**大数据编程语言**

**Scala**

|  |  |  |
| --- | --- | --- |
|  | 文件标识： |  |
| 当前版本： | 1.0 |
| 作 者： | 马卫花 |
| 完成日期： | 2022-10 |

**目 录**

[1 Scala简介 2](#_Toc1306)

[1.1 什么是Scala 2](#_Toc24103)

[1.2 为什么学习Scala 2](#_Toc11759)

[2 Scala环境搭建 3](#_Toc20730)

[2.1 Scala下载 3](#_Toc1826)

[2.1.1 安装Scala插件 3](#_Toc17283)

[2.1.2 安装Maven 7](#_Toc31682)

[2.1.3 创建Scala项目 11](#_Toc11534)

[3 Scala基本语法 17](#_Toc25729)

[3.1 Scala变量 17](#_Toc4045)

[2.2 Scala基本数据类型 17](#_Toc15851)

[3.3 Scala基本类型操作 18](#_Toc29882)

[3.4 运算符的优先级 19](#_Toc22854)

[4 程序控制结构 20](#_Toc27500)

[4.1 if语句 20](#_Toc9610)

[4.2 for循环语句 21](#_Toc21940)

[4.3 while循环语句 22](#_Toc22806)

[5 函数 23](#_Toc17662)

[5.1 基本语法 23](#_Toc18955)

[5.2 值函数 25](#_Toc8131)

[5.3 函数小结 26](#_Toc29297)

[5.4 练习作业 26](#_Toc14229)

# 1 Scala简介

## 什么是Scala

Scala是一门多范式（multi-paradigm）的编程语言，式Scalable Language的简写。Scala设计的初衷是要集成面向对象编程和函数式编程的各种特性。Scala运行在Java虚拟机上，并兼容现有的Java程序。Scala源代码被编译成Java字节码，所以它可运行在JVM之上，并可以调用现有的Java类库。

Scala官网网址：

<https://docs.scala-lang.org/zh-cn/tour/tour-of-scala.html#>

## 为什么学习Scala

Spark框架是用Scala编写的，在学习大数据分析案例时，采用与底层框架相同的编程语言有如下好处：

* 性能开销小

在JVM上运行R或Python编写的算法，我们必须要花费代价在不同环境中传递代码和数据，在数据转换的过程中有可能出现数据丢失，但如果采用Spark Scala API编写准确率和运行性能都能提高。

* 有助于理解Spark工作原理

即使在Python或R中调用了Spark，API的工作仍然反映底层工作原理，所以学习Scala更有助于“Spark思考”。

* 能用上最新的版本和最好的功能

Spark的机器学习、流处理和图分析库全都是用Scala编写的，所以学习Scala可以更快更容易的掌握Spark的最新版本，应用其最好的功能。

# 2 Scala环境搭建

## 2.1 Scala下载

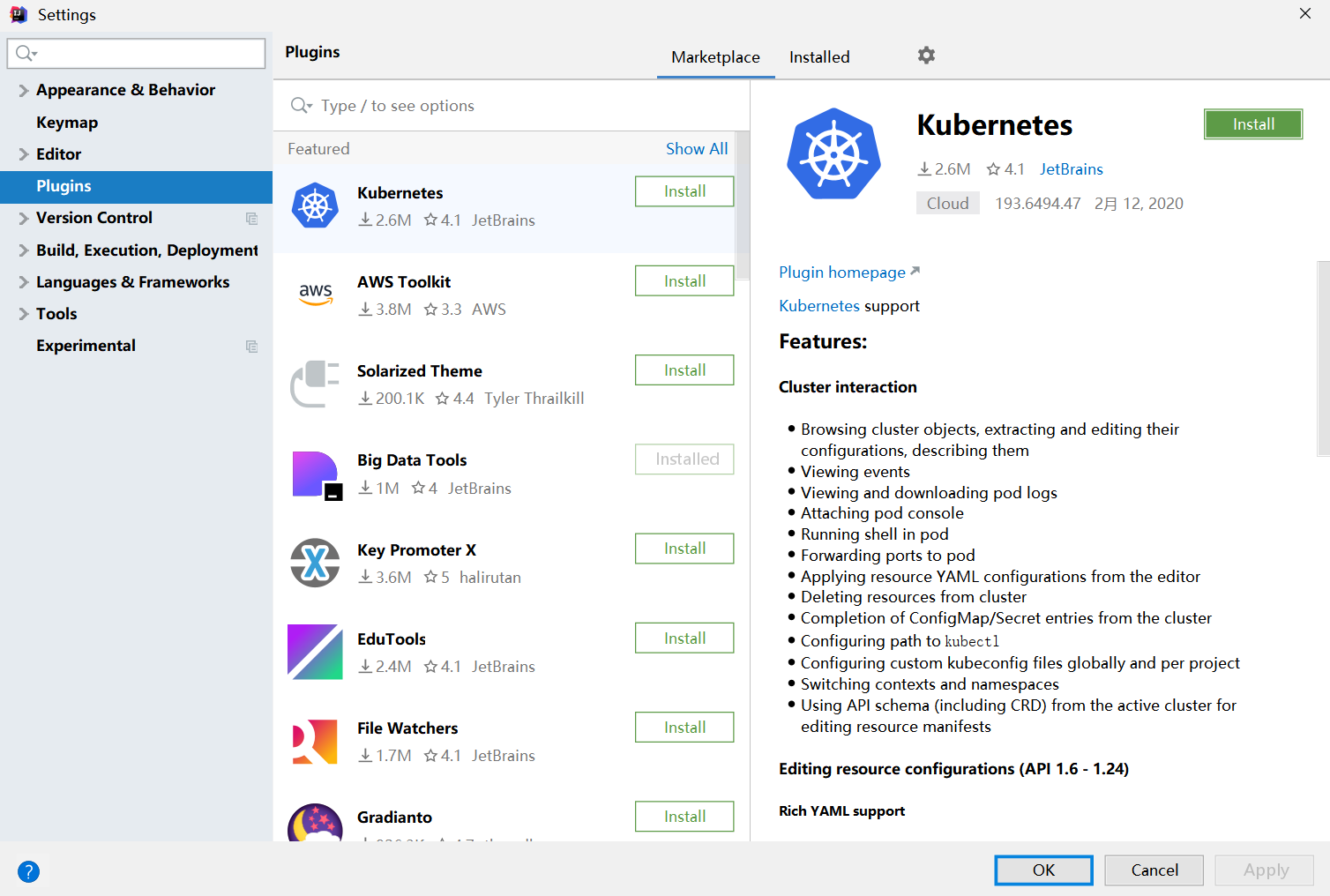
**2.1.1 安装Scala插件**

Scala的开发工具比较多，例如Eclipse、VS Code、IDEA等等，在企业中开发选择用IDEA开发工具，本案例就选用IDEA开发工具安装Scala插件。

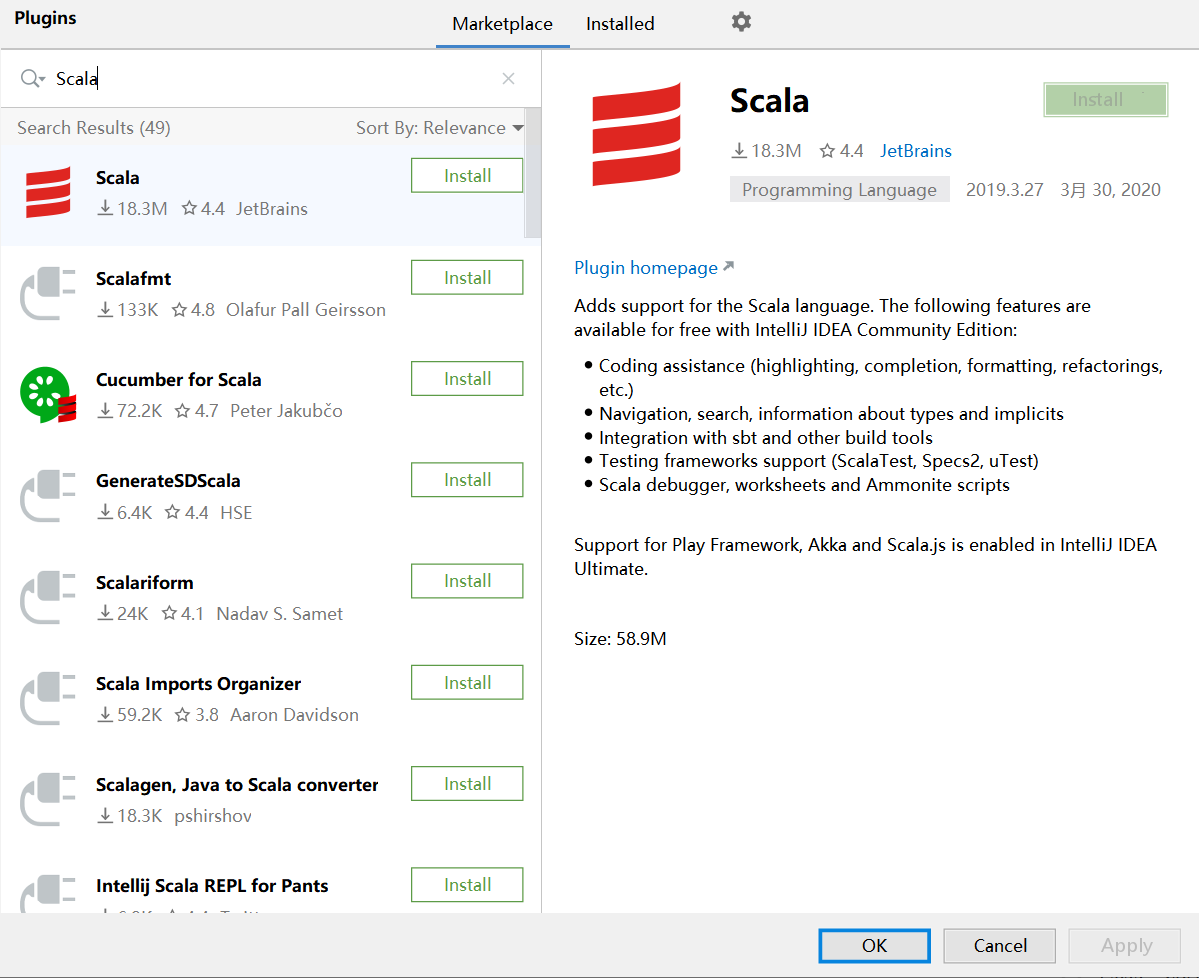
* 在线安装Scala插件

在线安装Scala插件的操作步骤如下所示：

1. 打开IDEA显示界面，点击工具栏中的File菜单，选择Settings菜单，然后再选择Plugins菜单，显示如下图所示的界面。

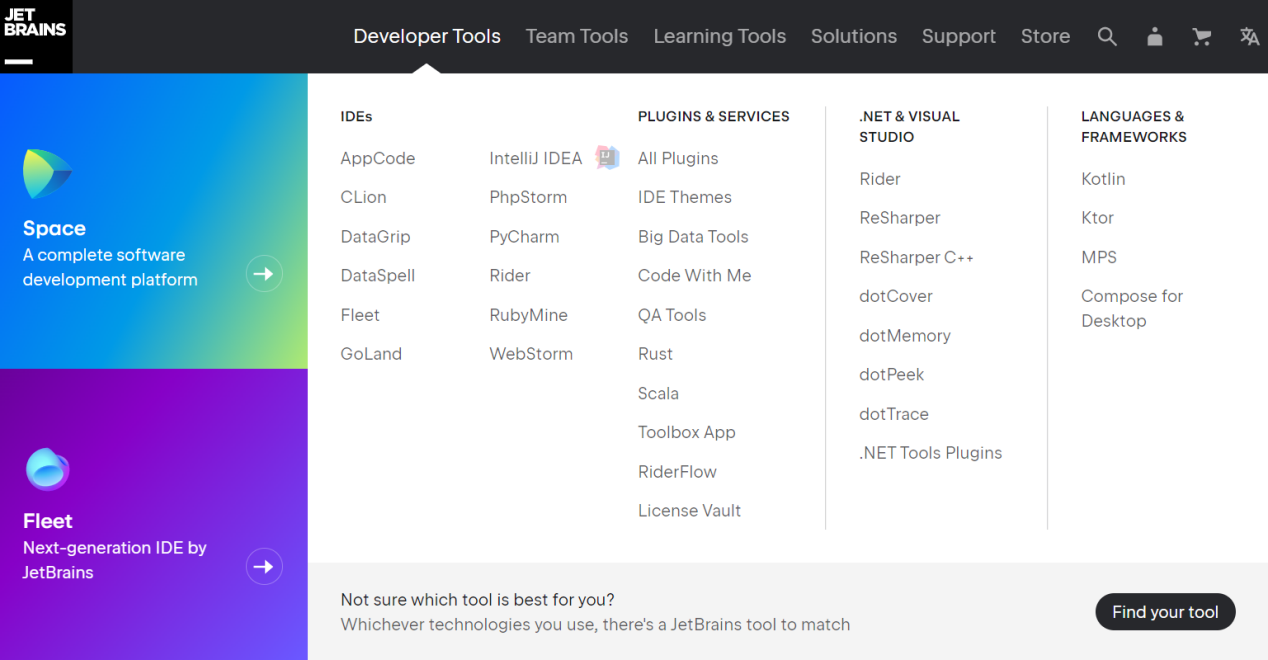


2）在Plugins显示主窗口的搜索框中输入Scala关键字如下图所示的界面，选择Scala插件，点击最右边界面上的Install按钮，完成Scala线上安装。Scala安装完成，重启IDEA。

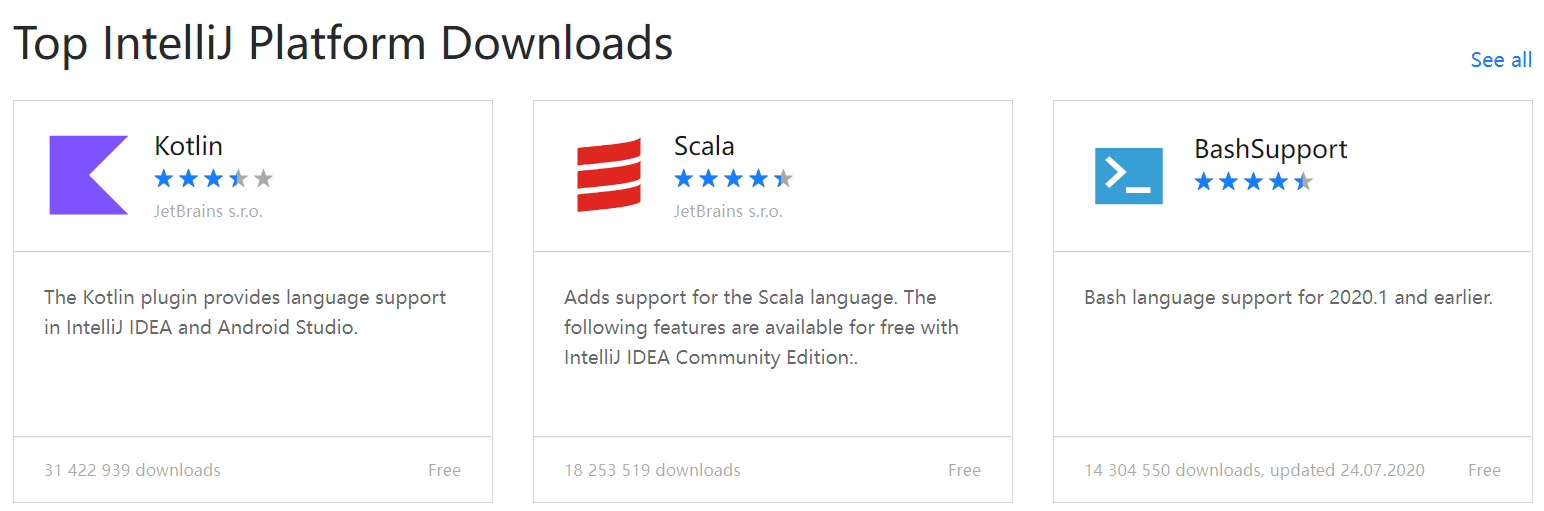


* 离线安装Scala插件

1. 访问IDEA的官网https://www.jetbrains.com/idea/，将鼠标移动到Developer Tools,如下图所示：



1. 点击界面中的All Plugins，鼠标向下滑动显示如下图所示的界面。



（3）点击界面中的Scala，页面跳转到介绍Scala插件页面，然后选择Versions选项卡如下图所示，在显示的版本信息页面中，鼠标向下滑动下载和本机安装的IDEA版本对应的Scala插件如下图2所示。IDEA2019.3版下载Scala插件2019.3.23版。

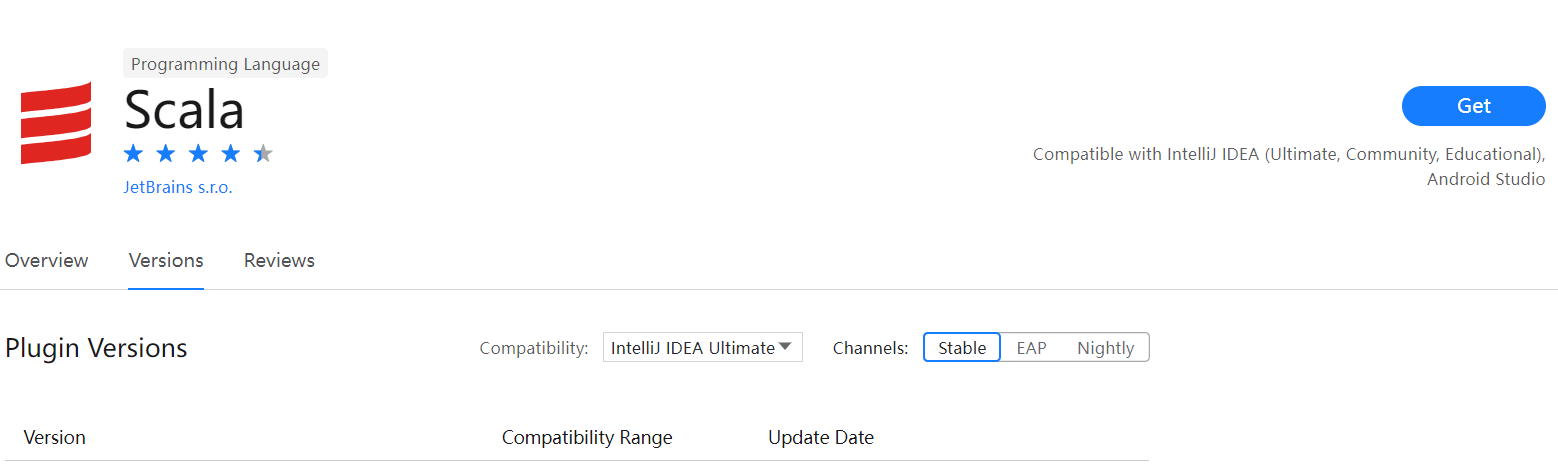


图1

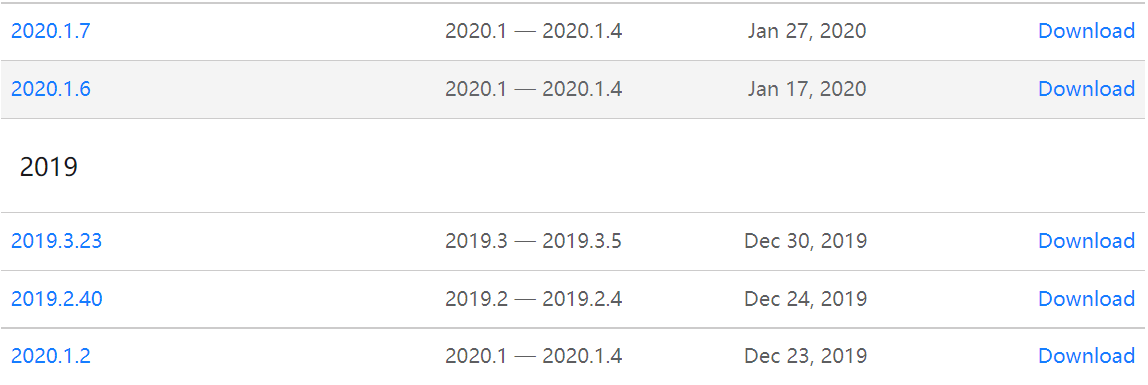
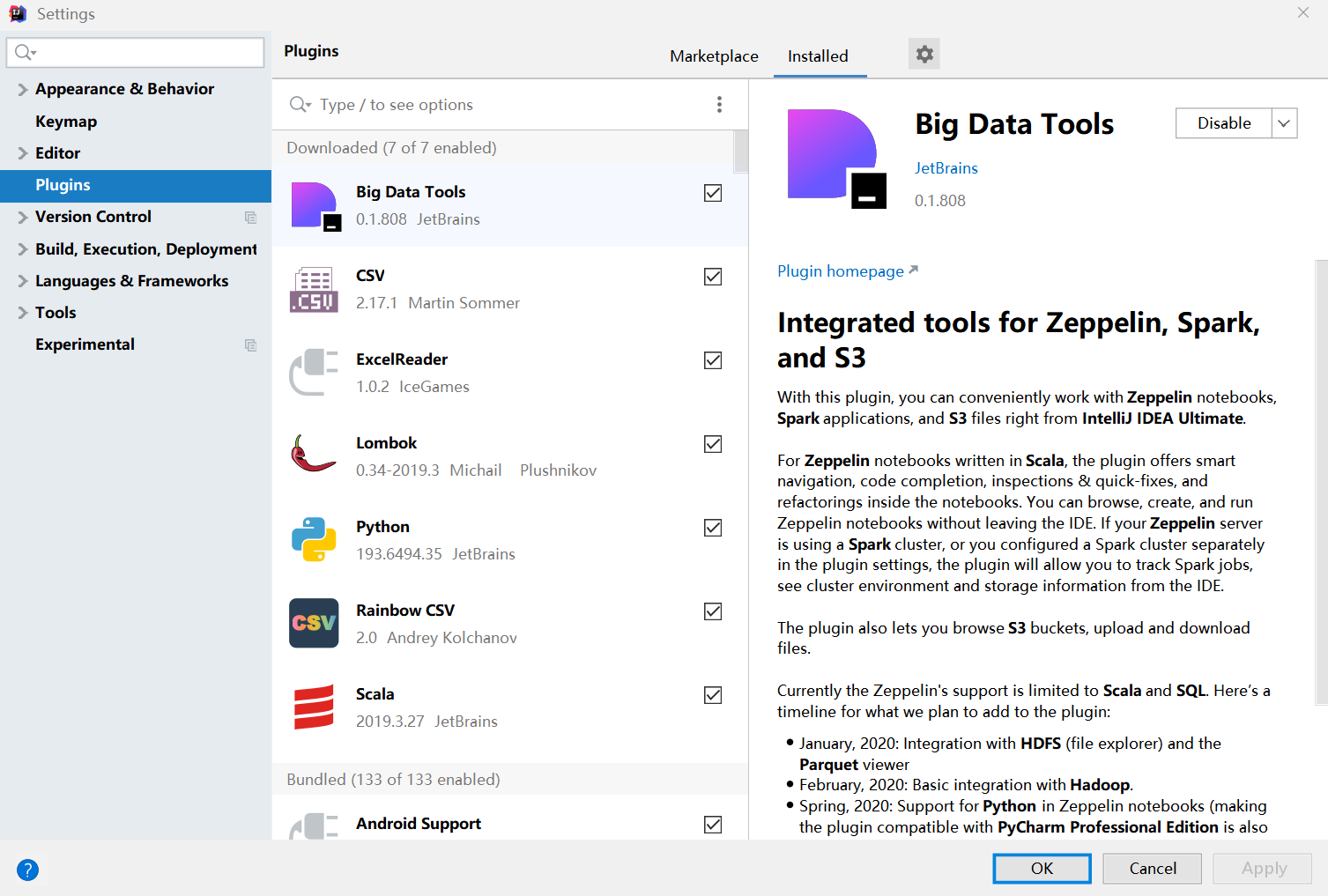
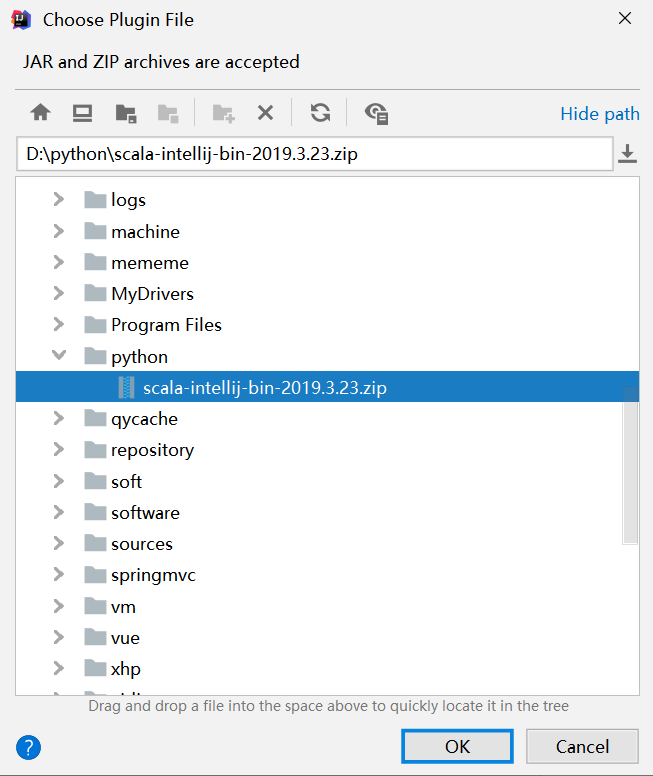


图2

1. 下载完成后，打开IDEA环境选择File菜单中的Settings，然后再Settings显示窗口中选择plugins如下图所示：



点击界面中的选项按钮，选择Install Plugin from Disk(选择从磁盘安装)，然后选择已下载下来的Scala插件，操作界面如下图所示，最后点击OK按钮。



选择完成后点击界面中的Restart IDE，重新启动IDEA就可完成离线Scala插件的安装。

**2.1.2 安装Maven**

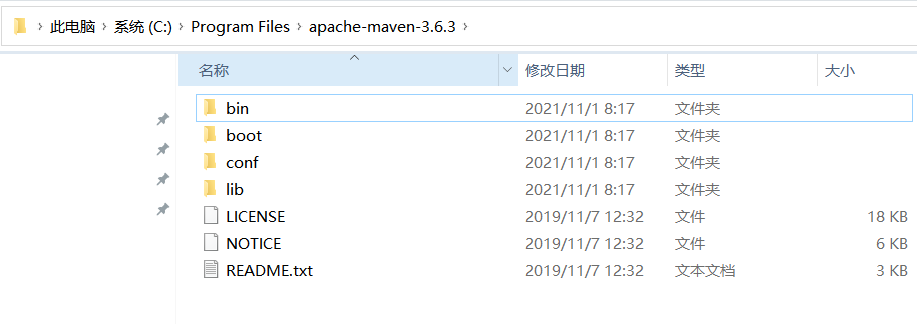
在企业开发中，大多数项目都会使用Maven作为项目的构建工具，Scala项目也可采用Maven构建。

1. 下载安装Maven

Maven下载的地址：[https://archive.apache.org/dist/maven/maven-3/，打开网页后选择要下载的版本地址，](https://archive.apache.org/dist/maven/maven-3/，打开网页后显示如下图所示的界面。)本书的实训案例下载的是Maven3.6.3版本，下载界面如下图所示。apache-maven-3.6.3-bin.tar.gz安装包适用于Linux、Mac操作系统，apache-maven-3.6.3-bin.zip安装包适用于Window操作系统，选择下载apache-maven-3.6.3-bin.zip安装包。



1. 解压下载的安装包，并修改解压后的文件夹名为:apache-maven-3.6.3如下图所示。



1. 修改Maven配置

进入Maven的安装目录下的conf文件夹，打开settings.xml文件，修改Maven本地仓库路径，如下图所示将53行的内容复制到55行，然后设置Maven本地仓库路径，如将D:\develop\repository设置为本地仓库，设置如下图所示。

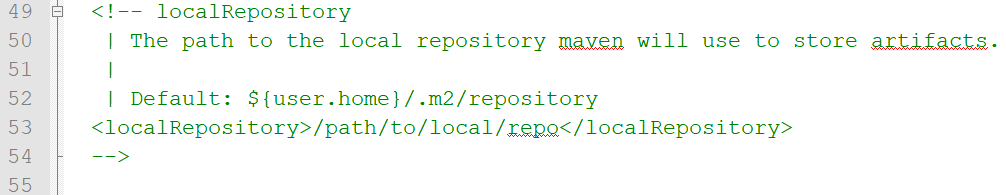


图1

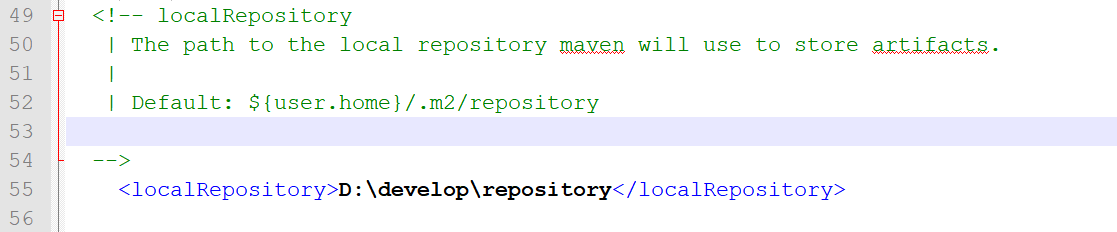
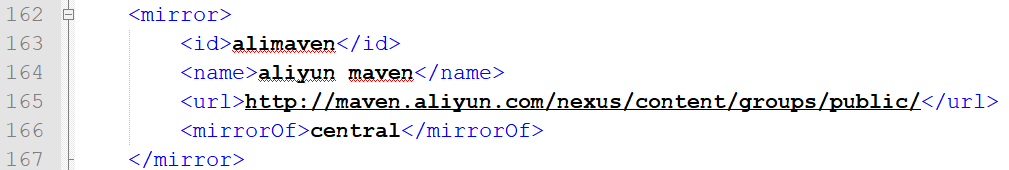


图2

1. 修改镜像源

在settings.xml文件的mirrors标签中添加子标签，设置国内的镜像源，构建项目时下载的速度比较快，如下图所示：



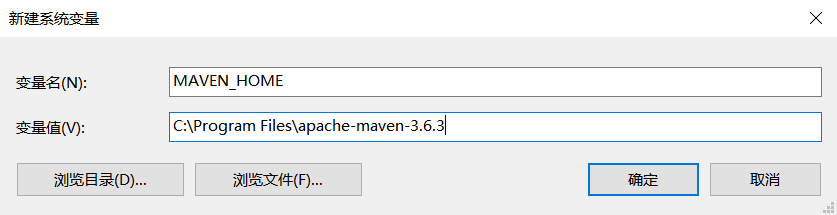
注意：mirror标签必须包含在mirrors标签元素内。

1. 配置环境变量

右键桌面上“我的电脑”选择“属性”，在弹出的页面上点击“高级系统设置”。如下图所示。



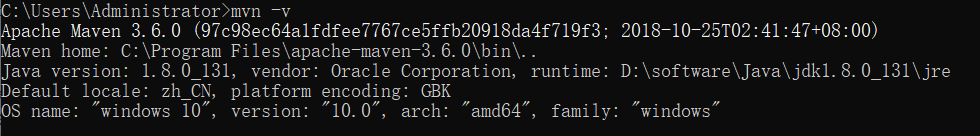
打开系统环境变量配置窗口，点击新建弹出如下图所示的窗口，变量名文本框中输入MAVEN\_HOME，变量值文本框中输入Maven解压后的路径，设置完成后点击确定按钮。



添加配置path，找到path系统变量，点击界面中的新建输入%MAVEN\_HOME%\bin然后点击保存。

1. 验证maven安装是否完成

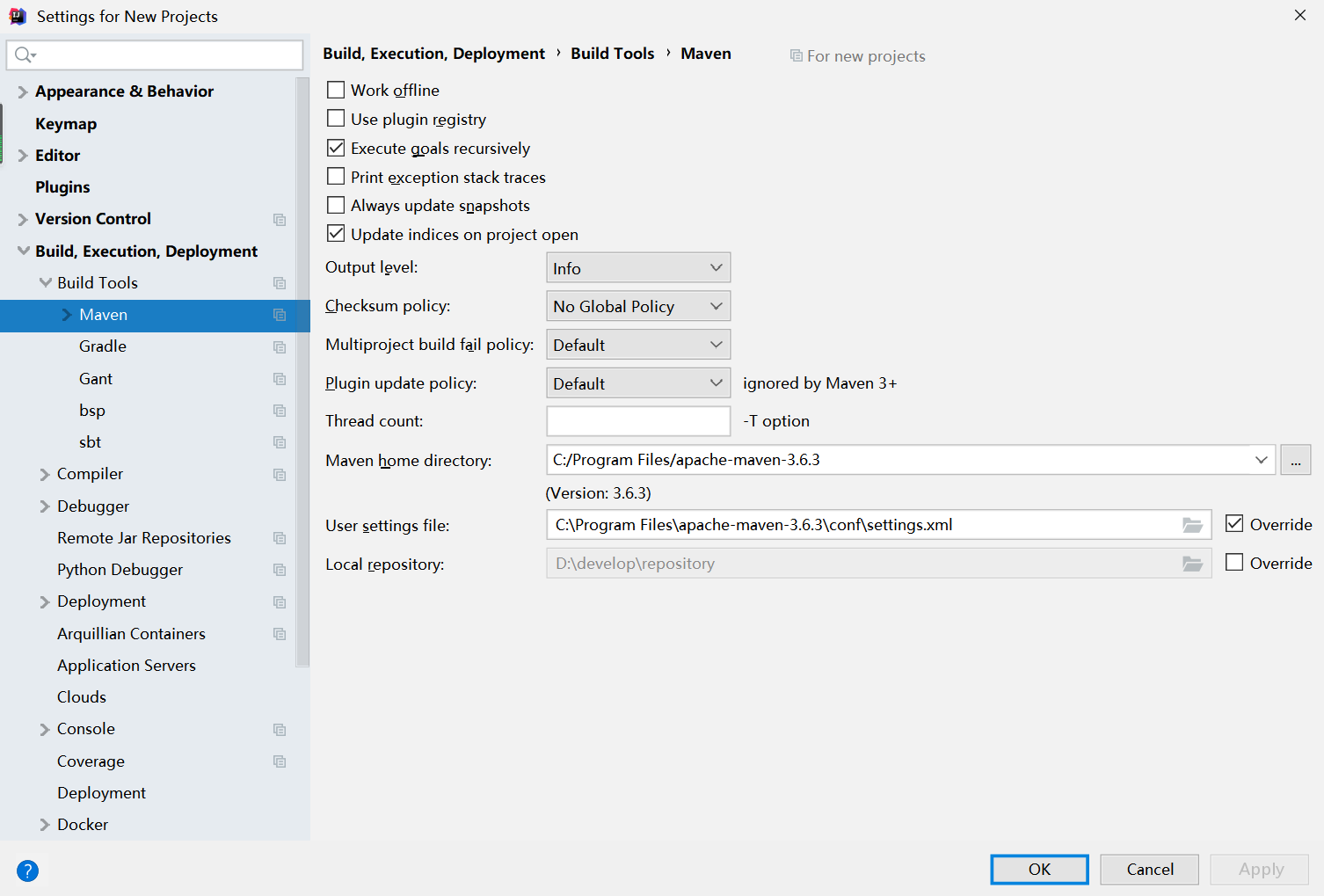
打开window命令窗口，输入mvn -v显示出maven的安装版本如下图所示。



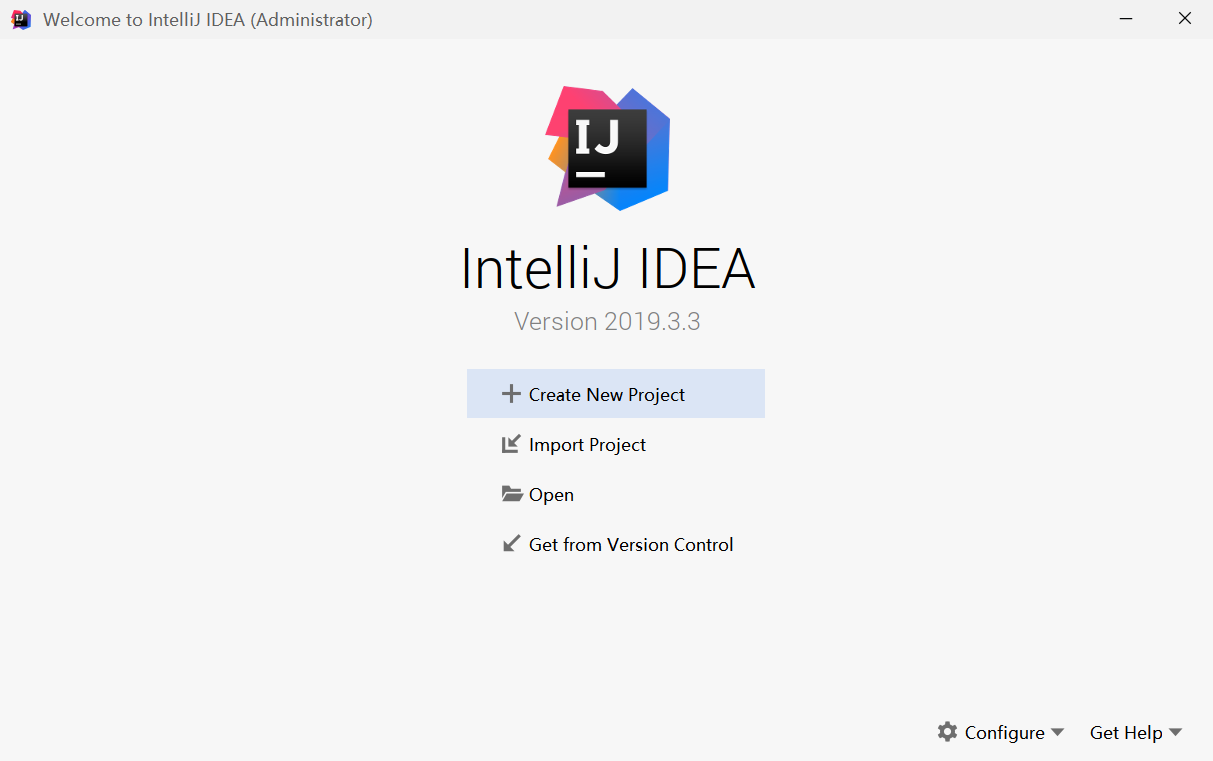
**2.1.3 创建Scala项目**

1）在IDEA中配置Maven

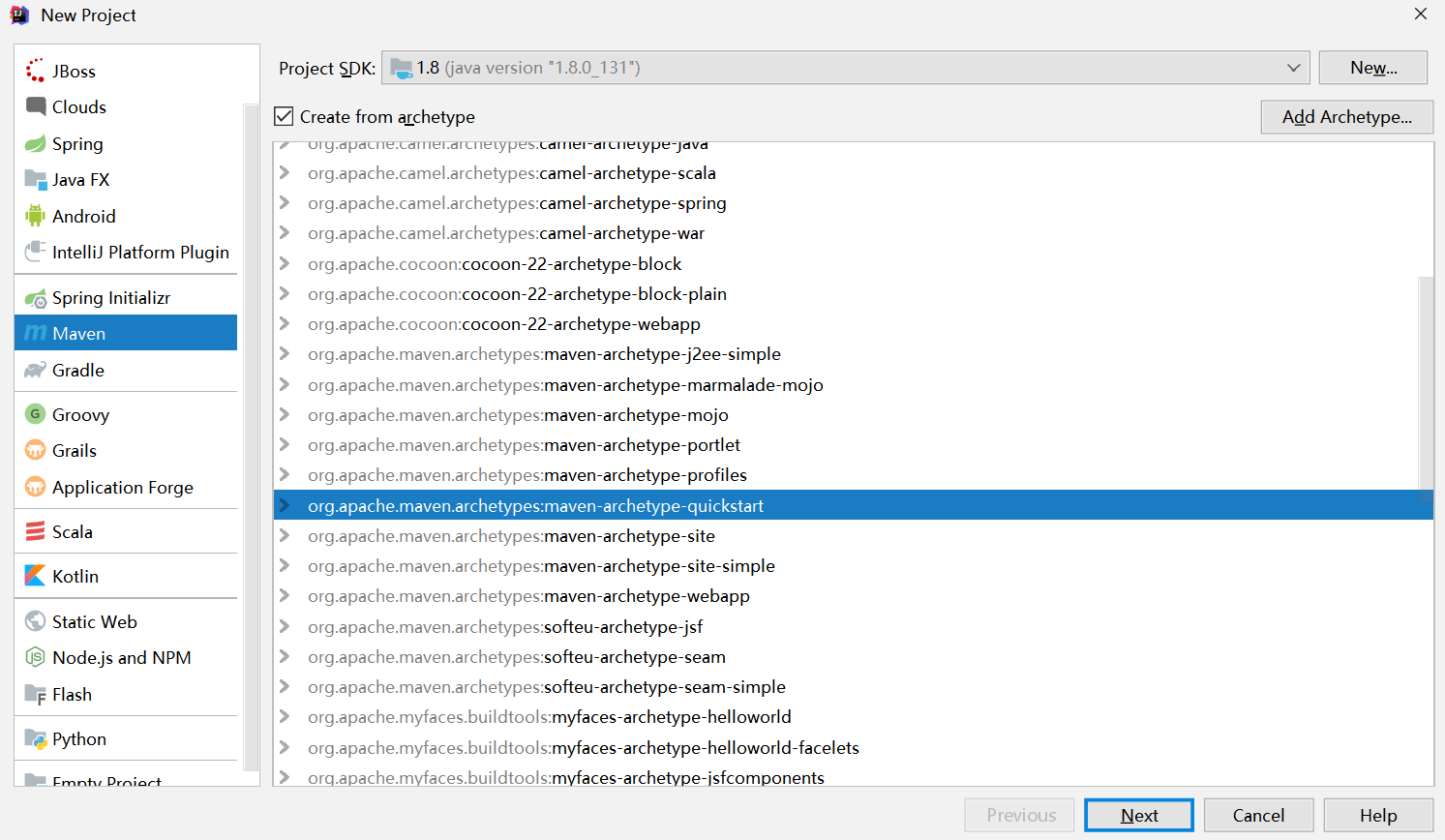
打开Idea欢迎界面如下图所示，点击Configure选择Settings菜单，显示Settings选项菜单界面，然后选择Build,Execution,Deployment，在选择Build Tools目录下的的Maven，显示界面如下图所示。Maven home directory选择本机安装的Maven地址，User settings file选择安装目录的settings.xml，Local repository显示settings.xml文件中设置的本地仓库路径。



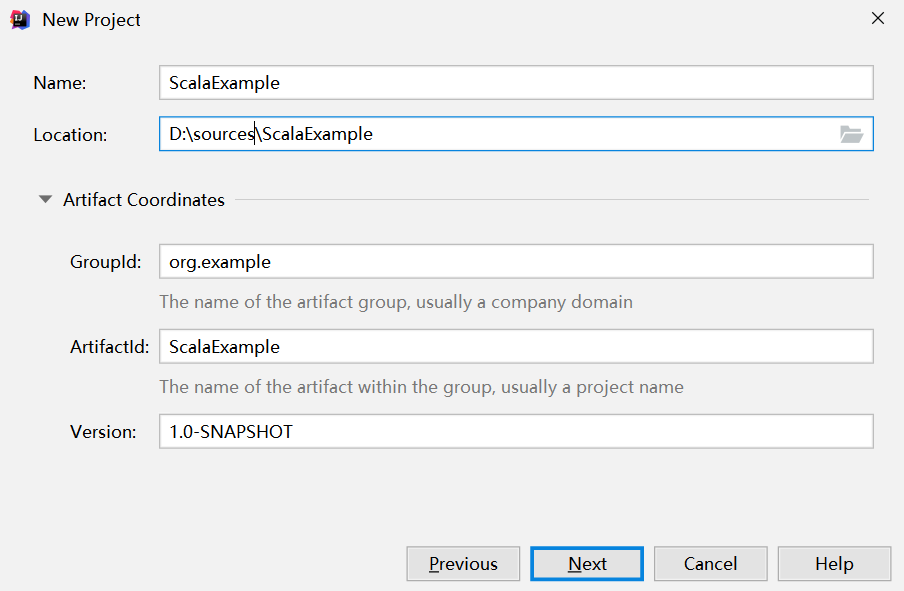
2）创建Maven项目

打开Idea欢迎界面如下图所示，点击界面中的create new project。

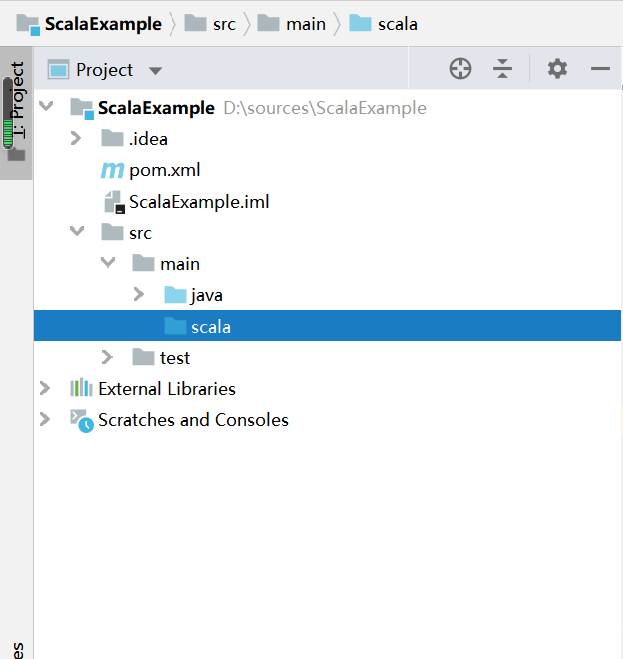
打开New Project对话框，选择左边导航栏中的Maven，在Project SDK选项框中选择JDK的版本，然后勾选Create frmo archetype，选择org.apache.maven.mchetypes.maven-archetype-quickstart，点击界面中的Next按钮。



创建一个ScalaExample Maven项目如下图所示，Name文本框中输入项目名，Location设置项目路径，GroupId输入组织ID（通常为组织域名的倒写），ArtifactId输入项目名称，Version为项目的版本。设置完成后点击next按钮，在选择Maven的安装目录以及相关选项，点击Finish按钮完成项目的创建。



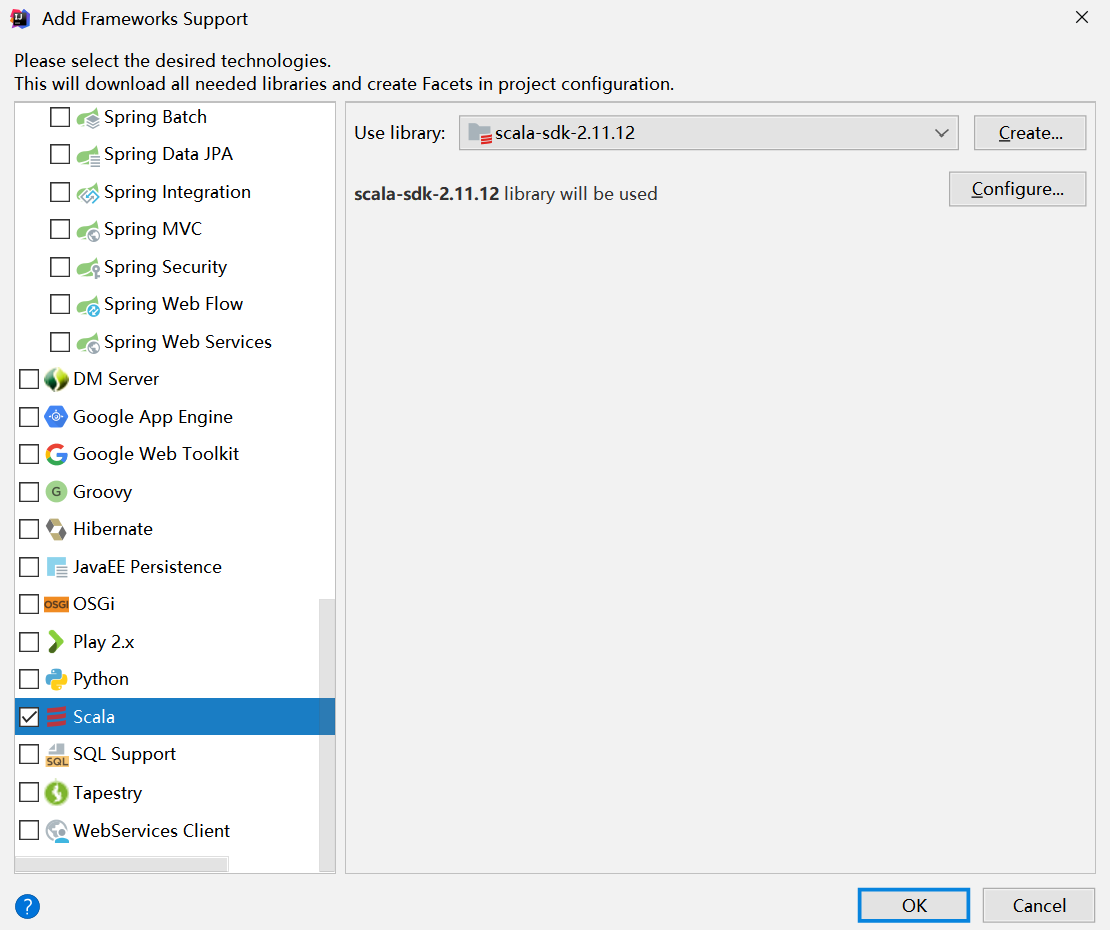
在新建项目的main目录下创建scala目录，并设置为源码目录的根路径，如下图所示：



3）添加scala支持

创建Maven项目后添加Scala框架支持，才能在项目中编写scala代码。

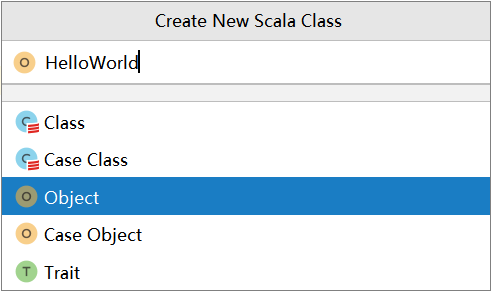
选择项目目录点击右键选择弹出菜单中的Add Framework Support，然后在选择scala操作界面如下图所示：



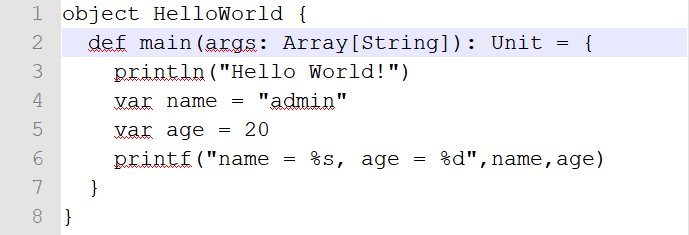
如果Use library没有Scala版本选项，点击Create按钮，选择需要下载的版本然后在选择。

4）编写scala代码

选择scala点击右键，选择新建Scala Class，在弹出的界面中选择Object，填写伴生类名HelloWorld，如下图所示。

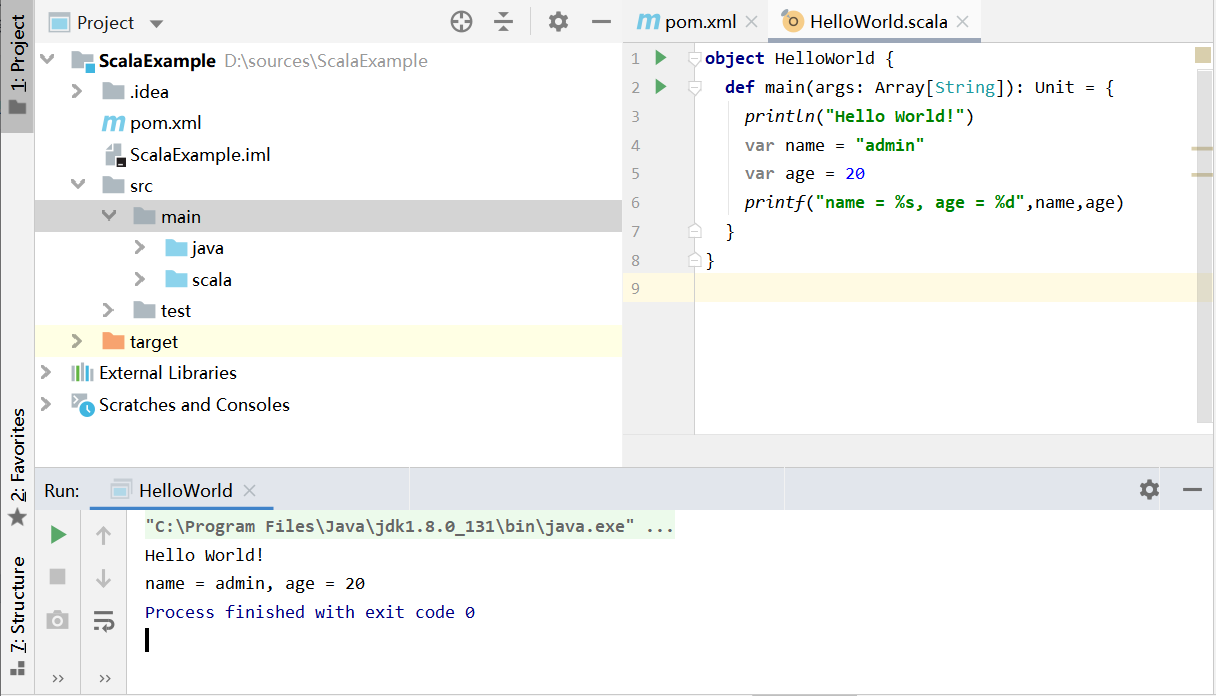


编写Scala代码。



1. object创建的是一个单例对象，也叫伴生对象。
2. main是Scala程序运行的主入口方法。
3. var 声明的变量可改变。

运行结果如下图所示。



# 3 Scala基本语法

## 3.1 Scala变量

变量是一种使用方便的占位符，用于引用计算机内存地址。

基本语法:

var/val 变量名[:变量类型]=变量值

(自动推导)

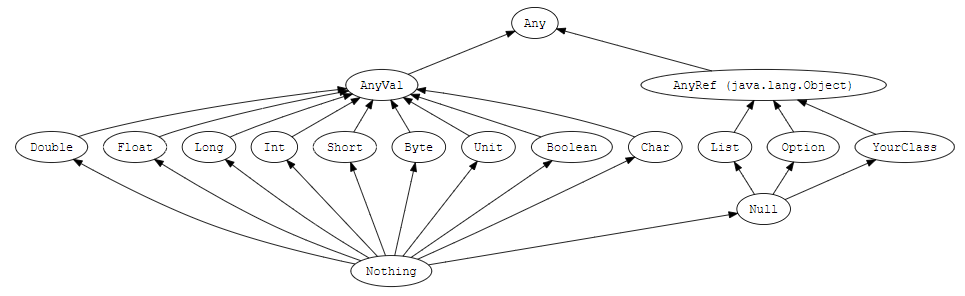
var修饰的变量可改变

val修饰的变量不可改变,val修饰的对象属性在编译后类似与Java中增加的final关键字。

注意：声明变量时，类型可以省略【类型推断】

## 2.2 Scala基本数据类型

Scala中的数据类型和Java中的数据类型相似,但是Scala中的所有数据类型都是对象(即object修饰的)，没有Java中的基本数据类型。



Scala中的数据类型主要分为两个大类型:AnyVal(值类型)和AnyRef(引用类型).Scala中的数据类型如下所示：



Scala中还有一个非常特殊的数据类型元组。元组是不同类型值的聚集，它可以将不同类型的值放在一个变量中保存。

元组示例:

object TuppleScala {

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

//元祖：将不同类型的数据保存为一个变量

//1. 创建元祖

val t1 = ("zhangsan",22,"bigdata")

//2. 访问元祖

println(t1.\_1)

println(t1.\_2)

//3.元祖中的数据最多22个数据，分别对应于Tuple1、Tuple2、....Tuple22

//4.循环遍历

for(e <- t1.productIterator){

println(e)

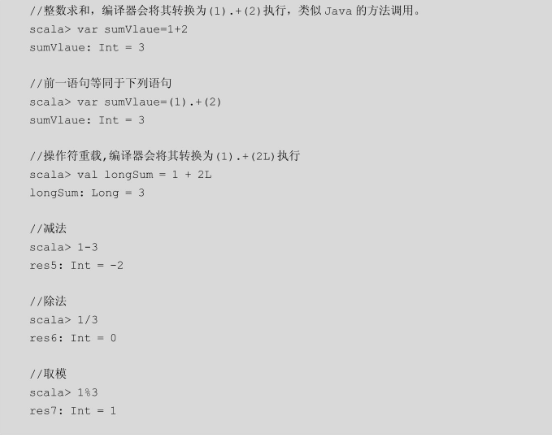
}

}

}

## 3.3 Scala基本类型操作

Scala是纯面向对象编程语言，在Scala中一切皆为对象，所有操作都是方法调用，Scala数据类型操作包括算数运算符、关系运算符、逻辑运算符、位运算操作、对象比较运算操作及字符串比较运算操作。



## 3.4 运算符的优先级

Scala中运算符的优先级如下表所示：



# 4 程序控制结构

## 4.1 if语句

* 单分支

val flag=true

if(flag){

print("true")

}

* 双分支

val flag=true

if(flag){

print("true")

}else{

print("false")

}

* 多分支

val score=75

if(score>90){

println("A")

}else if (score>80){

println("B")

}else if(score>60){

println("C")

}else{

println("D")

}

提示：

* 如果大括号{}内的逻辑代码只有一行，大括号可以省略
* Scala中任意表达式都是有返回值的，因此if else表达式也是有返回结果的，具体的返回结果的值取决于满足条件的执行语句的最后一行
* 有返回值的分支

val flag =true

val res1:Any = if(flag){

"hello"

}

println(res1)

val res2:Any=if(flag){

"hello"

}else{

"world"

}

println(res2)

## 4.2 for循环语句

Scala为for循环提供了非常多的特性，这些for循环特性被成为for推导式或者for表达式。

* to and until

for (i <- 1 to 5)

println(s"i=$i")

for (i <- 1 until 5)

println(s"i=$i")

//分析，这两个循环操作相同吗?

* Range

for (i <- Range(1,5)) //类似于until

println(s"i=$i")

for (i <- Range(1,5,2)) //步长为2

println(s"i=$i")

* 循环守卫

循环守卫就是增加条件判断，如果条件成立，则执行循环体，否则跳过循环体。

for(i <- 1 to 10 if i % 2 ==0){

println(s"i=$i")

* 带返回值的循环

val result1 = for(i <- 1 to 10){

println(s"i=$i")

}

println(result1)

val result2 = for(i <- 1 to 5) yield i \* 2

println(result2)

## 4.3 while循环语句

Scala也提供while循环语句，不过Scala弱化了while循环语句的作用，在程序中不推荐使用While循环，尽量使用for循环或者递归代替while循环语句。

**var i =1**

**var sum =0**

**while(i < 10){**

**sum - sum + i**

**i = i +1**

**}**

**练习：采用for,while循环计算 1!+2!+...5!**

# 5 函数

## 5.1 基本语法

在 Scala 当中，函数是一等公民，像变量一样，既可以作为函数的参数使用，也可以将函数赋值给一个变量. ，函数的创建不用依赖于类或者对象，而在 Java 当中，函数的创建则要依赖于类、抽象类或者接口。

**def 函数名 ([参数名: 参数类型], ...)[[: 返回值类型] =] {**

**语句...**

**[return] 返回值**

**}**

* 函数声明关键字为 def (definition)
* [参数名: 参数类型], ...：表示函数的输入(就是参数列表)没有。 如果有，多个参数使用逗号间隔
* 函数中的语句：表示为了实现某一功能代码块

**def gcd(x:Int,y:Int) :Int={**

**if (x % y ==0)**

**y**

**else**

**x**

**}**

**val result = gcd(20,10)**

**print(result)**

函数体中如果只有一条语句可以省略。

**def gcd(x:Int,y:Int) :Int= if (x % y ==0) y else x**

Scala具有类型推导功能，可以根据最终的返回值推到出函数的返回值类型，因此在实际应用中也常常会省略函数的返回值。

**def sum(x:Int,y:Int) => x+y**

**val result=sum(10,20)**

**Println(result)**

提示：类型推导有两个限制：

1. 如果需要return关键之指定返回值，则必须显式地指定函数返回值类型
2. 如果函数中存在递归调用，则必须显式地指定函数返回值的类型

**常见函数形式**

* 无参无返回值

**def f1( )={**

**println("hello")**

**}**

**f1( )**

**f1**

**def f2={**

**println("scala")**

**}**

**//f2() // ERROR**

**f2 //调用函数f2**

* 有参无返回值

**def f1(a:Int,b:Int)={**

**println(s"a+b=${a+b}")**

**}**

**f1(1,2)**

* 无参有返回值

**def f1()={**

**"scala"**

**}**

**println(f1)**

* 有参有返回值

**def** f1(a:Int,b:Int) ={

a + b

}

* 有参带默认值

**def** f1(a:Int,b:Int = 1) ={

a + b

}

*println*(f1(10,20)) //?

*println*(f1(10))

**def** f2(a:Int=10,b:Int)={

a + b

}

*println*(f2(b=20))

* 可变参数函数

**def** f1(a:Int\*) ={

**var** sum = 0

**for**(i <- a)

sum += 1

sum

}

*print*(f1(1,2,3,4,5))

print(f1()) //?

println(f1(1,2,3,4,5,6))

## 5.2 值函数

在Scala中函数也是对象,也可以像变量一样被赋值，把这种函数称为函数字面量(function literal)或值函数,标准的函数字面量定义如下图所示：

**val sum = (x:Int,y:Int) =>{x+ y}**

一行语句大括号可以省略，如下格式编写:

**val sum = (x:Int,y:Int)=>x+y**

**println(sum(10,20))**

形如(x:Int,y:Int)=>{println(x+y);x+y}的表达式被称为Lambda表达式.

注意：

* 值函数不能像普通函数可以指定返回值类型，编译器会自动根据类型推导确定函数的返回值类型。
* 值函数可以如一般的val变量一样使用lazy关键字修饰，使用lazy修饰的函数字面量的作用原理同一般的val变量一样，只有当程序真正使用到该函数字面量的时候才会被创建。

Eg:

**val f = (x:Double)=>x\*2**

**print(f(10))**

* 匿名函数

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

//对匿名函数的说明

//1. 不需要写 def 函数名

//2. 不需要写返回类型，使用类型推导

//3. = 变成 =>

//4. 如果有多行，则使用{} 包括

val triple = (x: Double) => {

println("x=" + x)

3 \* x

}

println("triple" + triple(3)) // 9.0

}

请编写一个匿名函数，可以返回 2 个整数的和，并输出该匿名函数的类型.

val f1 = (n1: Int, n2: Int ) => {

println("匿名函数被调用")

n1 + n2

}

println("f1 类型=" + f1)

println(f1(10, 30))

## 5.3 函数小结

* 函数可以有很多个参数，但是如果没有参数，那么在调用时可以不带()
* 函数的参数类型和返回值类型均可以是任意类型
* 函数的返回值类型可以根据返回值自动推断，因此返回值类型可以省略
* 函数如果指定了return关键字返回数据，那么返回值类型就不能省略
* 函数如果指定了返回值类型为Unit,那么无论是否有return关键字和最后一行是什么，返回值均为()
* Scala语法中任何的语法结构都可以嵌套其他语法结构，函数可以嵌套类、函数等，类中亦可嵌套其他的类和函数
* 函数可以赋值给一个变量
* 函数不可以重载

## 5.4 练习作业

1：编写scala程序，完成以下功能： 假设有整型变量x，判断x是否为偶数，若为偶数，则在控制台上打印“输入的数值是偶数”。无论x是否为偶数，最后都要在控制台上输出x的值？

2.有两个整型变量x，y，请在控制台上输出x与y中值较大的那个数。

3.打印100到1000的[水仙花数](https://so.csdn.net/so/search?q=%E6%B0%B4%E4%BB%99%E8%8A%B1%E6%95%B0&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_57239983/article/details/_blank)