# 实验五 第一个MapReduce程序WordCount

**·实验要求：**

以Hadoop-2.7.3下的README.txt文件内容为例，不考虑标点符号。统计该文本中的单词出现次数。

开发一个MapReduce程序，用来统计文本文件中单词出现的次数。并在hadoop集群中运行，查看运行结果。

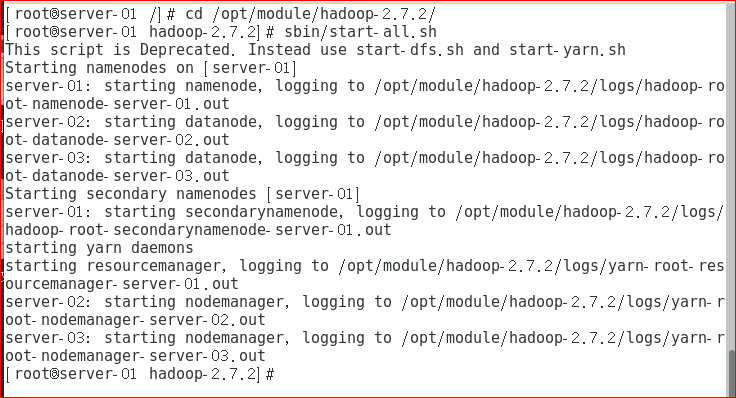
**·实验环境：**

Hadoop集群正常启动，eclipse工具。

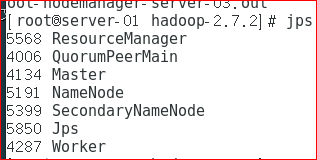
**·实验步骤：**

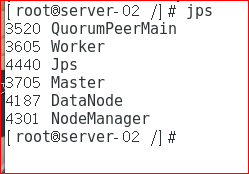
1. 启动hadoop服务：
   1. 启动hadoop集群：

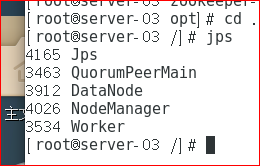
|  |
| --- |
| cd /opt/module/hadoop-2.7.2  sbin/./start-all.sh |



用jps命令查看已启动的服务：

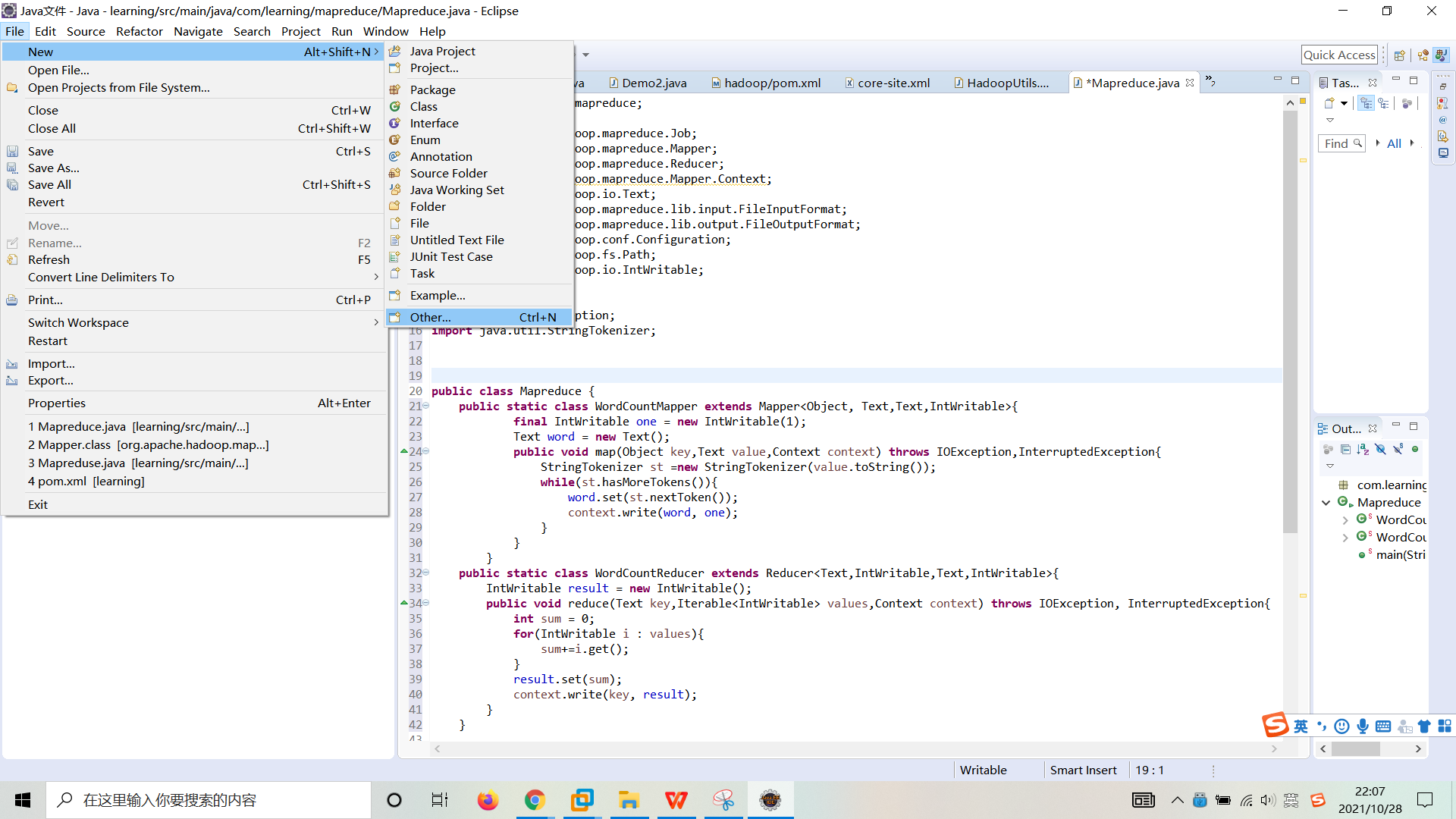


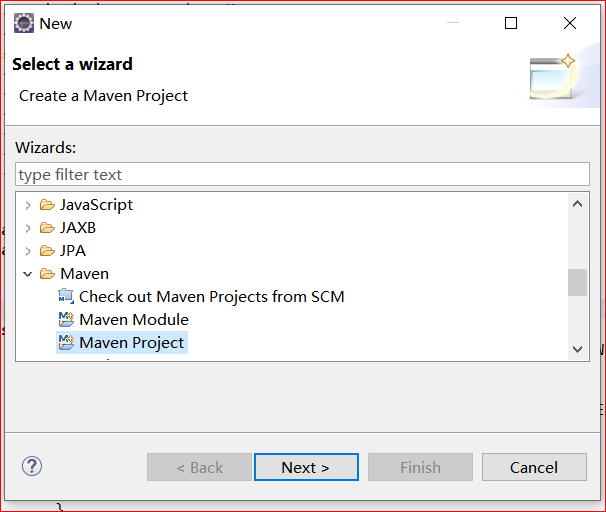


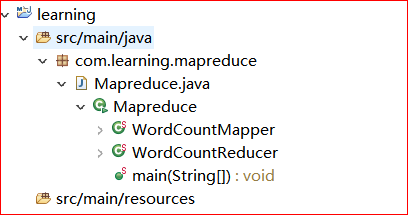
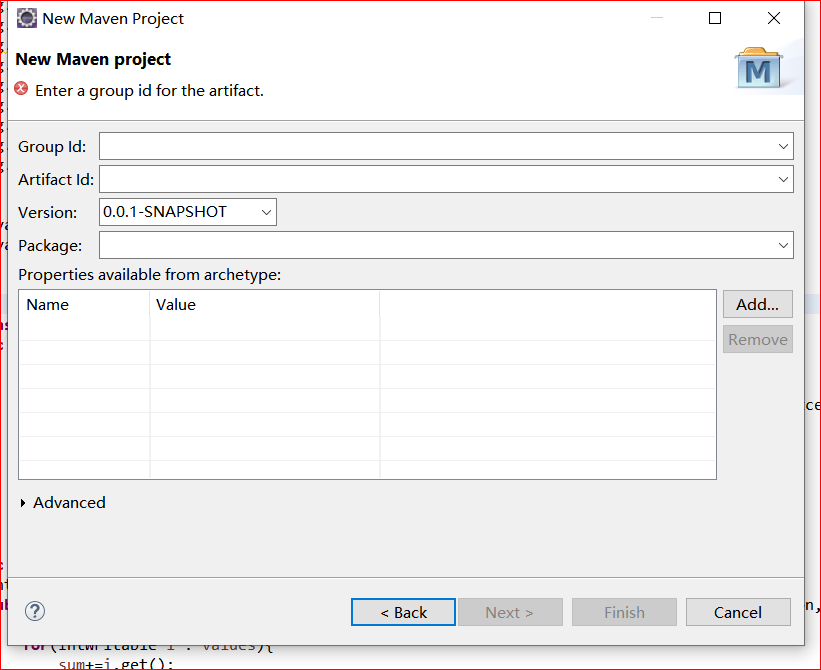
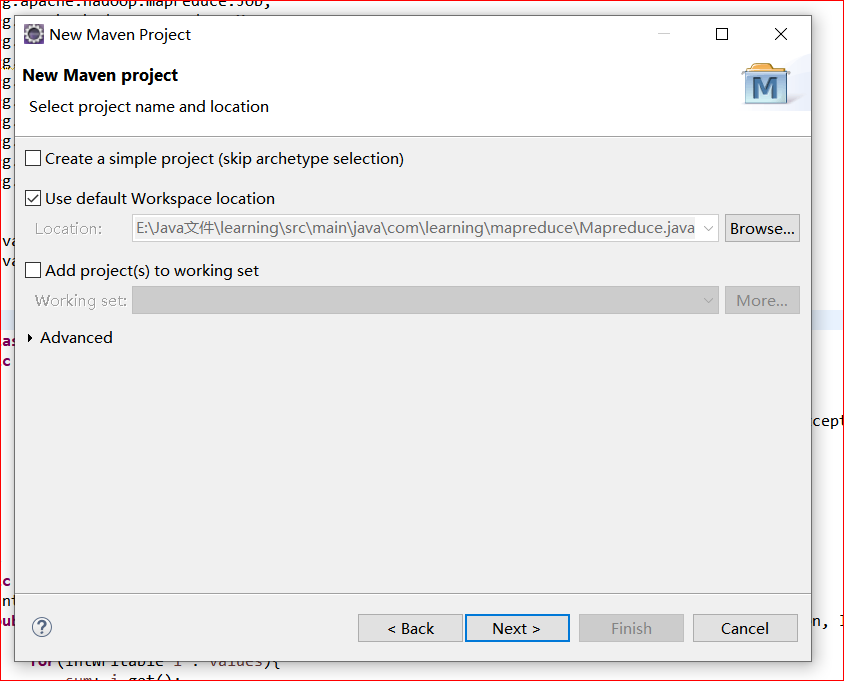


1. 开发WordCount程序

2.1、打开eclipse工具并新建Maven项目MapReduce，并配置buildpath（设置jdk版本为1.8）







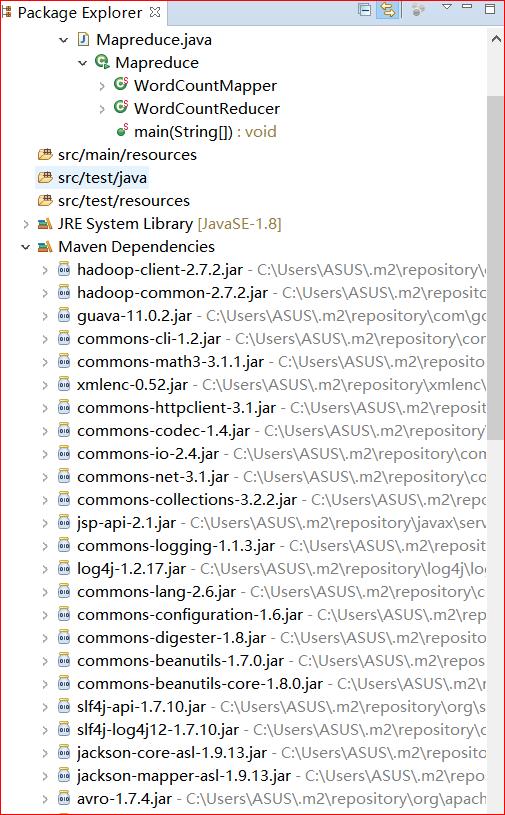
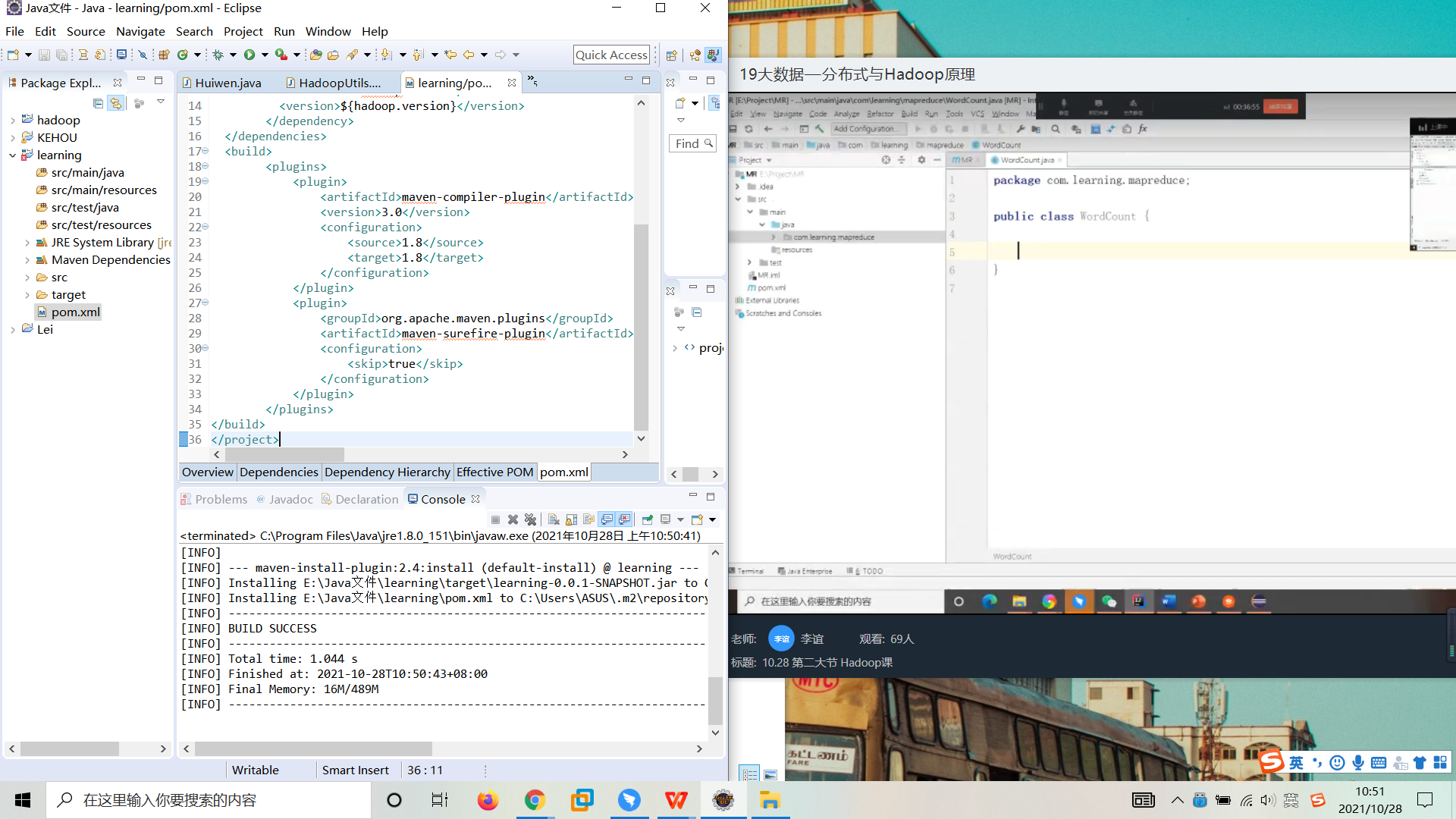
写入相关信息

新建Maven项目后，pom.xml内容：

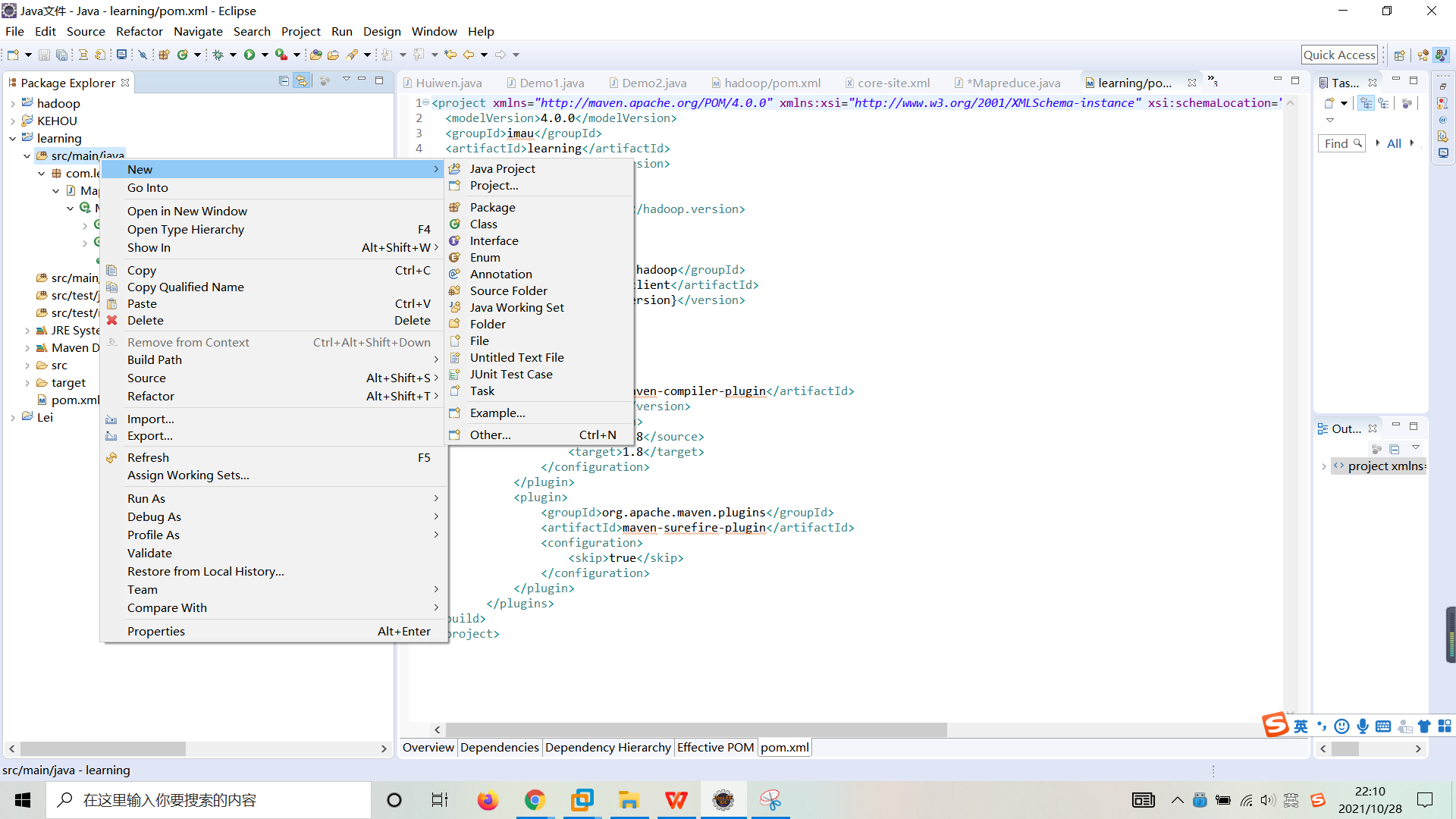
|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>imau</groupId>  <artifactId>learning</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>hadoop</name>  <properties>  <hadoop.version>2.7.2</hadoop.version>  </properties>  <dependencies>  <dependency>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-client</artifactId>  <version>${hadoop.version}</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.0</version>  <configuration>  <source>1.8</source>  <target>1.8</target>  </configuration>  </plugin>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-surefire-plugin</artifactId>  <configuration>  <skip>true</skip>  </configuration>  </plugin>  </plugins>  </build>  </project> |



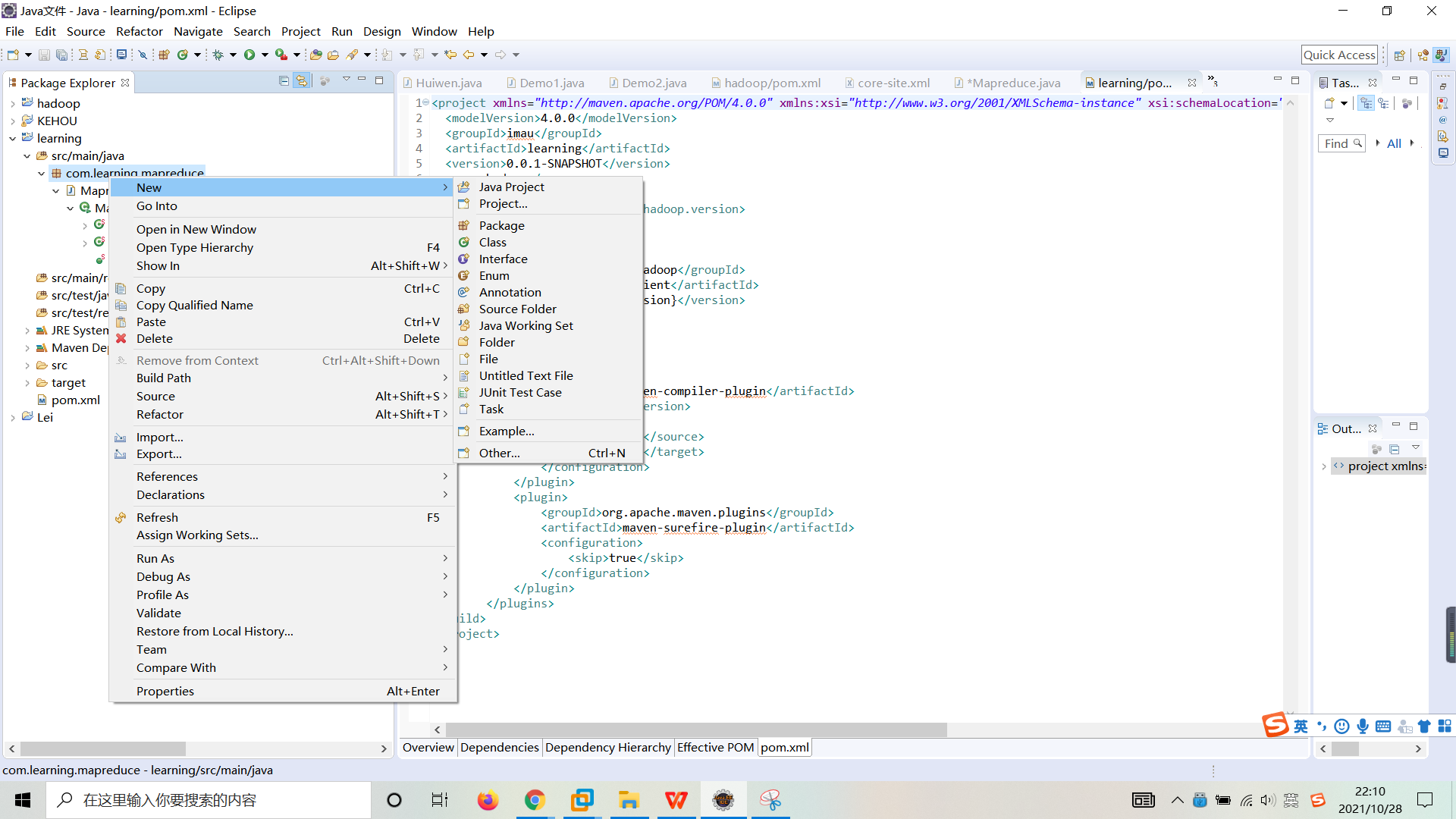
保存后会下载相关MAven包：



在src/main/java/下新建 com.learning.mapreduce 包：

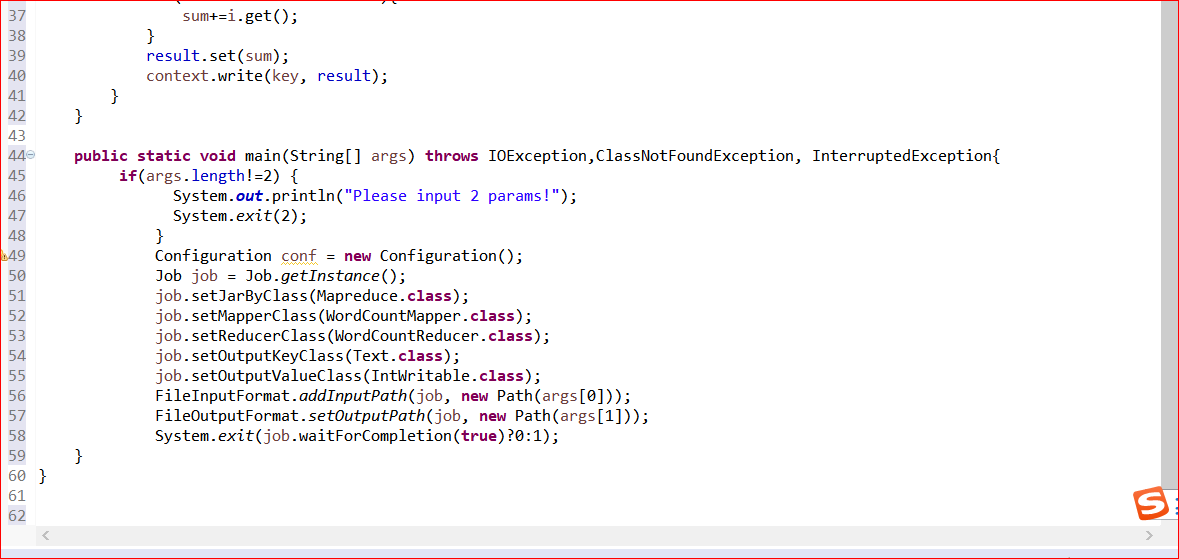
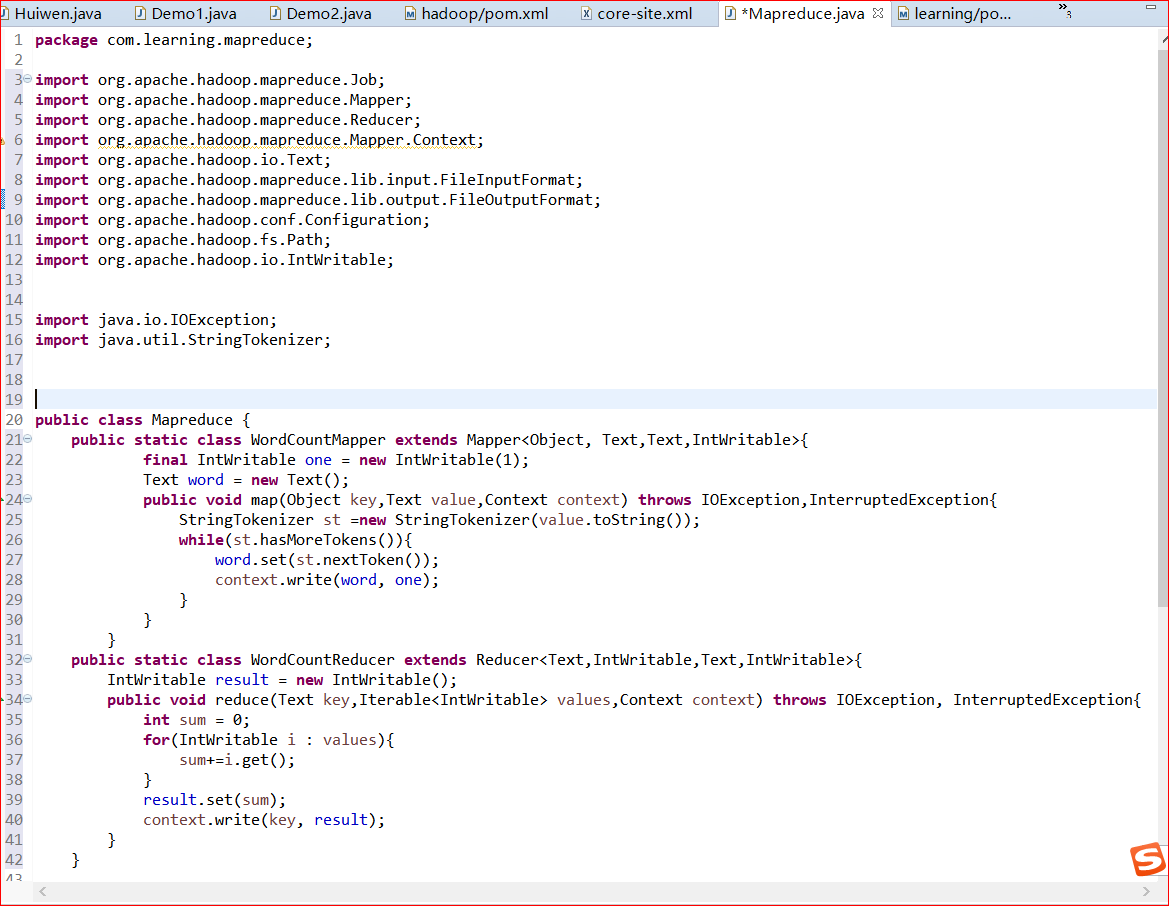


在com.learning.mapreduce下新建WordCount class文件：



MapReduce.java内容：

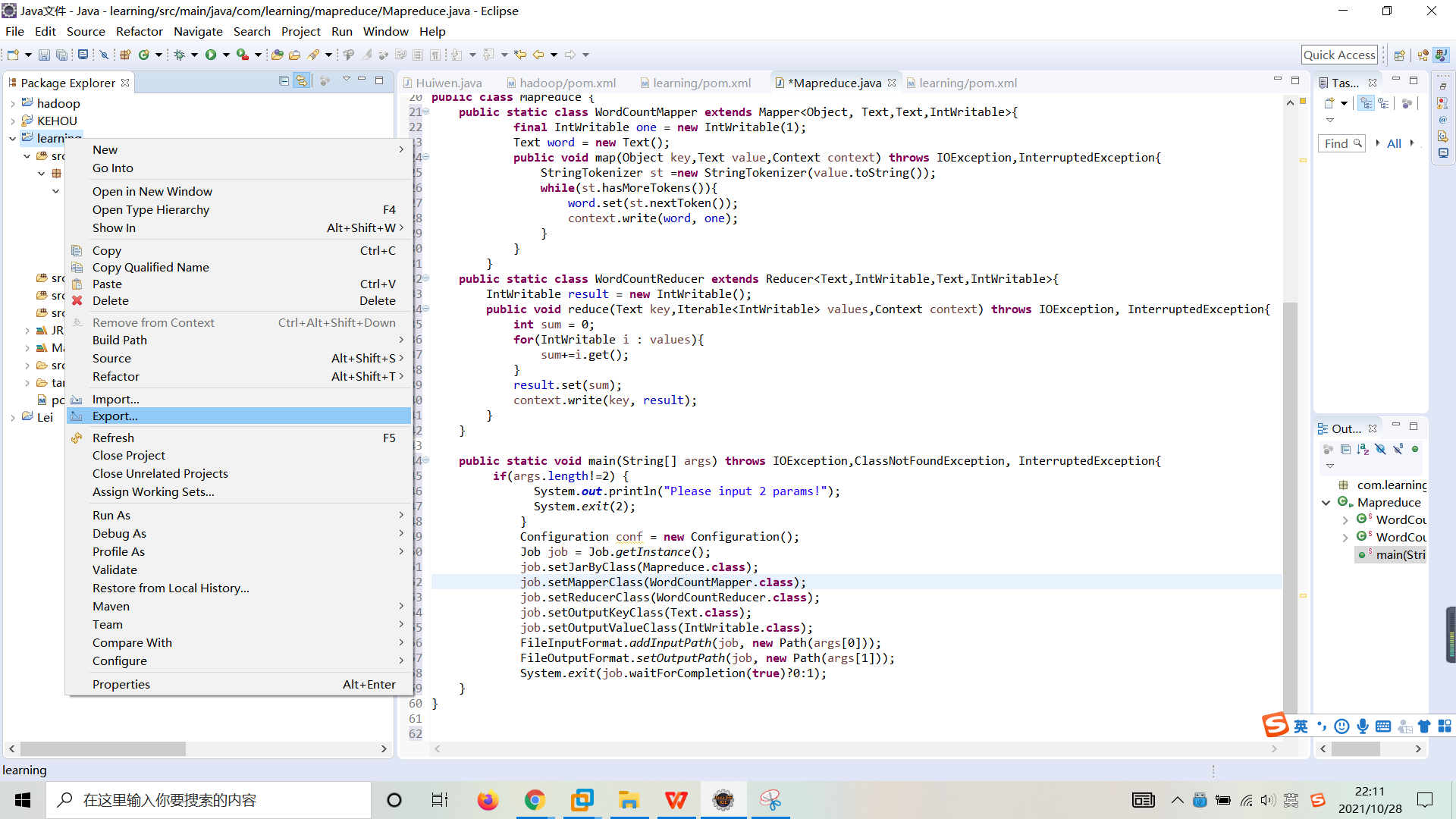
|  |
| --- |
| **package** com.learning.mapreduce;  **import** org.apache.hadoop.mapreduce.Job;  **import** org.apache.hadoop.mapreduce.Mapper;  **import** org.apache.hadoop.mapreduce.Reducer;  **import** org.apache.hadoop.mapreduce.Mapper.Context;  **import** org.apache.hadoop.io.Text;  **import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.Path;  **import** org.apache.hadoop.io.IntWritable;  **import** java.io.IOException;  **import** java.util.StringTokenizer;  **public** **class** Mapreduce {  **public** **static** **class** WordCountMapper **extends** Mapper<Object, Text,Text,IntWritable>{  **final** IntWritable one = **new** IntWritable(1);  Text word = **new** Text();  **public** **void** map(Object key,Text value,Context context) **throws** IOException,InterruptedException{  StringTokenizer st =**new** StringTokenizer(value.toString());  **while**(st.hasMoreTokens()){  word.set(st.nextToken());  context.write(word, one);  }  }  }  **public** **static** **class** WordCountReducer **extends** Reducer<Text,IntWritable,Text,IntWritable>{  IntWritable result = **new** IntWritable();  **public** **void** reduce(Text key,Iterable<IntWritable> values,Context context) **throws** IOException, InterruptedException{  **int** sum = 0;  **for**(IntWritable i : values){  sum+=i.get();  }  result.set(sum);  context.write(key, result);  }  }  **public** **static** **void** main(String[] args) **throws** IOException,ClassNotFoundException, InterruptedException{  **if**(args.length!=2) {  System.***out***.println("Please input 2 params!");  System.*exit*(2);  }  Configuration conf = **new** Configuration();  Job job = Job.*getInstance*();  job.setJarByClass(Mapreduce.**class**);  job.setMapperClass(WordCountMapper.**class**);  job.setReducerClass(WordCountReducer.**class**);  job.setOutputKeyClass(Text.**class**);  job.setOutputValueClass(IntWritable.**class**);  FileInputFormat.*addInputPath*(job, **new** Path(args[0]));  FileOutputFormat.*setOutputPath*(job, **new** Path(args[1]));  System.*exit*(job.waitForCompletion(**true**)?0:1);  }  } |



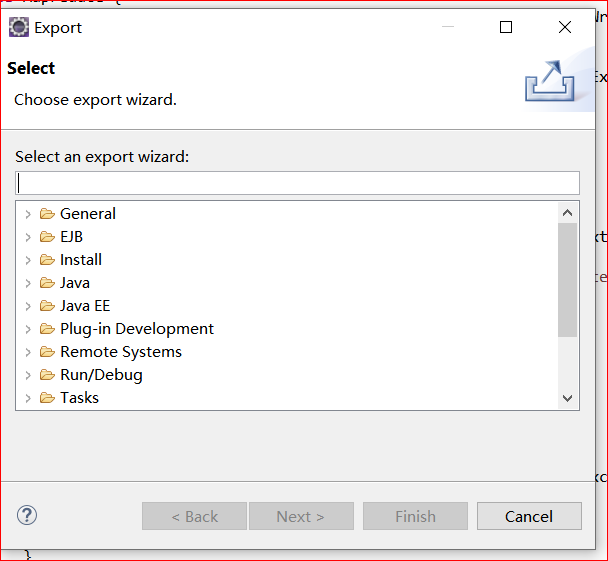
1. 到处jar包：

（\*我们需要将开发的WordCount.java编译后的class打成jar包，并上传到master服务器上才能运行。可以使用jar命令也可以使用Eclipse里的导出jar包功能。\*）

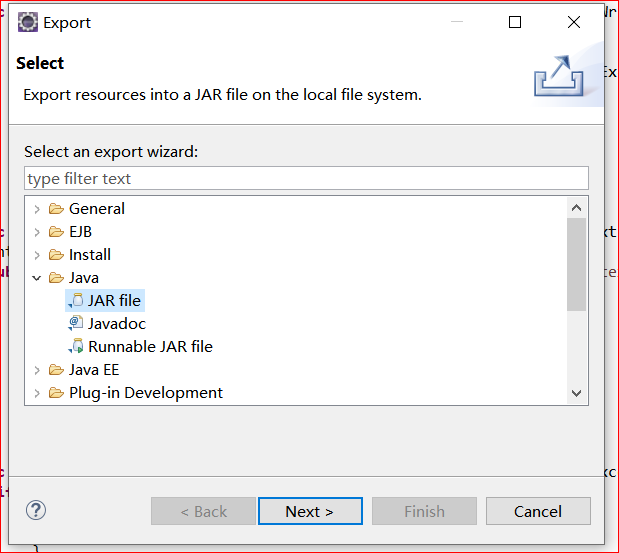
在MapReduce项目上单击右键，点击“Export”：



点击java：

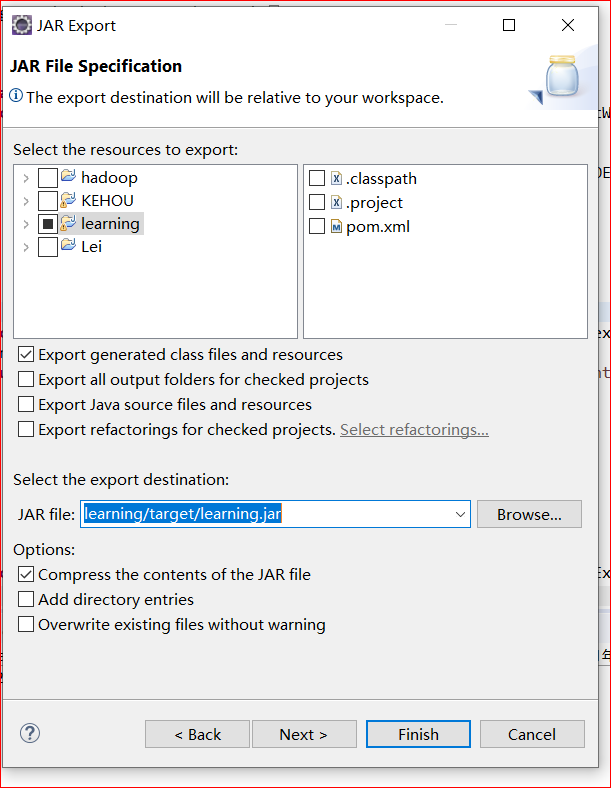


选中java file：



取消选中

点击“next”按钮，进入到下面的对话框，取消选中的三个选项并选择jar文件路径：



输出路径选择learning/target目录下，jar包名起为learning.jar.

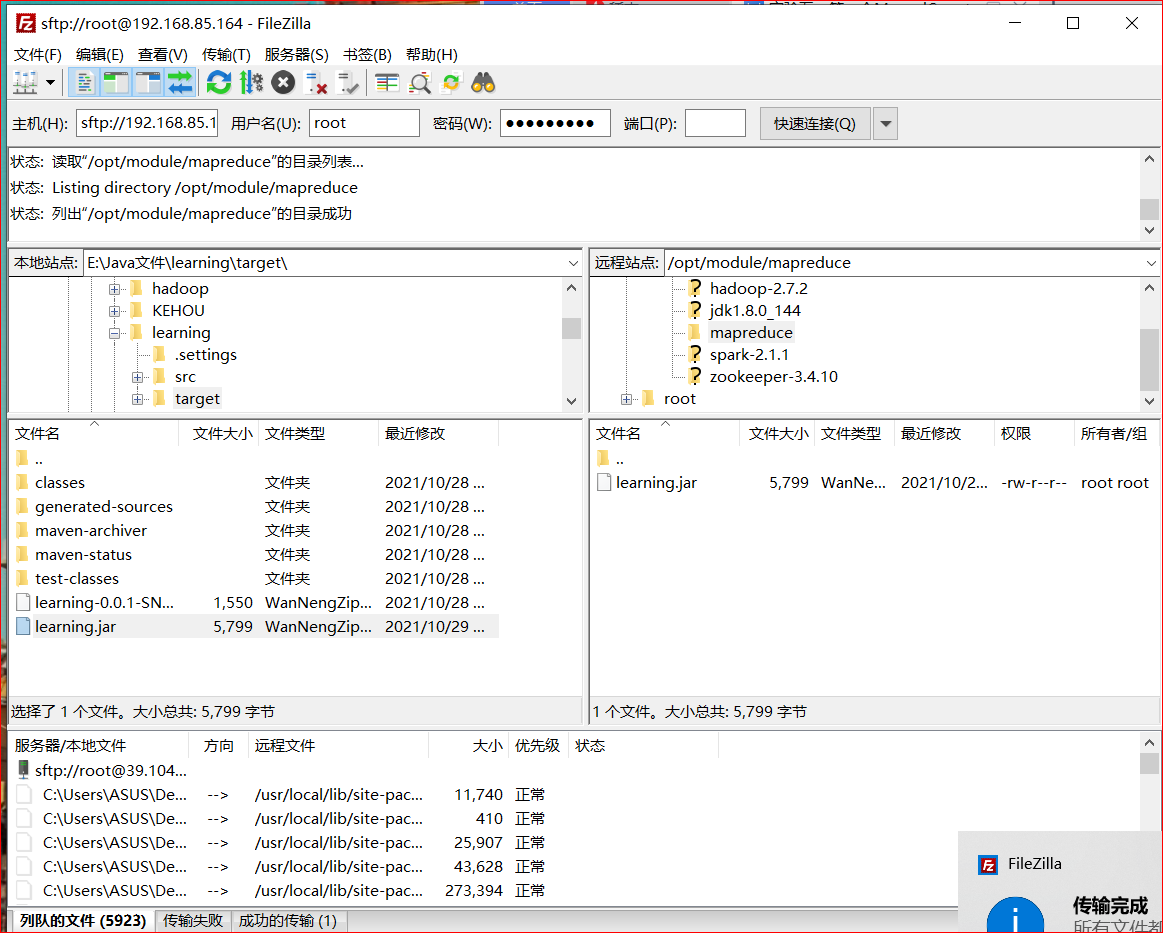
1. 将jar包文件上传到master服务器上

在server-01上的/opt/module/目录下新建目录MapReduce

|  |
| --- |
| mkdir /opt/module/mapreduce |

21

将上一步中导出的learning.jar文件使用filezilla上传到服务器的/opt/module/mapreduce下：



5、准备测试数据：

在HDFS集群中创建目录input，然后将文件/opt/module/hadoop2.7.2/README.txt上传到hdfs集群的/input下：

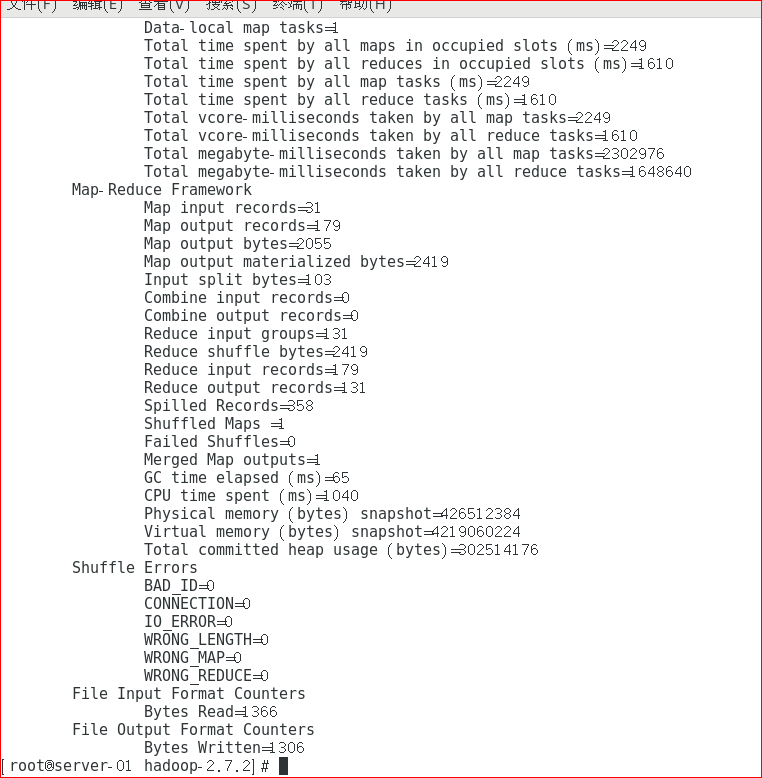
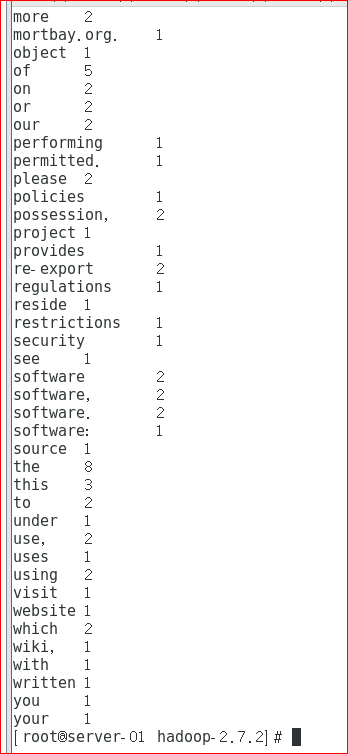
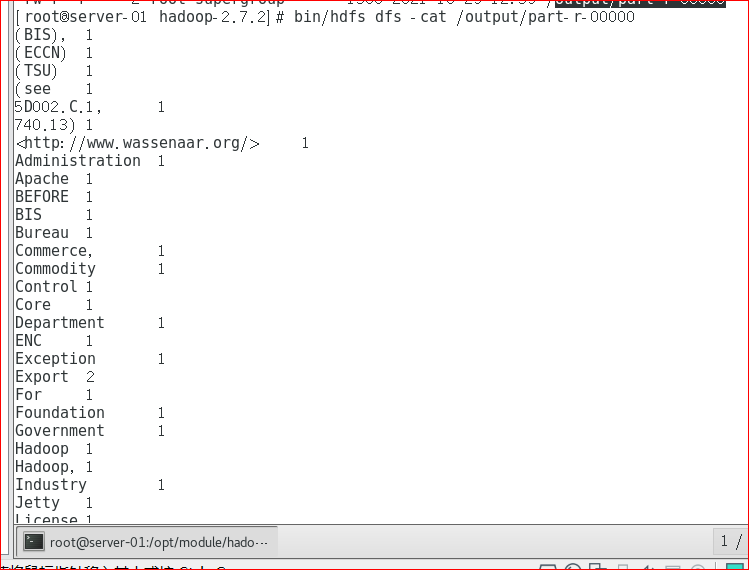
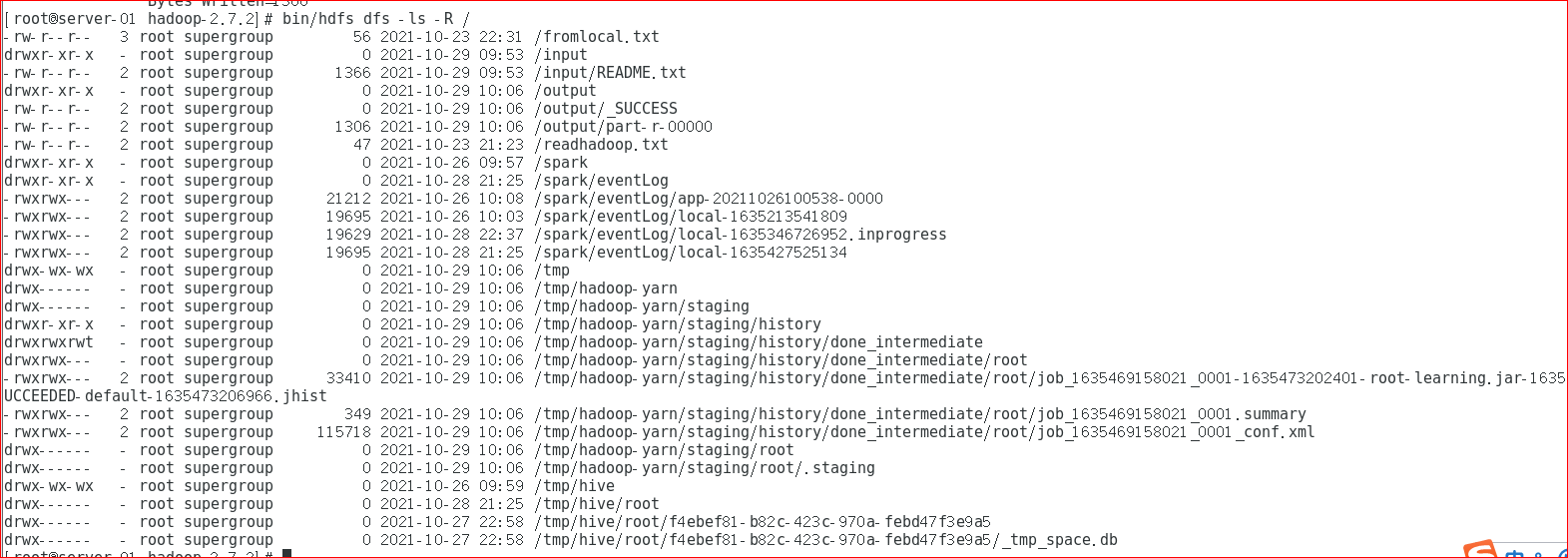
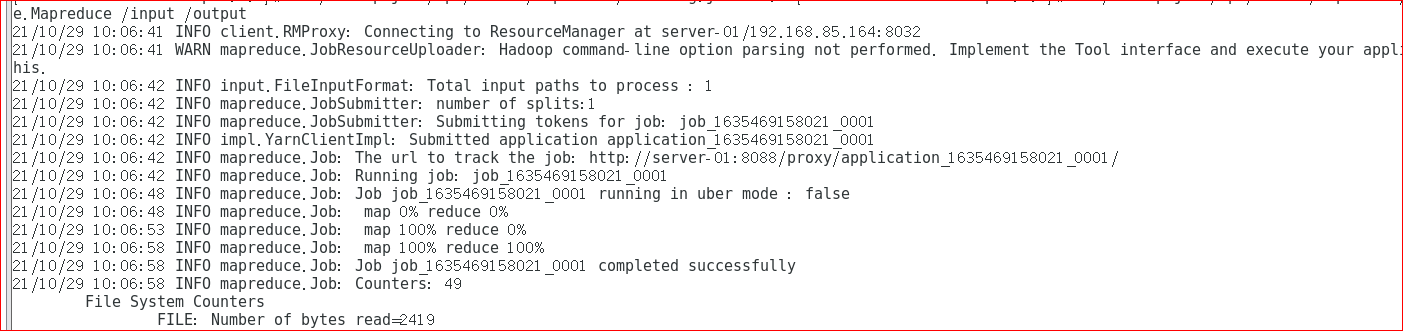
|  |
| --- |
| bin/hdfs dfs -mkdir /input  bin/hdfs dfs -put /opt/module/hadoop-2.7.2/README.txt /input/ |

2324

6、运行WordCount

使用命令：bin/hadoop jar运行WordCount程序；使用命令：bin/hdfs dfs -ls 查看运行成功之后生成了哪些文件和目录；使用命令：bin/hdfs dfs -cat来查看运行结果：

|  |
| --- |
| bin/hadoop jar /opt/module/mapreduce/learning.jar com.leaning.mapreduce.Mapreduce /input /output |
| bin/hdfs dfs -ls -R /output  bin/hdfs dfs -cat /output/part-r-00000 |

2625 

**·实验心得：**

学会了如何通过eclipse导出生成一个jar包，知道了java语句中导包的代码写法，知道了如何在hdfs集群中上传文件、目录，删除文件、目录，知道了如何在hdfs集群中运行WordCount程序，如何查看生成文件内容。