Idea 下用Maven[搭建](http://www.cnblogs.com/fillPv/p/5882364.html)Spark开发环境

## 安装jdk

jdk版本1.8以上，设置好环境变量，安装过程，略。

## 安装Maven

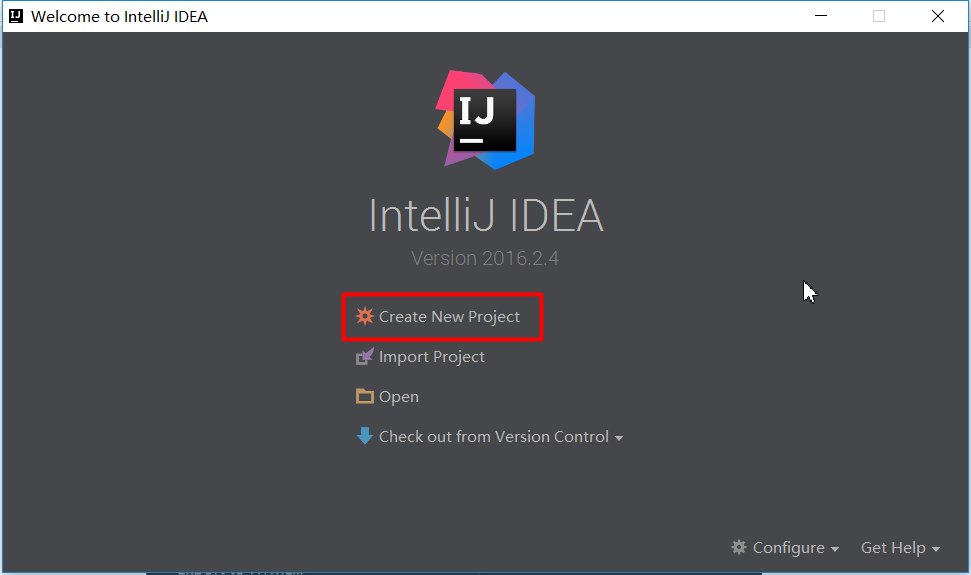
Maven版本3.5.0，安装过程，略。

## 安装Idea

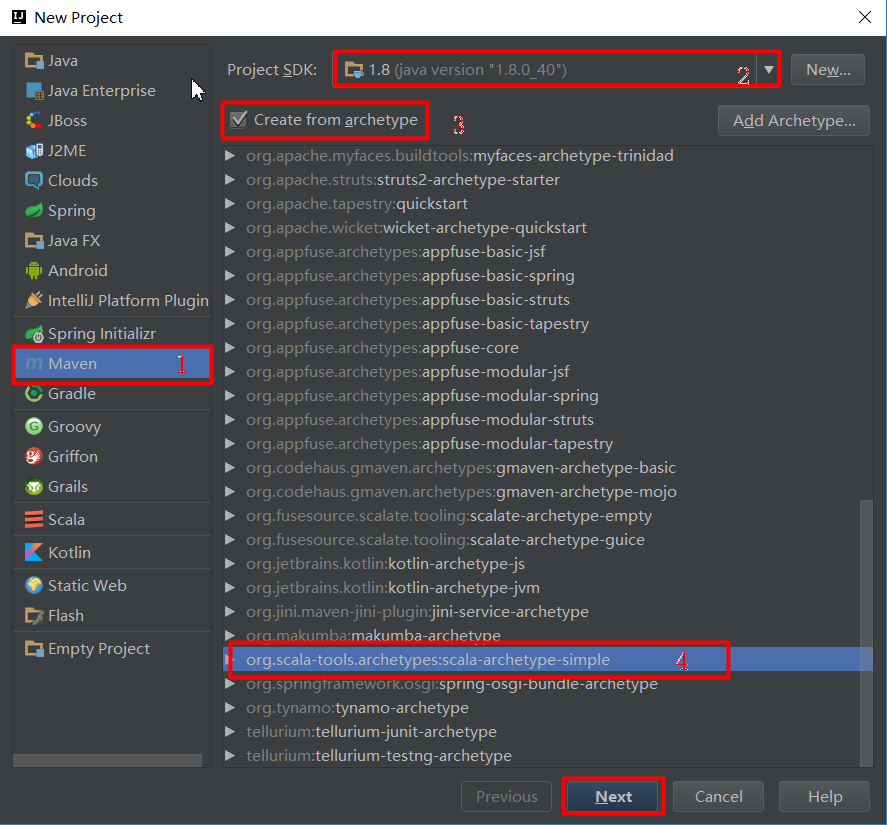
安装过程，略。

## 创建Spark项目

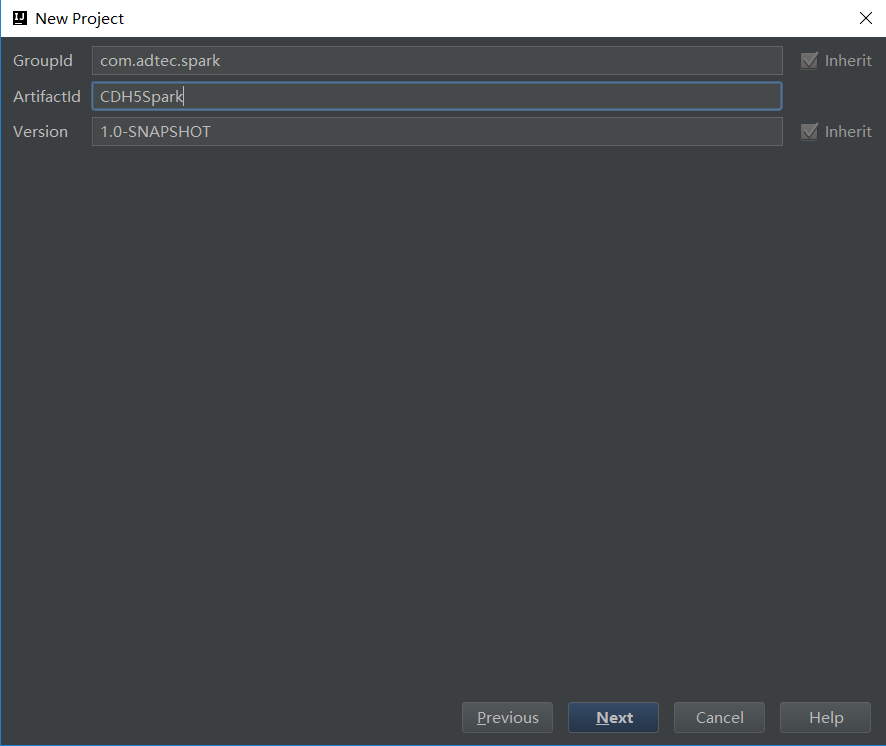
### **新建一个Spark项目**



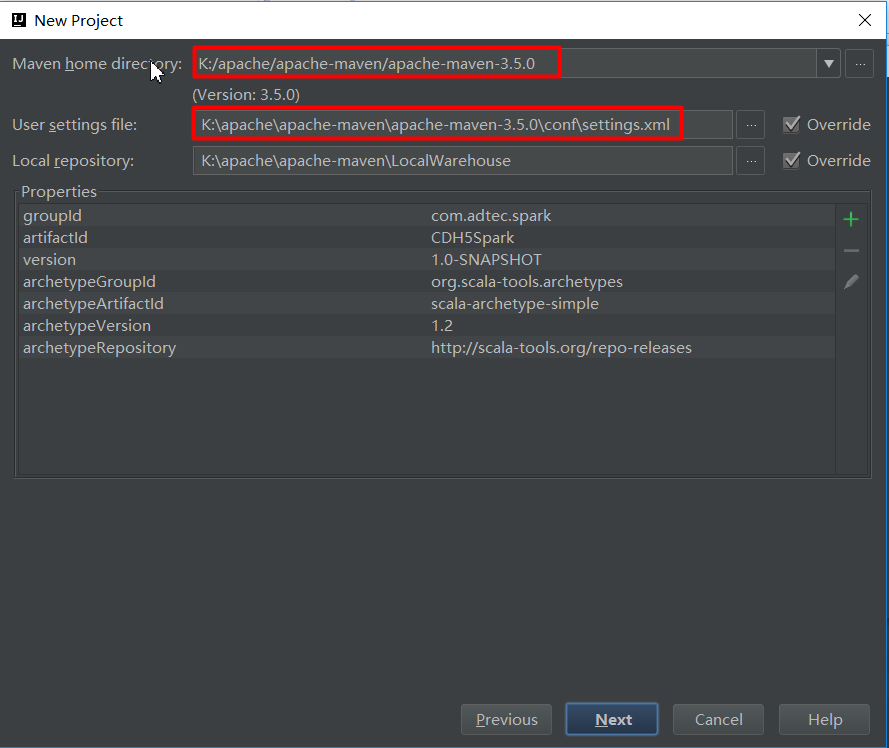
### **选择Maven，从模板创建项目**



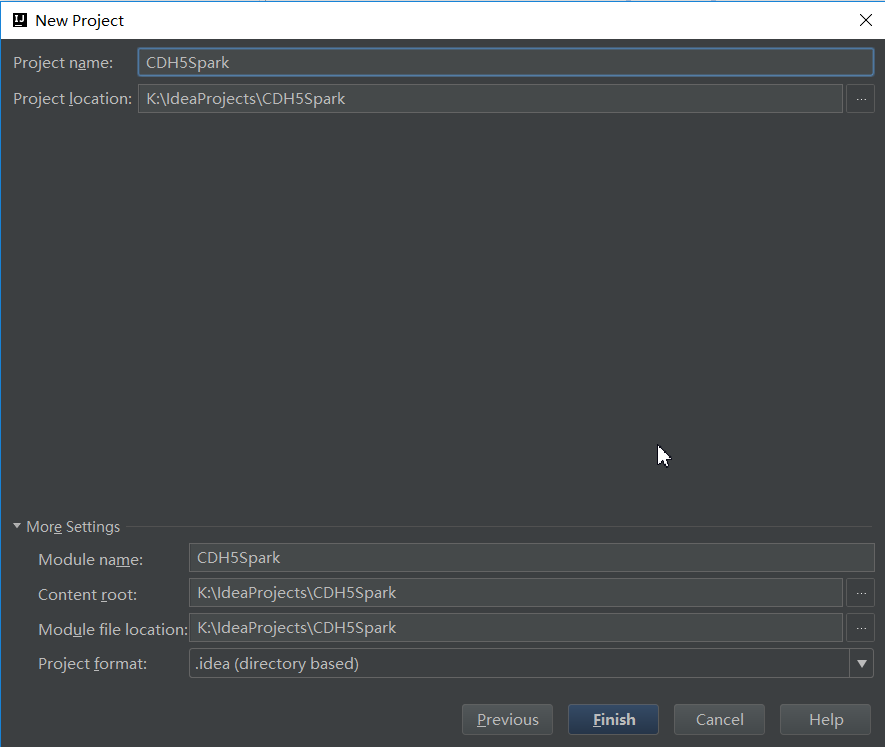
### **填写项目GroupId等，**



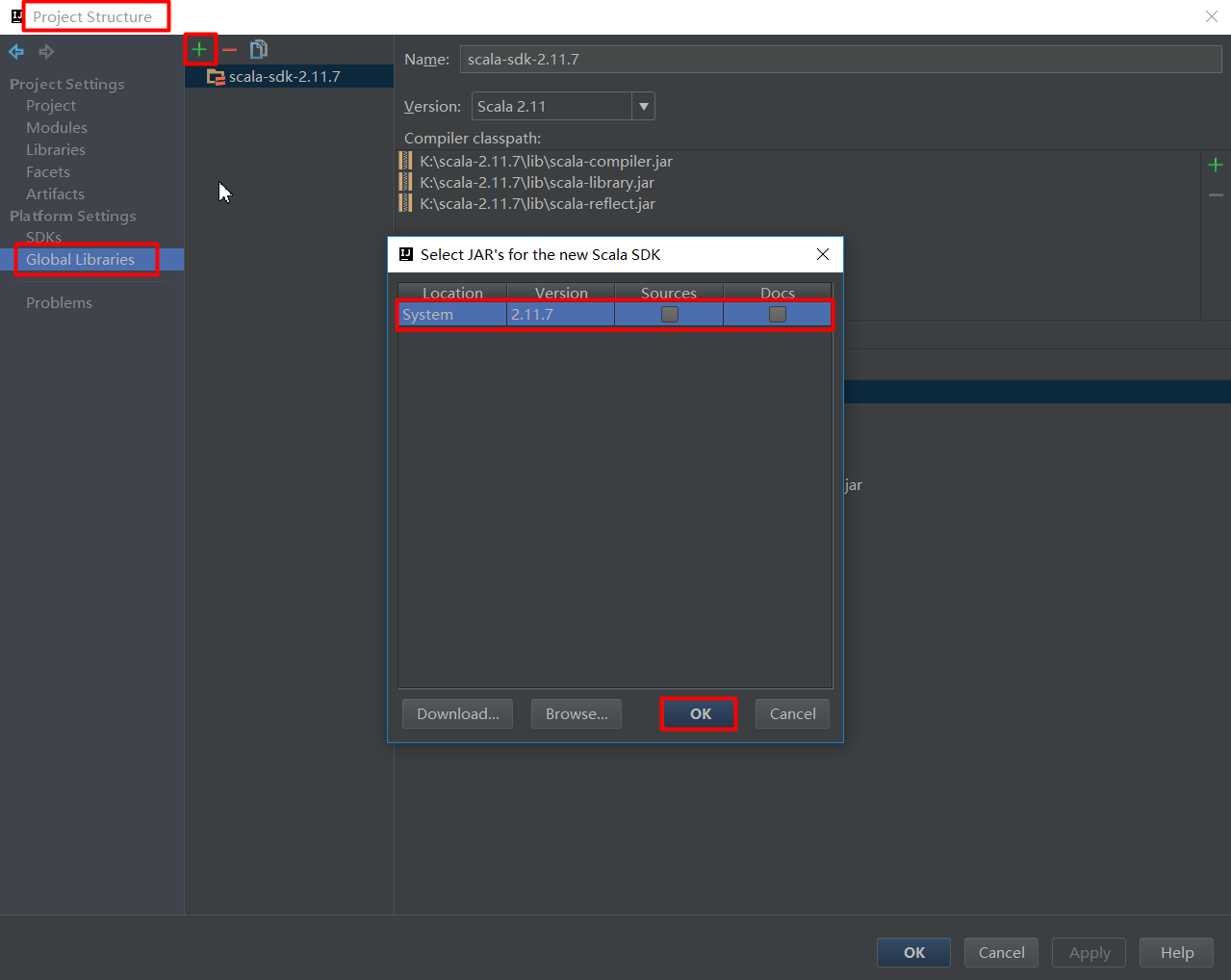
### 选择本地安装的Maven和Maven配置文件。



### 项目名称



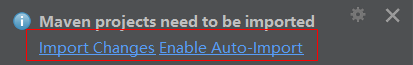
### 添加Scala开发库



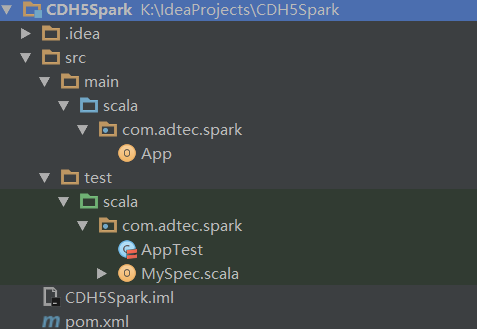
修改Scala.version：

<scala.version>2.11.7</scala.version>

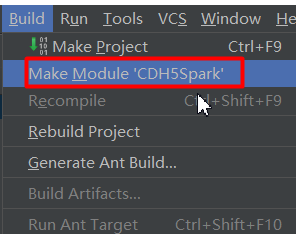
### 自动更新Maven pom文件



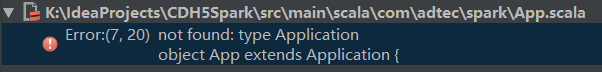
### 创建完毕，查看新项目结构：



### 编译项目

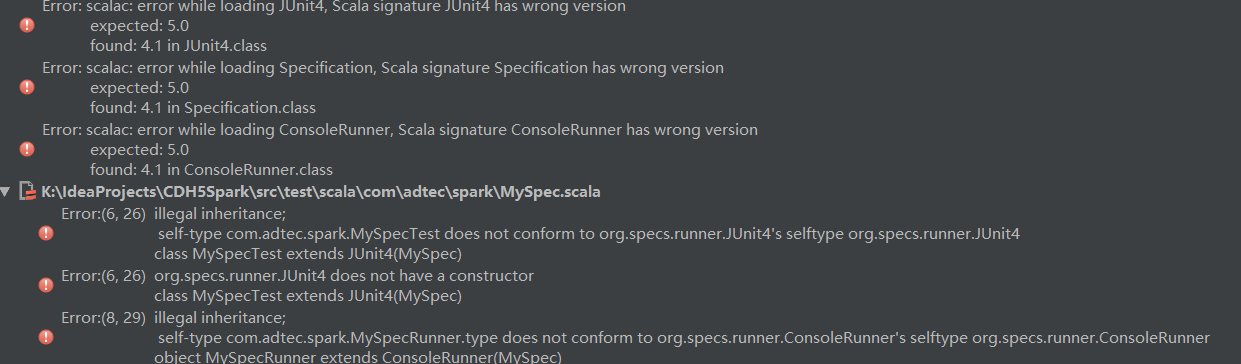


错误一：



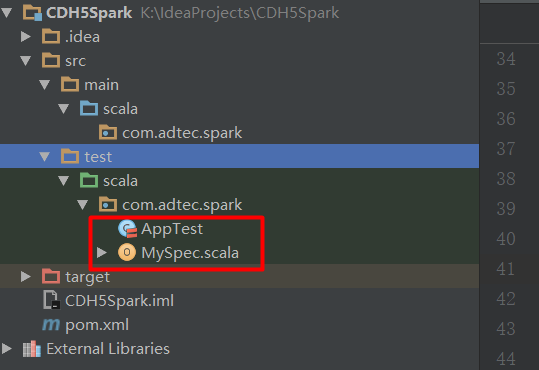
解决方法：删除App.scala

错误二：



原因：如果出现这种错误，这个错误是由于Junit版本造成的，可以删掉Test，和pom.xml文件中Junit的相关依赖

解决方法：删掉如下Scala类：



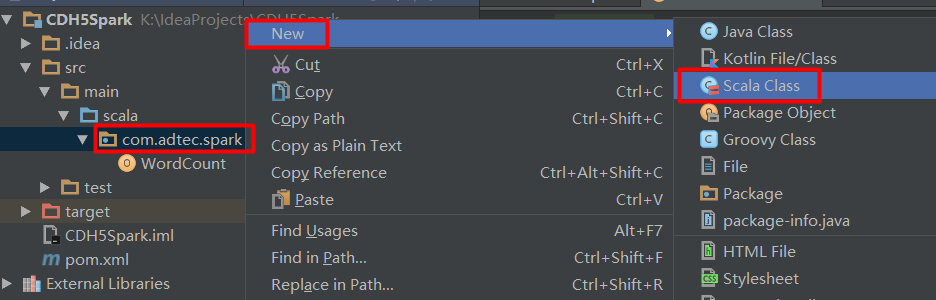
和pom.xml文件中的Junit依赖：

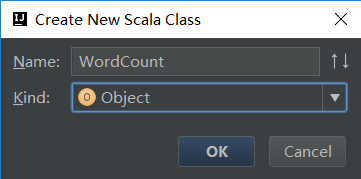
<dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.4</version>  
 <scope>test</scope>  
</dependency>  
<dependency>  
 <groupId>org.specs</groupId>  
 <artifactId>specs</artifactId>  
 <version>1.2.5</version>  
 <scope>test</scope>  
</dependency>

### 添加CDH5 Spark依赖包

参考《5\_Maven Artifacts for CDH.txt》、《pom.xml》

### 新建一个Scala Object。





代码为：

package com.adtec.spark

import org.apache.spark.{SparkConf, SparkContext}

import org.apache.spark.SparkContext.\_

/\*\*

\* Created by 陈少钦 on 2017/9/16.

\*/

object WordCount {

def main(args: Array[String]) {

/\*\*

\* 第1步；创建Spark的配置对象SparkConf，设置Spark程序运行时的配置信息

\* 例如 setAppName用来设置应用程序的名称，在程序运行的监控界面可以看到该名称，

\* setMaster设置程序运行在本地还是运行在集群中，运行在本地可是使用local参数，也可以使用local[K]/local[\*],

\* 可以去spark官网查看它们不同的意义。 如果要运行在集群中，以Standalone模式运行的话，需要使用spark://HOST:PORT

\* 的形式指定master的IP和端口号，默认是7077

\*/

val conf = new SparkConf().setAppName("WordCount").setMaster("local")

//val conf = new SparkConf().setAppName("WordCount").setMaster("spark://master:7077") // 运行在集群中

/\*\*

\* 第2步：创建SparkContext 对象

\* SparkContext是Spark程序所有功能的唯一入口

\* SparkContext核心作用： 初始化Spark应用程序运行所需要的核心组件，包括DAGScheduler、TaskScheduler、SchedulerBackend

\* 同时还会负责Spark程序往Master注册程序

\*

\* 通过传入SparkConf实例来定制Spark运行的具体参数和配置信息

\*/

val sc = new SparkContext(conf)

/\*\*

\* 第3步： 根据具体的数据来源(HDFS、 HBase、Local FS、DB、 S3等)通过SparkContext来创建RDD

\* RDD 的创建基本有三种方式： 根据外部的数据来源(例如HDFS)、根据Scala集合使用SparkContext的parallelize方法、

\* 由其他的RDD操作产生

\* 数据会被RDD划分成为一系列的Partitions，分配到每个Partition的数据属于一个Task的处理范畴

\*/

// val lines = sc.textFile("D:/resources/README.md") // 读取本地文件

// val lines = sc.textFile("/library/wordcount/input") // 读取HDFS文件，并切分成不同的Partition

val lines = sc.textFile("hdfs://192.168.88.201/tmp/genji/abc.txt") // 或者明确指明是从HDFS上获取数据

/\*\*

\* 第4步： 对初始的RDD进行Transformation级别的处理，例如 map、filter等高阶函数来进行具体的数据计算

\*/

val words = lines.flatMap(\_.split(" ")).filter(\_ != " ") // 拆分单词，并过滤掉空格，当然还可以继续进行过滤，如去掉标点符号

val pairs = words.map((\_, 1)) // 在单词拆分的基础上对每个单词实例计数为1, 也就是 word => (word, 1)

val wordscount = pairs.reduceByKey(\_ + \_) // 在每个单词实例计数为1的基础之上统计每个单词在文件中出现的总次数, 即key相同的value相加

//wordscount.collect.foreach(println) // 打印结果，使用collect会将集群中的数据收集到当前运行drive的机器上，需要保证单台机器能放得下所有数据

wordscount.saveAsTextFile("hdfs://192.168.88.201/tmp/genji/result")

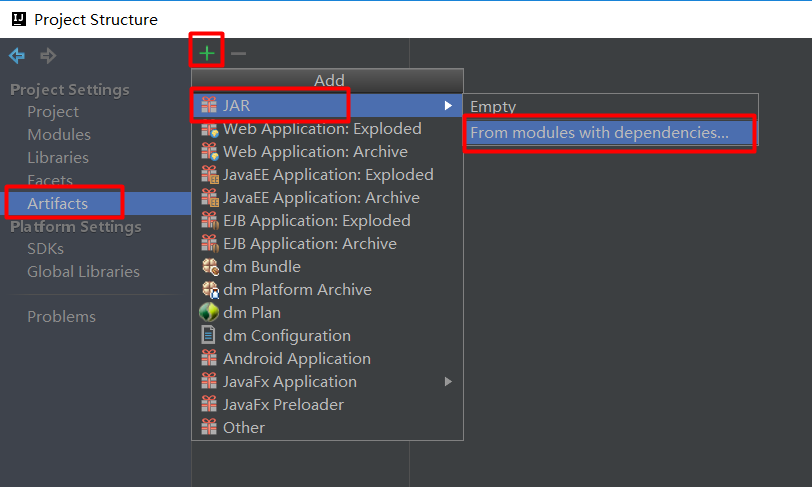
sc.stop() // 释放资源

}

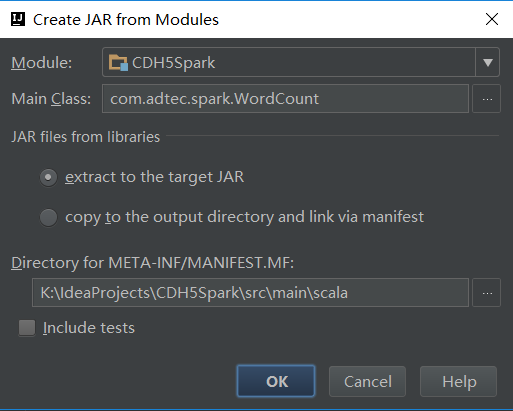
}

### 打jar包

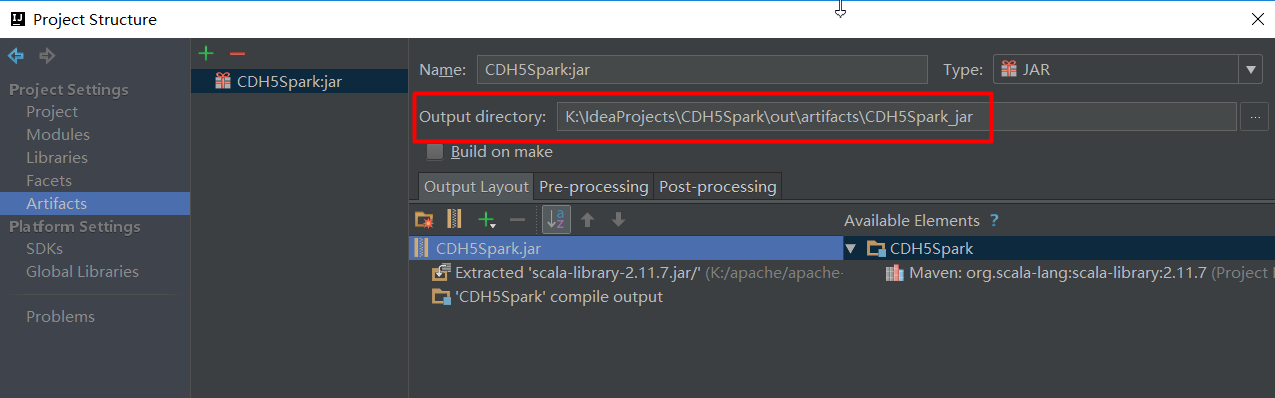
File -》Project Structure



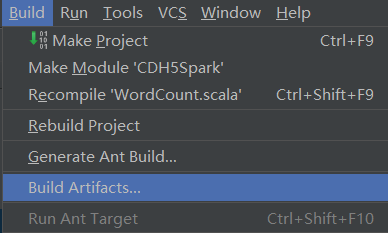
点击ok

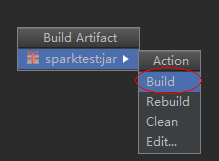


设置jar包输出目录：



build Artifact





### 运行：

把测试文件放到HDFS的/tmp/genji/abc.txt 目录下，提交：

[root@master genji]# spark-submit --class com.adtec.spark.WordCount --executor-memory 500m --total-executor-cores 2 /tmp/genji/CDH5Spark.jar

…

17/08/07 09:02:27 INFO storage.BlockManagerMasterActor: Stopping BlockManagerMaster

17/08/07 09:02:27 INFO storage.BlockManagerMaster: BlockManagerMaster stopped

17/08/07 09:02:27 INFO spark.SparkContext: Successfully stopped SparkContext

17/08/07 09:02:27 INFO remote.RemoteActorRefProvider$RemotingTerminator: Shutting down remote daemon.

17/08/07 09:02:27 INFO remote.RemoteActorRefProvider$RemotingTerminator: Remote daemon shut down; proceeding with flushing remote transports.

17/08/07 09:02:27 INFO Remoting: Remoting shut down

### 查看结果：

