**大数据分析技术与应用实验报告**

**实验二:MapReduce实例实现**

目 录

[1 实验任务 1](#_Toc107143904)

[2 实验过程 1](#_Toc107143905)

[2.1 Hadoop、Eclipse编程环境搭建 1](#_Toc107143906)

[1. 启动Hadoop 1](#_Toc107143907)

[2. 配置Hadoop-Eclipse-Plugin 1](#_Toc107143908)

[3. 在Eclipse中操作HDFS文件 3](#_Toc107143909)

[2.2 Eclipse运行Wordcount实例 7](#_Toc107143910)

[1. 在Eclipse创建MapReduce项目 7](#_Toc107143911)

[2. 创建WordCount.java文件，并编写WordCount程序 9](#_Toc107143912)

[3. 在Eclipse中运行WordCount程序 12](#_Toc107143913)

[2.3基于Hadoop的数据去重实例实现 14](#_Toc107143914)

[1. 在Eclipse中的HDFS上创建data文件夹 14](#_Toc107143915)

[2. 往data文件夹中上传数据文件data1.txt和data2.txt 14](#_Toc107143916)

[3. 创建数据去重程序 15](#_Toc107143917)

[4. 将数据去重项目打包成一个jar包，放到本身项目的jar包库中 16](#_Toc107143918)

[5. 运行数据去重程序 17](#_Toc107143919)

[6. 试验运行结果 18](#_Toc107143920)

[3 实验总结 18](#_Toc107143921)

# 1 实验任务

1. 搭建Hadoop、Eclipse编程环境
2. 在Eclipse中操作HDFS
3. 在Eclipse中运行Wordcount程序
4. 参照Wordcount程序，编程实现数据去重程序

# 2 实验过程

## 2.1 Hadoop、Eclipse编程环境搭建

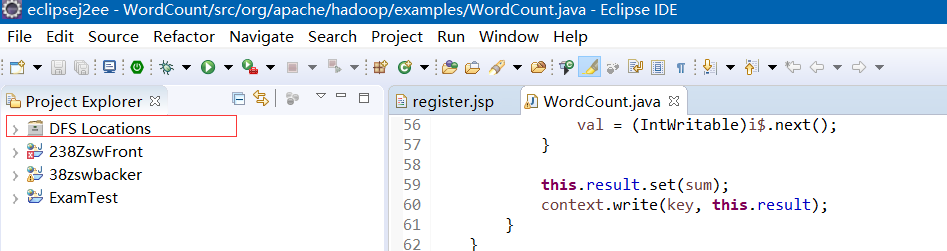
### 启动Hadoop

运行 %HADOOP\_HOME%/sbin/start-all.cmd

### 配置Hadoop-Eclipse-Plugin

把下载好的hadoop-eclipse-plugin-2.7.3.jar文件放在eclipse安装目录下的plugins文件夹下。

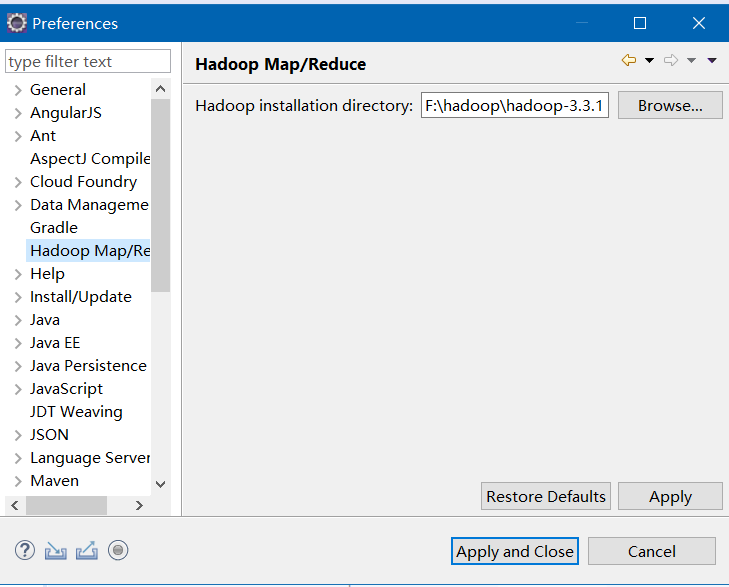
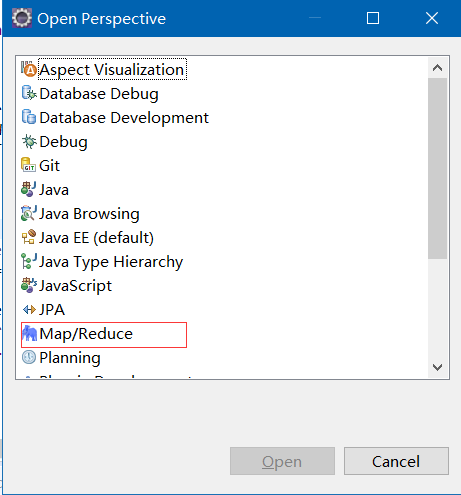
安装hadoop-eclipse-plugin插件

启动Eclipse可以看到Project Explorer下的DFS Locations。

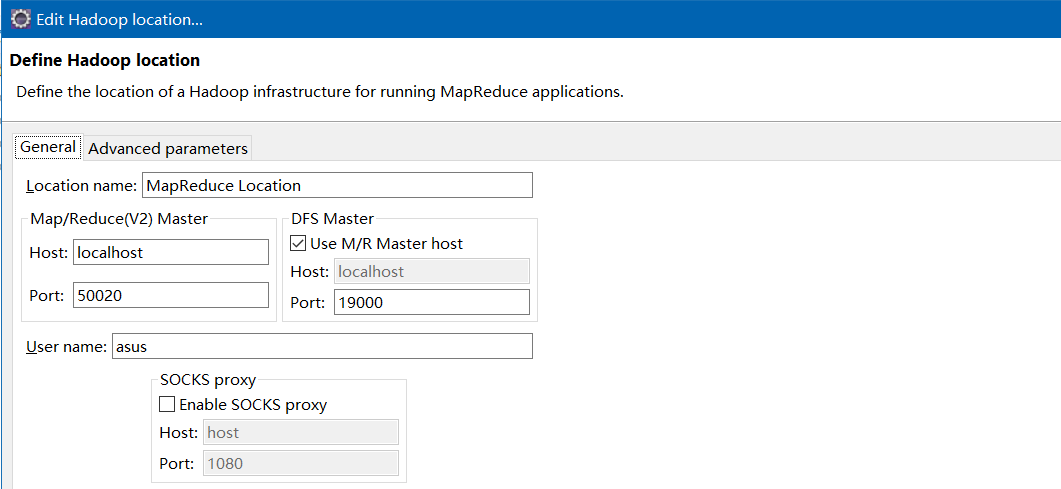
安装好hadoop-eclipse-plugin插件的效果

在eclipse中配置Hadoop安装目录，选择Windows->Preferences->Hadoop Map/Reduce 选择Hadoop安装目录。

选择Hadoop安装目录

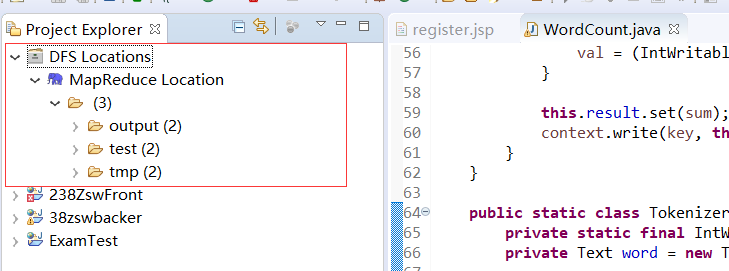
打开Map/Reduce开发视图，选择Window->Perspective->Open Perspective->Other，选择Map/Reduce。

打开Map/Reduce开发视图

配置环境与启动的Hadoop进行连接，点击Eclipse下的Map/Reduce Locations，右击选择New Hadoop Location。填入Hadoop的配置信息。

配置与Hadoop连接

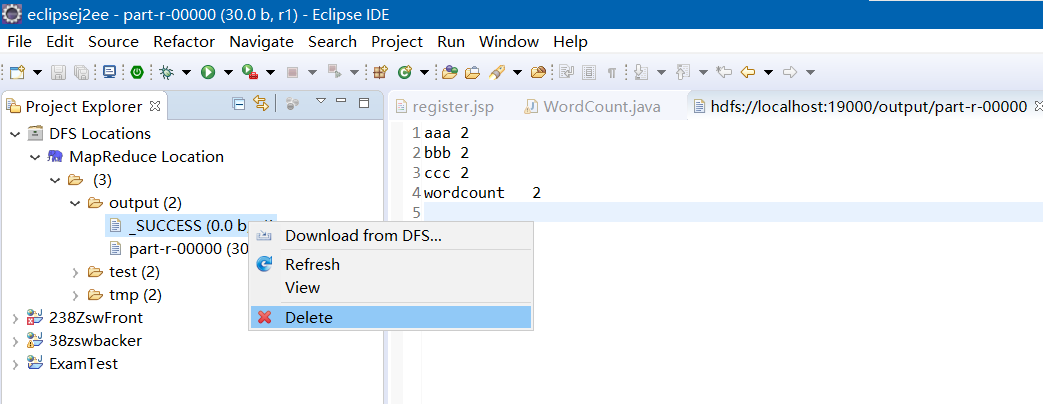
### 在Eclipse中操作HDFS文件

点击Project Explorer中的DFS Locations，显示配置好的MapReduce Location，再点击MapReduce Location显示连接上的Hadoop的文件系统目录。

Hadoop文件系统目录展示

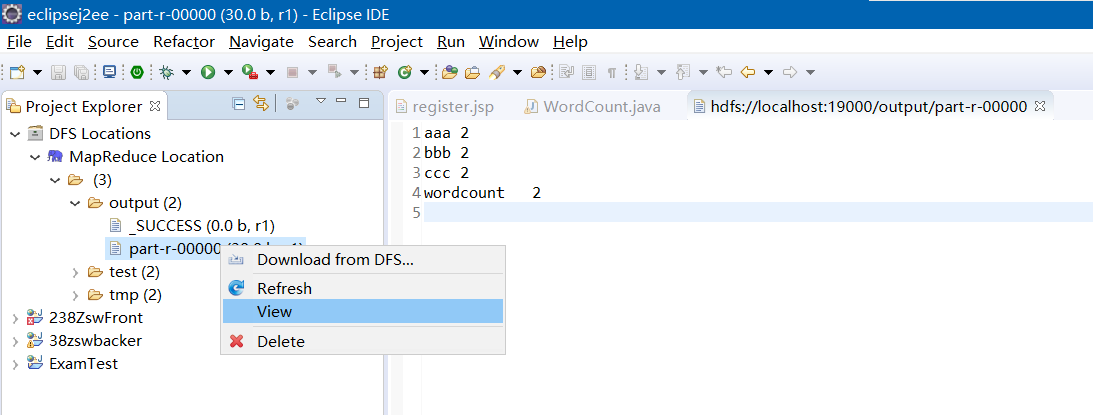
**删除文件**

左键点击output文件，右键点击SUCEESS文件，选择Delete对文件进行删除



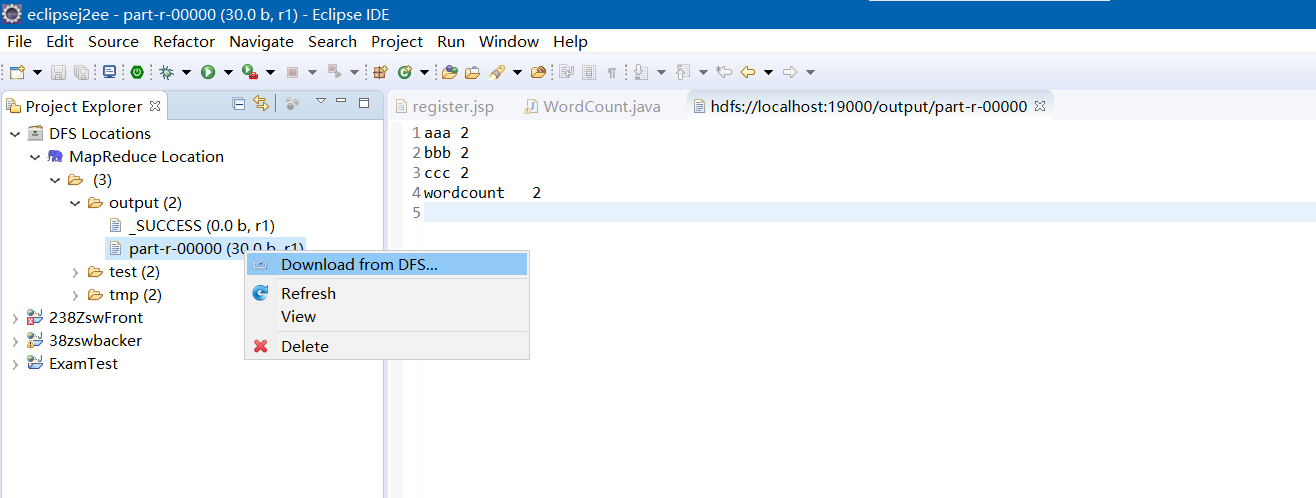
文件删除

**查看文件操作**

左键点击output文件，右键点击View按钮，实现对文件查看

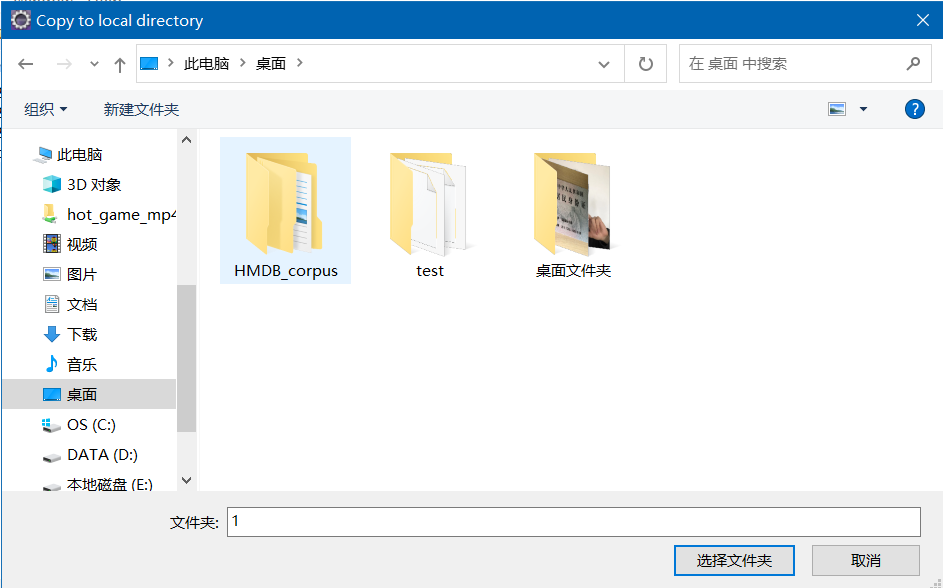
文件查看

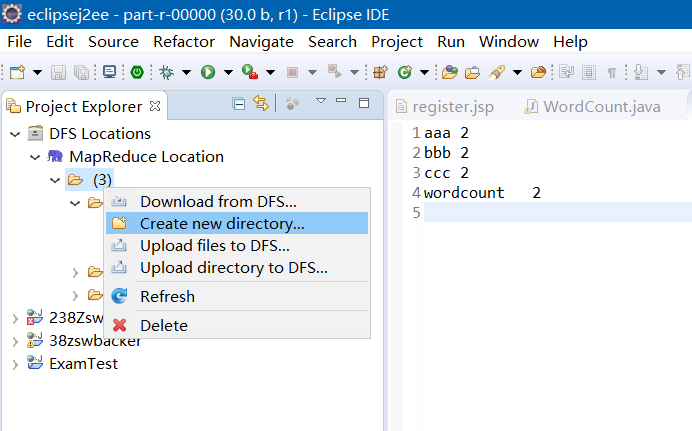
**下载文件**

左键点击output文件，右键点击Download from DFS，弹出下载位置窗口，选择下载位置，文件可以从HDFS下载。

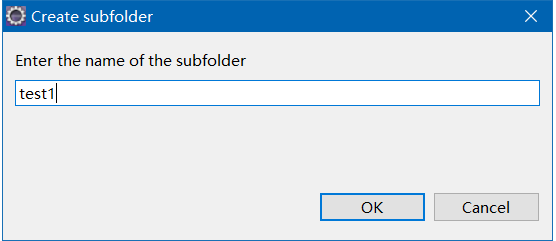
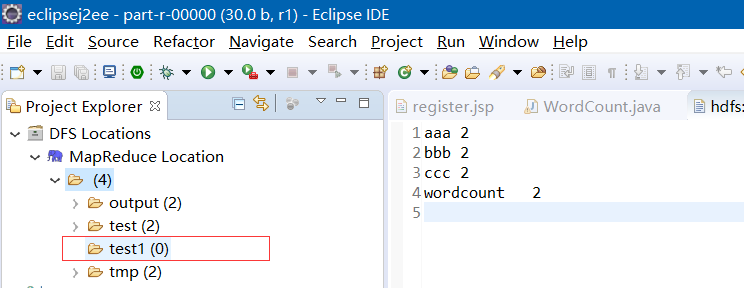
文件下载

文件下载窗口

**新建文件夹test1**

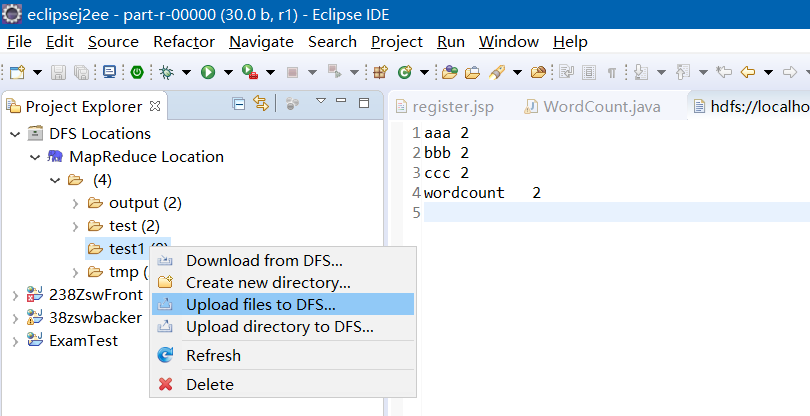
点击文件右键，并选择Create new directory

新建文件夹

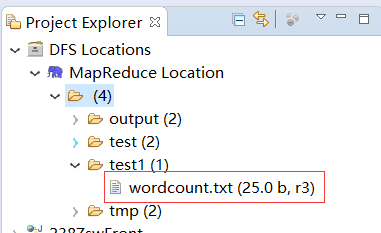
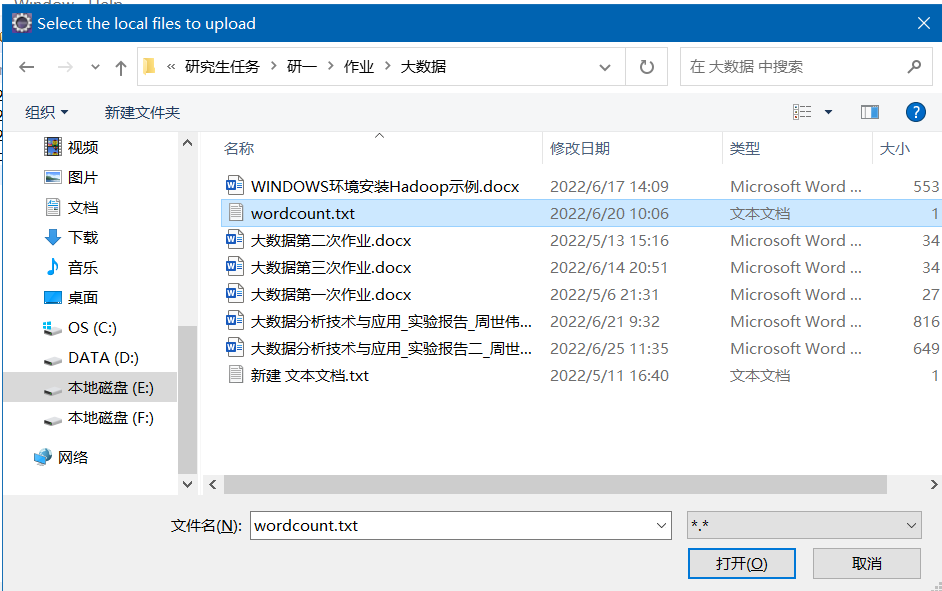
新建test1文件夹

test1文件夹

**上传文件到HDFS**

右键test1文件夹，选择Upload files to DFS

上传文件到HDFS

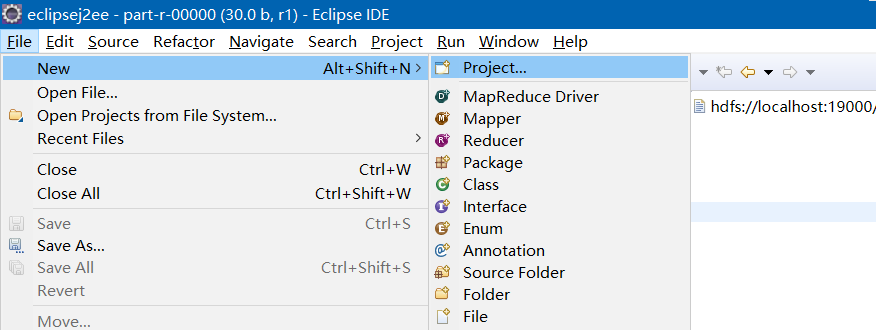
选择wordcount.txt文件上传

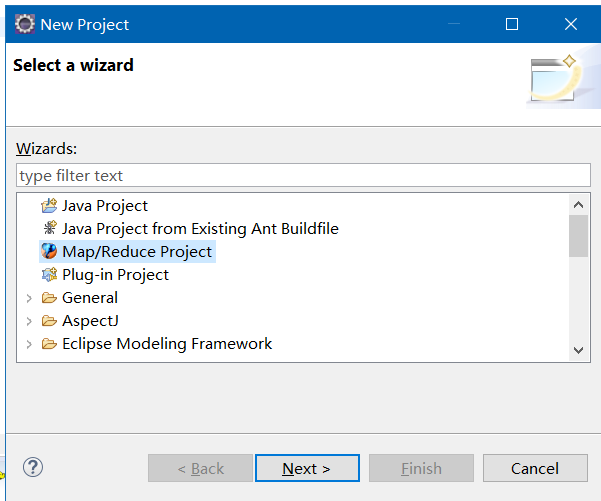
上传的wordcount.txt文件夹

## Eclipse运行Wordcount实例

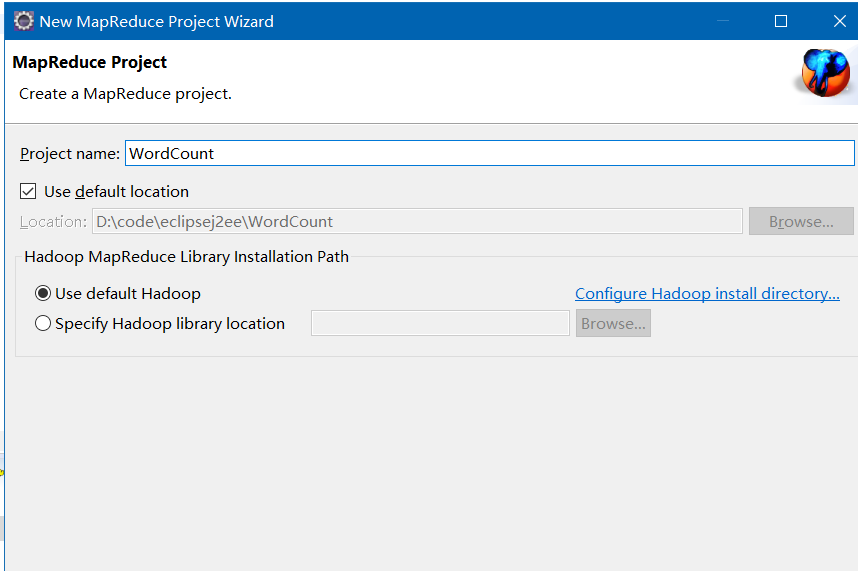
### 在Eclipse创建MapReduce项目

点击Eclipse左上角File->New->Project,选择Map/Reduce Project,填写项目名称WordCount,然后点击Finish

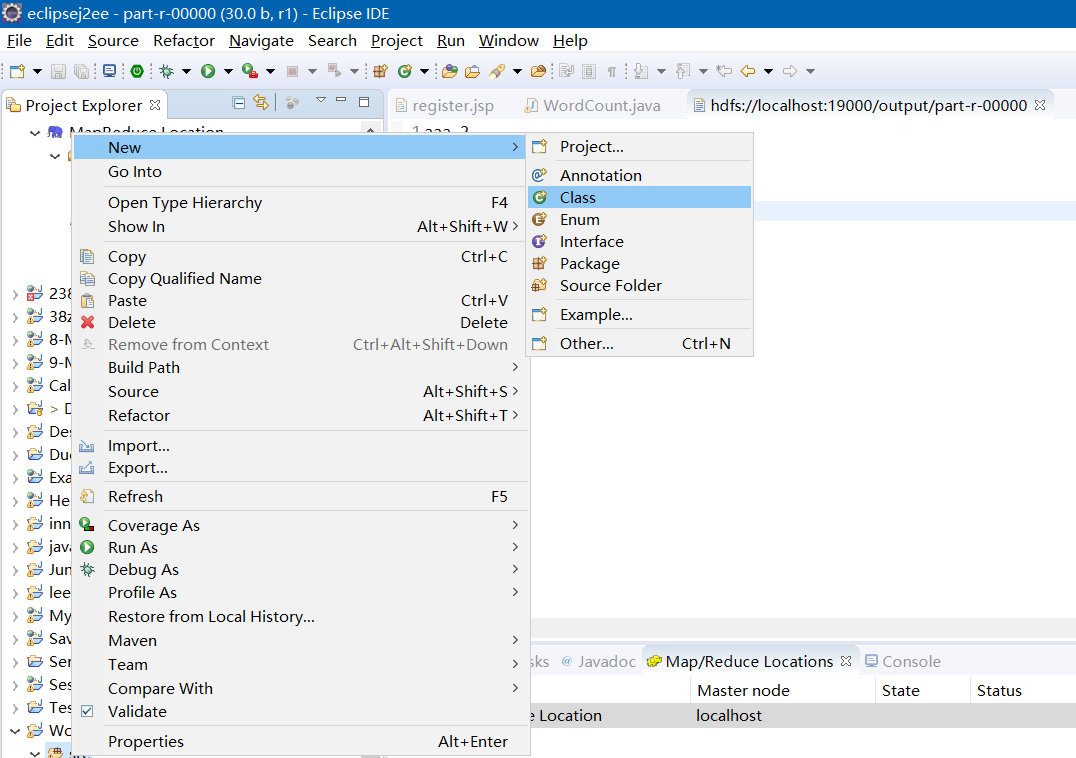


创建Map/Reduce Project

创建Map/Reduce Project

创建Map/Reduce Project

### 创建WordCount.java文件，并编写WordCount程序

右击src文件夹，选择New->Class，创建WordCount类

创建WordCount类

在WordClass中，编写如下代码

package org.apache.hadoop.examples;

import java.io.IOException;

import java.util.Iterator;

import java.util.StringTokenizer;

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.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

public WordCount() {

}

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

Configuration conf = new Configuration();

String[] otherArgs = (new GenericOptionsParser(conf, args)).getRemainingArgs();

if(otherArgs.length < 2) {

System.err.println("Usage: wordcount <in> [<in>...] <out>");

System.exit(2);

}

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(WordCount.class);

job.setMapperClass(WordCount.TokenizerMapper.class);

job.setCombinerClass(WordCount.IntSumReducer.class);

job.setReducerClass(WordCount.IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

for(int i = 0; i < otherArgs.length - 1; ++i) {

FileInputFormat.addInputPath(job, new Path(otherArgs[i]));

}

FileOutputFormat.setOutputPath(job, new Path(otherArgs[otherArgs.length - 1]));

System.exit(job.waitForCompletion(true)?0:1);

}

public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

private IntWritable result = new IntWritable();

public IntSumReducer() {

}

public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {

int sum = 0;

IntWritable val;

for(Iterator i$ = values.iterator(); i$.hasNext(); sum += val.get()) {

val = (IntWritable)i$.next();

}

this.result.set(sum);

context.write(key, this.result);

}

}

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

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

private Text word = new Text();

public TokenizerMapper() {

}

public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while(itr.hasMoreTokens()) {

this.word.set(itr.nextToken());

context.write(this.word, one);

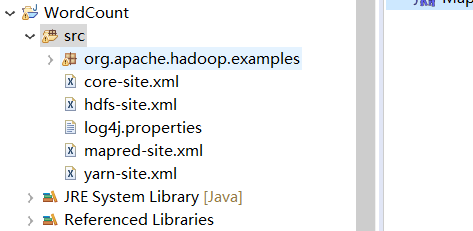
}

}

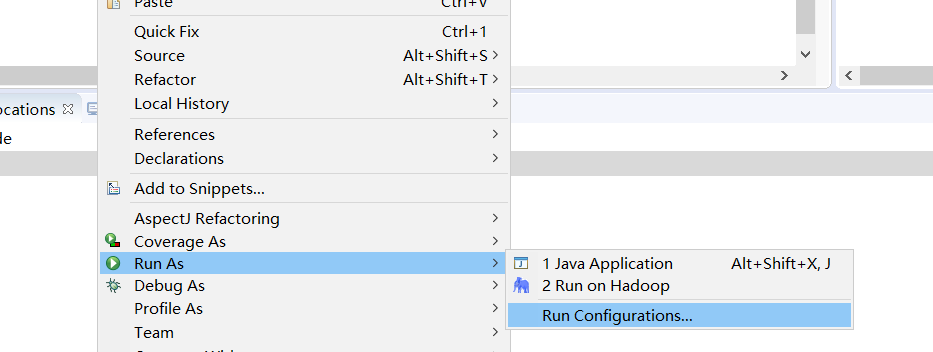
}

}

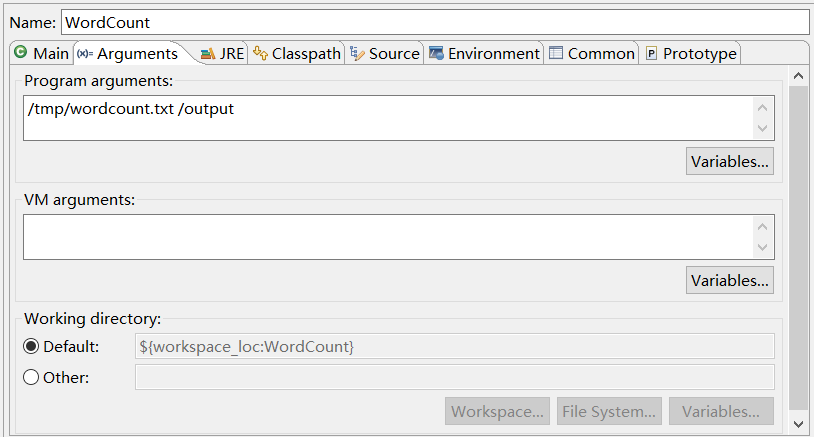
### 在Eclipse中运行WordCount程序

将hadoop目录下的core-site.xml、hdfs-site.xml、log4j.properties、mapred-site.xml、yarn-site.xml复制到WordCount下的src目录下

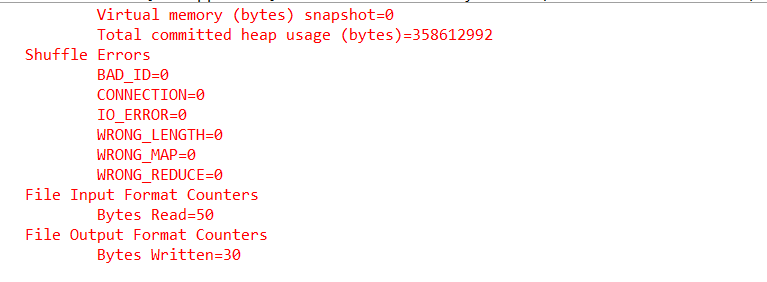
复制配置文件到WordCount项目

右击程序，选择Run As->Run Configurations

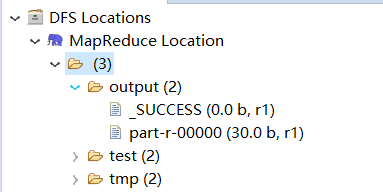
运行WordCount程序

设置运行参数，第一个参数是要统计的单词信息的文件路径，第二个是输出结果地址。

配置运行参数

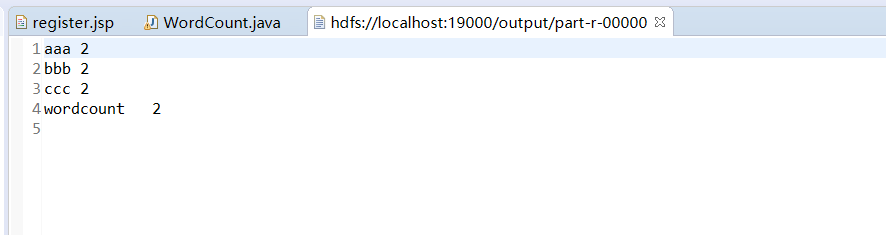
运行结果如下：

运行结果

生成的output文件夹在DFS中如图所示：

output文件夹

生成的统计结果文件如图所示：

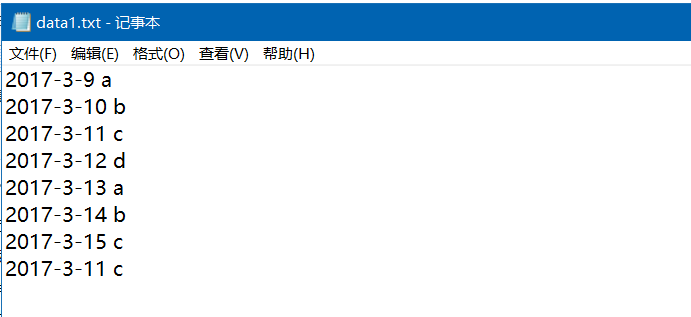
统计结果文件

## 2.3基于Hadoop的数据去重实例实现

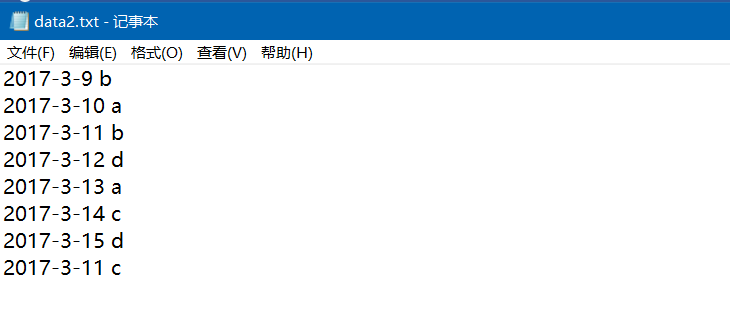
### 在Eclipse中的HDFS上创建data文件夹

HDFS上的data文件夹

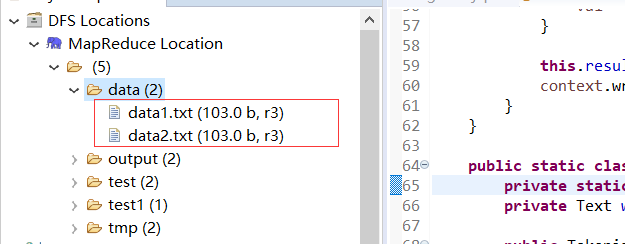
### 往data文件夹中上传数据文件data1.txt和data2.txt

data1.txt的文件内容如下:

data1.txt文件内容

data2.txt的文件内容如下：

data2.txt文件内容

将data1.txt和data2.txt通过Eclipse的HDFS上传到data文件夹下

上传data1.txt和data2.txt到data文件夹

### 创建数据去重程序

public class Process {

public Process() {

}

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

// TODO Auto-generated method stub

Configuration conf = new Configuration();

String[] otherArgs = new GenericOptionsParser(conf,args).getRemainingArgs();

if(otherArgs.length != 2){

System.err.println("Usage WordCount <int> <out>");

System.exit(2);

}

Job job = Job.getInstance(conf, "Data Process");

job.setJarByClass(Process.class);

job.setMapperClass(Process.DedupMapper.class);

job.setCombinerClass(Process.DedupReducer.class);

job.setReducerClass(Process.DedupReducer.class);

job.setOutputKeyClass(Text.class);

//设置输出类型

job.setOutputValueClass(NullWritable.class);

FileInputFormat.addInputPath(job, new Path(otherArgs[0]));

FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

public static class DedupMapper extends Mapper<LongWritable, Text, Text, NullWritable> {

private Text line = new Text();//每行数据

@Override

protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, NullWritable>.Context context)throws IOException, InterruptedException {

line = value;

context.write(line,NullWritable.get());

}

}

public static class DedupReducer extends Reducer<Text, NullWritable, Text, NullWritable> {

//重写reduce()方法

@Override

protected void reduce(Text key, Iterable<NullWritable> values, Reducer<Text, NullWritable, Text, NullWritable>.Context context) throws IOException, InterruptedException {

context.write(key,NullWritable.get());

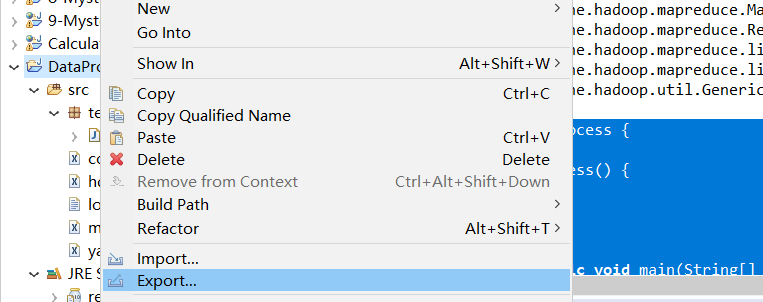
}

}

}

### 将数据去重项目打包成一个jar包，放到本身项目的jar包库中

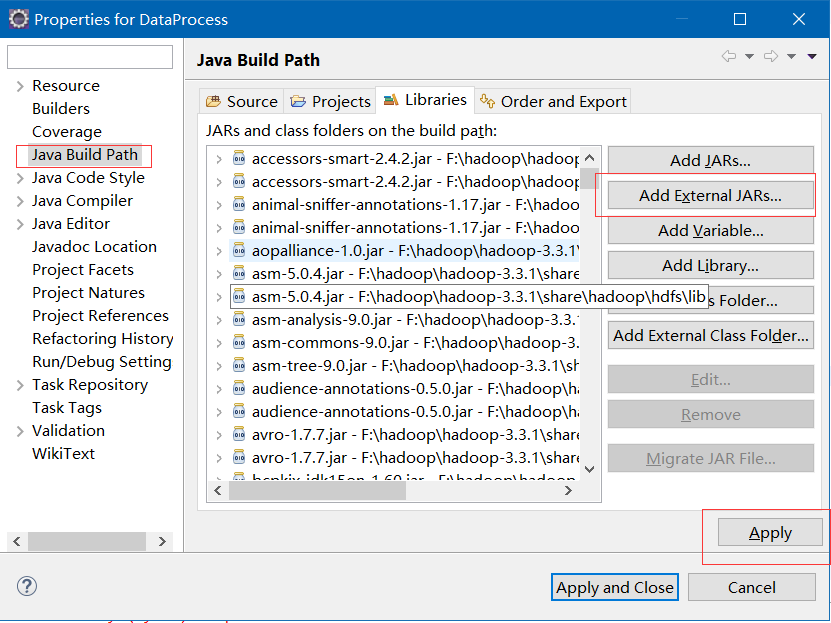
这样操作的目的是防止Map/Reduce找不到自己重写的map和reduce类

右击项目本身，选择Export

导出项目jar包

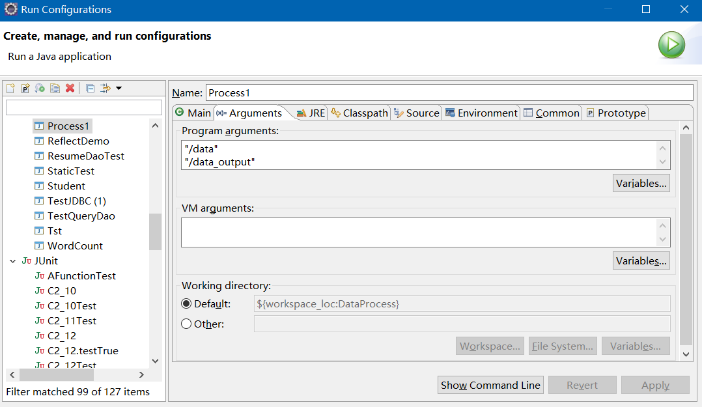
选择导出位置和文件类型。

右击项目本身，点击Properties，选择Java Build Path，点击Add External JARS，选择导出的JAR包，导入项目本身。



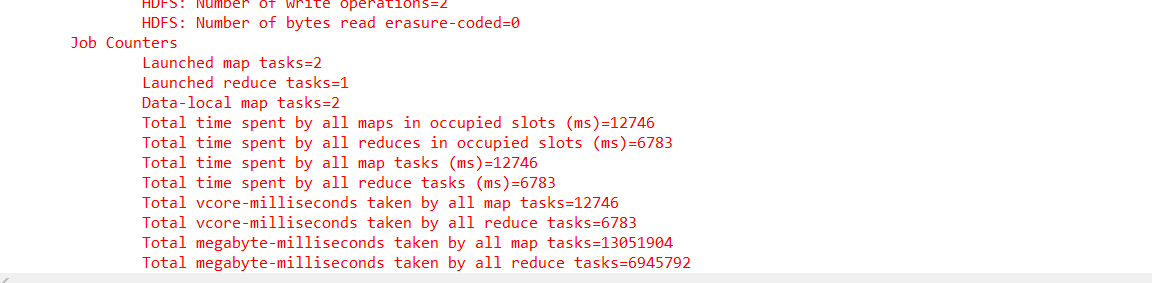
项目导入jar包

### 运行数据去重程序

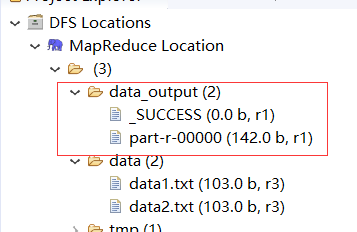
右击代码，选择Run As ->Run Configures，输入运行参数，第一个参数为输入文件路径，第二个参数为输出文件路径。

输入运行参数

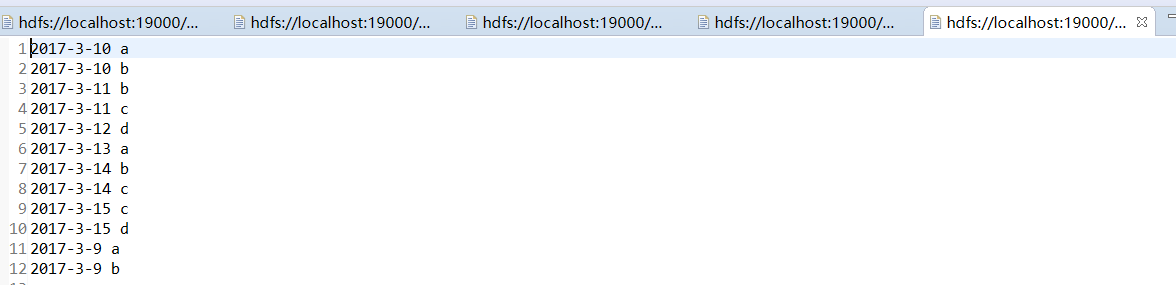
### 试验运行结果

运行过程如图所示：

运行过程

生成的data\_out文件夹如图所示：

输出文件夹

结果文件如图所示：

结果文件

# 实验总结

通过此次实验，我学会了很多知识，收获颇丰。我学会了搭建Hadoop、Eclipse环境，学会了在Eclipse上运行wordcount程序，学会了在Eclipse操作HDFS文件目录，熟悉了Hadoop、Eclipse编程流程及思想，学会了编写Hadoop数据去重程序。老师提供的实验指导书内容详尽，帮助我解决了许多棘手的问题。

# 附录

数据去重程序

public class Process {

public Process() {

}

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

// TODO Auto-generated method stub

Configuration conf = new Configuration();

String[] otherArgs = new GenericOptionsParser(conf,args).getRemainingArgs();

if(otherArgs.length != 2){

System.err.println("Usage WordCount <int> <out>");

System.exit(2);

}

Job job = Job.getInstance(conf, "Data Process");

job.setJarByClass(Process.class);

job.setMapperClass(Process.DedupMapper.class);

job.setCombinerClass(Process.DedupReducer.class);

job.setReducerClass(Process.DedupReducer.class);

job.setOutputKeyClass(Text.class);

//设置输出类型

job.setOutputValueClass(NullWritable.class);

FileInputFormat.addInputPath(job, new Path(otherArgs[0]));

FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

public static class DedupMapper extends Mapper<LongWritable, Text, Text, NullWritable> {

private Text line = new Text();//每行数据

@Override

protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, NullWritable>.Context context)throws IOException, InterruptedException {

line = value;

context.write(line,NullWritable.get());

}

}

public static class DedupReducer extends Reducer<Text, NullWritable, Text, NullWritable> {

//重写reduce()方法

@Override

protected void reduce(Text key, Iterable<NullWritable> values, Reducer<Text, NullWritable, Text, NullWritable>.Context context) throws IOException, InterruptedException {

context.write(key,NullWritable.get());

}

}

}