# 基于Window和 IntelliJ构建Spark本地开发调试环境

## 引言

用于在window下调试spark代码。

环境：window7 + java1.8 + scala10.6

安装：IntelliJ，<https://www.jetbrains.com/idea/download/#section=windows>

选window版本，可以选择是社区版还是付费版，付费版只能用一个月，之后每次只能用半小时，建议社区版。

下载：spark，<http://spark.apache.org/downloads.html>

Spark1.6.0，下载了两个版本一个hadoop-2.4，一个source code。

## 搭建环境

Step1.建立文件，并建立output 的文件夹，通常是project父目录下建立一个out文件夹，作为本项目的输出文件夹。

Step2.建立project，设定jdk和scala，以及output path



备注：这里output path如果不设定，可能后面导致。Cannot start compilation: the output path is not specified for module的错误。

Step3.加入scala到 library



Step4.将spark的jar加入到library



在spark也可以放入libararies里，这里导入spark-assembly的jar即可。另外在右边的红圈的“+”可以导入soure code，以便后面可以查看spark的源码。此处source code导入的文件夹为引言中下载的source code并解压的文件夹。

建议：可以在project文件夹下建立一个lib文件夹，将需要的jar等文件放进去，将解压的source code放进去。

Step5.建立module





Step6.建立Package和Class或Object





Step7.测试编译环境

在Object中输入代码

package testCore

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

import org.apache.spark.sql.SQLContext

import org.apache.spark.streaming.{Seconds, StreamingContext}

/\*\*

\* Created by DataShoe on 2018/1/5.

\*/

object testCoreCompile {

def main(args: Array[String]): Unit = {

// Test compile.

println("Hello world!")

// Test SparkContext

val conf = new SparkConf().setMaster("local").setAppName("DataShoe-Learning")

val sc = new SparkContext(conf)

val sqlc = new SQLContext(sc)

val rdd = sc.parallelize((0 until 100).toList)

val rdd\_sum = rdd.reduce(\_ + \_)

println(rdd\_sum)

}

}

查看，编译是否成功

备注1：如果测试的object能够编译成功，但测试SparkContext时不成功，报错



则是scala和spark的版本不匹配，该版本spark可能要求2.10版本的scala。

备注2：如果出现找不到spark对应类的情况



请在module中加入libray



## 建立一个自动设定SparkContext的抽象类

## 背景

如果每次都设定master和appName太麻烦，可以建立一个基本的module，里面建立一个抽象类myApp，使得其他module可以引用该，不用每次都设定SparkContext。

## 步骤

Step1.建立新的module，名称为base



Step2.将spark加入到该module

将需要的spark的lib add到该module中来,方法同前面一样。Libraries/global libray => add to modules

Step3.建立对应package和测试package，并在其中建立抽象类的和对应的测试object

1）建立两个package：com.self.myApp和com.self.testMyApp，分别用于建立抽象类和测试抽象类

2）在新建的module base中建立com.self.myApp的Package，在其中建立类myAPP

package com.self.myApp

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

import org.apache.spark.sql.SQLContext

import org.apache.spark.streaming.{Seconds, StreamingContext}

/\*\*

\* Created by DataShoe on 2018/1/5.

\*/

abstract class myAPP {

lazy val conf = new SparkConf().setMaster("local").setAppName("DataShoe-Learning")

lazy val sc = new SparkContext(conf)

lazy val sqlc = new SQLContext(sc)

lazy val smc : StreamingContext = new StreamingContext(conf, Seconds(1))

def run(): Unit

def main(args: Array[String]): Unit = {

run()

}

}

3）在另一个Package，com.self.testMyApp中建立Object——testMyAPP测试该类是否成功

package com.self.testMyApp

import com.self.myApp.myAPP

/\*\*

\* Created by DataShoe on 2018/1/5.

\*/

object testMyAPP extends myAPP{

override def run(): Unit = {

// Test compile.

println("Hello World")

// Test SparkContext.

val rdd = sc.parallelize((0 until 100).toList)

val rdd\_sum = rdd.reduce(\_ + \_)

println(rdd\_sum)

}

}

Step4.将base加入core的依赖中。



Step5.在core中建立一个object测试是否成功

在core中建立一个常用的package：com.self.core，在其中建立子package，testBaseApp，此时目录并在一块，不用担心，IntelliJ会自动合并只有单个子目录或子文件的目录，下次如果想继续在com.self.core下建子目录，比如newPackage，直接在core下面新建一个com.self.core. newPackage即可。

将代码贴进来



发现没有import myApp，Alt+Enter即可。

编译之，成功。