

Life is short, you need Spark!



从**零**开始

不需要任何基础，带领您无痛入门 Spark

云计算分布式大数据 Spark 实战高手之路

王家林著

Spark 亚太研究院系列丛书 版权所有

伴随着大数据相关技术和产业的逐步成熟，继 Hadoop 之后，Spark 技术以其无可比拟的优势，发展迅速，将成为替代 Hadoop 的下一代云计算、大数据核心技术。

本书特点

- ▶ 云计算分布式大数据 Spark 实战高手之路三部曲之第一部
- ▶ 网络发布版为图文并茂方式，边学习，边演练
- ▶ 不需要任何前置知识，从零开始，循序渐进

本书作者



王家林，Spark 亚太研究院院长和首席专家，中国目前唯一的移动互联网和云计算大数据集大成者。在 Spark、Hadoop、Android 等方面有丰富的源码、实务和性能优化经验。彻底研究了 Spark 从 0.5.0 到 0.9.1 共 13 个版本的 Spark 源码，并已完成 2014 年 5 月 31 日发布的 Spark1.0 源码研究。

Hadoop 源码级专家，曾负责某知名公司的类 Hadoop 框架开发工作，专注于 Hadoop 一站式解决方案的提供，同时也是云计算分布式大数据处理的最早实践者之一。

Android 架构师、高级工程师、咨询顾问、培训专家。

通晓 Spark、Hadoop、Android、HTML5，迷恋英语播音和健美。

“真相会使你获得自由。”

— 耶稣《圣经》约翰 8:32KJV

“所有人类的幸福都来源于不能直面事实。”

— 释迦摩尼

“道法自然”

— 老子《道德经》第 25 章

《云计算分布式大数据 Spark 实战高手之路》

系列丛书三部曲

《云计算分布式大数据 Spark 实战高手之路---从零开始》：

不需要任何基础，带领您无痛入门 Spark 并能够轻松处理 Spark 工程师的日常编程工作，内容包括 Spark 集群的构建、Spark 架构设计、RDD、Shark/SparkSQL、机器学习、图计算、实时流处理、Spark on Yarn、JobServer、Spark 测试、Spark 优化等。

《云计算分布式大数据 Spark 实战高手之路---高手崛起》：

大话 Spark 源码，全世界最有情趣的源码解析，过程中伴随诸多实验，解析 Spark 1.0 的任何一句源码！更重要的是，思考源码背后的问题场景和解决问题的设计哲学和实现招式。

《云计算分布式大数据 Spark 实战高手之路---高手之巅》：

通过当今主流的 Spark 商业使用方法和最成功的 Hadoop 大型案例让您直达高手之巅，从此一览众山小。



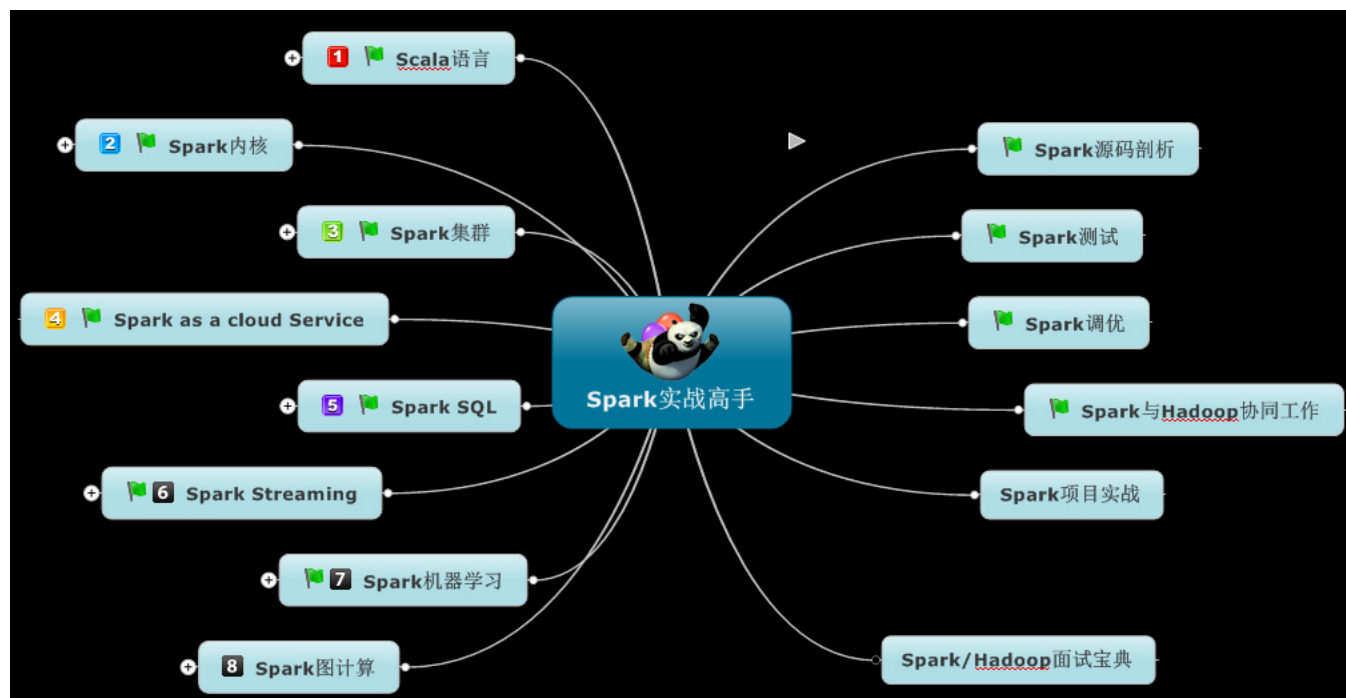
《前言》

Spark采用一个统一的技术堆栈解决了云计算大数据的如流处理、图技术、机器学习、NoSQL查询等方面的所有核心问题，具有完善的生态系统，这直接奠定了其一统云计算大数据领域的霸主地位；

要想成为Spark高手，需要经历六大阶段



Spark 实战高手之核心技能点



第一阶段：熟练的掌握Scala语言

1. Spark 框架是采用 Scala 语言编写的，精致而优雅。要想成为 Spark 高手，你就必须阅读 Spark 的源代码，就必须掌握 Scala；
 2. 虽然说现在的 Spark 可以采用多语言 Java、Python 等进行应用程序开发，但是最快速的和支持最好的开发 API 依然并将永远是 Scala 方式的 API，所以你必须掌握 Scala 来编写复杂的和高性能的 Spark 分布式程序；
 3. 尤其要熟练掌握 Scala 的 trait、apply、函数式编程、泛型、逆变与协变等；
- 推荐课程：“精通Spark的开发语言：Scala最佳实践”

第二阶段：精通Spark平台本身提供给开发者API

1. 掌握 Spark 中面向 RDD 的开发模式 掌握各种 transformation 和 action 函数的使用；
 2. 掌握 Spark 中的宽依赖和窄依赖以及 lineage 机制；
 3. 掌握 RDD 的计算流程，例如 Stage 的划分、Spark 应用程序提交给集群的基本过程和 Worker 节点基础的工作原理等
- 推荐课程：“18 小时内掌握Spark：把云计算大数据速度提高 100 倍以上!”

第三阶段：深入Spark内核

此阶段主要是通过 Spark 框架的源码研读来深入 Spark 内核部分：

1. 通过源码掌握 Spark 的任务提交过程；
2. 通过源码掌握 Spark 集群的任务调度；
3. 尤其要精通 DAGScheduler、TaskScheduler 和 Worker 节点内部的工作的每一步的细节；

推荐课程：[“Spark 1.0.0 企业级开发动手：实战世界上第一个Spark 1.0.0 课程，涵盖Spark 1.0.0 所有的企业级开发技术”](#)

第四阶段:掌握基于Spark上的核心框架的使用

Spark 作为云计算大数据时代的集大成者，在实时流处理、图技术、机器学习、NoSQL 查询等方面具有显著的优势，我们使用 Spark 的时候大部分时间都是在使用其上的框架例如 Shark、Spark Streaming 等：

1. Spark Streaming 是非常出色的实时流处理框架，要掌握其 DStream、transformation 和 checkpoint 等；
2. Spark 的离线统计分析功能，Spark 1.0.0 版本在 Shark 的基础上推出了 Spark SQL，离线统计分析的功能的效率有显著的提升，需要重点掌握；
3. 对于 Spark 的机器学习和 GraphX 等要掌握其原理和用法；

推荐课程：[“Spark企业级开发最佳实践”](#)

第五阶段:做商业级别的Spark项目

通过一个完整的具有代表性的 Spark 项目来贯穿 Spark 的方方面面，包括项目的架构设计、用到的技术的剖析、开发实现、运维等，完整掌握其中的每一个阶段和细节，这样就可以让您以后可以从容面对绝大多数 Spark 项目。

推荐课程：[“Spark架构案例鉴赏：Conviva、Yahoo！、优酷土豆、网易、腾讯、淘宝等公司的实际Spark案例”](#)

第六阶段：提供Spark解决方案

1. 彻底掌握 Spark 框架源码的每一个细节；
2. 根据不同的业务场景的需要提供 Spark 在不同场景的下的解决方案；
3. 根据实际需要，在 Spark 框架基础上进行二次开发，打造自己的 Spark 框架；

推荐课程：[“精通Spark：Spark内核剖析、源码解读、性能优化和商业案例实战”](#)

《第二章：动手实战 Scala》

Scala 一门面向对象和函数编程完美结合的语言，特别适合于构建大型项目和密集而复杂的数据处理。

Spark 是基于 Scala 语言开发的大数据通用处理平台，于此同时在 Spark 上虽然可以使用 Scala、Java、Python 三种语言进行分布式应用程序开发，但 Scala 确实支持最好的首选开发语言，所以无论是从阅读源码的角度来讲还是开发程序的角度来讲，对 Scala 的掌握都是必要且至关重要的。

本章的内容主要是从 Spark 的角度来讲解 Scala，以动手实战为核心，从零开始，循序渐进的掌握 Scala 函数式编程和面向对象编程。

从零起步，动手实战 Scala 三部曲：

- 第一步：动手体验 Scala
- 第二步：动手实战 Scala 面向对象编程
- 第三步：动手实战 Scala 函数式编程

本将是动手实战 Scala 三部曲的第一步：动手体验 Scala，具体内容如下所示：

- 1，在命令行和 Scala IDE for Eclipse 中动手体验 Scala；
- 2，Scala 函数特性编程实战；
- 3，Scala 中的表达式实战；

注：本章所讲解的 Scala 的运行的操作系统是 Window 7，IDE 使用的 Scala IDE for Eclipse，关于 Ubuntu 上 Scala 与 IDEA 的安装和使用请参考本书的第一章。

不需任何前置知识，从零开始，循序渐进，成为 Spark 高手！



1, 在命令行和 Scala IDE for Eclipse 中动手体验 Scala

Scala 一门基于 JVM 的面向对象和函数式编程相结合的静态语言，安装和运行 Scala 首先需要 Java 虚拟环境，直接到官方网站下载和安装 Java 即可：

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

因为 Spark 的最新的 1.0.2 版本支持的是 Scala 2.10.X 版本，而 Scala 2.10.X 是不支持 Java 8 的，所以下载的时候需要下载 Java 6 或者 Java 7 的版本(需要注意系统的版本，家林的系统是 Window 7 的 64 位的版本)，安装并配置好之后需要在命令行下验证一下：

```
C:\Users\Administrator>java -version
java version "1.7.0_67"
Java(TM) SE Runtime Environment (build 1.7.0_67-b01)
Java HotSpot(TM) 64-Bit Server VM (build 24.65-b04, mixed mode)
```

接下来安装 Scala 2.10.4，下载地址是：

<http://www.scala-lang.org/download/2.10.4.html>

下载下来之后点击安装即可，安装之后需要把 scala 的 bin 目录配置到环境变量 path 中，安装配置完成后用命令行验证一下：

```
C:\Users\Administrator>scala -version
Scala code runner version 2.10.4 -- Copyright 2002-2013, LAMP/EPFL
```

进入 Scala 命令行：

```
C:\Users\Administrator>scala
Welcome to Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67).
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

在 Scala 中定义一个常量是使用 val：

```
scala> val name = "rocky"
name: String = rocky
```

如果此时相对 name 的值进行修改就会报错：

```
scala> name = "Say hi to rocky"
<console>:8: error: reassignment to val
    name = "Say hi to rocky"
      ^
```

使用 var 声明一个变量：

```
scala> var hobby = "music"
hobby: String = music
```

此时修改 hobby 这个变量：

```
scala> hobby = "sports"
hobby: String = sports
```

可以发现我们在定义常量和变量的时候并未定义其类型，但是命令终端却显示出了类型，这是 Scala 类型的自我推到能力，Scala 可以根据变量和常量值的类型来推到变量和常量本身的类型。

当然，你可以显示的指定类型：

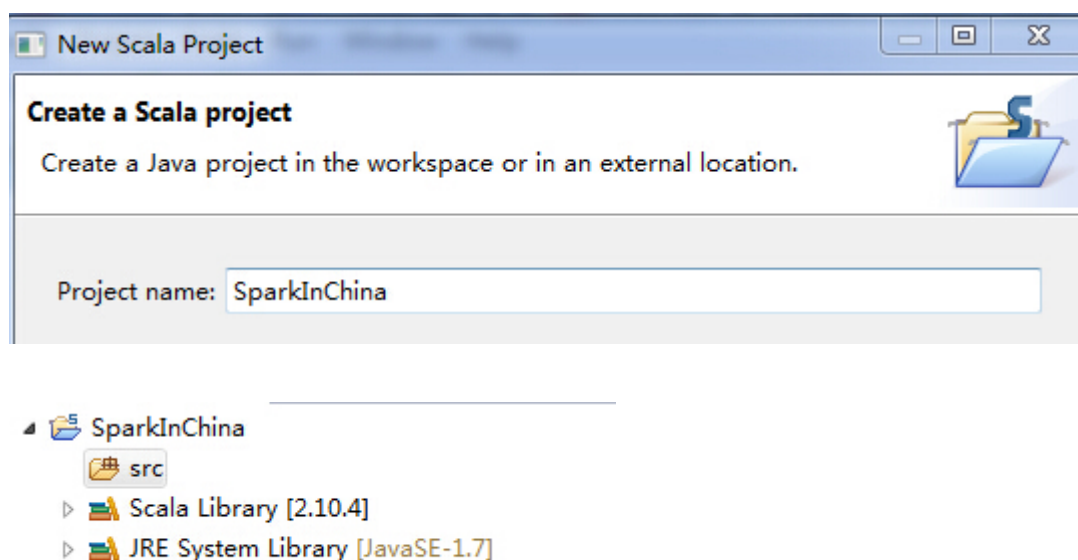
```
scala> var job : String = "Software engineer"
job: String = Software engineer
```

接下来下载 Scala IDE for Eclipse，看一下在 Eclipse 中如何写 Scala 代码，下载 Scala2.10.4 的版本即可，下载地址如下：

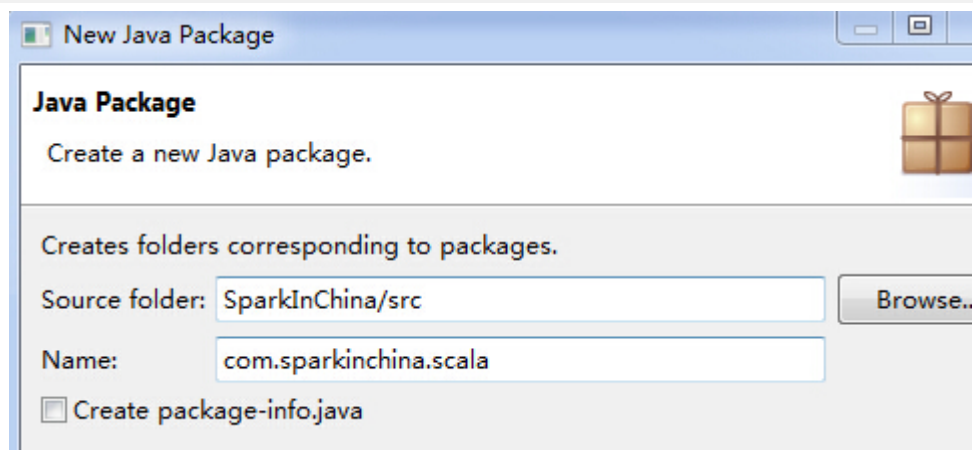
<http://scala-ide.org/download/sdk.html>

下载后直接解压启动即可。

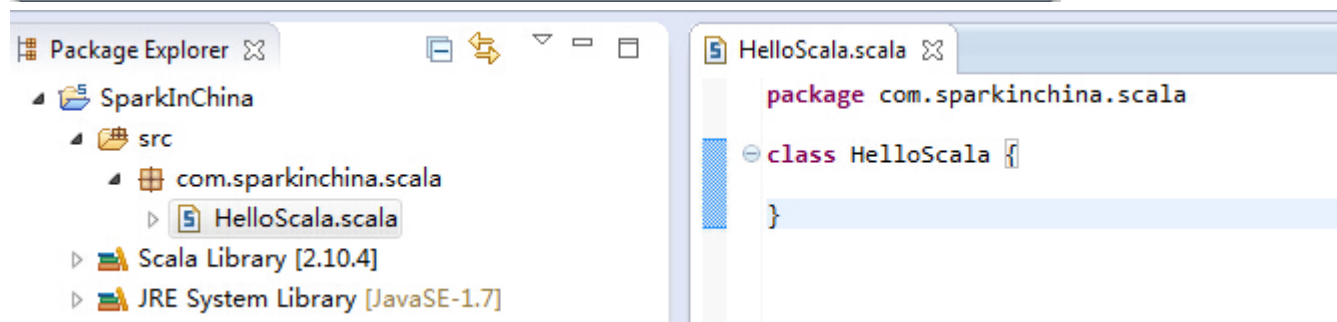
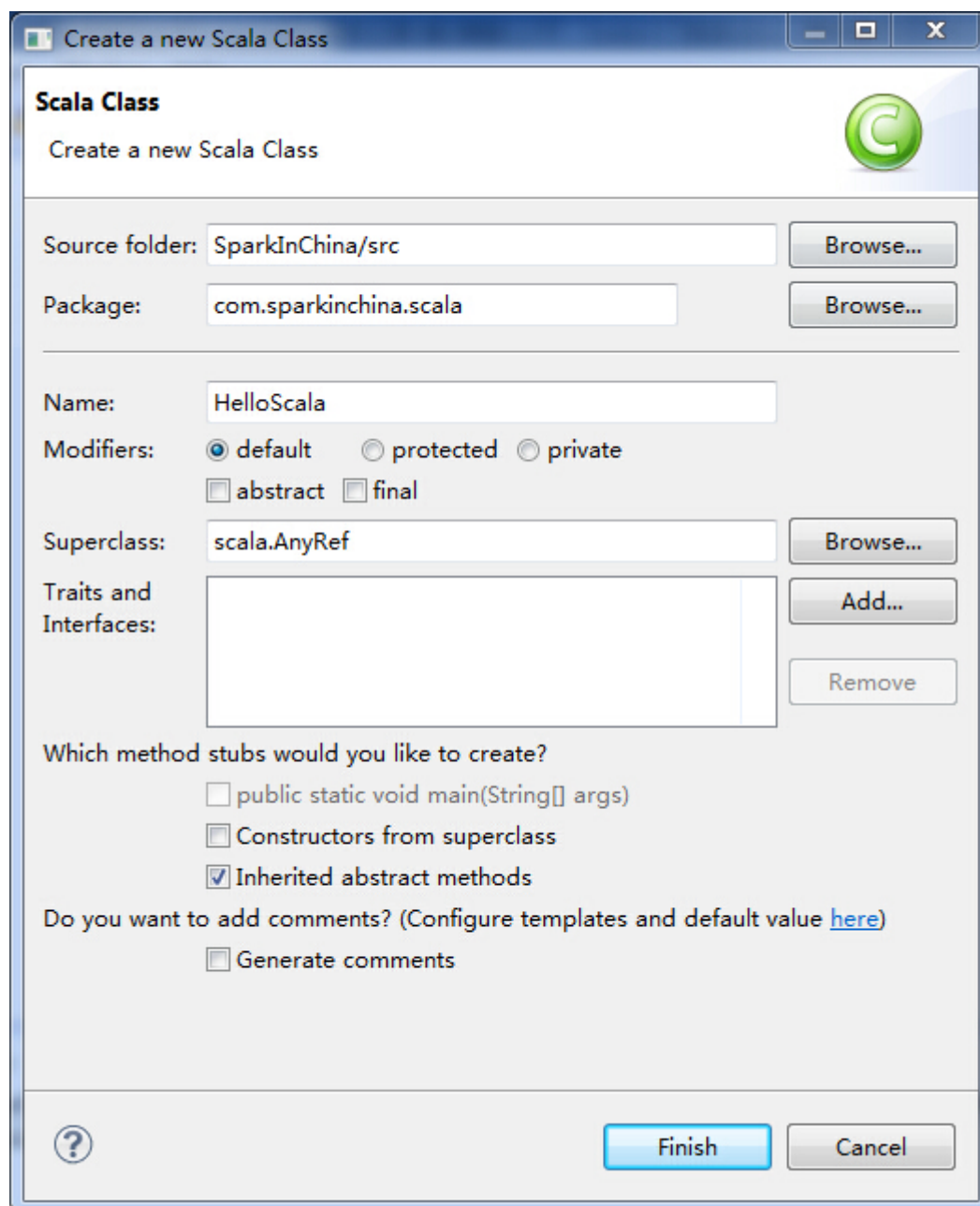
创建一个 Scala 项目：



在 src 下创建一个 scala 的包：



接下来创建一个名称为 “HelloScala” 的 Scala 类：



接下来写我们的 main 函数，需要注意的是在 Scala 中 main 函数需要存在于 object 对象中，所有我们需要一个 object HelloScala 并在其中编写 main 方法：

```
HelloScala.scala
package com.sparkinchina.scala

class HelloScala {

}

object HelloScala{

  def main(args: Array[String]) {
    println("Hello Scala!!!")
  }
}
```

编写完毕之后选择“Run as”为“Scala Application”即可，运行结果如下所示：

```
Problems Tasks Console
<terminated> HelloScala$ (1) [Scala Application] C:\Program Files\Ja
Hello Scala!!!
```

接下来使用 def 定义一个函数 “hello”：

```
package com.sparkinchina.scala

class HelloScala {

}

object HelloScala{

  def hello(name : String) : String = {
    "Hello" + name
  }

  def main(args: Array[String]) {
    println("Hello Scala!!!")
  }
}
```

Scala 中有两点需要注意：

- 1，函数体的最后一行的值就是整个函数的返回值；
- 2，类型的声明是位于变量或者函数或者类的后面的；

把函数的执行结果打印出来：

```
package com.sparkinchina.scala

class HelloScala {

}

object HelloScala {

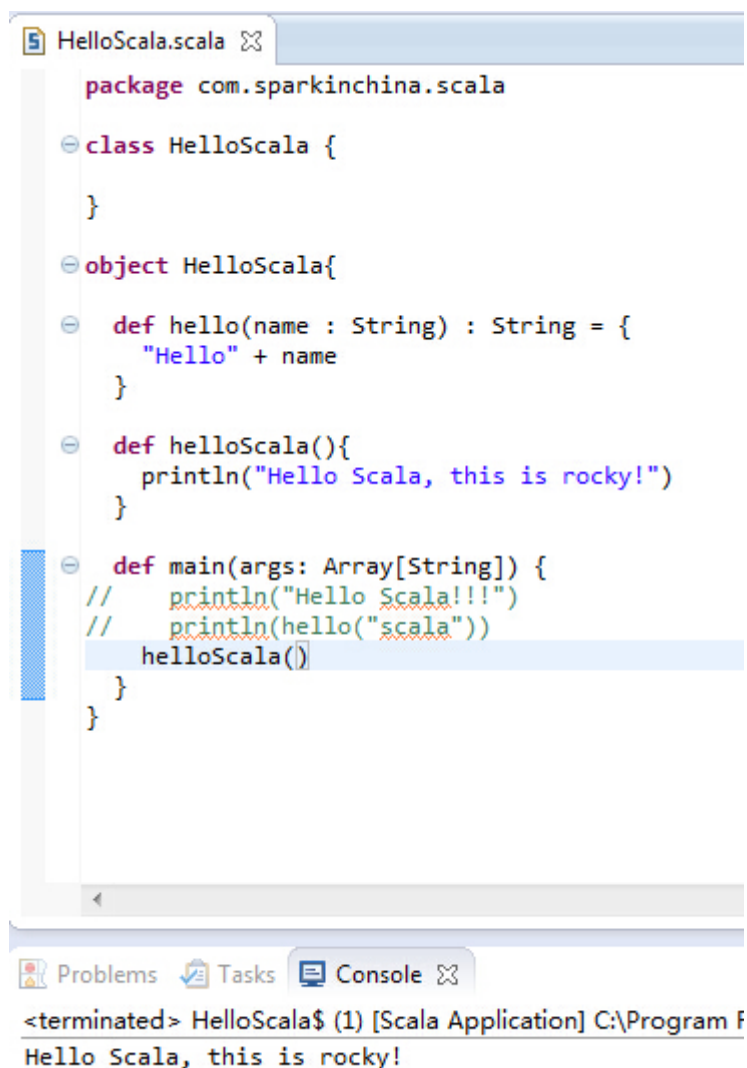
  def hello(name : String) : String = {
    "Hello" + name
  }

  def main(args: Array[String]) {
    // println("Hello Scala!!!")
    println(hello("scala"))
  }
}
```

执行程序：

```
Problems Tasks Console
<terminated> HelloScala$ (1) [Scala Application] C:\Progr
Helloworld
```

下面我们看另外一个方法的定义、使用和执行：



```
package com.sparkinchina.scala

class HelloScala {

}

object HelloScala{

  def hello(name : String) : String = {
    "Hello" + name
  }

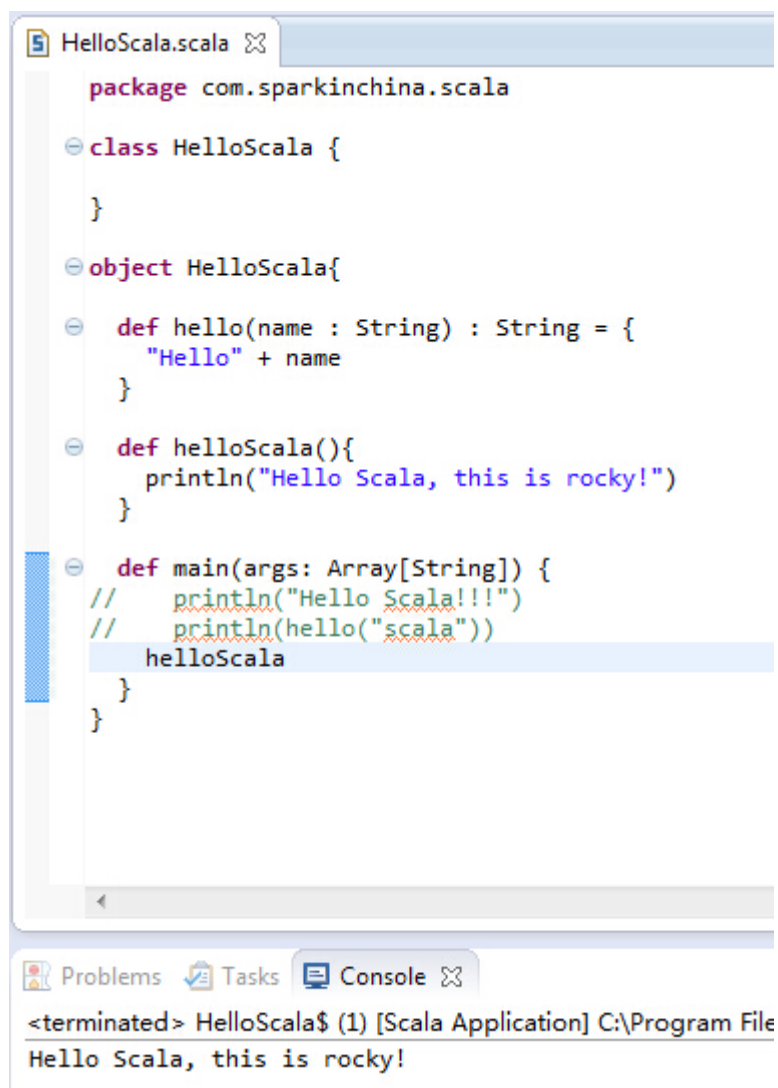
  def helloScala(){
    println("Hello Scala, this is rocky!")
  }

  def main(args: Array[String]) {
    // println("Hello Scala!!!")
    // println(hello("scala"))
    helloScala()
  }
}
```

<terminated> HelloScala\$ (1) [Scala Application] C:\Program F
Hello Scala, this is rocky!

2 , Scala 函数特性编程实战

当函数不带参数的使用，我们调用的时候可以省略括号：



```
package com.sparkinchina.scala

class HelloScala {

}

object HelloScala{

  def hello(name : String) : String = {
    "Hello" + name
  }

  def helloScala(){
    println("Hello Scala, this is rocky!")
  }

  def main(args: Array[String]) {
    // println("Hello Scala!!!")
    // println(hello("scala"))
    helloScala
  }
}
```

Problems Tasks Console

<terminated> HelloScala\$ (1) [Scala Application] C:\Program File
Hello Scala, this is rocky!

接下来定义一个匿名函数、调用并运行：


```
package com.sparkinchina.scala

class HelloScala {

}

object HelloScala{

  def hello(name : String) : String = {
    "Hello" + name
  }

  def helloScala(){
    println("Hello Scala, this is rocky!")
  }

  def add = (x : Int, y : Int) => x + y

  def main(args: Array[String]) {
    // println("Hello Scala!!!")
    // println(hello("scala"))
    // helloScala
    println(add(1 , 2))
  }
}

<terminated> HelloScala$ (1) [Scala Application] C:\Program
3
```

在 Scala 总函数式一等公民，所以我们可以把函数付给一个常：

```
package com.sparkinchina.scala

class HelloScala {
}

object HelloScala{

  def hello(name : String) : String = {
    "Hello" + name
  }

  def helloScala(){
    println("Hello Scala, this is rocky!")
  }

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

  def main(args: Array[String]) {
    // println("Hello Scala!!!")
    // println(hello("scala"))
    // helloScala
    // println(add(1 , 2))
    println(sum(1 , 2))
  }
}

<terminated> HelloScala$ (1) [Scala Application] C:\Program
3
```

另外一个特性是 Scala 函数编程的柯里化，这个特性允许函数定义的时候有两个括号：

```
class HelloScala {  
}  
  
object HelloScala{  
  def hello(name : String) : String = {  
    "Hello" + name  
  }  
  
  def helloScala(){  
    println("Hello Scala, this is rocky!")  
  }  
  
  def add = (x : Int, y : Int) => x + y  
  val sum = (x : Int, y : Int) => x + y  
  
  def sum2(x : Int)(y : Int) = x + y  
  
  def main(args: Array[String]) {  
    // println("Hello Scala!!!")  
    // println(hello("scala"))  
    // helloScala  
    // println(add(1 , 2))  
    // println(sum(1 , 2))  
    println(sum2(2)(3))  
  }  
}
```

Problems Tasks Console

<terminated> HelloScala\$ (1) [Scala Application] C:\Progra
5

而这种代码在 Spark 的源码中是非常常见的。

下面看一下 Scala 中的可变参数的用法：

```
def hello(name : String) : String = {
  "Hello" + name
}

def helloScala(){
  println("Hello Scala, this is rocky!")
}

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

def sum2(x : Int)(y : Int) = x + y

def variableParameter(s : String*) = {
  s.foreach(x => println(x))
}

def main(args: Array[String]) {
  // println("Hello Scala!!!")
  // println(hello("scala"))
  // helloScala
  // println(add(1 , 2))
  // println(sum(1 , 2))
  // println(sum2(2)(3))
  variableParameter("I", "LOVE", "Spark")
}
```

Problems Tasks Console

<terminated> HelloScala\$ (1) [Scala Application] C:\Program File

I
LOVE
Spark

接下来看一下默认参数：

```
def helloDefault(name : String = "www.sparkinchina.com") : String = {
    "Hello" + name
}

def main(args: Array[String]) {
    // println("Hello Scala!!!")
    // println(hello("scala"))
    // helloScala
    // println(add(1 , 2))
    // println(sum(1 , 2))
    // println(sum2(2)(3))
    // variableParameter("I", "LOVE", "Spark")
    println(helloDefault())
}
```

Problems Tasks Console

<terminated> HelloScala\$ (1) [Scala Application] C:\Program Files\Java\jdk1.7.0_67\bin\javav
Hellowww.sparkinchina.com

3 , Scala 中的表达式实战

首先看一下 Scala 中优雅的条件的表达式：

```
def main(args: Array[String]) {
    // println("Hello Scala!!!")
    // println(hello("scala"))
    // helloScala
    // println(add(1 , 2))
    // println(sum(1 , 2))
    // println(sum2(2)(3))
    // variableParameter("I", "LOVE", "Spark")
    // println(helloDefault())

    val max = 1
    val result = if(max > 0) 1 else 0
    println(result)
}

}
```

Problems Tasks Console

<terminated> HelloScala\$ (1) [Scala Application] C:\Program
1

接下来看一下 while 循环表达式：

```

def main(args: Array[String]) {
  // println("Hello Scala!!!")
  // println(hello("scala"))
  // helloScala
  // println(add(1, 2))
  // println(sum(1, 2))
  // println(sum2(2)(3))
  // variableParameter("I", "LOVE", "Spark")
  // println(helloDefault())

  // val max = 1
  // val result = if(max > 0) 1 else 0
  // println(result)

  var (n, r) = (10, 0)
  while(n > 0){
    r = r + n
    n = n - 1
  }
  println(r)
}

```

Problems Tasks Console

<terminated> HelloScala\$ (1) [Scala Application] C:\Program File
55

下面看一下 for 条件表达式：

```

for(i <- 1 to 10){
  println(i)
}

```

Problems Tasks Console

<terminated> HelloScala\$ (1) [Scala Application] C:\Pro

1
2
3
4
5
6
7
8
9
10

在 for 中也可以使用 until：


```
for(i <- 1 until 10){  
  println(i)  
}
```

Problems Tasks Console

<terminated> HelloScala\$ (1) [Scala Application]

1
2
3
4
5
6
7
8
9

如果想在条件表达式中筛选出 1 到 10 中所有偶数，可以采用下面的写法：

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

Problems Tasks Console

<terminated> HelloScala\$ (1) [Scala Application] C:\Program Files\Java'

2
4
6
8
10