来源：https://blog.csdn.net/rtuujnncc/article/details/85012716

1.开发前准备

(1)安装好Hadoop集群

(2)准备winutils

winutils的下载地址：

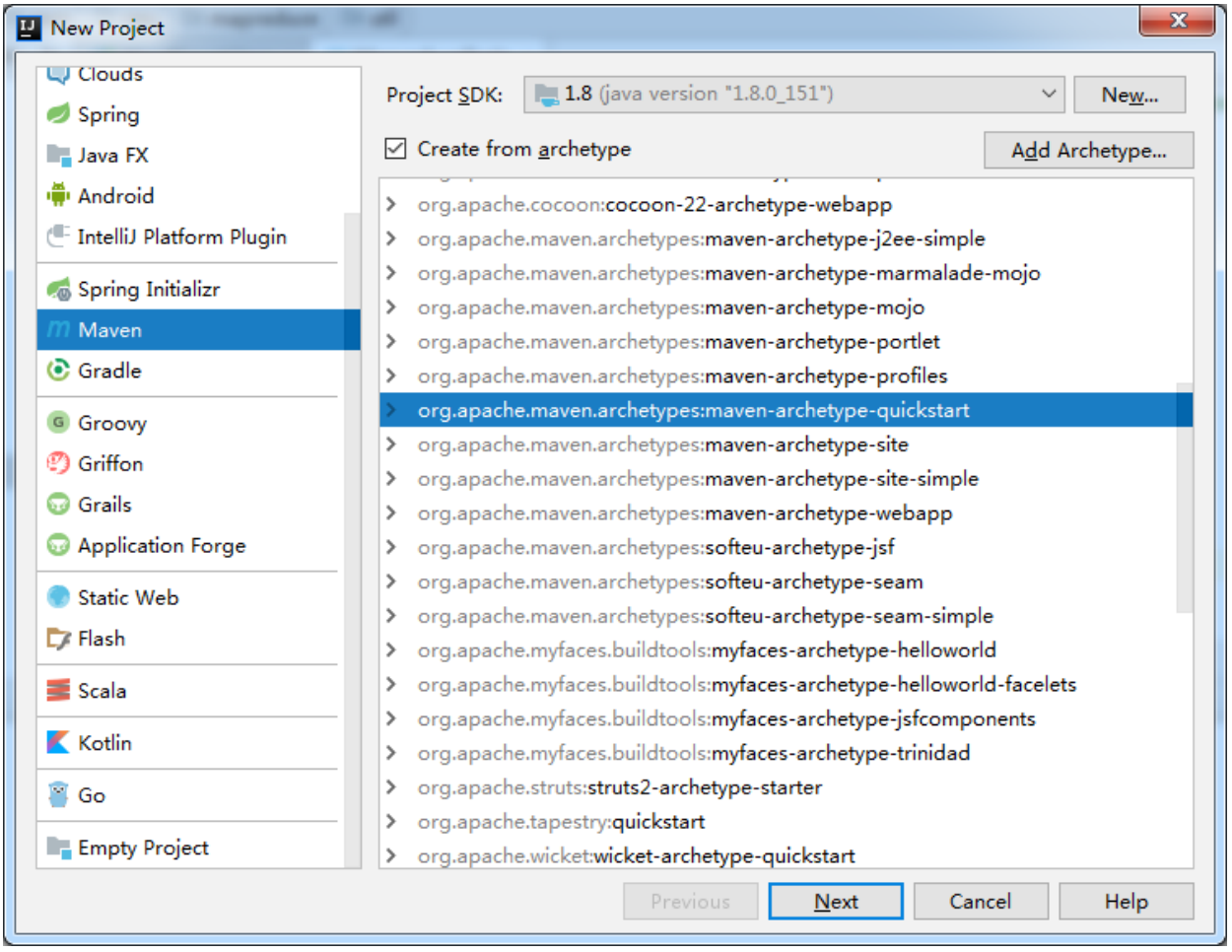
https://github.com/steveloughran/winutils/tree/master/hadoop-2.8.3/bin

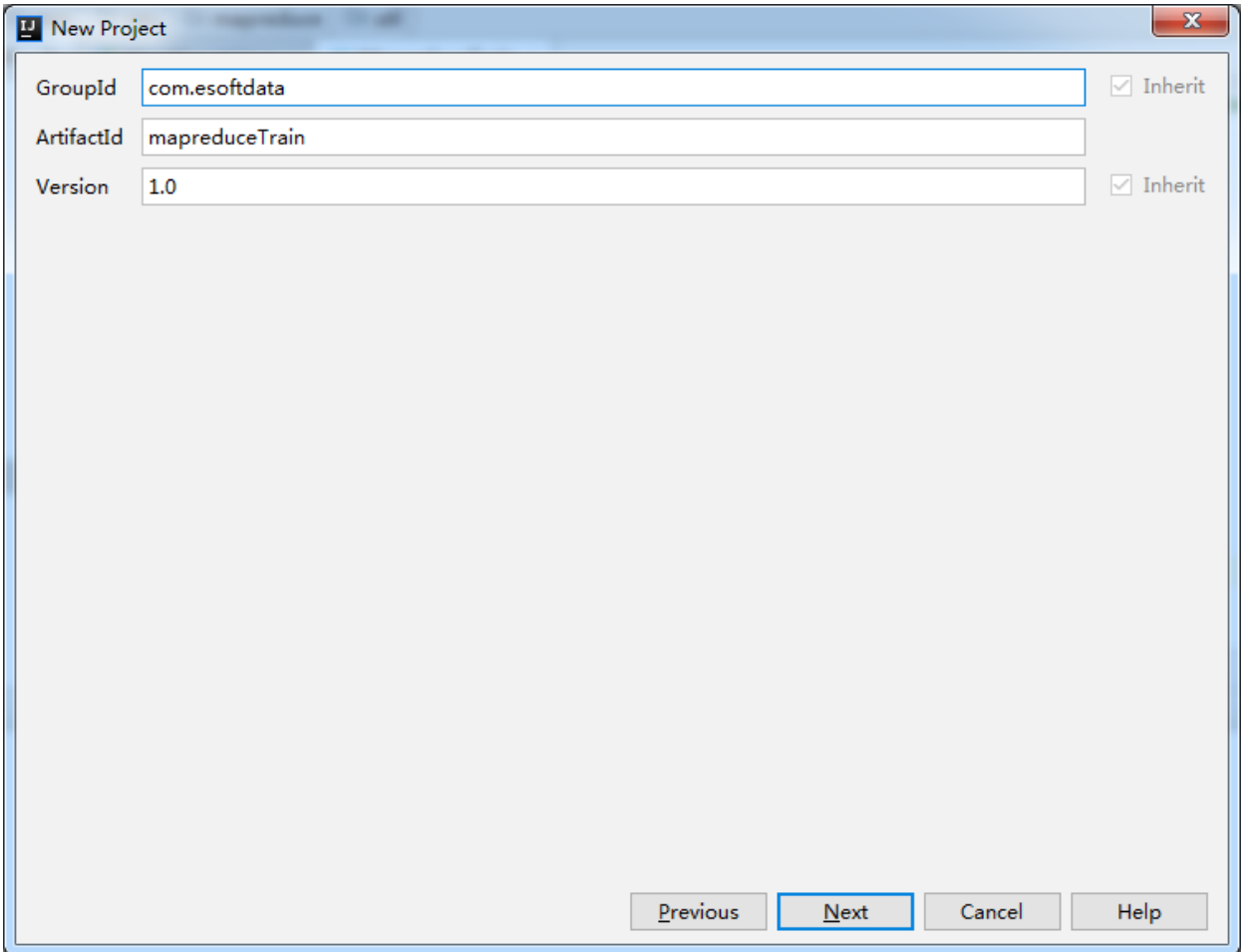
(3)将winutils解压后，将bin目录下的hadoop.dll复制到C:\Windows\System32

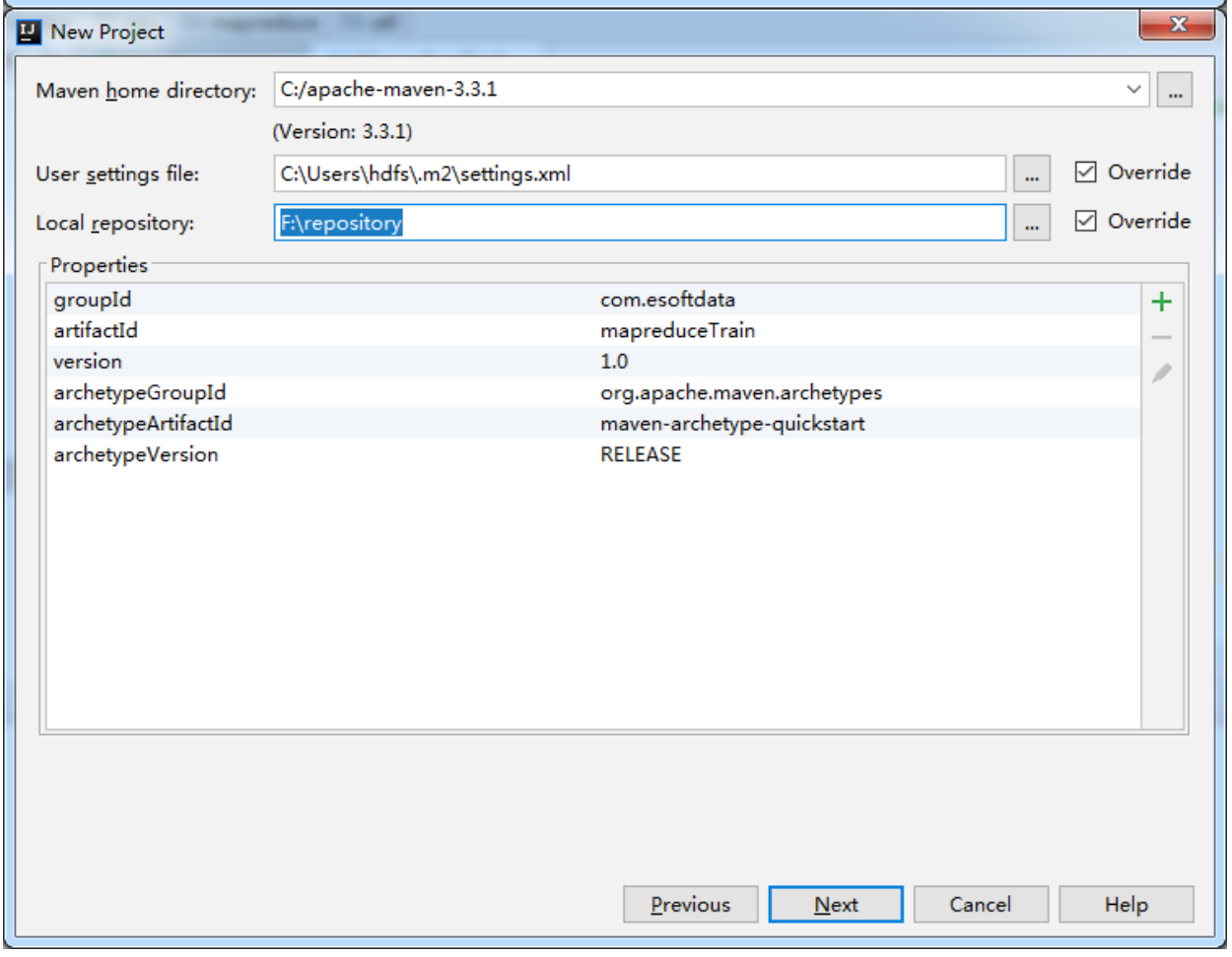
2.Idea创建Maven项目

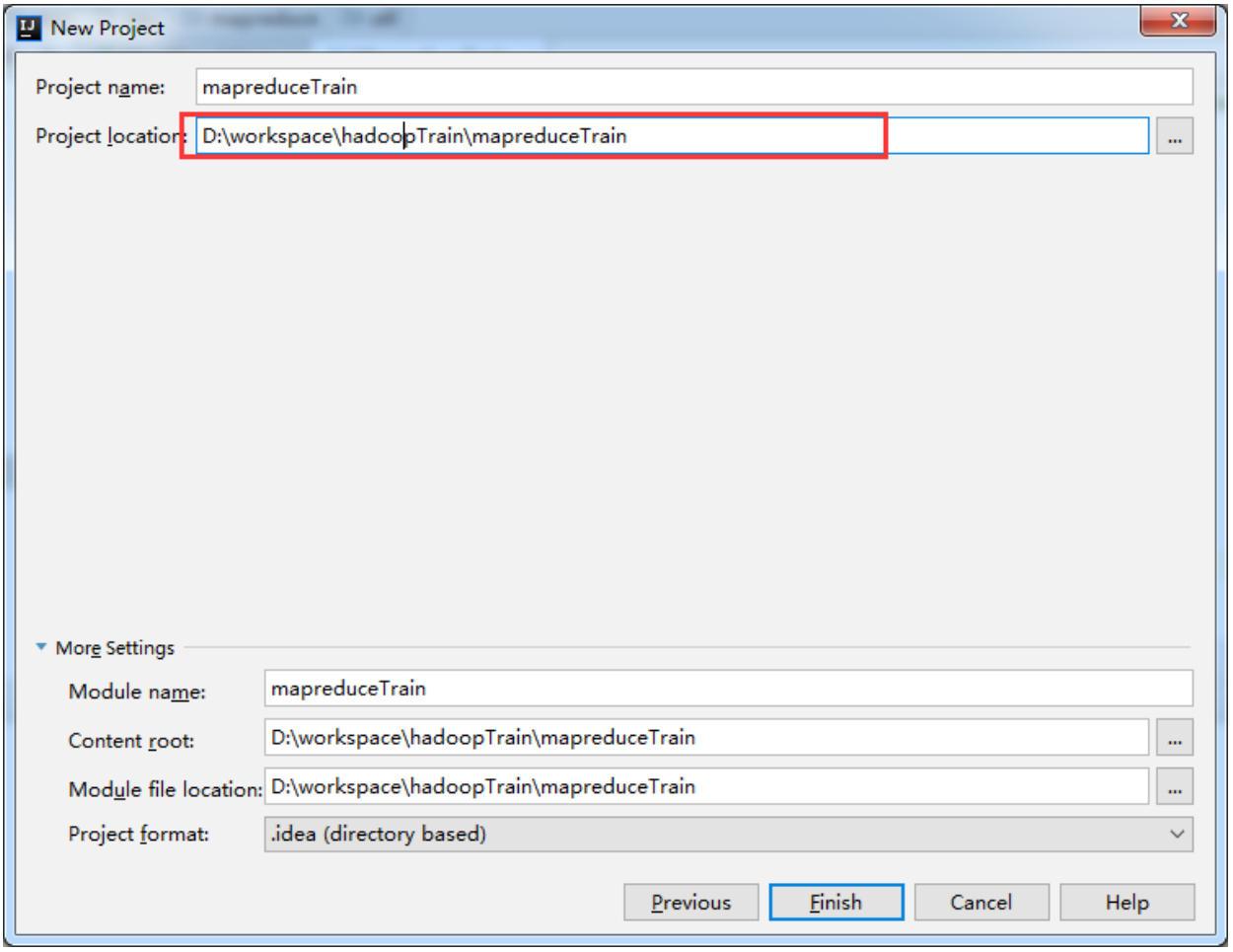
(1)新建maven工程，命名为mapreduceTrain

File-->new-->Project，选择maven。如下图所示。




New Project
GroupId
com.esoftdata
Inherit
ArtifactId
mapreduceTrain
Version
1.0
Inherit
Previous
Next
Cancel
Help


New Project
Maven home directory:
C:/apache-maven-3.3.1
(Version: 3.3.1)
User settings file:
C:\Users\hdfs\.m2\settings.xml
Override
Local repository:
F:\repository
Override
Properties
groupId
com.esoftdata
artifactId
mapreduceTrain
version
1.0
archetypeGroupId
org.apache.maven.archetypes
archetypeArtifactId
maven-archetype-quickstart
archetypeVersion
RELEASE
Previous
Next
Cancel
Help



(2)创建好maven项目后，在pom.xml文件中添加maven依赖。

```
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-hdfs</artifactId>
<version>${hadoop.version}</version>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-common</artifactId>
<version>${hadoop.version}</version>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-core</artifactId>
<version>${hadoop.version}</version>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-client</artifactId>
<version>${hadoop.version}</version>
</dependency>
<dependency>
```

```
<groupId>jdk.tools</groupId>
<artifactId>jdk.tools</artifactId>
<version>1.7</version>
</dependency>
```

(3)在resources下添加core-site.xml和hdfs-site.xml

(4)编写Mapreduce程序。

```
import java.io.IOException;
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.LongWritable;
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.mapreduce.lib.partition.HashPartitioner;
import org.apache.hadoop.util.GenericOptionsParser;

import com.neu.mapreduce.util.HDFSUtils;

public class WordCount {
//继承mapper接口，设置map的输入类型为<Object,Text>
//输出类型为<Text,IntWritable>
public static class Map extends Mapper<LongWritable,Text,Text,IntWritable>{
//one表示单词出现一次
private static IntWritable one = new IntWritable(1);
//word存储切下的单词
private Text word = new Text();
public void map(LongWritable key,Text value,Context context) throws IOExcep
  tion,InterruptedException{
//对输入的行切词
StringTokenizer st = new StringTokenizer(value.toString());
while(st.hasMoreTokens()){
word.set(st.nextToken());//切下的单词存入word
context.write(word, one);
}
}
}
//继承reducer接口，设置reduce的输入类型<Text,IntWritable>
```

```
//输出类型为<Text,IntWritable>
public static class Reduce extends Reducer<Text,IntWritable,Text,IntWritable>{
//result记录单词的频数
private static IntWritable result = new IntWritable();
public void reduce(Text key,Iterable<IntWritable> values,Context context) throws IOException,InterruptedException{
int sum = 0;
//对获取的<key,value-list>计算value的和
for(IntWritable val:values){
sum += val.get();
}
//将频数设置到result
result.set(sum);
//收集结果
context.write(key, result);
}
}
public static class WordPartitioner extends HashPartitioner<Text, IntWritable> {

@Override
public int getPartition(Text key, IntWritable value, int numReduceTasks) {
if(key.equals(new Text("Bootstrap"))) {
return 2;
}
return super.getPartition(key, value, numReduceTasks - 1);
}
}
/**
* @param args
*/
public static void main(String[] args) throws Exception{
Configuration conf = new Configuration();
//conf.set("mapreduce.output.fileoutputformat.compress", "true");
//conf.set("mapreduce.job.queuename ", "spark");
//检查运行命令
String[] otherArgs = new GenericOptionsParser(conf,args).getRemainingArgs();
if(otherArgs.length != 2){
System.err.println("Usage WordCount <int> <out>");
System.exit(2);
}
HDFSUtils hdfs = new HDFSUtils(conf);
```

```
hdfs.deleteDir(args[1]);
//配置作业名
Job job = Job.getInstance(conf, "word count");
//配置作业各个类
job.setJarByClass(WordCount.class);
job.setMapperClass(Map.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(IntWritable.class);
// job.setCombinerClass(Reduce.class);
job.setReducerClass(Reduce.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
job.setNumReduceTasks(3);
// job.setPartitionerClass(WordPartitioner.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
// FileInputFormat.setMaxInputSplitSize(job, 10);
// FileInputFormat.setMinInputSplitSize(job, 10);
FileOutputFormat.setOutputPath(job, new Path(args[1]));
// FileOutputFormat.setCompressOutput(job, true);
// FileOutputFormat.setOutputCompressorClass(job, BZip2Codec.class);
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

(5)执行Mapreduce程序。