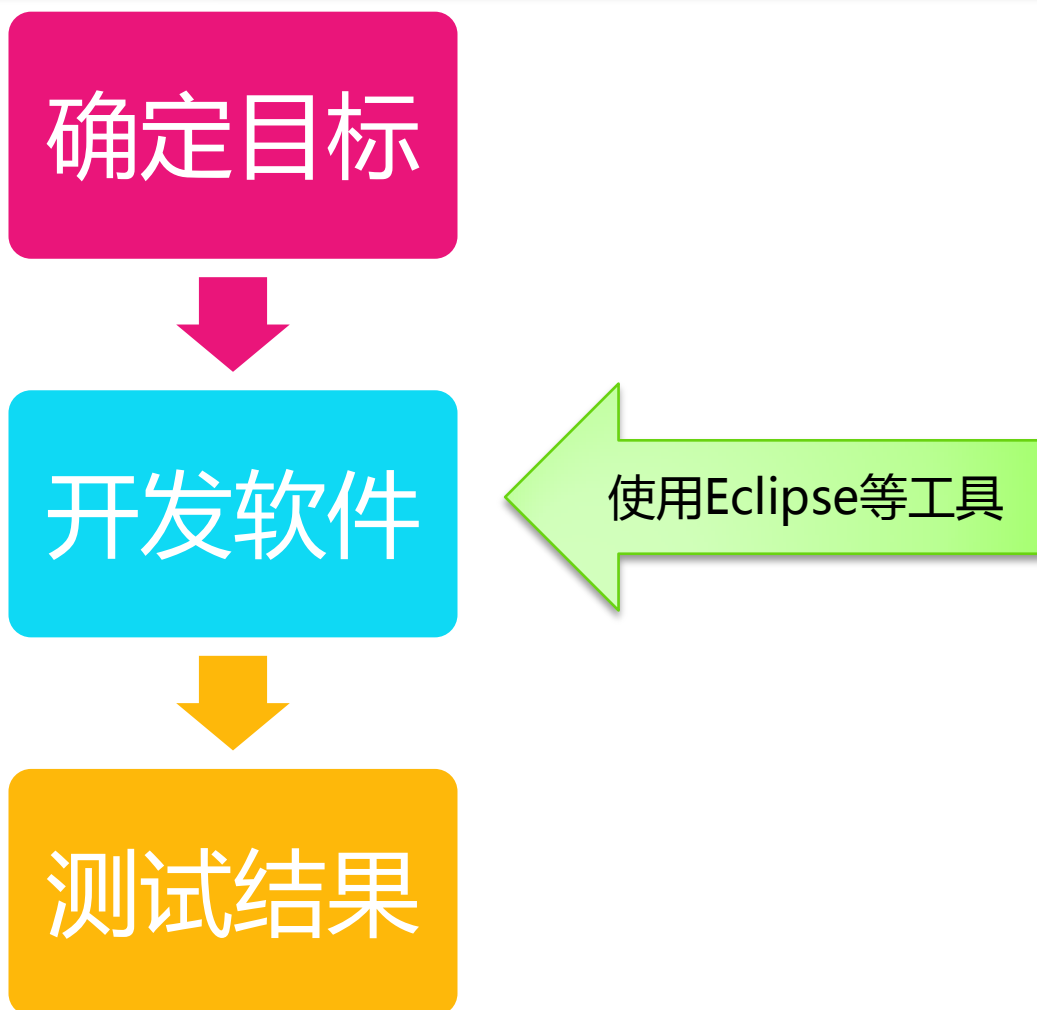




# Hadoop数据分析平台 第5周

2012.9.18



2012.9.18

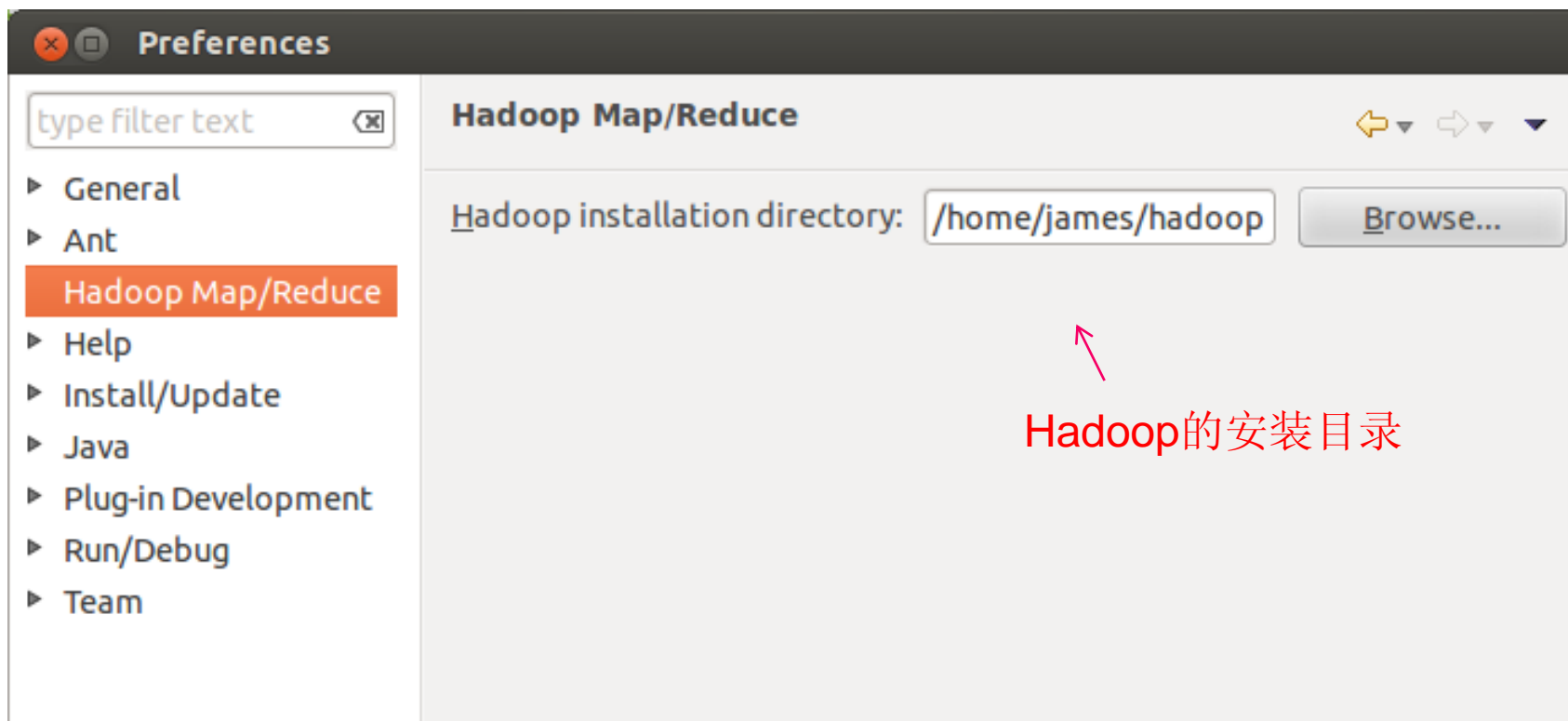
# Eclipse的Hadoop插件

- 专门对于Hadoop的插件
- 提供一个目录树用于管理HDFS文件系统
  - 可以创建和删除目录
  - 可以直接上传文件而不需输入命令
- 提供良好的编程环境
  - 自动提示
  - 能够直接在Eclipse上测试程序而不需要输入命令

- 把Hadoop安装目录下的contrib/eclipse-plugin文件复制到eclipse安装目录的plugins目录下

```
james@james-ubuntu (192.168.1.100) - byobu
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
james@james-ubuntu:~$ sudo cp $HADOOP_HOME/contrib/eclipse-plugin/hadoop-*-eclipse-plugin.jar /usr/lib/eclipse/plugins
james@james-ubuntu:~$
```

- 打开Window-->Preference
- 选择Hadoop Map/Reduce选项



# 安装方法

- 在Window-->Show View中打开Map/Reduce Locations。
- 在下方点选右键-->New Hadoop Location

General Advanced parameters

Location name: Ubuntu 随意填

Map/Reduce Master

Host: localhost

Port: 9001

DFS Master

☒ Use M/R Master host

Host: localhost

Port: 9000

User name: james 用户名

SOCKS proxy

☐ Enable SOCKS proxy

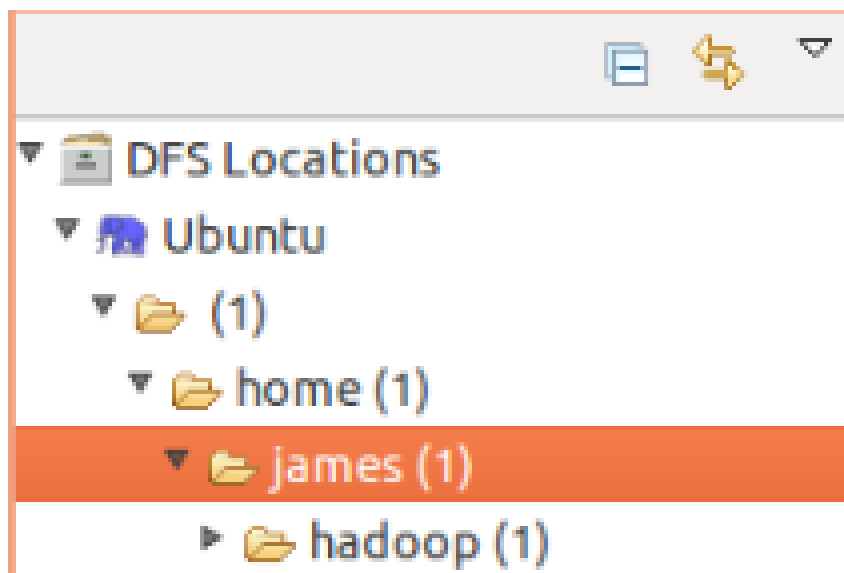
Host: host

Port: 1080

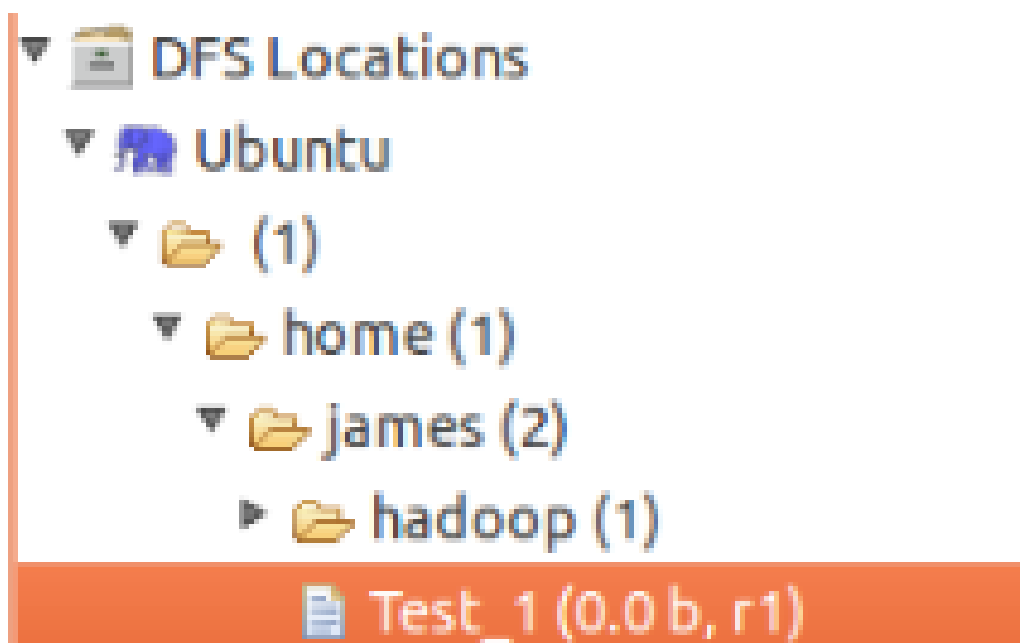
和mapred-site.xml以及core-site.xml的内容必须一致

2012.9.18

- 对着左边的DFS Locations下面的主机点选右键刷新
- 成功安装的话就可以看得到HDFS目录树



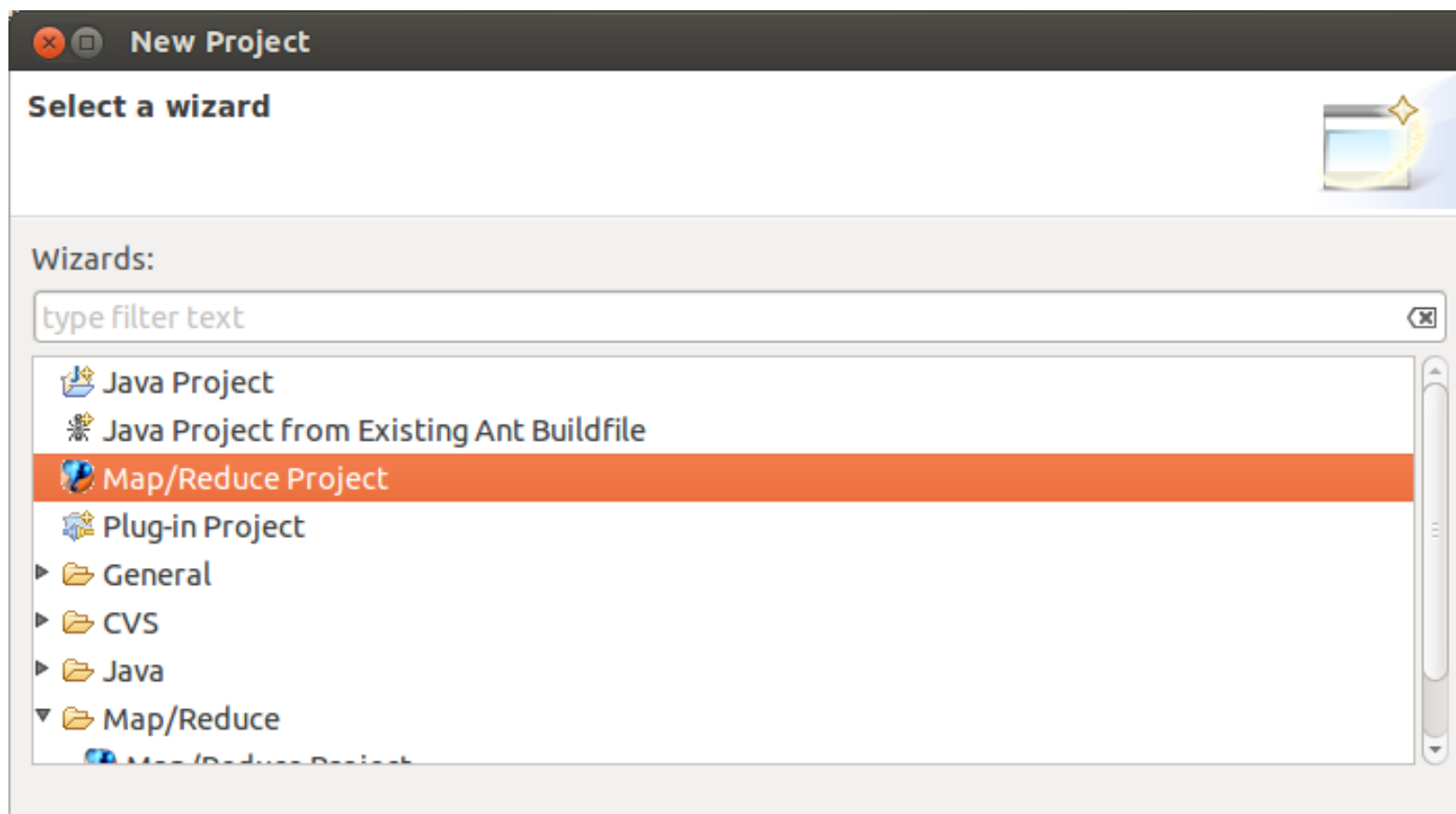
- 对左侧的目录树中的目录点选右键，选择 update files to DFS,然后把上述路由日志文本上传HDFS
- 也可以采用传统的命令行上传方法





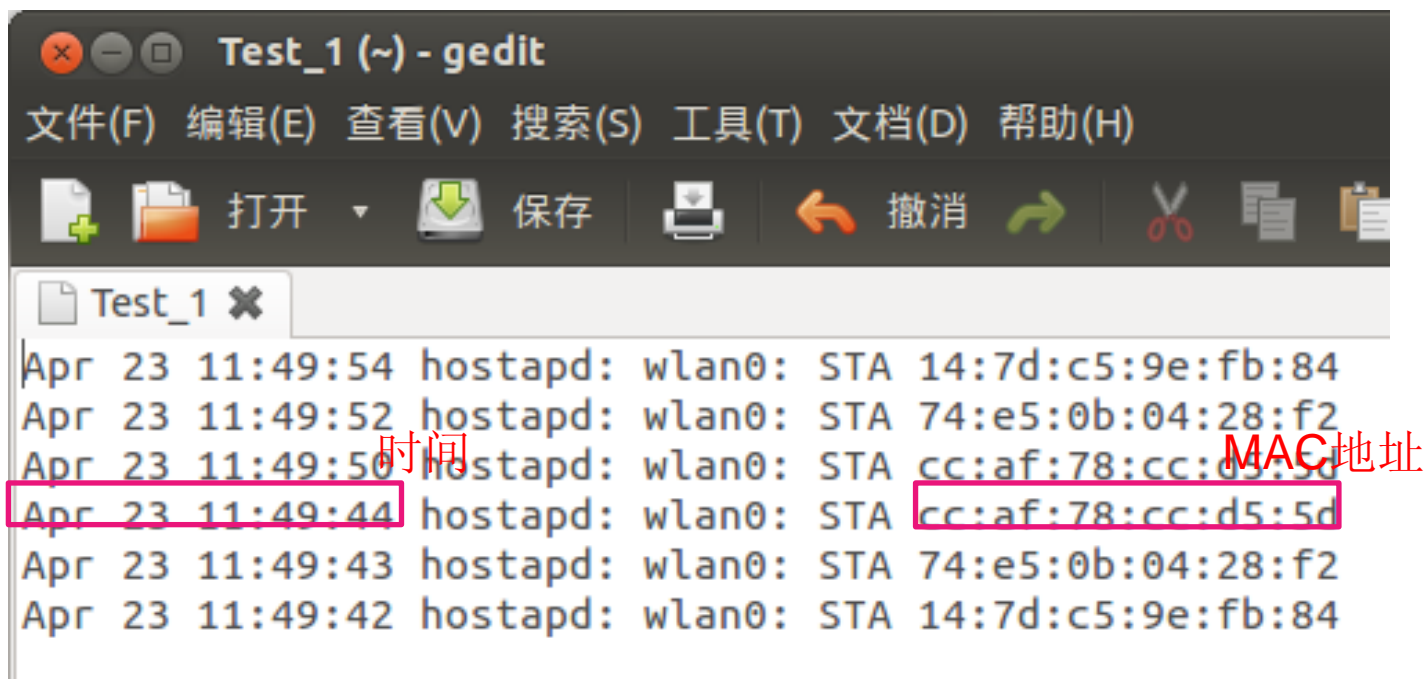
# 创建MapReduce Project

- 安装插件之后，可以在New Project页面建立M/R Project，便能自带编程所需API



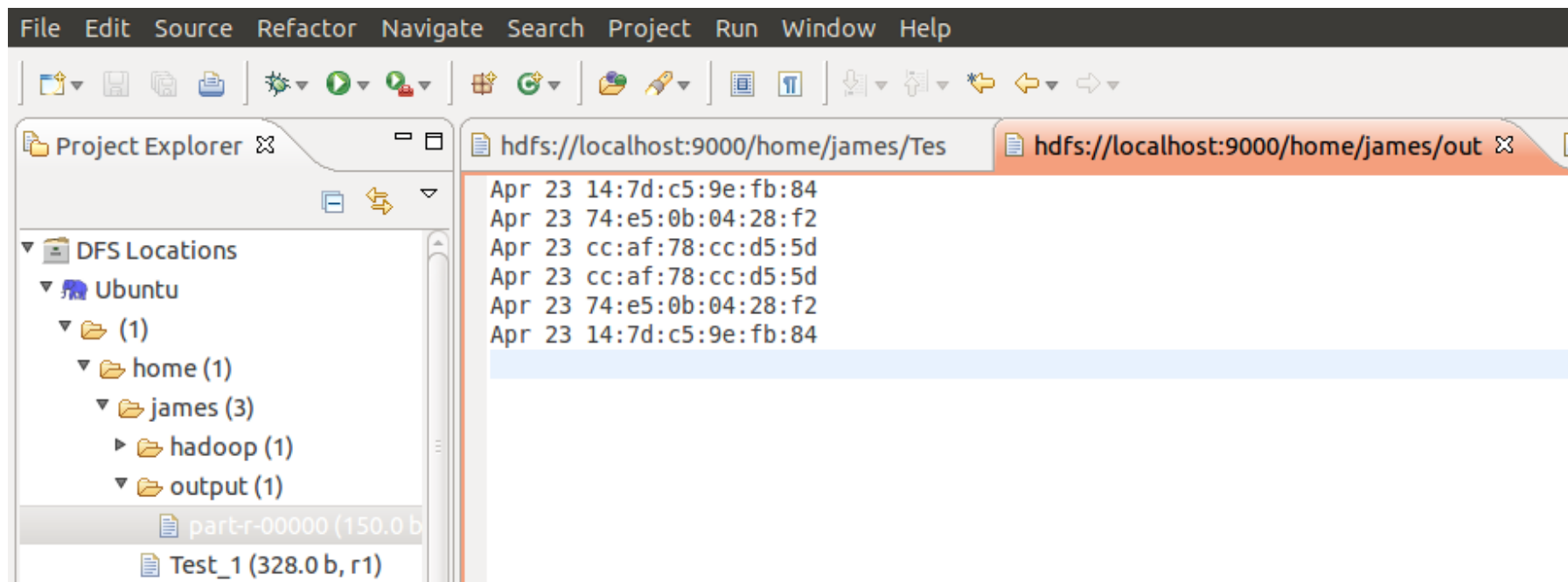
## ■ 任务要求：

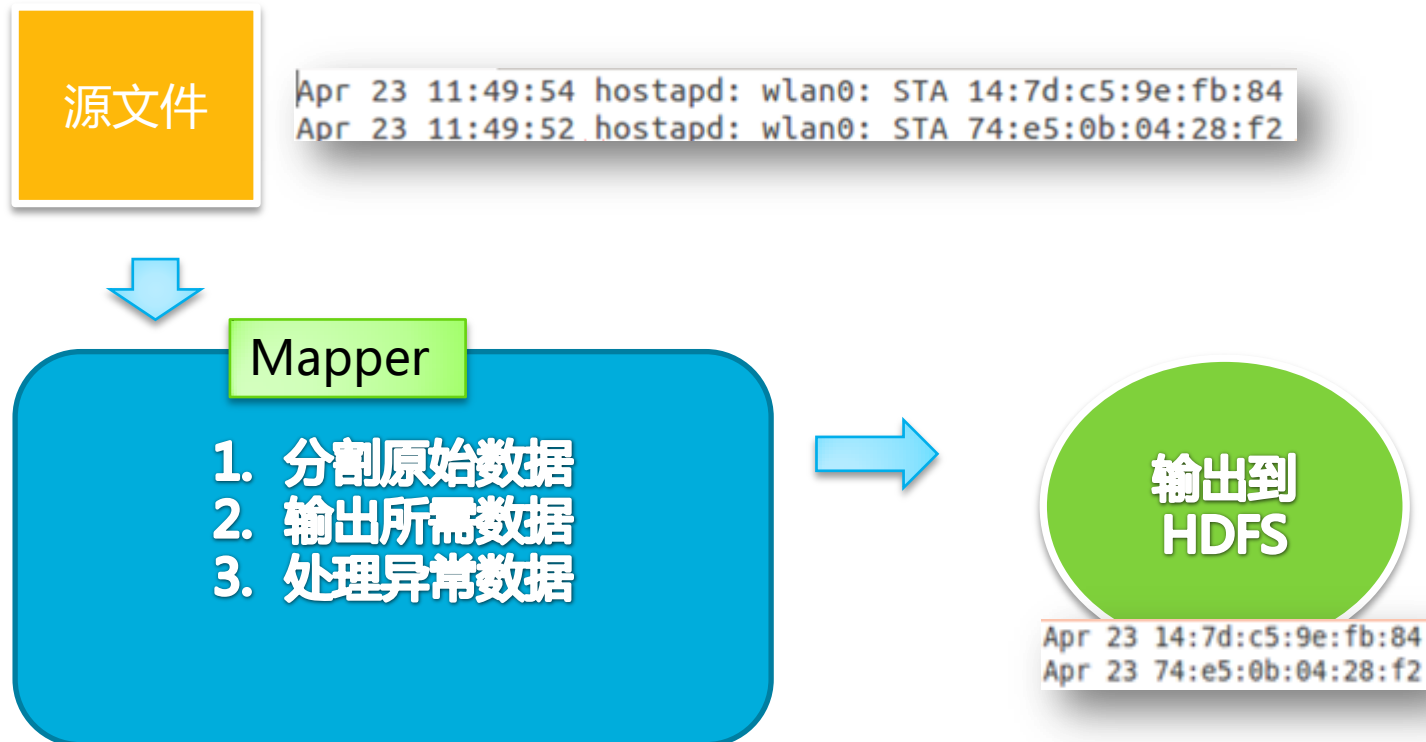
- 现有一批路由日志（有删减），需要提取MAC地址和时间，删去其他内容



```
Test_1 (~) - gedit
文件(F) 编辑(E) 查看(V) 搜索(S) 工具(T) 文档(D) 帮助(H)
打开 保存 撤消
Test_1
Apr 23 11:49:54 hostapd: wlan0: STA 14:7d:c5:9e:fb:84
Apr 23 11:49:52 hostapd: wlan0: STA 74:e5:0b:04:28:f2
Apr 23 11:49:50 hostapd: wlan0: STA cc:af:78:cc:d5:5d
Apr 23 11:49:44 hostapd: wlan0: STA cc:af:78:cc:d5:5d
Apr 23 11:49:43 hostapd: wlan0: STA 74:e5:0b:04:28:f2
Apr 23 11:49:42 hostapd: wlan0: STA 14:7d:c5:9e:fb:84
```

- 输出结果如下图所示，分别是时间和Mac地址





- MapReduce程序包括一个Map函数，一个Reduce函数，以及Main函数
- Reduce函数是可选的，当不指定Reduce的实现时，系统自动使用缺省的Reduce函数
- 部分程序代码不赘述，如以下Counter

```
public class Test_1 extends Configured implements Tool {  
    * 计数器  
    enum Counter  
    {  
        LINESKIP, //出错的行  
    }  
}
```

Counter即是一个计数器  
可以记录这个程序的一些数据用于统计

# Map函数

\* MAP任务

```
public static class Map extends Mapper<LongWritable, Text, NullWritable, Text>
{
    public void map ( LongWritable key, Text value, Context context ) throws IOException
    {
        String line = value.toString();           //读取源数据

        try    读取源文件，line得到的就是输入文件的一行数据
        {
            //数据处理
            String [] lineSplit = line.split(" ");    对源数据进行分割和重组
            String month = lineSplit[0];
            String time = lineSplit[1];
            String mac = lineSplit[6];
            Text out = new Text(month + ' ' + time + ' ' + mac);

            context.write( NullWritable.get(), out);    //输出    把两个参数分别作为KEY和VALUE输出
        }
        catch ( java.lang.ArrayIndexOutOfBoundsException e )
        {
            context.getCounter(Counter.LINESKIP).increment(1);    //出错令计数器+1
            return;
        }
    }
}
```

如果发生异常，则指定计数器中的LINESKIP自增

2012.9.18

# Map函数

```
* MAP任务[]
public static class Map extends Mapper<LongWritable, Text, NullWritable, Text>
{
    public void map ( LongWritable key, Text value, Context context ) throws IOException
    {
        String line = value.toString();           //读取源数据

        try
        {
            //数据处理
            String [] lineSplit = line.split(" ");
            String month = lineSplit[0];
            String time = lineSplit[1];
            String mac = lineSplit[6];
            Text out = new Text(month + ' ' + time + ' ' + mac);

            context.write( NullWritable.get(), out);    //输出
        }
        catch ( java.lang.ArrayIndexOutOfBoundsException e )
        {
            context.getCounter(Counter.LINESKIP).increment(1); //出错令计数器+1
            return;
        }
    }
}
```

输入格式，必须上下一致

输出KEY格式 输出VALUE格式

输出的KEY和VALUE必须与上述两个格式一致

- Run方法是运行程序的一种实现
- 在Run方法可以设定一些基本数据，从而让系统了解该如何运行整个任务
- 为了更好理解任务，此Run方法在屏幕上输出了一些基本信息



```
@Override
public int run(String[] args) throws Exception
{
    Configuration conf = getConf();

    Job job = new Job(conf, "Test_1"); //任务名
    job.setJarByClass(Test_1.class); //指定Class 必须是当前所在的Class名

    FileInputFormat.addInputPath( job, new Path(args[0]) ); //输入路径
    FileOutputFormat.setOutputPath( job, new Path(args[1]) ); //输出路径

    job.setMapperClass( Map.class ); //调用上面Map类作为Map任务代码
    job.setOutputFormatClass( TextOutputFormat.class );
    job.setOutputKeyClass( NullWritable.class ); //指定输出的KEY的格式
    job.setOutputValueClass( Text.class ); //指定输出的VALUE的格式 必须与上一面的输出格式一致

    job.waitForCompletion(true);

    //输出任务完成情况
    System.out.println( "任务名称: " + job.getJobName() );
    System.out.println( "任务成功: " + ( job.isSuccessful()? "是": "否" ) );
    System.out.println( "输入行数: " + job.getCounters().findCounter("org.apache.hadoop.mapred.Task
    System.out.println( "输出行数: " + job.getCounters().findCounter("org.apache.hadoop.mapred.Task
    System.out.println( "跳过的行: " + job.getCounters().findCounter(Counter.LINESKIP).getValue() )

    return job.isSuccessful() ? 0 : 1;
}
```

- 只需在Main函数调用Run方法，系统就会启动一个MapReduce任务

```
//运行任务  
int res = ToolRunner.run(new Configuration(), new Test_1(), args);
```

# 运行程序

新建配置

Name: Test\_1

Main Arguments JRE Classpath Source Environment Common

Program arguments:

hdfs://localhost:9000/home/james/Test\_1 hdfs://localhost:9000/home/james/output

输入路径

输出路径

Variables...

VM arguments:

Variables...

Working directory:

☒ Default:

\${workspace\_loc:Test\_1}

☐ Other:

Workspace...

File System...

Variables...

Apply

Revert

Close

Run

任务名称: Test\_1

任务成功: 是

输入行数: 6

输出行数: 6

跳过的行: 0

任务开始: 2012-09-18 10:16:46

任务结束: 2012-09-18 10:16:48

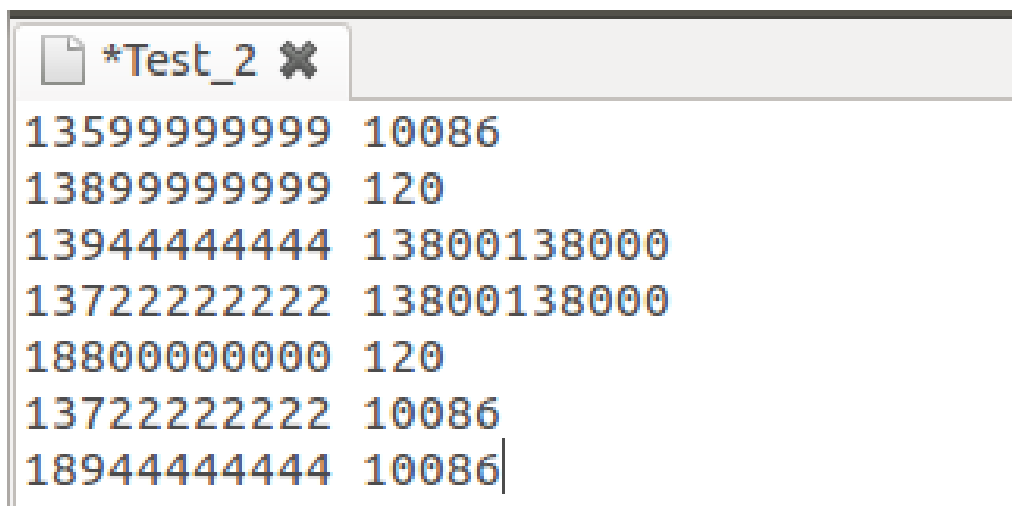
任务耗时: 0.026183333 分钟

即Counter.LINESKIP的计数，也即出错行数



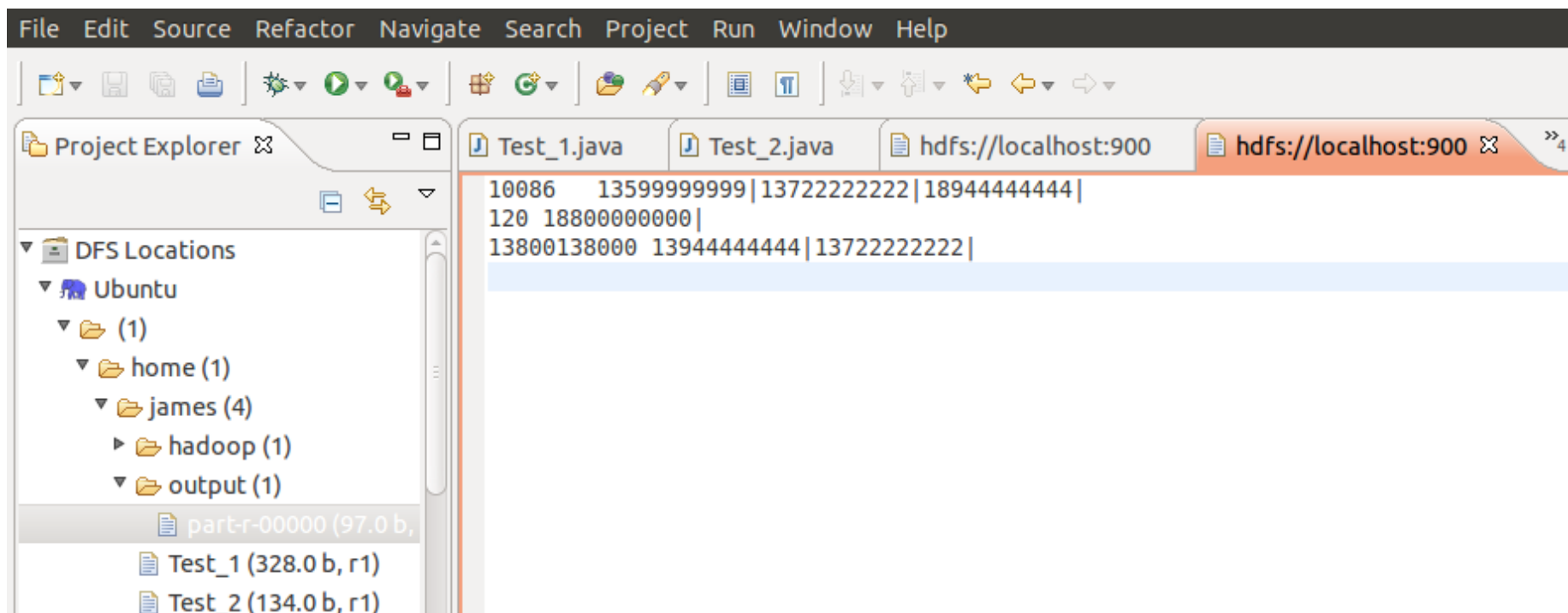
## ■ 任务要求

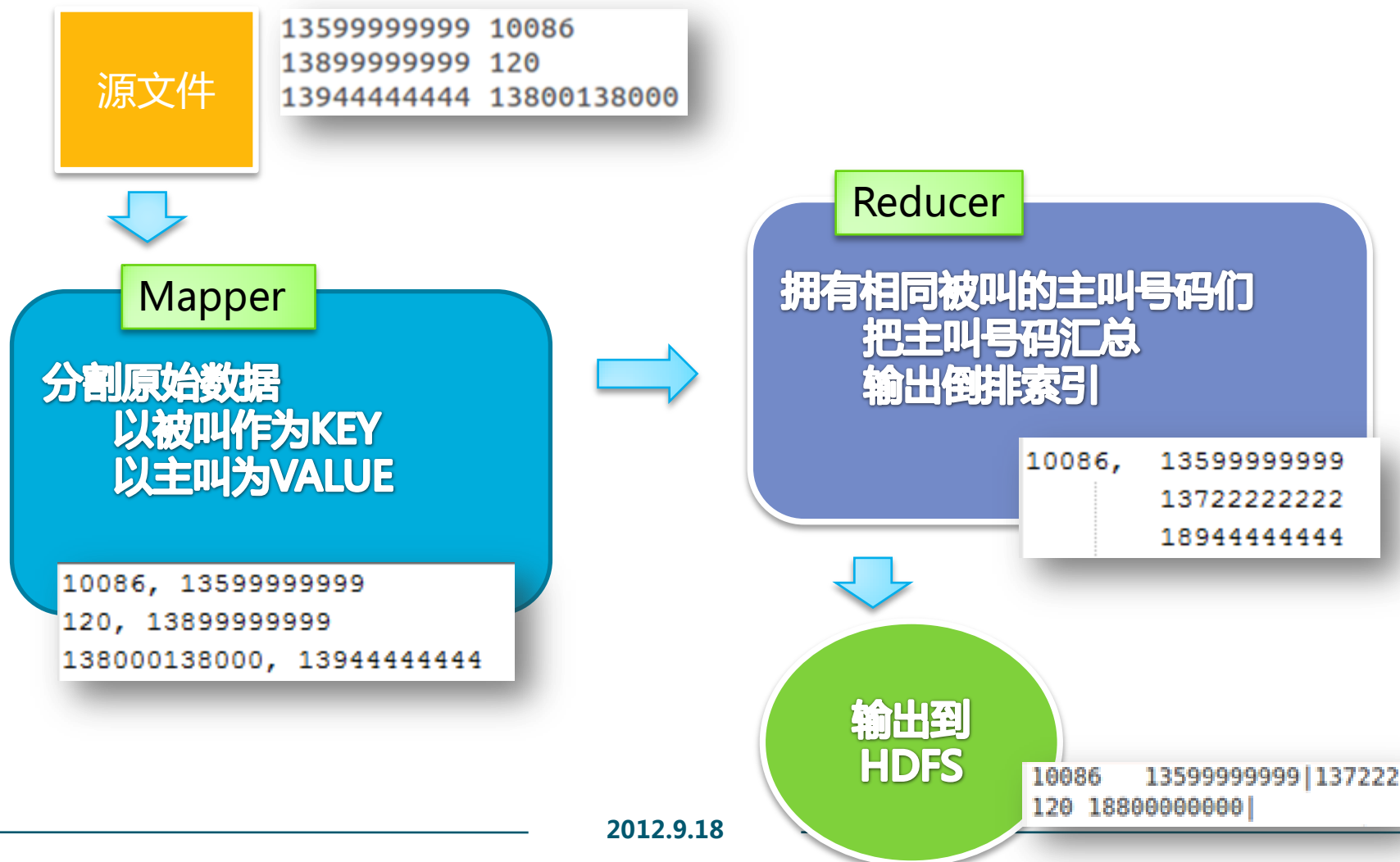
- 现有一批电话通信清单，记录了用户A拨打用户B的记录
- 需要做一个倒排索引，记录拨打给用户B的所有用户A



```
*Test_2 ✕  
13599999999 10086  
13899999999 120  
13944444444 13800138000  
13722222222 13800138000  
18800000000 120  
13722222222 10086  
18944444444 10086
```

- 任务输出必须如下所示，主叫以 ‘|’ 分割





2012.9.18

## 带有Reduce的任务

- 示例程序1不带Reduce任务，系统自动把Map函数的输出发送到输出文件，Map函数的输出格式必须与程序输出格式一致
- 示例程序2带有Reduce任务，系统首先把Mapper的输出中Key相同的部分都发送到同一个Reducer，然后再把Reduce函数的结果输出，Map函数的输出格式必须和Reduce函数的输入格式一致



- 此Map函数的主要作用是把两个号码分割，然后被叫作为Key，主叫作为Value

\* MAP任务

```
public static class Map extends Mapper<LongWritable, Text, Text, Text> {
    public void map ( LongWritable key, Text value, Context context ) throws IOException, InterruptedException
    {
        String line = value.toString();           //读取源数据

        try
        {
            //数据处理
            String [] lineSplit = line.split(" ");
            String anum = lineSplit[0];
            String bnum = lineSplit[1];

            context.write( new Text(bnum), new Text(anum) );    //输出
        }
        catch ( java.lang.ArrayIndexOutOfBoundsException e )
        {
            context.getCounter(Counter.LINESKIP).increment(1); //出错令计数器+1
            return;
        }
    }
}
```

输出格式

被叫

主叫

输入格式  
必须与Map函数的输出一致

```
* REDUCE任务
public static class Reduce extends Reducer<Text, Text, Text, Text>
{
    public void reduce ( Text key, Iterable<Text> values, Context context
    {
        String valueString;
        String out = "";

        for ( Text value : values )
        {
            valueString = value.toString();
            out += valueString + "|";
        }

        context.write( key, new Text(out) );
    }
}
```

每一个Value代表Map函数发送的一个Value  
在这里代表拨打了这个被叫号码的一个主叫

# Run方法

```
@Override
public int run(String[] args) throws Exception
{
    Configuration conf = getConf();

    Job job = new Job(conf, "Test_2");                //任务名
    job.setJarByClass(Test_2.class);                  //指定Class

    FileInputFormat.addInputPath( job, new Path(args[0]) );    //输入路径
    FileOutputFormat.setOutputPath( job, new Path(args[1]) );  //输出路径

    job.setMapperClass( Map.class );                  //调用上面Map类作为Map任务代码
    job.setReducerClass ( Reduce.class );             //调用上面Reduce类作为Reduce任务代码
    job.setOutputFormatClass( TextOutputFormat.class );
    job.setOutputKeyClass( Text.class );              //指定输出的KEY的格式
    job.setOutputValueClass( Text.class );            //指定输出的VALUE的格式

    job.waitForCompletion(true);

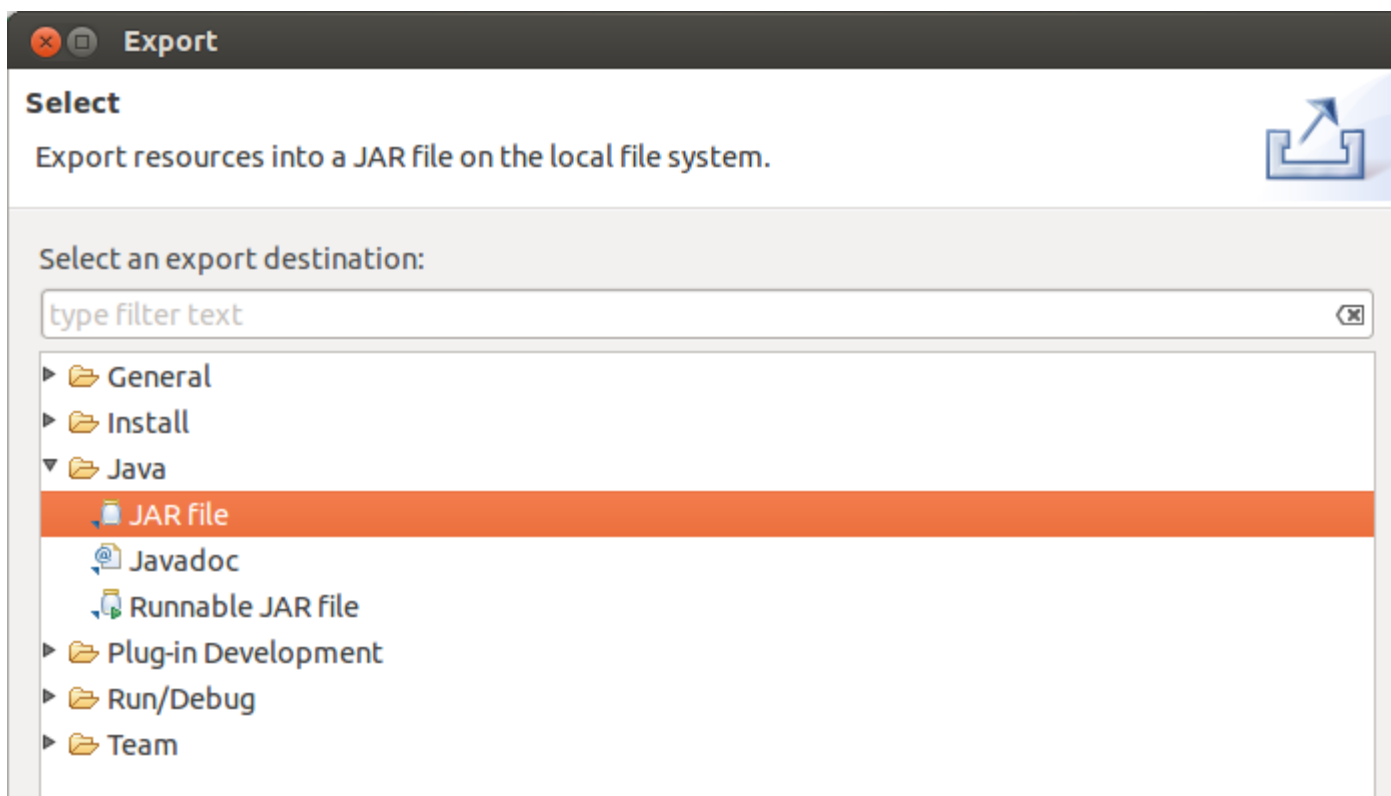
    //输出任务完成情况
    System.out.println( "任务名称: " + job.getJobName() );
    System.out.println( "任务成功: " + ( job.isSuccessful()? "是": "否" ) );
    System.out.println( "输入行数: " + job.getCounters().findCounter("org.apache.hadoop.mapred.Task$Coun
    System.out.println( "输出行数: " + job.getCounters().findCounter("org.apache.hadoop.mapred.Task$Coun
    System.out.println( "跳过的行: " + job.getCounters().findCounter(Counter.LINESKIP).getValue() );

    return job.isSuccessful() ? 0 : 1;
}
```

指定Reduce

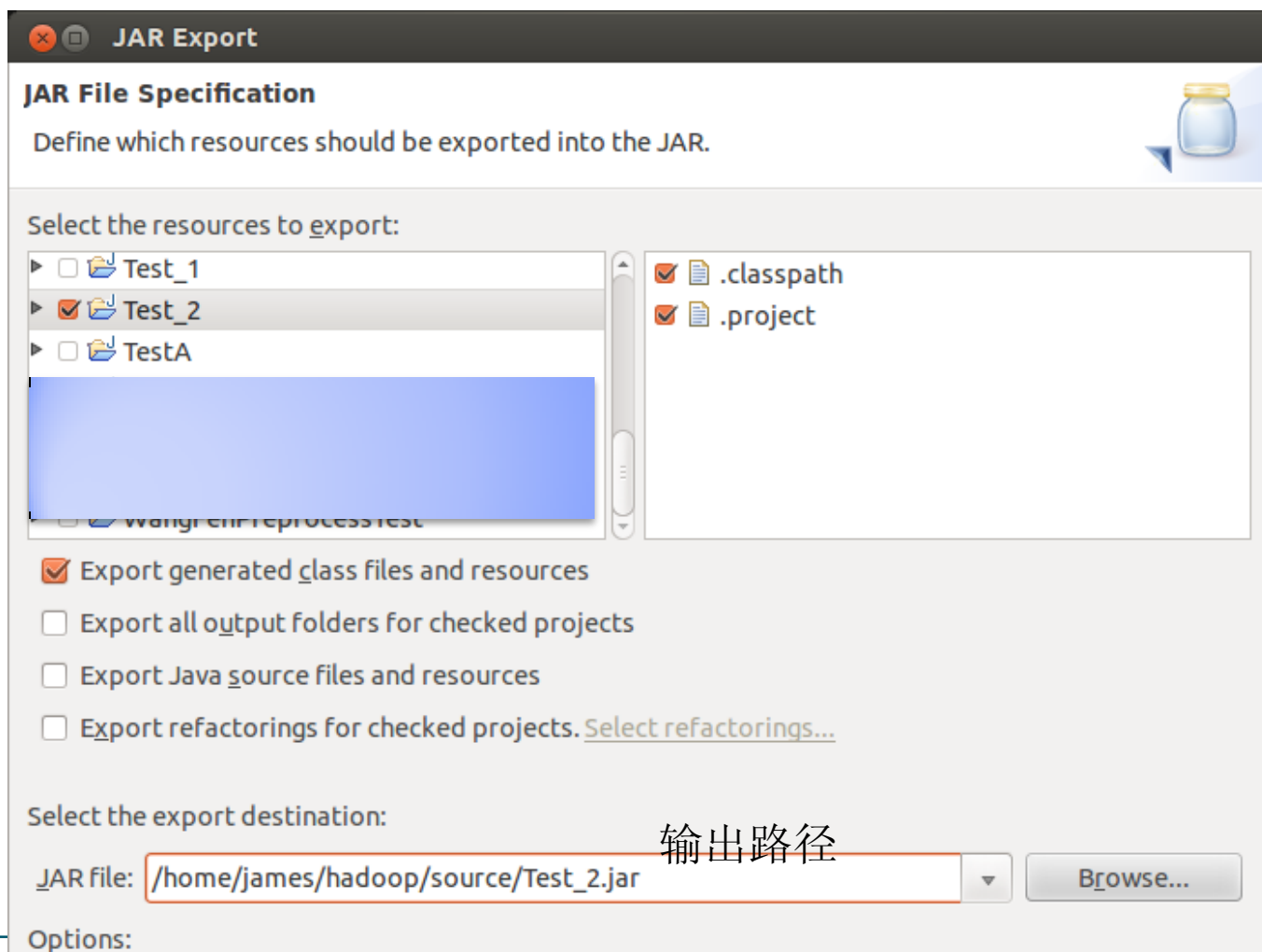
# 程序的导出

- 对Project点选右键 --> Export



2012.9.18

# 程序的导出



# 程序的导出

JAR Export

**JAR Manifest Specification**

Customize the manifest file for the JAR file.

Specify the manifest:

☒ Generate the manifest file

☐ Save the manifest in the workspace

☐ Use the saved manifest in the generated JAR description file

Manifest file:

☐ Use existing manifest from workspace

Manifest file:

Seal contents:

☐ Seal the JAR

☒ Seal some packages Nothing sealed

Select the class of the application entry point:

Main class:

指定MainClass

# 命令行运行程序

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
james@james-ubuntu:~$ hadoop jar /home/james/hadoop/source/Test_2.jar /home/james/Test_2 /home/james/output
```

```
12/09/18 12:08:25 INFO input.FileInputFormat: Total input paths to process : 1
```

```
12/09/18 12:08:25 INFO mapred.JobClient: Running job: job_201209180930_0001
```

```
12/09/18 12:08:26 INFO mapred.JobClient: map 0% reduce 0%
```

```
12/09/18 12:08:33 INFO mapred.JobClient: map 100% reduce 0%
```

```
12/09/18 12:08:45 INFO mapred.JobClient: map 100% reduce 100%
```

```
12/09/18 12:08:47 INFO mapred.JobClient: Job complete: job_201209180930_0001
```

```
12/09/18 12:08:47 INFO mapred.JobClient: Counters: 18
```

```
12/09/18 12:08:47 INFO mapred.JobClient: Job Counters
```

```
12/09/18 12:08:47 INFO mapred.JobClient: Launched reduce tasks=1
```

```
12/09/18 12:08:47 INFO mapred.JobClient: Launched map tasks=1
```

```
12/09/18 12:08:47 INFO mapred.JobClient: Data-local map tasks=1
```

```
12/09/18 12:08:47 INFO mapred.JobClient: FileSystemCounters
```

```
12/09/18 12:08:47 INFO mapred.JobClient: FILE_BYTES_READ=136
```

```
12/09/18 12:08:47 INFO mapred.JobClient: HDFS_BYTES_READ=134
```

```
12/09/18 12:08:47 INFO mapred.JobClient: FILE_BYTES_WRITTEN=304
```

```
12/09/18 12:08:47 INFO mapred.JobClient: HDFS_BYTES_WRITTEN=97
```

```
12/09/18 12:08:47 INFO mapred.JobClient: Test_2$Counter
```

```
12/09/18 12:08:47 INFO mapred.JobClient: LINESKIP=1
```

```
12/09/18 12:08:47 INFO mapred.JobClient: Map-Reduce Framework
```

```
12/09/18 12:08:47 INFO mapred.JobClient: Reduce input groups=3
```

2012.9.18





# Thanks

## FAQ时间