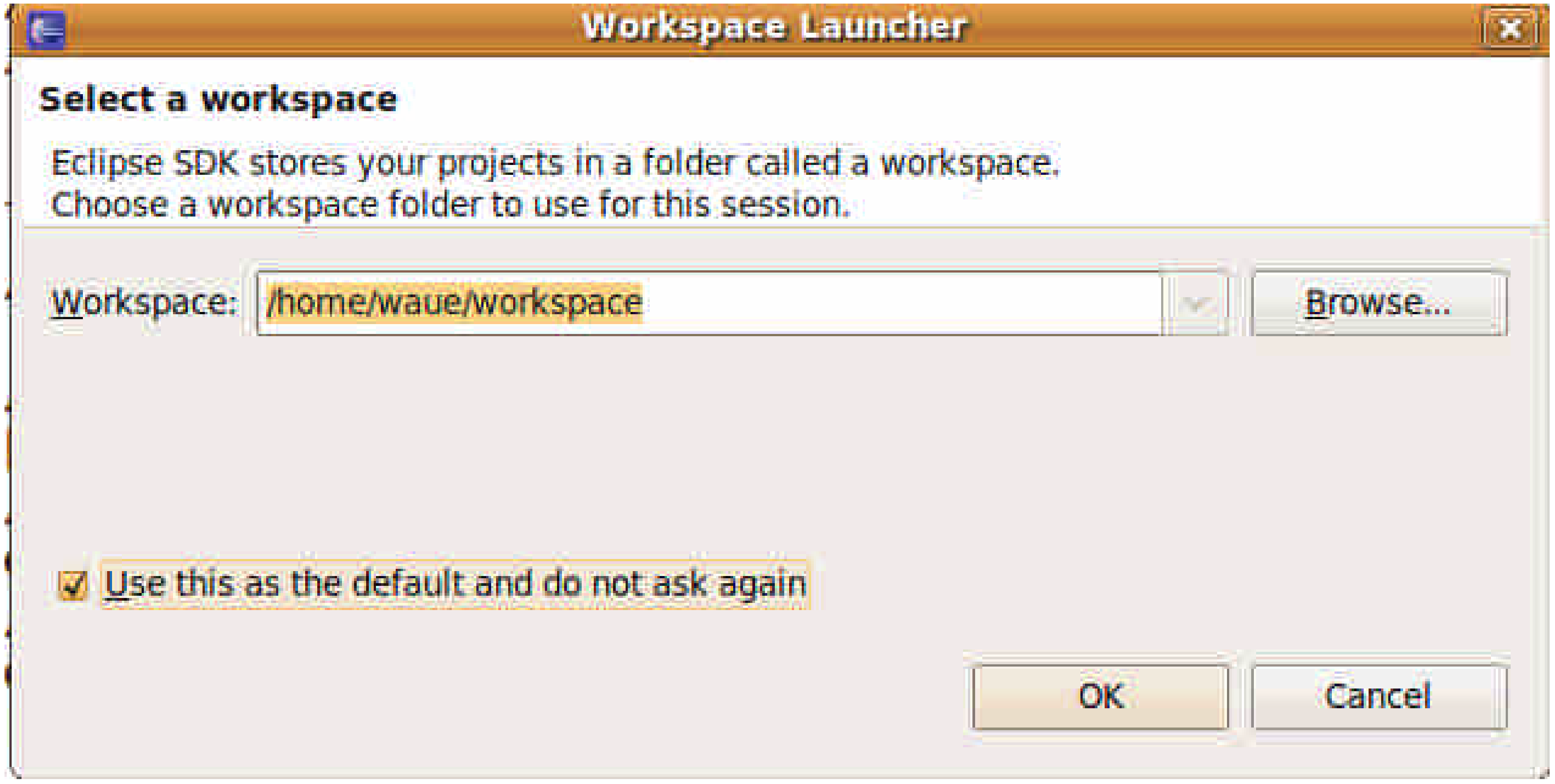


附錄：Hadoop Programming with Eclipse



財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

1 打開Eclipse, 設定專案目錄


Workspace Launcher

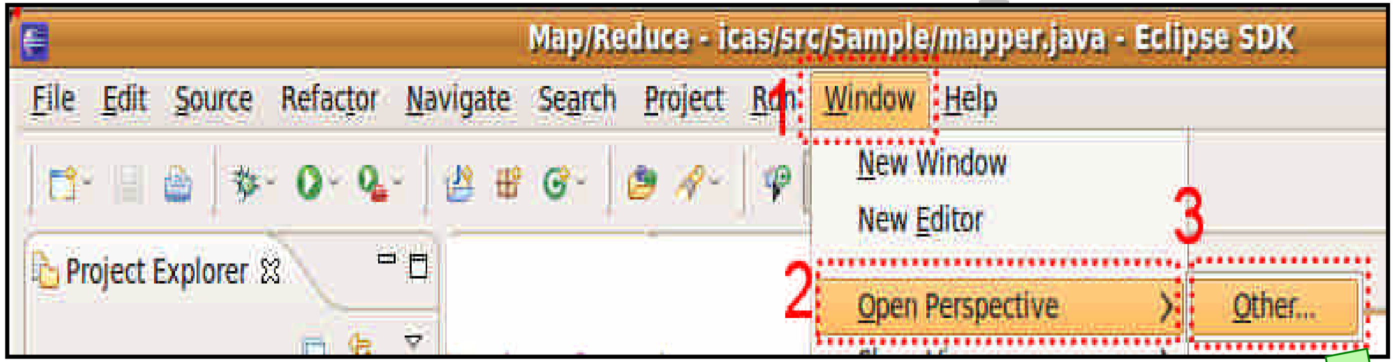
Select a workspace

Eclipse SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

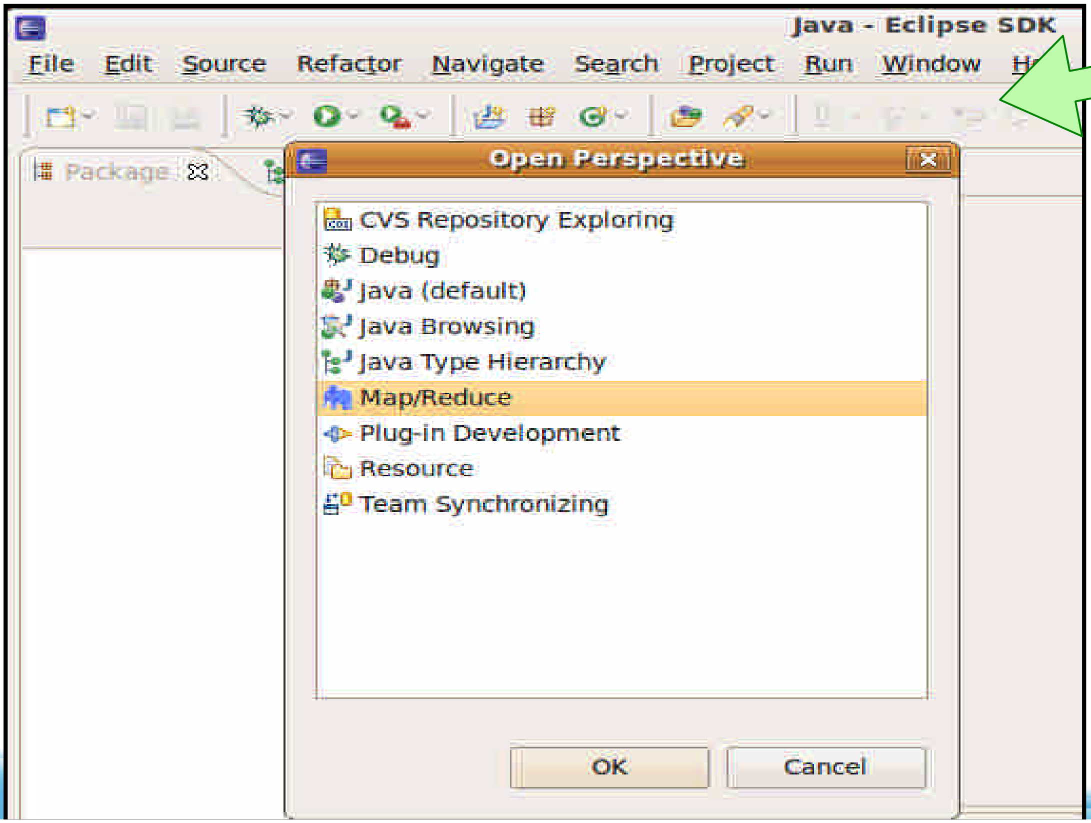
Workspace: /home/waue/workspace

[✓] Use this as the default and do not ask again

2. 使用Hadoop mode視野

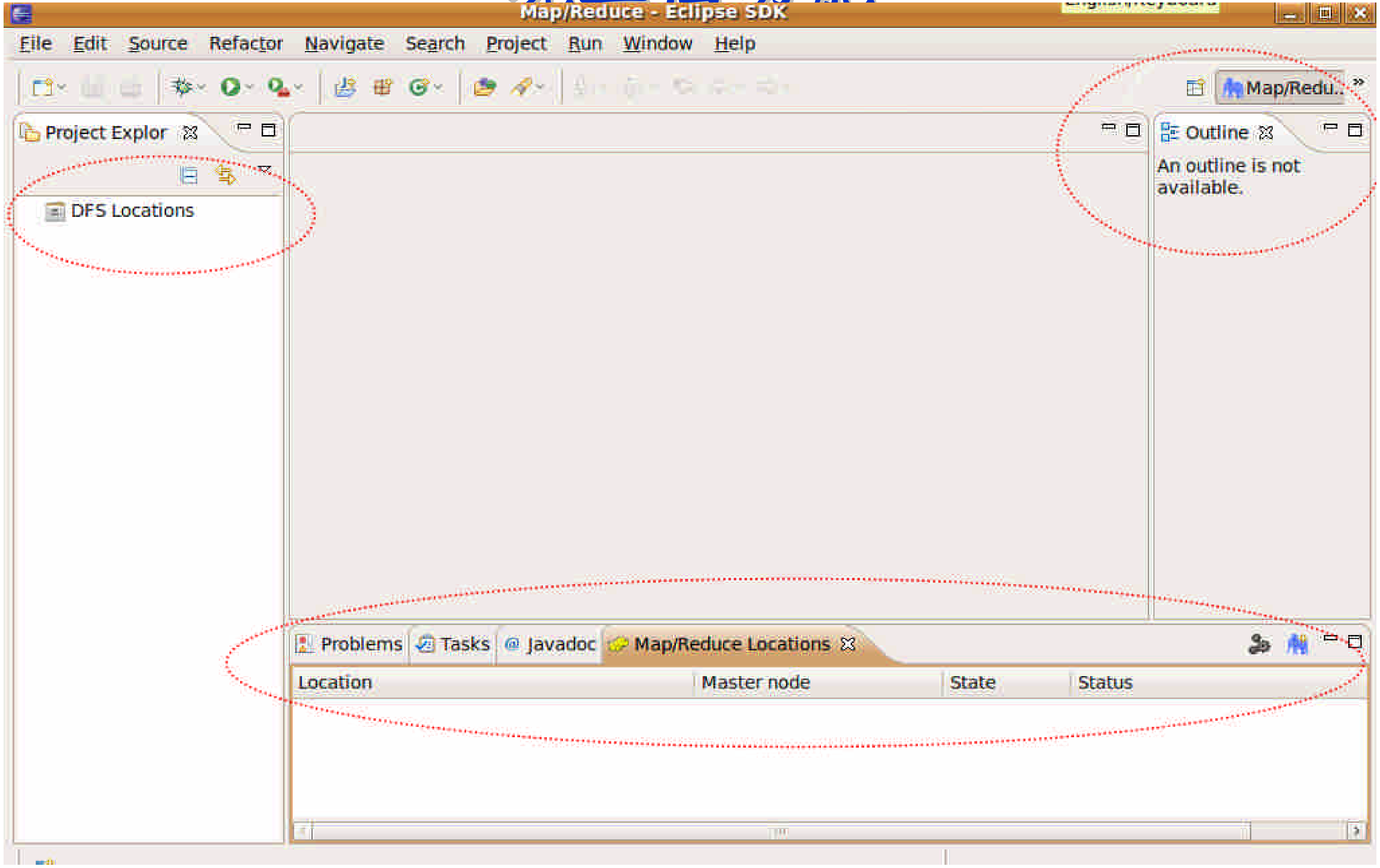


Window → Open Perspective → Other

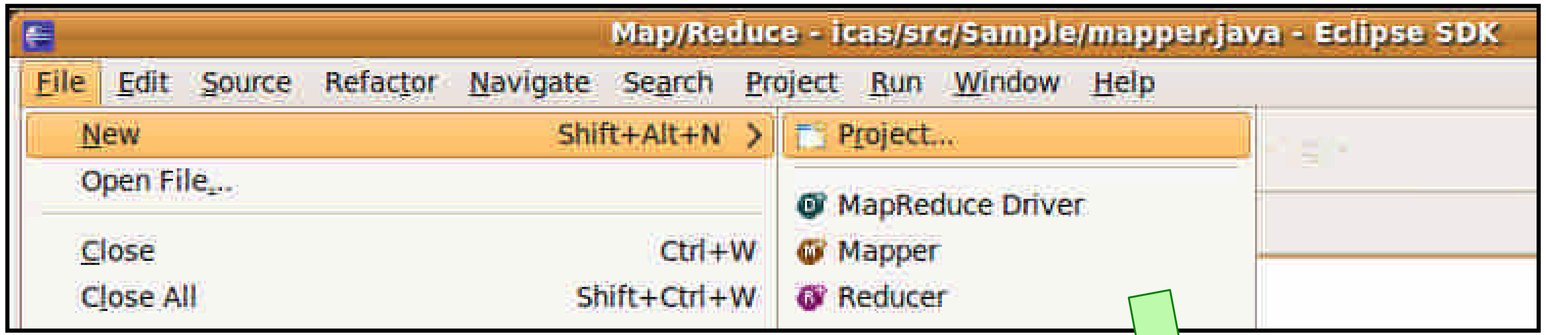


若有看到MapReduce的大象圖示代表Hadoop Eclipse plugin有安裝成功， 若沒有請檢查是否有安之裝正確

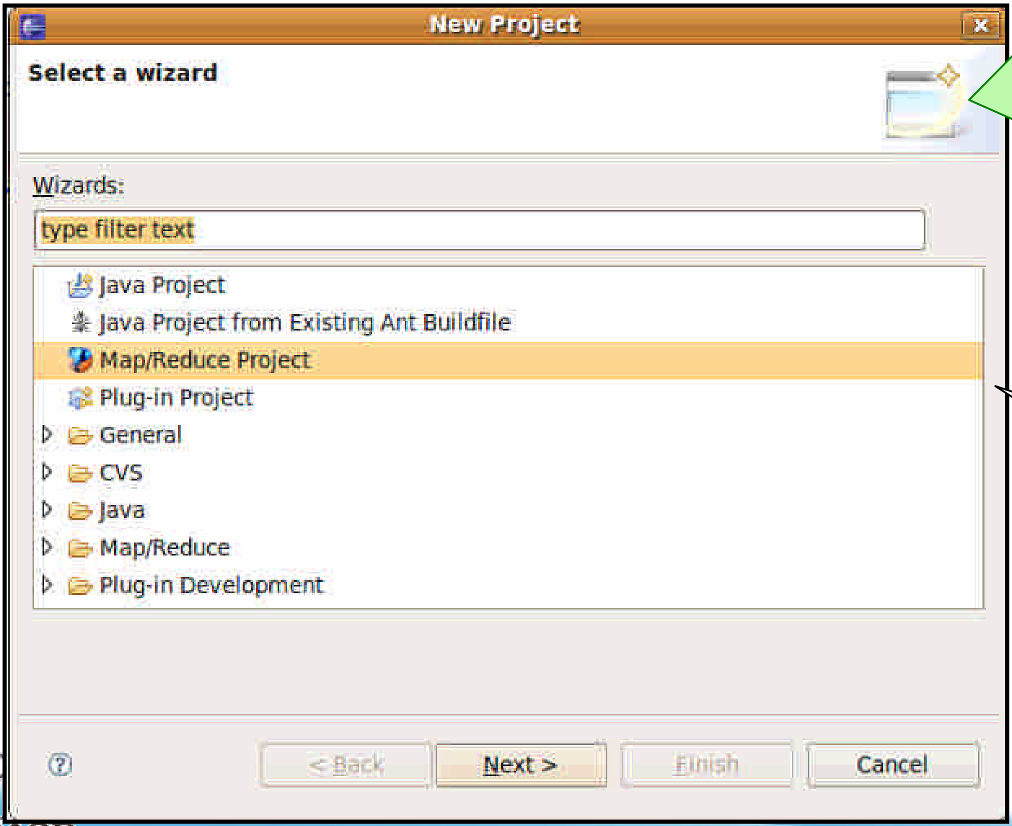
3. 使用Hadoop視野，主畫面將出現三個功能



4.建立一個Hadoop專案



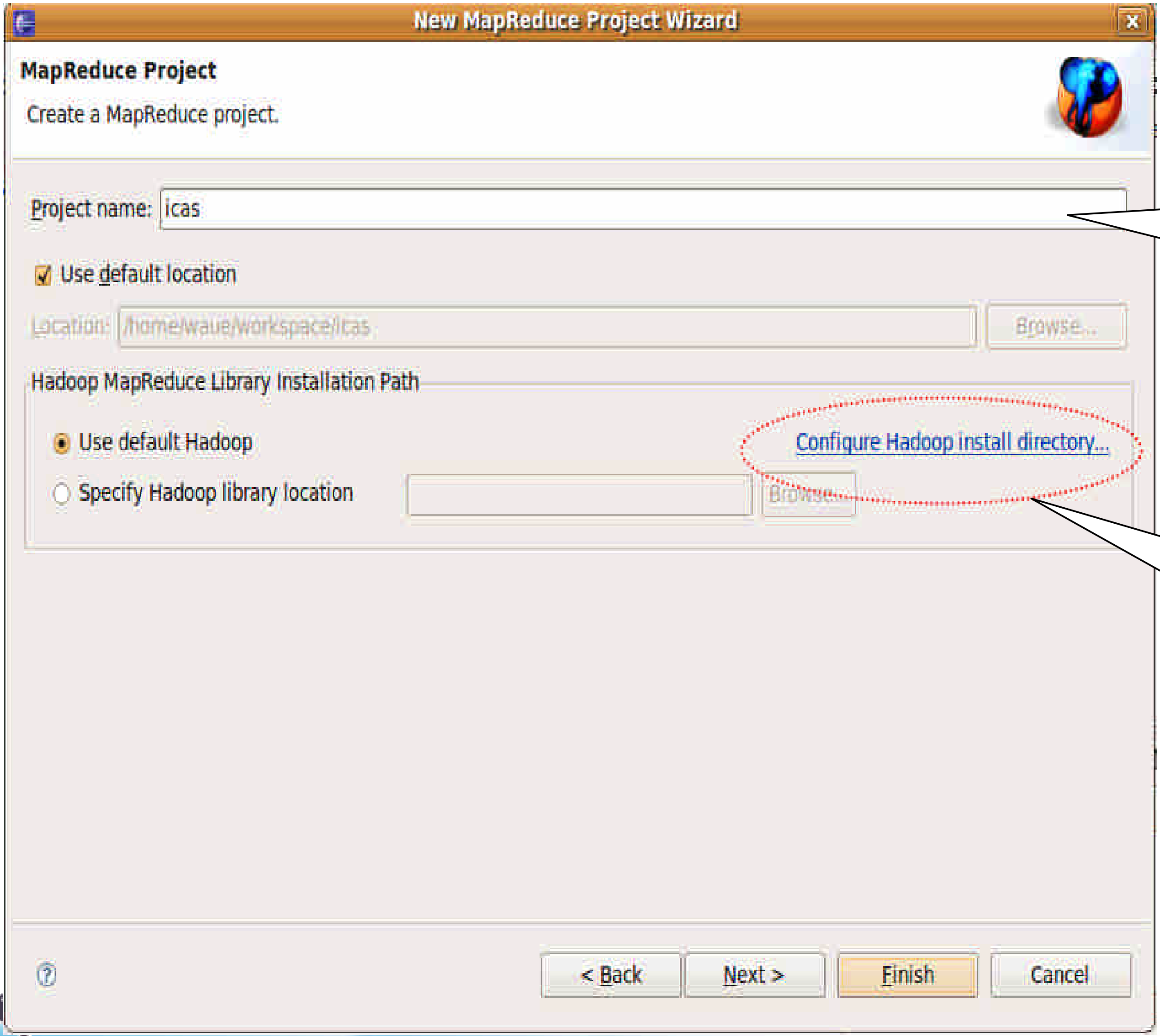
Map/Reduce - icas/src/Sample/mapper.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
New Shift+Alt+N
Open File...
Close Ctrl+W
Close All Shift+Ctrl+W
Project...
MapReduce Driver
Mapper
Reducer



開出新專案

選擇Map/Reduce
專案

4-1. 輸入專案名稱並點選設定 Hadoop安裝路徑



New MapReduce Project Wizard

MapReduce Project

Create a MapReduce project.

Project name: icas

[✓] Use default location

Location: /home/waue/workspace/icas

Hadoop MapReduce Library Installation Path

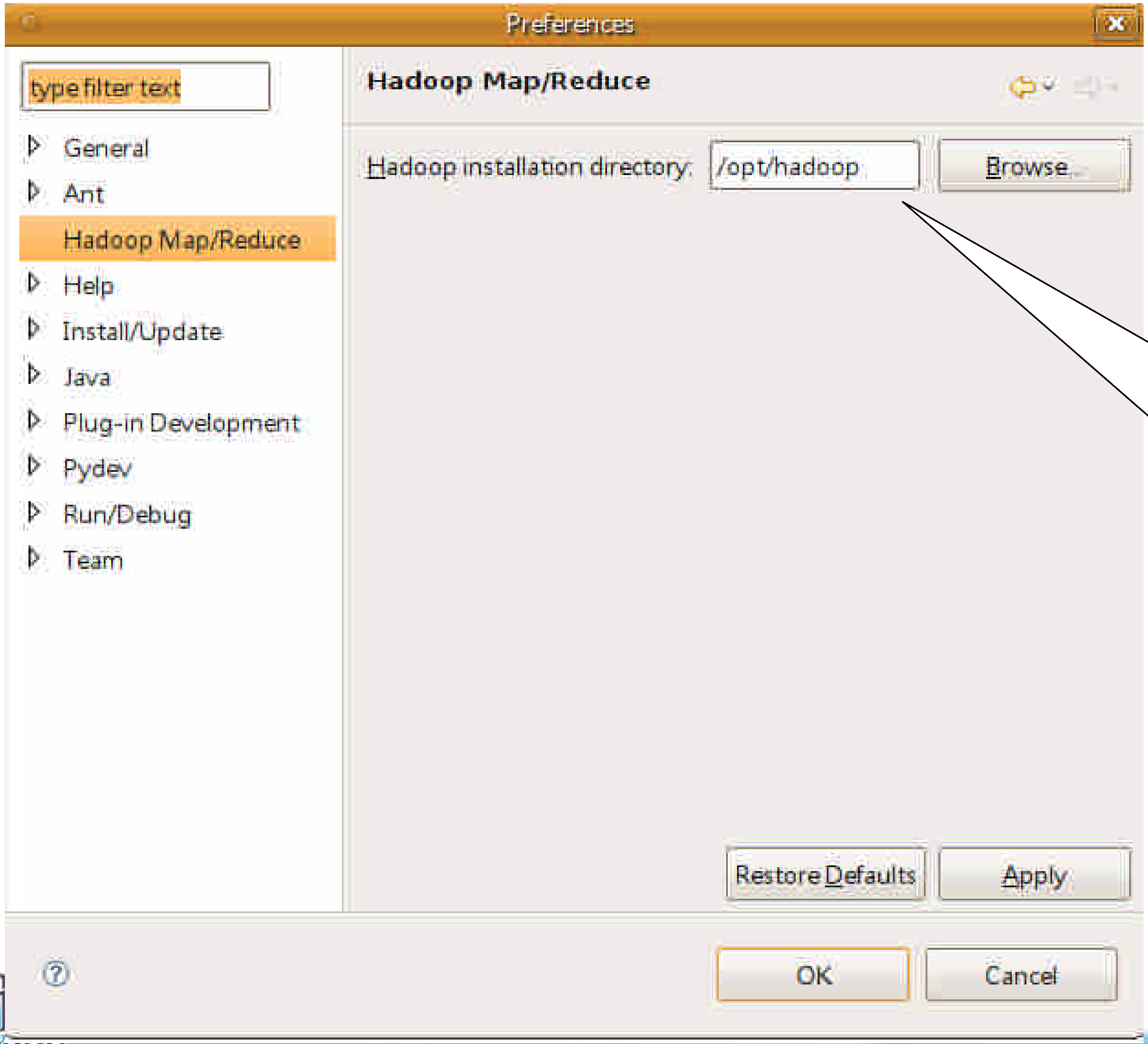
(●) Use default Hadoop

( ) Specify Hadoop library location

由此設定專案名稱

由此設定Hadoop的安裝路徑

4-1-1. 填入Hadoop安裝路徑



Preferences

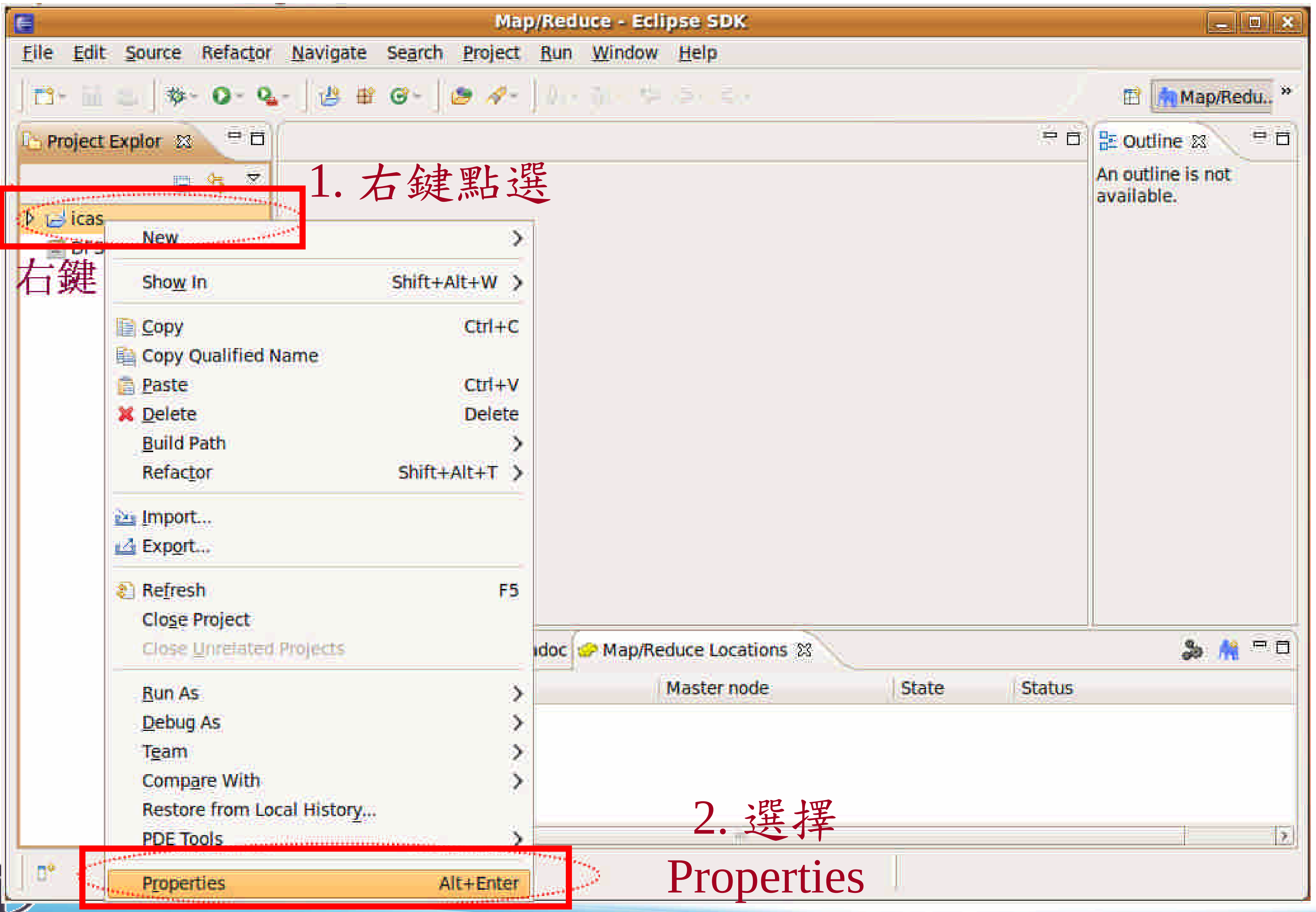
type filter text

Hadoop Map/Reduce

Hadoop installation directory: /opt/hadoop

於此輸入您Hadoop的安裝路徑，之後選擇 ok

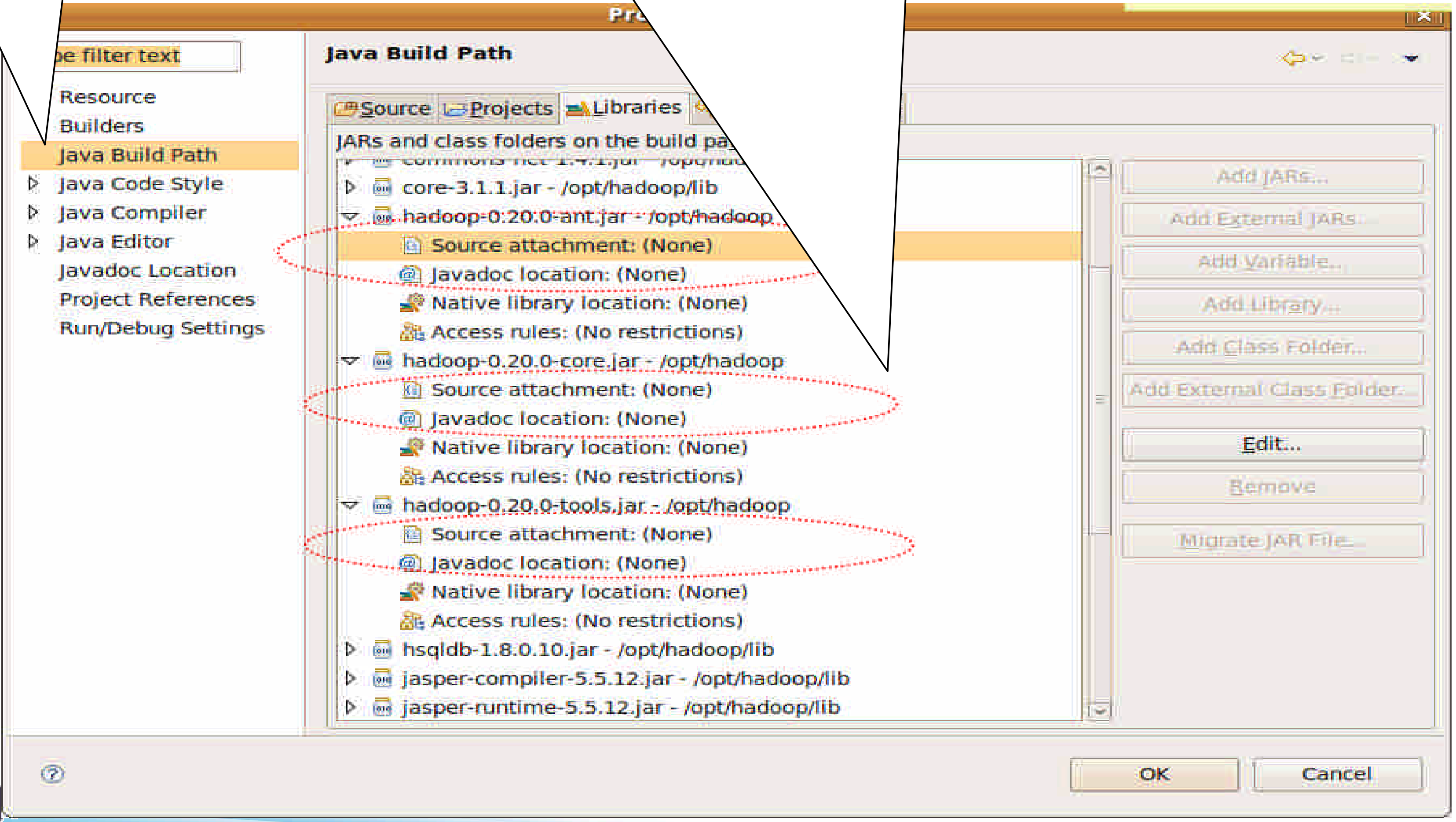
5. 設定Hadoop專案細節



5-1. 設定原始碼與文件路徑

選擇 Java Build Path

以下請輸入正確的Hadoop原始碼與API文件檔路徑，如
source ：/opt/hadoop/src/core/
javadoc：file:/opt/hadoop/docs/api/

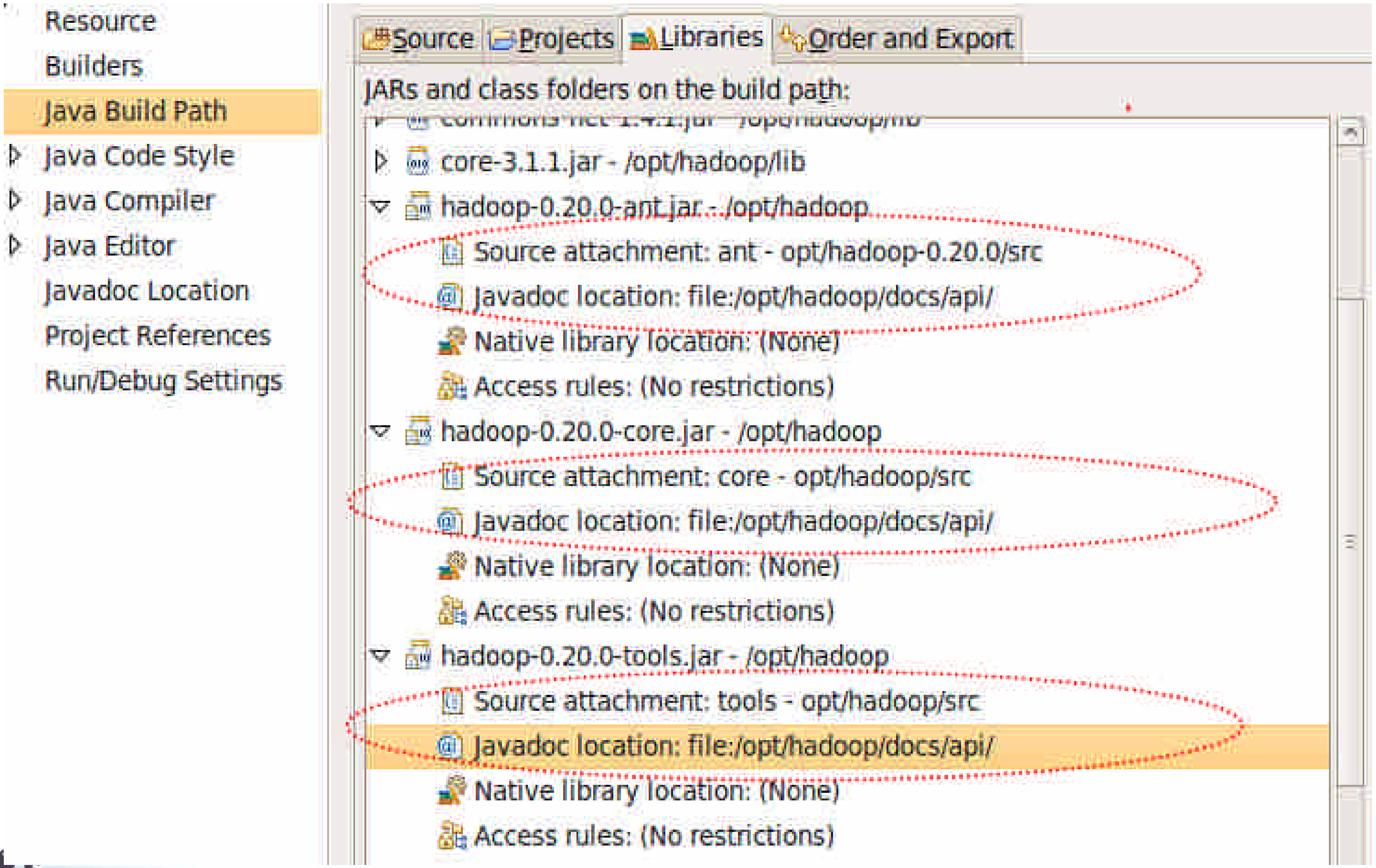

Java Build Path

Source | Projects | Libraries

JARs and class folders on the build pa

```
▷ core-3.1.1.jar - /opt/hadoop/lib
▽ hadoop-0.20.0-ant.jar - /opt/hadoop
    Source attachment: (None)
    Javadoc location: (None)
    Native library location: (None)
    Access rules: (No restrictions)
▽ hadoop-0.20.0-core.jar - /opt/hadoop
    Source attachment: (None)
    Javadoc location: (None)
    Native library location: (None)
    Access rules: (No restrictions)
▽ hadoop-0.20.0-tools.jar - /opt/hadoop
    Source attachment: (None)
    Javadoc location: (None)
    Native library location: (None)
    Access rules: (No restrictions)
▷ hsqldb-1.8.0.10.jar - /opt/hadoop/lib
▷ jasper-compiler-5.5.12.jar - /opt/hadoop/lib
▷ jasper-runtime-5.5.12.jar - /opt/hadoop/lib
```

5-1-1. 完成圖

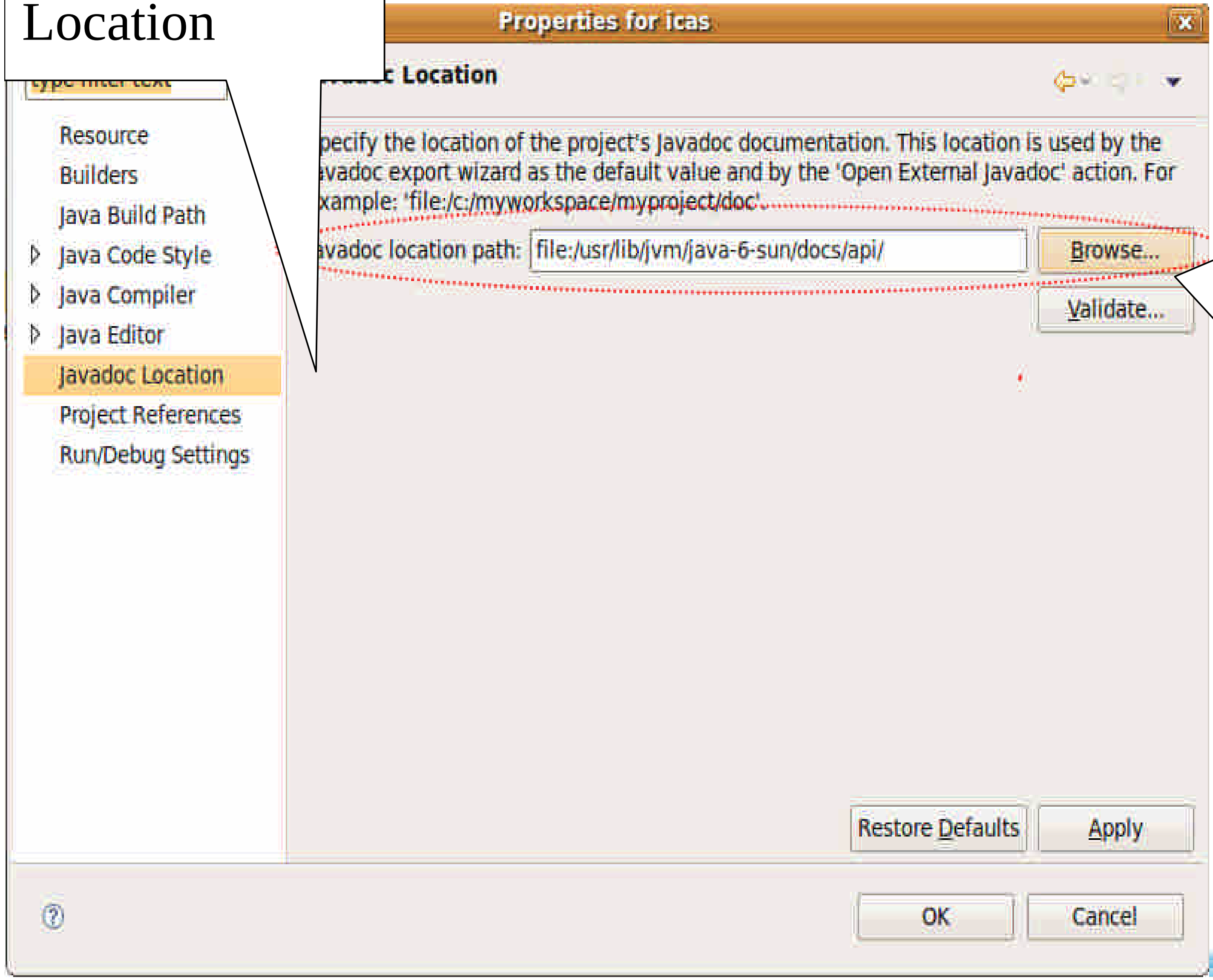


JARs and class folders on the build path:

```
▷ commons-net-1.4.1.jar - /opt/hadoop/lib
▷ core-3.1.1.jar - /opt/hadoop/lib
▽ hadoop-0.20.0-ant.jar - /opt/hadoop
    Source attachment: ant - opt/hadoop-0.20.0/src
    Javadoc location: file:/opt/hadoop/docs/api/
    Native library location: (None)
    Access rules: (No restrictions)
▽ hadoop-0.20.0-core.jar - /opt/hadoop
    Source attachment: core - opt/hadoop/src
    Javadoc location: file:/opt/hadoop/docs/api/
    Native library location: (None)
    Access rules: (No restrictions)
▽ hadoop-0.20.0-tools.jar - /opt/hadoop
    Source attachment: tools - opt/hadoop/src
    Javadoc location: file:/opt/hadoop/docs/api/
    Native library location: (None)
    Access rules: (No restrictions)
```

5-2. 設定java doc的完整路徑

選擇 Javadoc Location

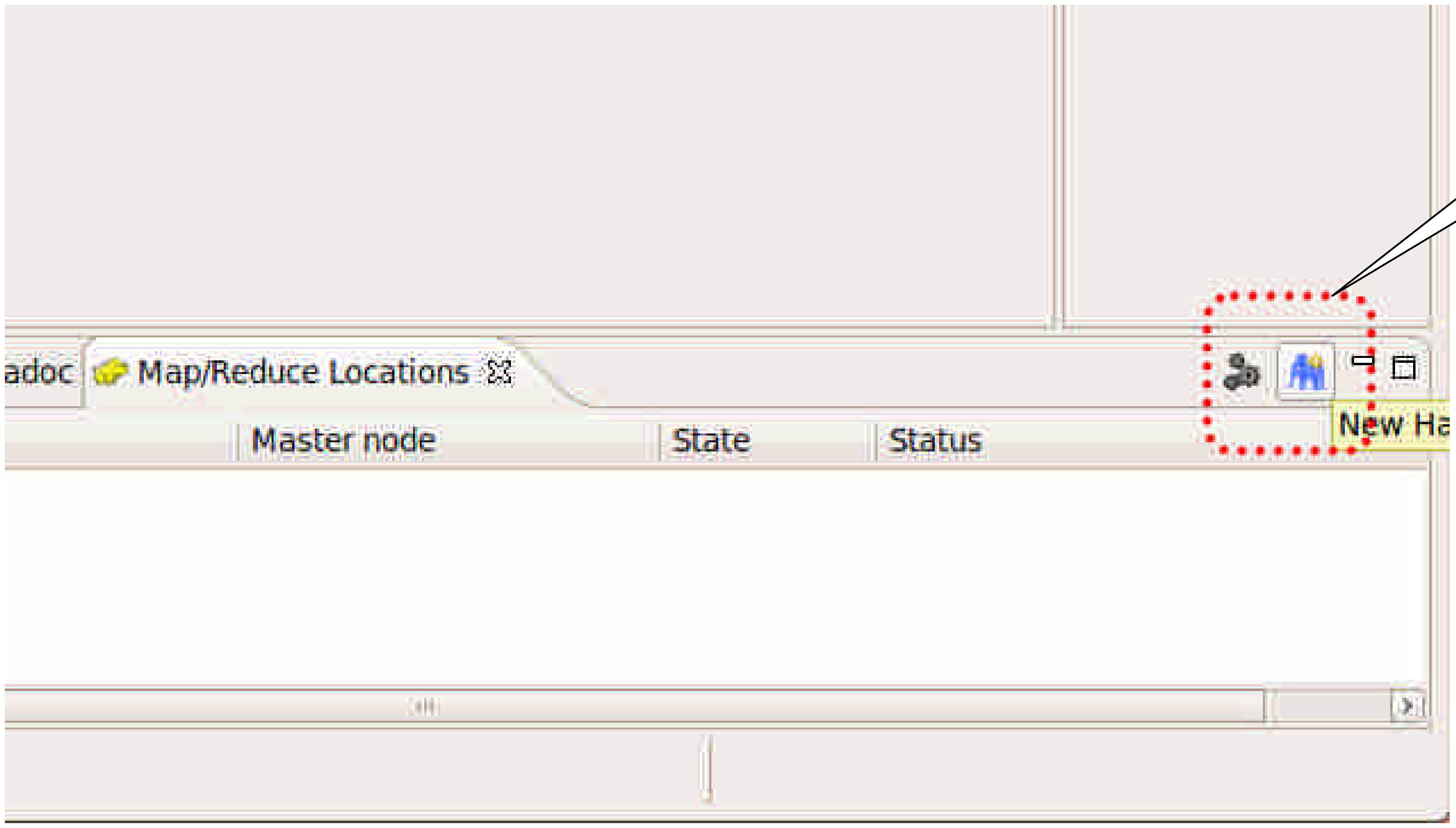


pecify the location of the project's Javadoc documentation. This location is used by the avadoc export wizard as the default value and by the 'Open External Javadoc' action. For xample: 'file:/c:/myworkspace/myproject/doc'.

avadoc location path: file:/usr/lib/jvm/java-6-sun/docs/api/

輸入java 6 的API正確路徑，輸入完後可選擇validate以驗證是否正確

6. 連結Hadoop Server與Eclipse



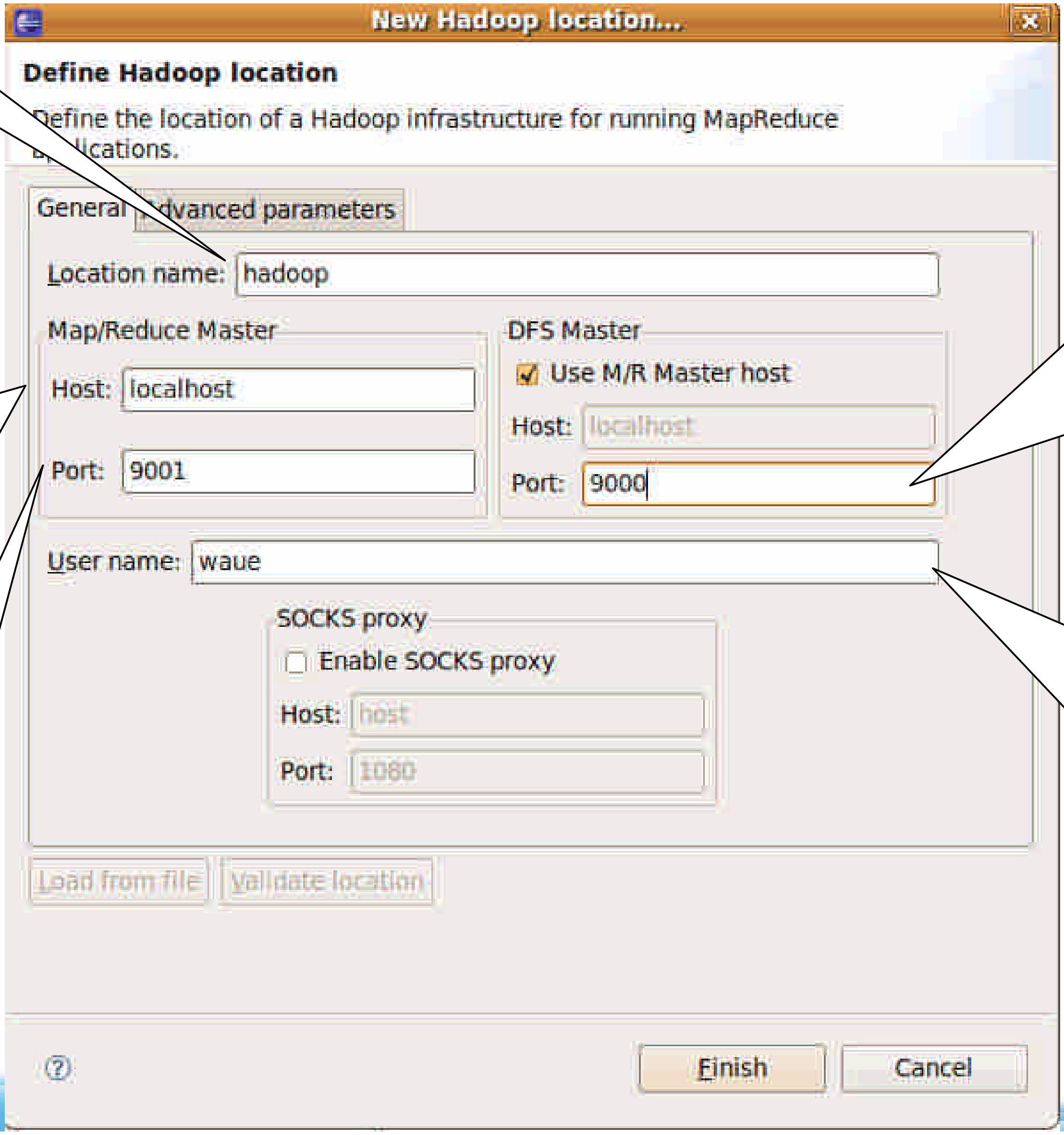
點選此
圖示

6-1 . 設定你要連接的Hadoop主機

任意填一個名稱

輸入主機位址或domain name

MapReduce 監聽的Port (設定於mapred-site.xml)



New Hadoop location...

Define Hadoop location

Define the location of a Hadoop infrastructure for running MapReduce applications.

Location name: hadoop

Map/Reduce Master
- Host: localhost
- Port: 9001

DFS Master
- [✓] Use M/R Master host
- Host: localhost
- Port: 9000

User name: waue

SOCKS proxy
- [ ] Enable SOCKS proxy
- Host: host
- Port: 1080

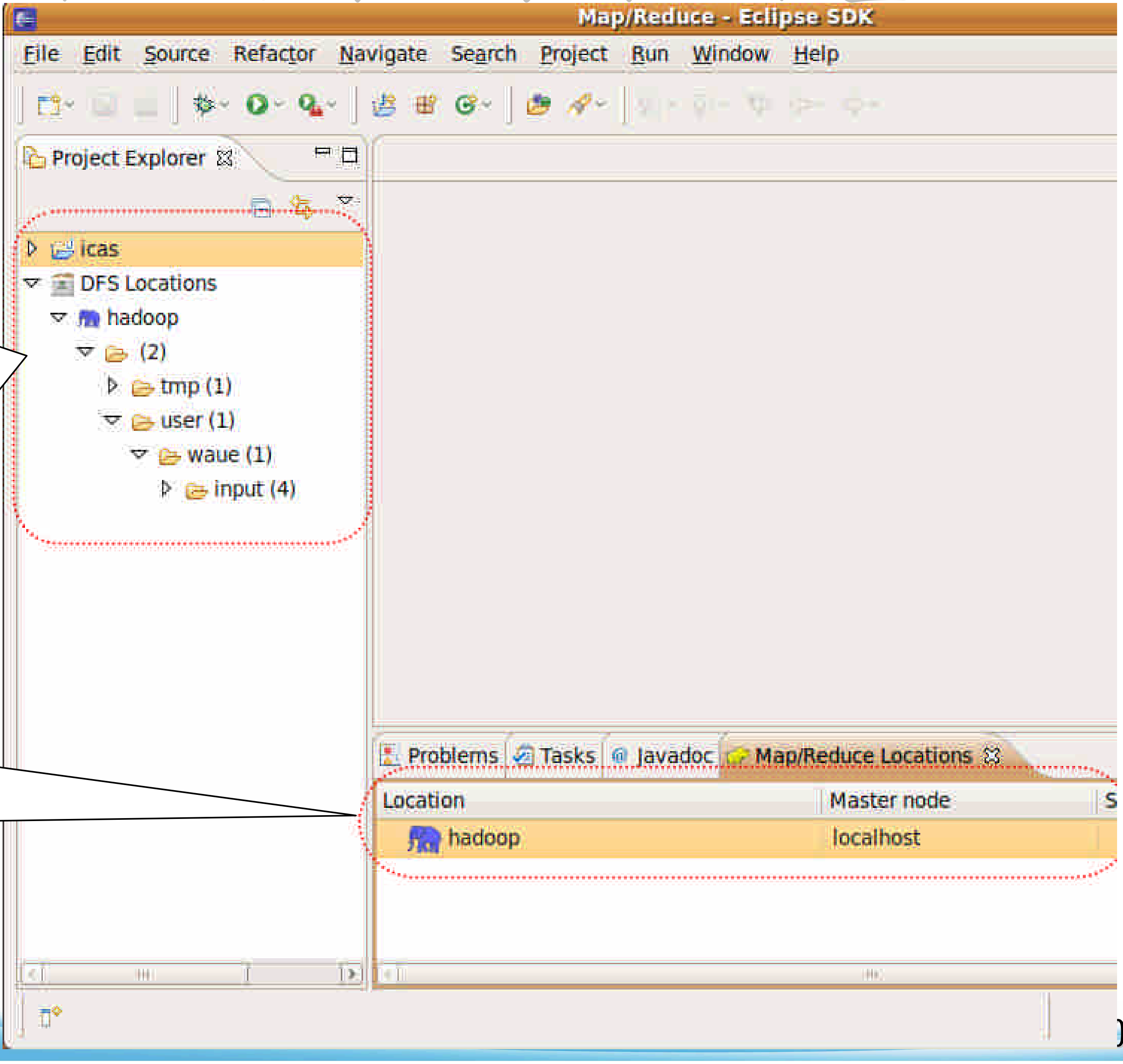
HDFS監聽的Port (設定於core-site.xml)

你在此Hadoop Server上的Username

6-2 若正確設定則可得到以下畫面

HDFS的資訊，可直接於此操作檢視、新增、上傳、刪除等命令

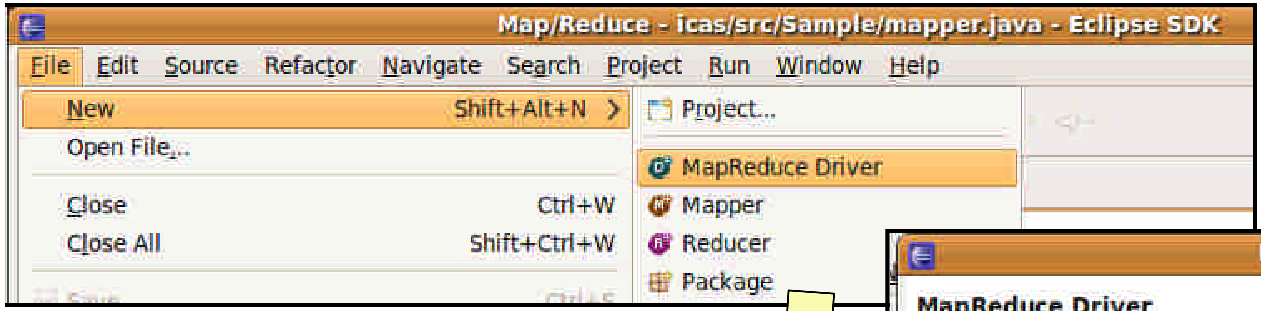
若有Job運作，可於此視窗檢視

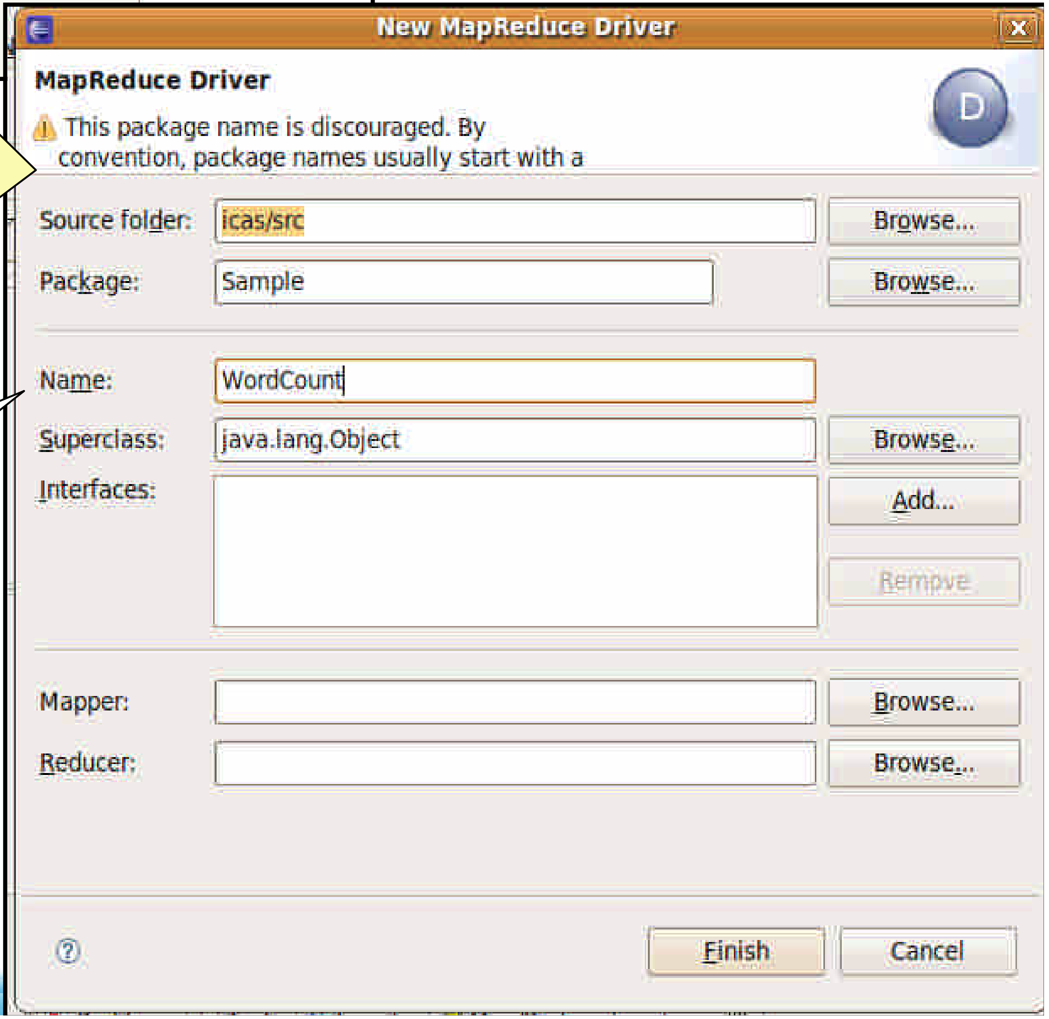


Map/Reduce - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
icas
DFS Locations
hadoop
(2)
tmp (1)
user (1)
waue (1)
input (4)
Problems Tasks Javadoc Map/Reduce Locations

Location	Master node
hadoop	localhost

7. 新增一個Hadoop程式


Map/Reduce - icas/src/Sample/mapper.java - Eclipse SDK


New MapReduce Driver

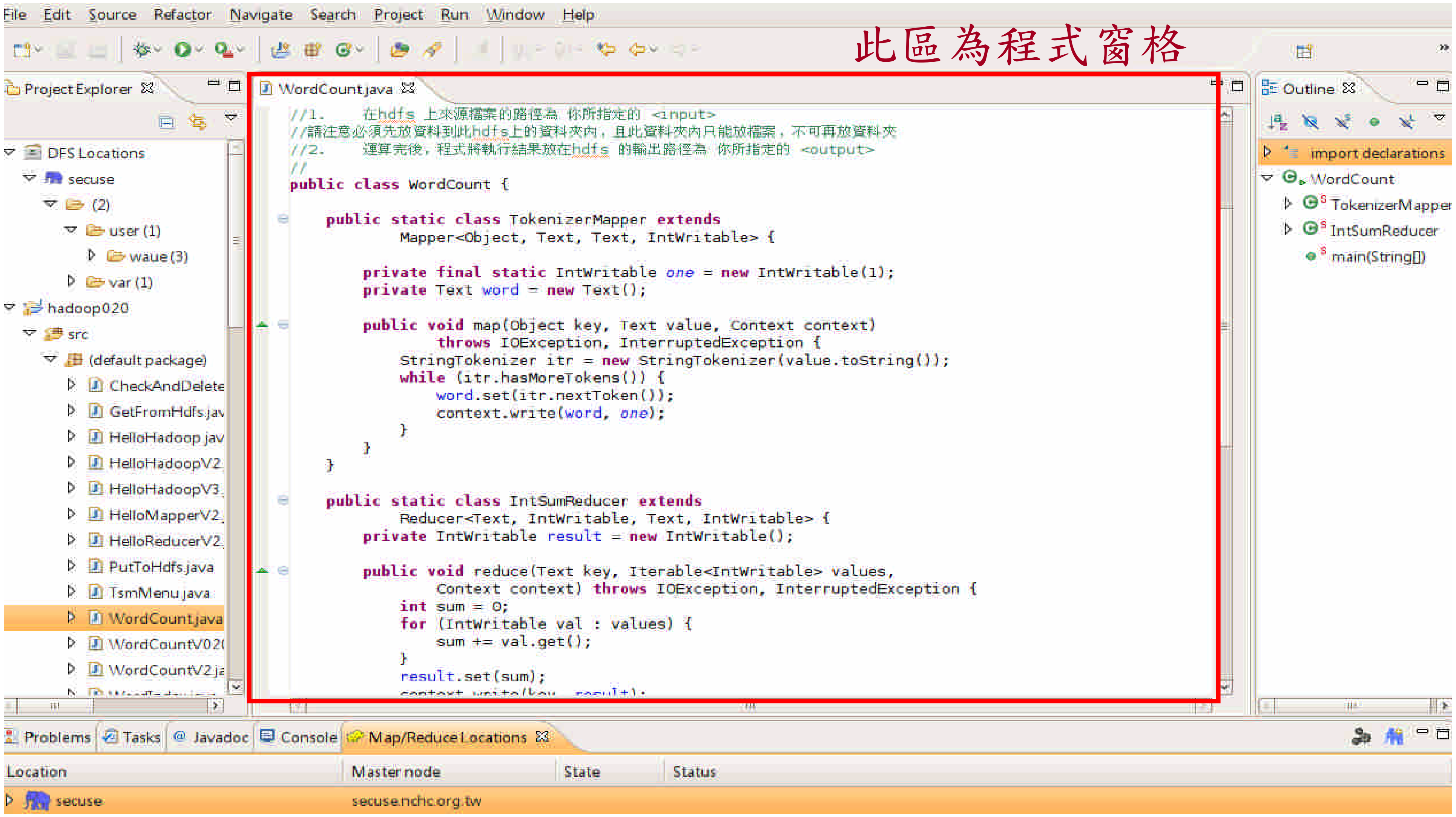
MapReduce Driver

This package name is discouraged. By convention, package names usually start with a

Source folder:	icas/src
Package:	Sample
Name:	WordCount
Superclass:	java.lang.Object
Interfaces:	
Mapper:	
Reducer:	

首先先建立一個WordCount程式，其他欄位任意

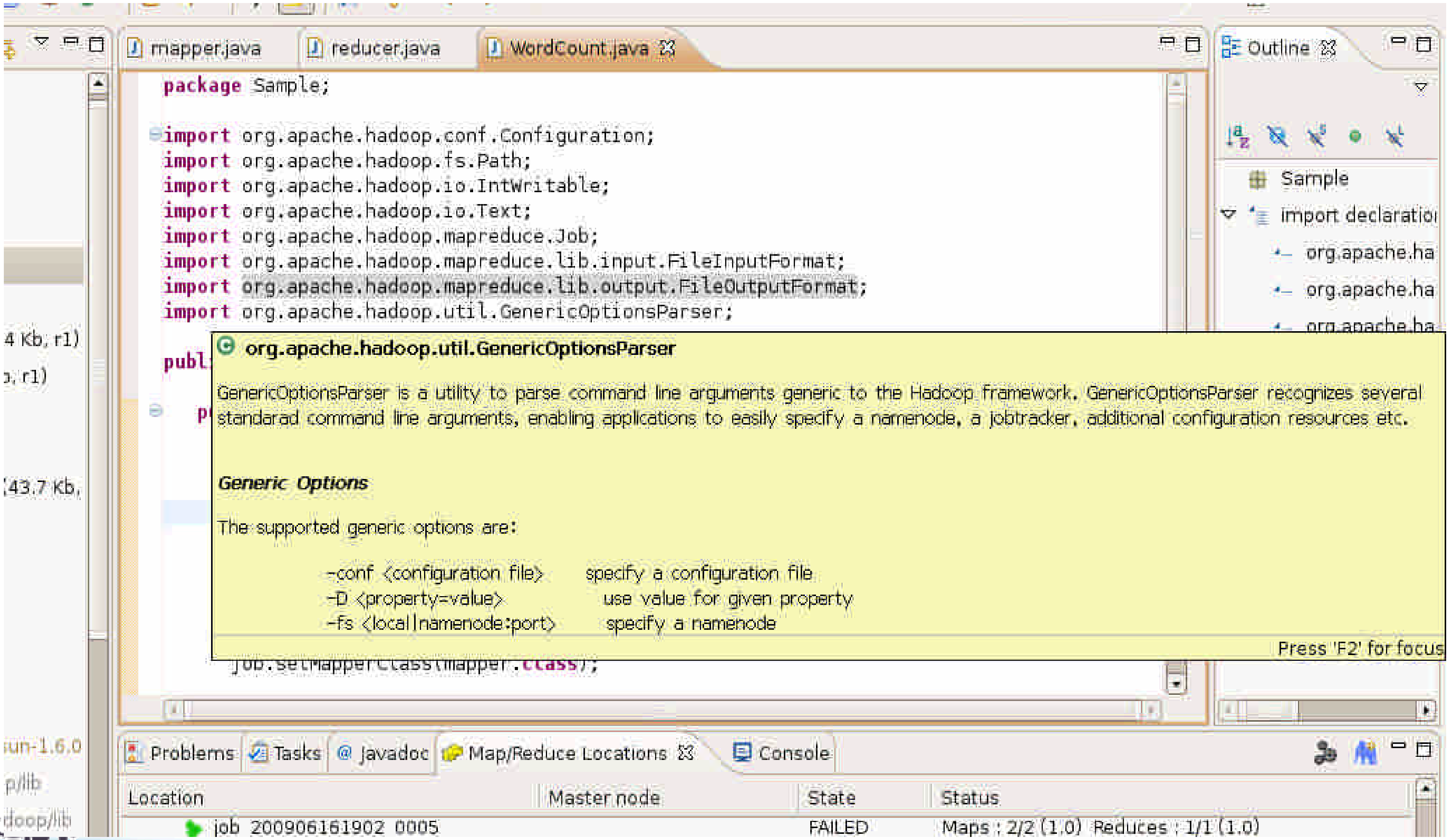
7.1 於程式窗格內輸入程式碼



```java
//1.     在hdfs 上來源檔案的路徑為 你所指定的 <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
//2.     運算完後，程式將執行結果放在hdfs 的輸出路徑為 你所指定的 <output>
//
public class WordCount {

    public static class TokenizerMapper extends
            Mapper<Object, Text, Text, IntWritable> {

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context)
                throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer extends
            Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values,
                Context context) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
```

Location	Master node	State	Status
secuse	secuse.nchc.org.tw		

7.2 補充：若之前doc部份設定正確，則滑鼠移至程式碼可取得API完整說明



mapper.java | reducer.java | WordCount.java | Outline

```
package Sample;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

publ
    p
        job.setMapperClass(mapper.class);
```

org.apache.hadoop.util.GenericOptionsParser

GenericOptionsParser is a utility to parse command line arguments generic to the Hadoop framework. GenericOptionsParser recognizes several standarad command line arguments, enabling applications to easily specify a namenode, a jobtracker, additional configuration resources etc.

Generic Options

The supported generic options are:

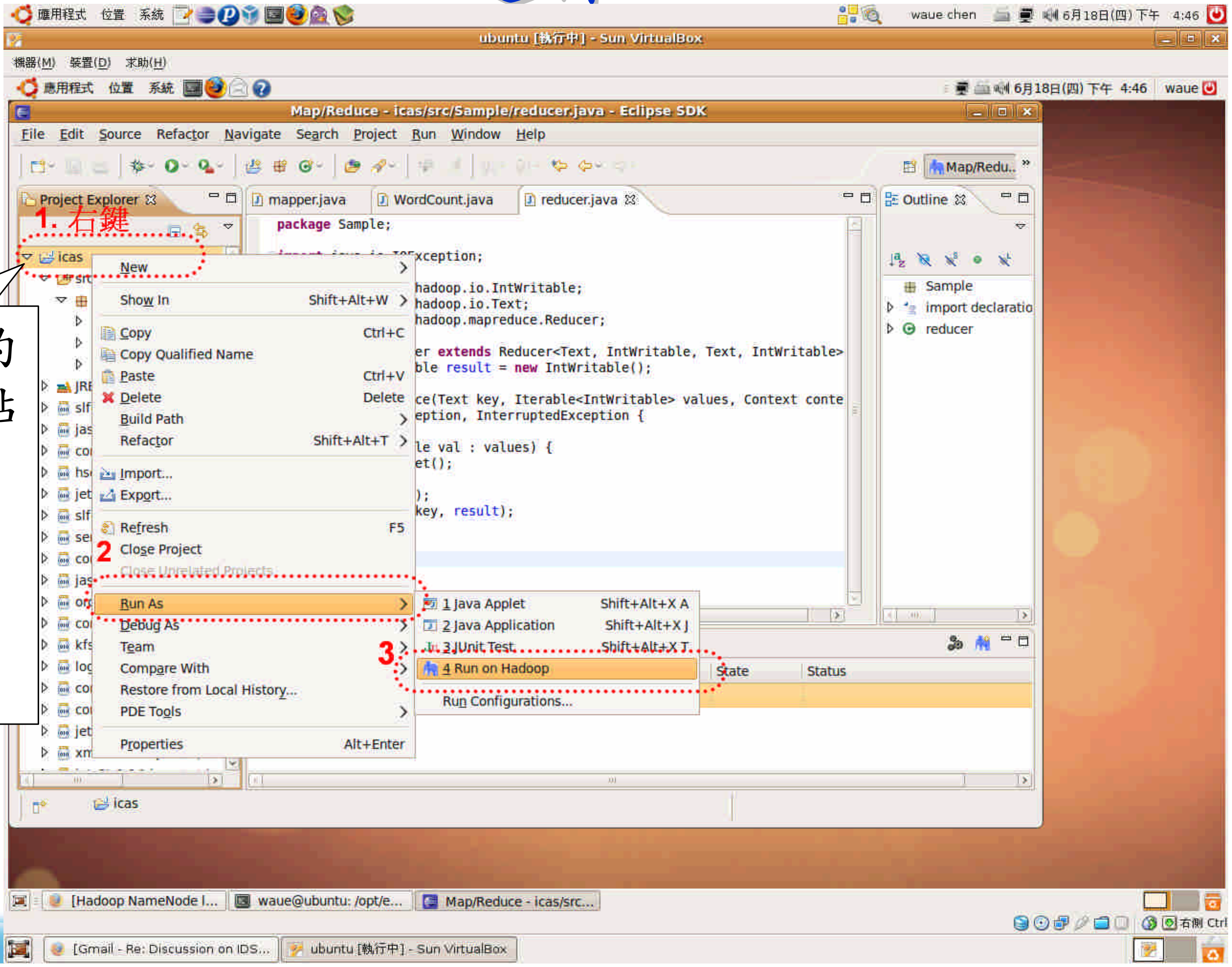
```
-conf <configuration file>     specify a configuration file
-D <property=value>              use value for given property
-fs <local|namenode:port>        specify a namenode
```

Press 'F2' for focus

Location	Master node	State	Status
job_200906161902_0005		FAILED	Maps : 2/2 (1.0) Reduces : 1/1 (1.0)

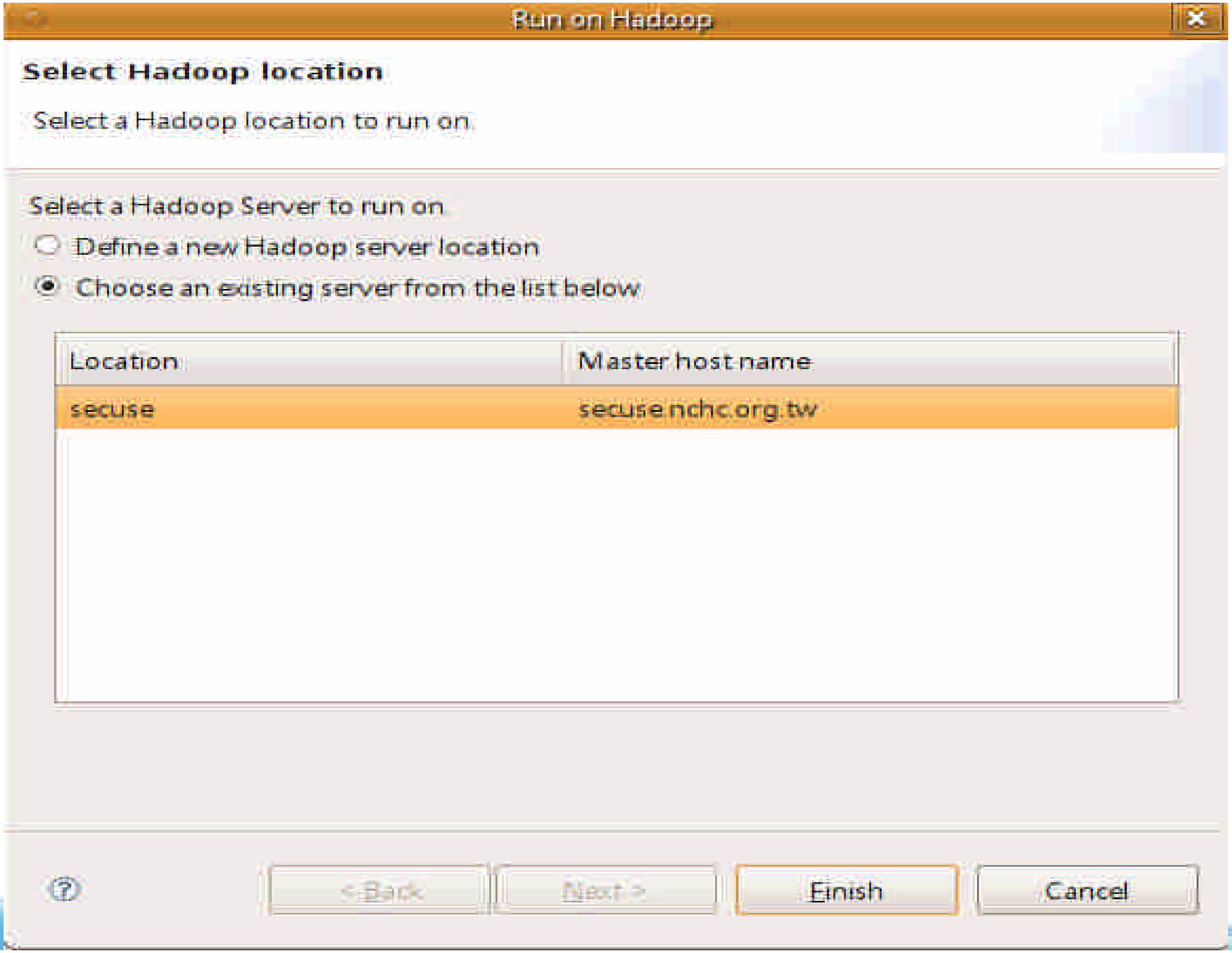
8. 運作

於欲運算的
程式碼處點
選右鍵 →
Run As →
Run on Hadoop



```
package Sample;

import java.io.IOException;
...hadoop.io.IntWritable;
...hadoop.io.Text;
...hadoop.mapreduce.Reducer;

...er extends Reducer<Text, IntWritable, Text, IntWritable>
...ble result = new IntWritable();

...ce(Text key, Iterable<IntWritable> values, Context conte
...eption, InterruptedException {
...le val : values) {
...et();

...);
...key, result);
```

8-1 選擇之前設定好所要運算的主機



Run on Hadoop

Select Hadoop location

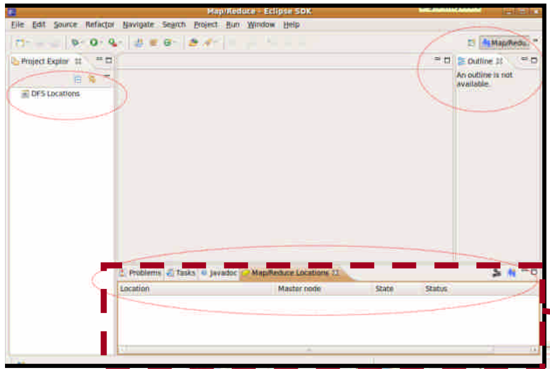
Select a Hadoop location to run on

Select a Hadoop Server to run on

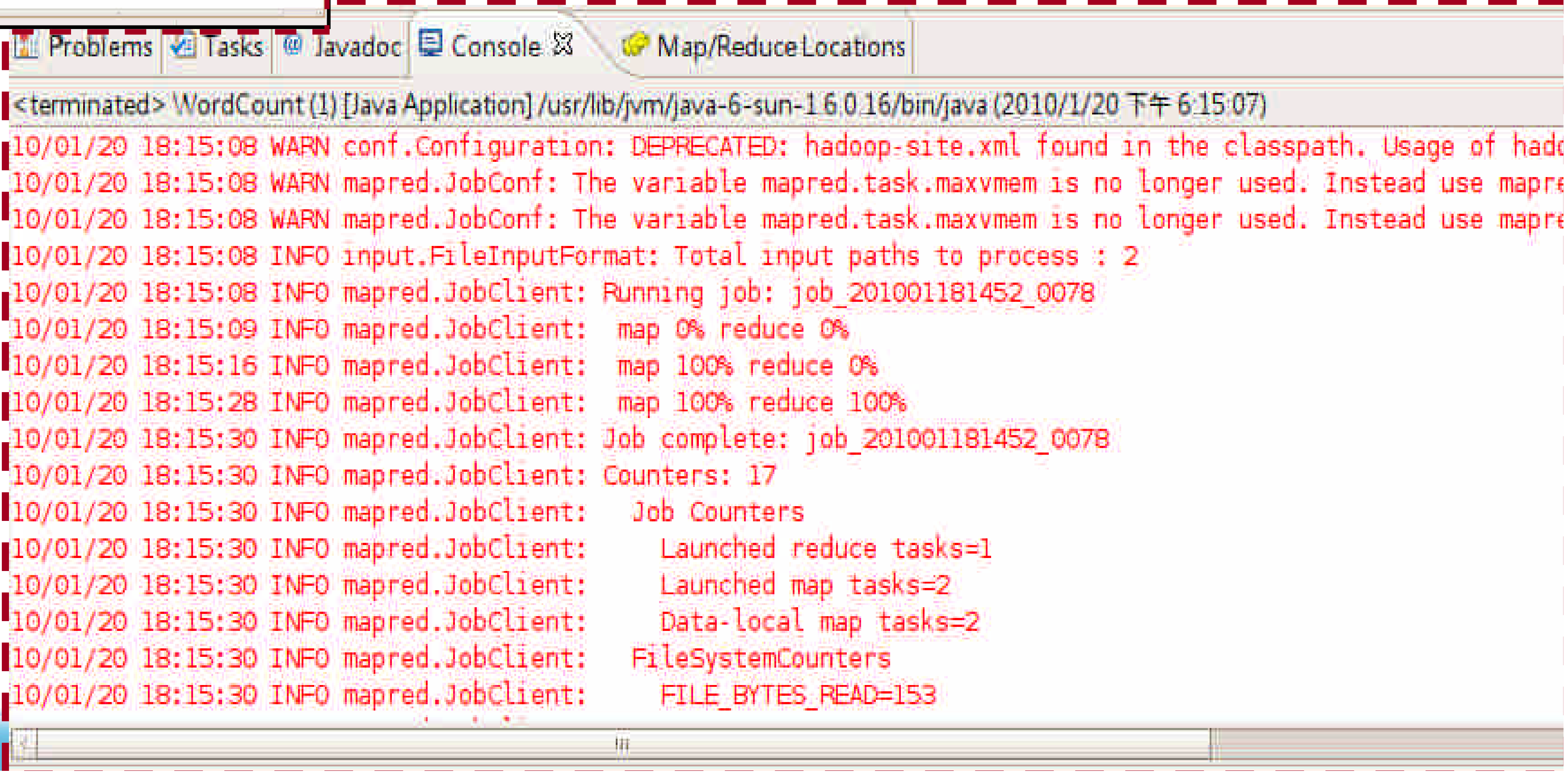
- [ ] Define a new Hadoop server location
- [✓ Choose an existing server from the list below]

Location	Master host name
secuse	secuse.nchc.org.tw

8.2 運算資訊出現於Eclipse 右下方的Console 視窗

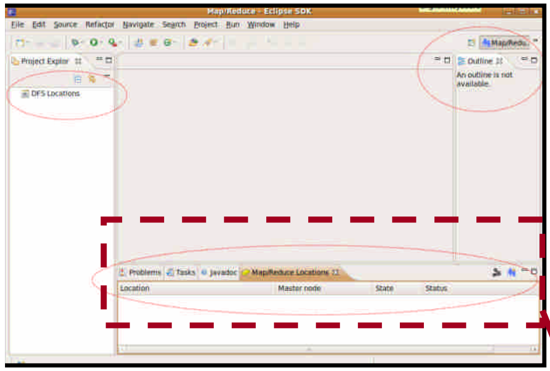

Map/Reduce - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explor
DFS Locations
Outline
An outline is not available.
Problems Tasks Javadoc Map/Reduce Locations
Location Master node State Status

放大

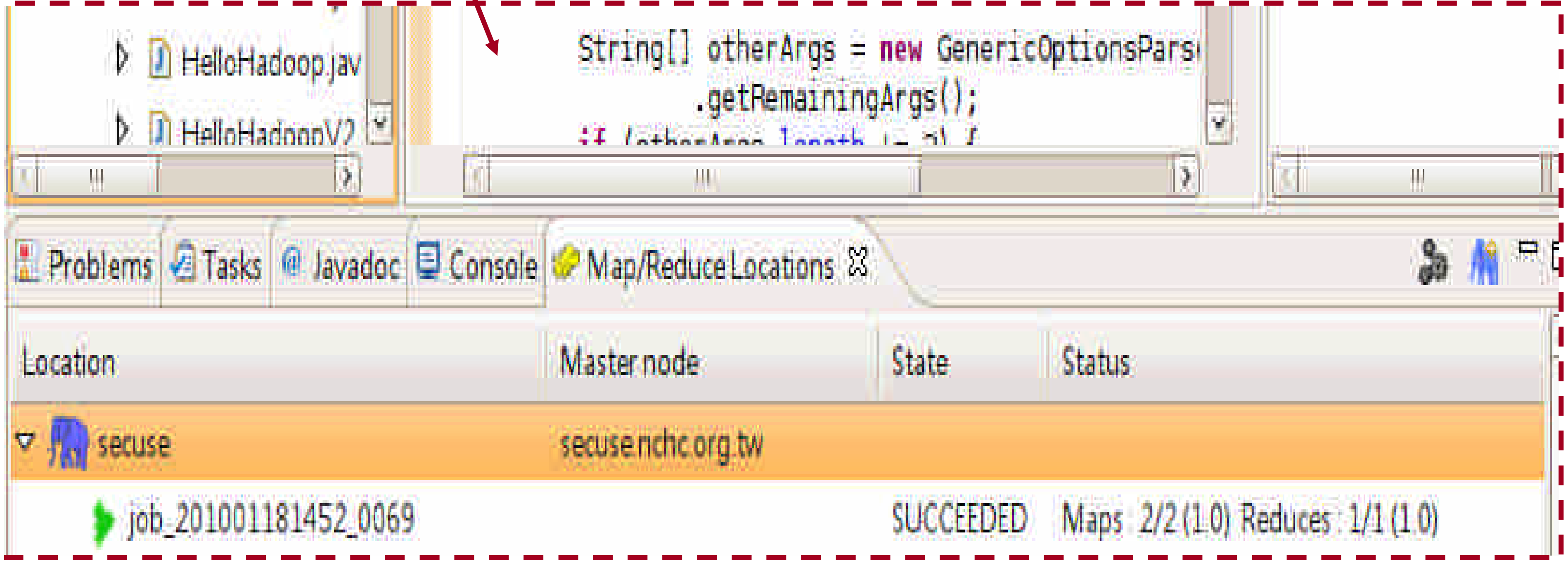


```
<terminated> WordCount (1) [Java Application] /usr/lib/jvm/java-6-sun-1.6.0.16/bin/java (2010/1/20 下午 6:15:07)
10/01/20 18:15:08 WARN conf.Configuration: DEPRECATED: hadoop-site.xml found in the classpath. Usage of hado
10/01/20 18:15:08 WARN mapred.JobConf: The variable mapred.task.maxvmem is no longer used. Instead use mapre
10/01/20 18:15:08 WARN mapred.JobConf: The variable mapred.task.maxvmem is no longer used. Instead use mapre
10/01/20 18:15:08 INFO input.FileInputFormat: Total input paths to process : 2
10/01/20 18:15:08 INFO mapred.JobClient: Running job: job_201001181452_0078
10/01/20 18:15:09 INFO mapred.JobClient:  map 0% reduce 0%
10/01/20 18:15:16 INFO mapred.JobClient:  map 100% reduce 0%
10/01/20 18:15:28 INFO mapred.JobClient:  map 100% reduce 100%
10/01/20 18:15:30 INFO mapred.JobClient: Job complete: job_201001181452_0078
10/01/20 18:15:30 INFO mapred.JobClient: Counters: 17
10/01/20 18:15:30 INFO mapred.JobClient:   Job Counters
10/01/20 18:15:30 INFO mapred.JobClient:     Launched reduce tasks=1
10/01/20 18:15:30 INFO mapred.JobClient:     Launched map tasks=2
10/01/20 18:15:30 INFO mapred.JobClient:     Data-local map tasks=2
10/01/20 18:15:30 INFO mapred.JobClient:   FileSystemCounters
10/01/20 18:15:30 INFO mapred.JobClient:     FILE_BYTES_READ=153
```

8.3 剛剛運算的結果出現如下圖



放大



```
String[] otherArgs = new GenericOptionsPars
        .getRemainingArgs();
```

Location	Master node	State	Status
secuse	secuse.nchc.org.tw		
job_201001181452_0069		SUCCEEDED	Maps : 2/2 (1.0) Reduces : 1/1 (1.0)