scala简介

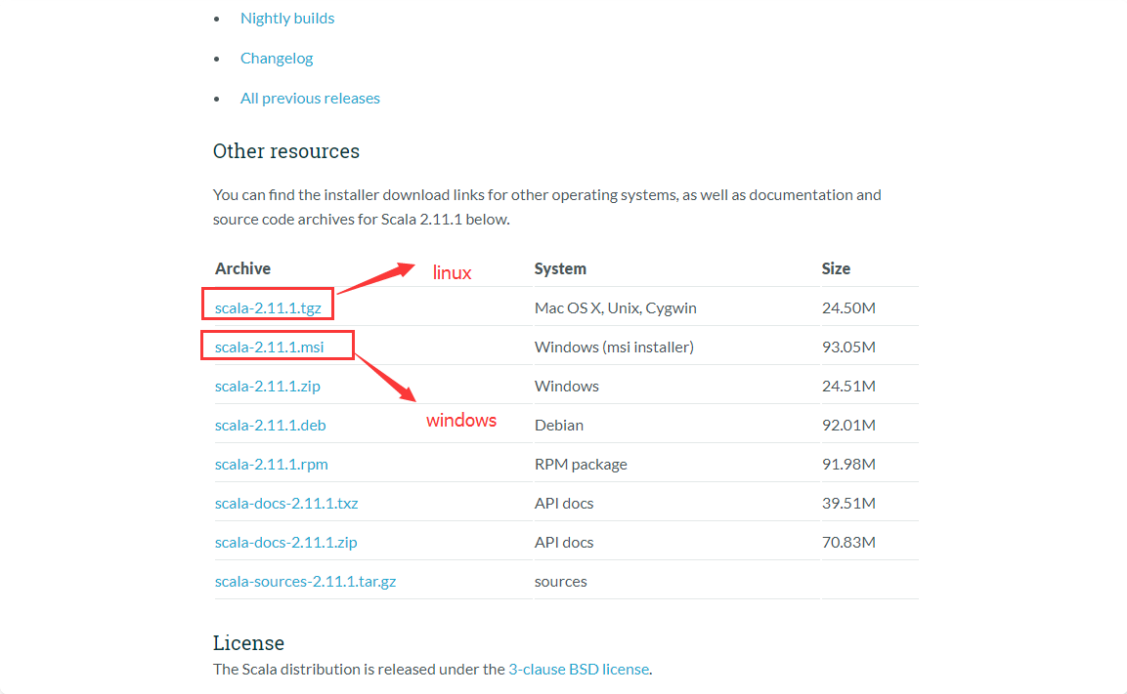
scala spark

scala介绍

> Scala是一门多范式的编程语言，一种类似java的编程语言，设计初衷是实现可伸缩的语言、并集成面向对象编程和函数式编程的各种特性。

scala 安装

1. 下载 https://www.scala-lang.org/download/2.11.1.html、

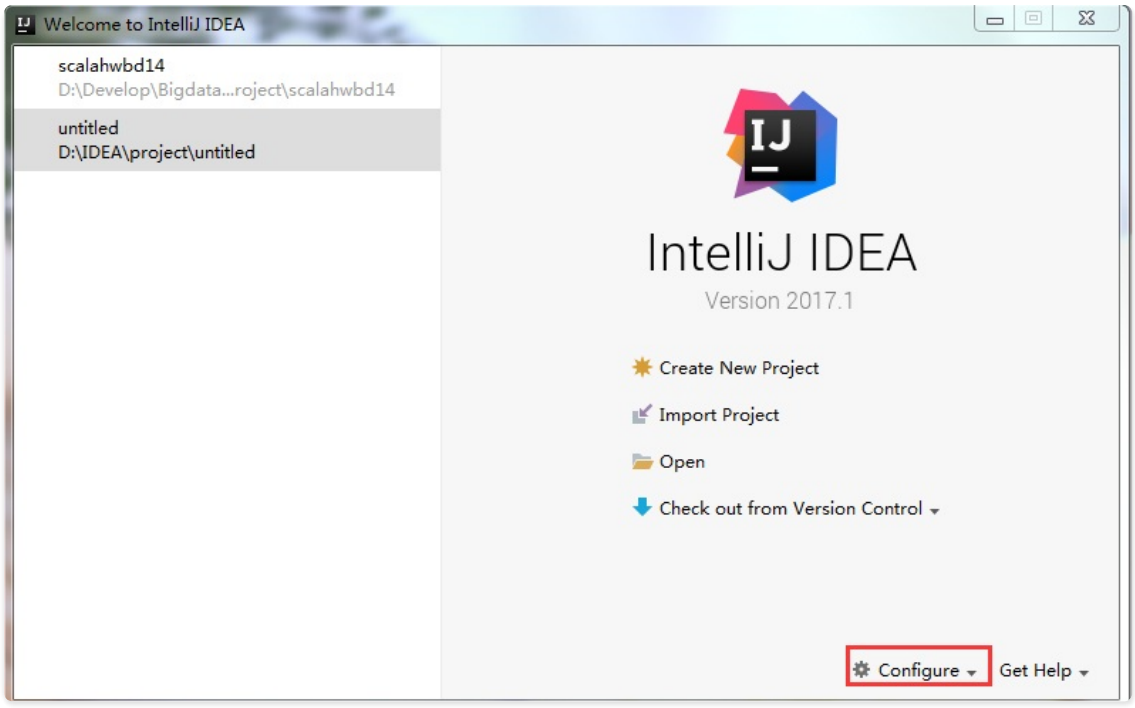


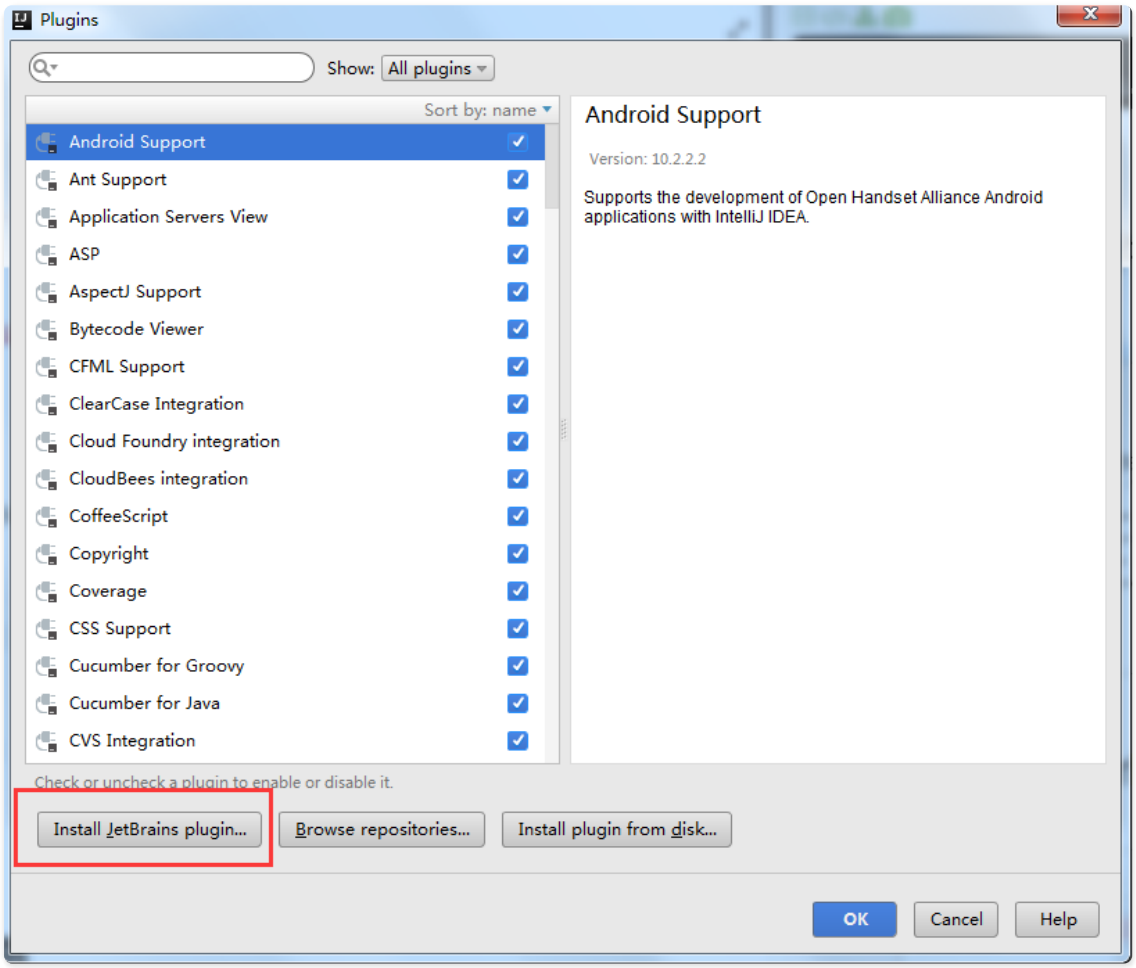
2. 安装，直接下一步就可以了
3. 配置环境变量

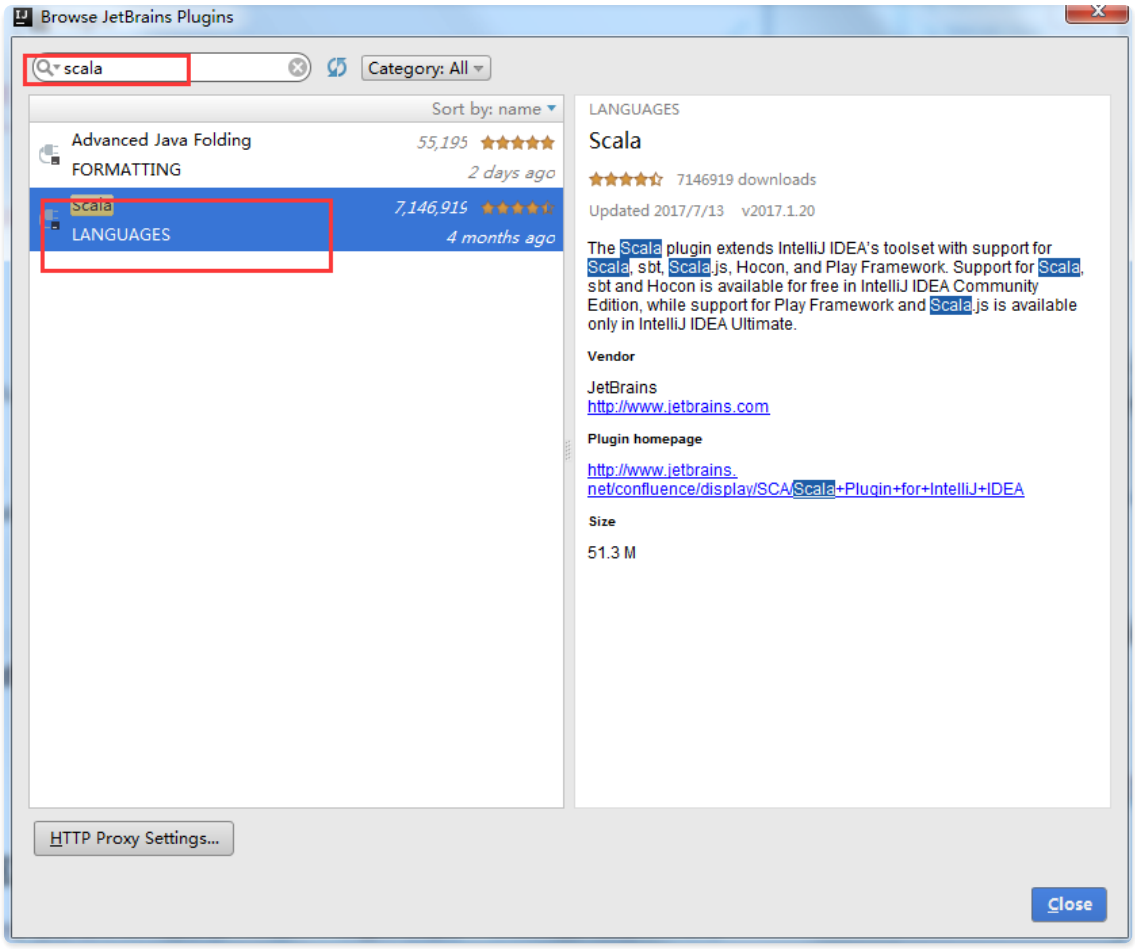
```
SCALA_HOME=scala安装路径
path= ;%SCALA_HOME%\bin;%SCALA_HOME%\jre\bin
```

idea安装scala插件

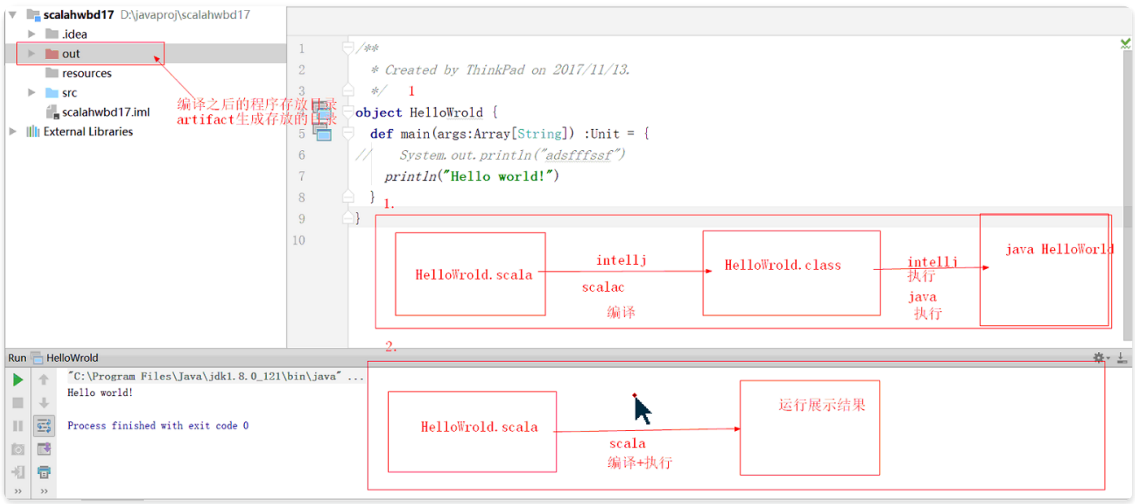
依次点击[configure/plugins/Install JetBrains plugin]


Welcome to IntelliJ IDEA
scalahwbd14
D:\Develop\Bigdata...roject\scalahwbd14
untitled
D:\IDEA\project\untitled
IntelliJ IDEA
Version 2017.1
Create New Project
Import Project
Open
Check out from Version Control
Configure
Get Help


Plugins
Show: All plugins
Sort by: name
Android Support
Ant Support
Application Servers View
ASP
AspectJ Support
Bytecode Viewer
CFML Support
ClearCase Integration
Cloud Foundry integration
CloudBees integration
CoffeeScript
Copyright
Coverage
CSS Support
Cucumber for Groovy
Cucumber for Java
CVS Integration
Android Support
Version: 10.2.2.2
Supports the development of Open Handset Alliance Android applications with IntelliJ IDEA.
Check or uncheck a plugin to enable or disable it.
Install JetBrains plugin...
Browse repositories...
Install plugin from disk...
OK
Cancel
Help



scala之Hello World



强类型与弱类型的区别

强类型语言：定义对象或变量时们需要指定其归属类型，一旦一个变量类型确定，它所归属的类型不可改变

弱类型语言：定义变量时不用指定变量类型，在程序运行中，可以改变变量的归属类型

scala基本语法

1. scala变量定义：
   var str = "abcd"
   这种写法不是没有指定str的类型，而是没有显式的指定str的类型，它隐式的从变量值中自动判断

显式写法：

var str:String = "abcd"

2. 声明变量有两种修饰符
   var： 变量可被重新赋值
   val(变量)不可被重新赋值
   在编程中，能使用val的地方不使用var

3. 基础数据类型

Value type	Range
Byte	8-bit signed two's complement integer (-2^7 to $2^7 - 1$, inclusive)
Short	16-bit signed two's complement integer (-2^{15} to $2^{15} - 1$, inclusive)
Int	32-bit signed two's complement integer (-2^{31} to $2^{31} - 1$, inclusive)
Long	64-bit signed two's complement integer (-2^{63} to $2^{63} - 1$, inclusive)
Char	16-bit unsigned Unicode character (0 to $2^{16} - 1$, inclusive)
String	a sequence of Chars
Float	32-bit IEEE 754 single-precision float
Double	64-bit IEEE 754 double-precision float
Boolean	true or false

java基础数据类型，它对应的变量，不是对象，不能通过“.”运算符来访问对象的方法。
scala对应的基础数据类型，对应的变量可使用"."操作来访问对象的方法

```
val a = 123
a.toDouble
a.toLong
```

scala字面量

```
val intval = 23
val doubleval = 23.0
val floatval = 23L
val str =“aaa”
```

4. string 类型的字面量

```
val s1 = "abcd"
 val s2 = "ab\"cd"
 val s3 = """ab".e0.*ed"""
```

字符串模板嵌套

```
println(s"name:$name,age:$age")
println(s"name:$name",age:${age}aa")
println(s"""name:$name
 age:$age
 over
 """
)
```

5. 基础数据类型之间的转换方法

对象.to类型

```
123.toDouble
"123".toInt
123.33.toString
```

6. scala中的运算符
   scala中运算符不是语法，而是函数(对象)
   `a + b ===> a.+(b)` 前者是后者的简写形式，当一个对象通过点调用其方法的时候，如果该方法只有一个参数，name点号可以省略，对象、方法、参数用空格隔开即可

==运算符(java)

```
enter code here
```

== 在scala中方法，这个方法等同于equal方法

```
val a = new String("abc")
val b = new String("abc")
a == b
```

7. 标识符 符合java规范
   类的标识符驼峰式命名首字母大写
   变量 方法标识符，驼峰式命名，首字母小写
   包表示符，全小写，层级使用点分隔

> 注意：val 在scala中虽然定义的是常量，但是一般使用变量的规则来命名标识符
> 运算符中没有++、--

8. 注释
   和java中一样

9. 语句块

java中的语句块全部都是过程，没有返回值，只有方法语句块中return才有返回值
scala中大部分的语句块都是有返回值的，而且不需要return
java中语句块的作用主要来划分作用域
scala中的语句块除了划分作用域之外还可以带返回值

```
val str1 = "111"
val str2 = {
 val str3 = s"${str1}defg"
 str3
}
println(str3) // 访问不到
scala中语句块的最后一句，就是该语句的返回值
```

scala控制结构

条件表达式 if...else...

> scala的if/else语法结构和java或者c++是一样的。

```
/**
  * 接收分数，判断给出分数的优良中差
  */
object IfElse {

  def main(args: Array[String]): Unit = {
    val score = args(0).toInt
    // 类似java语法
    if(score >= 90){
      println("优秀")
    }else if(score >= 80){
      println("良好")
    }else if(score >= 60){
      println("中等")
    }else{
      println("差")
    }

    // java三木运算 String result = score > 60 ? "优秀" : "差"
    // scala没有三木运算符 if(score > 60) "优秀"else "差"
    //scala的用法
    val result = if(score >= 90)
      "优秀"
    else if(score >= 80)
      "良好"
    else if(score >= 60)
      "中等"
    else
      "差"
    println(result)
  }
}
```

while循环

语句块中是没有返回值的

```
/**
  * while 循环
  */
object WhileTest {

  def main(args: Array[String]): Unit = {
    var times = args(0).toInt

    while (times > 0){
      println(s"第${times}此打印")
```

```
      times = times - 1
    }

    var times2 = args(0).toInt
    do{
      println(s"doWhile第${times2}此打印")
      times2 -= 1
    }while(times2 > 0)

    // 构建死循环
    while (true){
      // 循环体,轮巡
    }
  }
}
```

for循环

for也是scala中少数没有返回值的语句块之一
但是scala提供了一种方式(yield)让其具有返回值的能力

```
for(int i = 0; i < 10; i++){
 // 循环体
}
for(String i : sList){}
```

scala中的for循环更像foreach

```
for(i <- list){}
```

通过守卫循环遍历

```
package top.xiesen.bd14

/**
  * for循环
  */
object ForTest {

  def main(args: Array[String]): Unit = {
    val times = args(0).toInt

    for (i <- 1 to times) {
      println(s"1print: $i")
    }

    // 循环守卫通知
    for (i <- 1 to times if i % 2 == 0) println(s"1print$i")
    for (i <- 1 to times if i % 2 == 0 && i > 5) println(s"2println$i")

    // 多重循环(嵌套循环)
      for(i <- 1 to times){
        for(j <- 1 to times){
          println(s"i:$i,j:$j")
        }
      }


    for (i <- 1 to times; j <- 1 to times) {
```

```
      println(s"i:$i,j:$j")
    }

    // 嵌套限定条件
    for (i <- 1 to times; j <- 1 to times if i % 2 == 1 && j % 2 == 0) {
      println(s"i:$i,j:$j")
    }

    // i代表长，j代表宽，打印面积大于25的
    for(i <- 1 to times; j <- 1 to times if i * j > 25){
      println(s"长: $i, 宽: $j, 面积: ${i * j}")
    }

    // 对于条件复杂的for循环，可以把小括号写成大括号
    for {
      i <- 1 to times
      j <- 1 to times
      x = i * j
      if x > 25
    } println(s"长: $i, 宽: $j, 面积: ${i * j}")

 // 打印乘法口诀表，不能使用var
    for(i <- 1 to 9){
      for(j <- 1 to i){
        print(s"$j * $i = ${i * j}\t")
      }
      println()
    }

    for (i <- 1 to 9; j <- 1 to i) {
      print(s"$j * $i = ${i * j}\t")
      if (i == j) println()
    }

    for(i <- 1 to 9; j <- 1 to i) print(s"$j * $i = ${i * j} ${if(i == j) "\n" else ""
}")

 // 把1-10之间的偶数以集合的形式返回，让for循环具备返回值的能力
    // yield后面的语句块的返回值就是for yield的返回值
    // yield后面的语句块一定有返回值即使没有返回值，它会以Unit的对象"()"作为返回值
    val result = for (i <- 1 to 10) yield {
      if (i % 2 == 0) i
    }
    println(result)

    val result2 = for (i <- 1 to 10 if (i % 2 == 0)) yield i
    println(result2)

    // 构建一个集合对象 1-10，对每个元素都加上5，返回发音新的结果
    val result3 = for (i <- 1 to 10) yield i + 5
    println(result3)
  }
}
```

Unit类型

java中无返回值的方法类型是void
scala中没有void，它使用Unit类型来代替
Unit的实例就是“()”

idea常用快捷键

1.Ctrl＋E，可以显示最近编辑的文件列表

2.Shift＋Click可以关闭文件
3.Ctrl＋[或]可以跳到大括号的开头结尾
4.Ctrl＋Shift＋Backspace可以跳转到上次编辑的地方
5.Ctrl＋F12，可以显示当前文件的结构
6.Ctrl＋F7可以查询当前元素在当前文件中的引用，然后按F3可以选择
7.Ctrl＋N，可以快速打开类
8.Ctrl＋Shift＋N，可以快速打开文件
9.Alt＋Q可以看到当前方法的声明
10.Ctrl＋W可以选择单词继而语句继而行继而函数
11.Alt＋F1可以将正在编辑的元素在各个面板中定位
12.Ctrl＋P，可以显示参数信息
13.Ctrl＋Shift＋Insert可以选择剪贴板内容并插入
14.Alt＋Insert可以生成构造器/Getter/Setter等
15.Ctrl＋Alt＋V 可以引入变量。例如把括号内的SQL赋成一个变量
16.Ctrl＋Alt＋T可以把代码包在一块内，例如try/catch
17.Alt＋Up and Alt＋Down可在方法间快速移动