DT 大数据梦工厂
DATA TECHNOLOGY DREAMWORK

Spark

# Spark 零基础实战

王家林　孔祥瑞　等编著

- 本书完全从零起步，通过代码实战以及Spark源码阅读的方式对函数式编程与面向对象完美结合的Scala语言进行了详细讲解，以此帮助读者快速掌握Scala语言
- 完全的零基础实战的方式对Hadoop和Spark集群安装部署以及Spark在不同集成开发环境中的开发做出了详细的讲解
- 深入浅出的基于大量的实战案例来讲解Spark核心RDD编程并深度解密RDD的密码，且通过实战的方式讲解了TopN等高级算法在Spark 中的实现代码
- 为了让读者彻底了解Spark，本书用了大量的篇章详细解密了Spark的高可用性、内核架构、运行机制等内容

化学工业出版社

# Spark 零基础实战

王家林　孔祥瑞　等编著

化学工业出版社

·北京·

Spark是业界公认的近几年发展最快、最受关注度的一体化多元化的大数据计算技术，可以同时满足不同业务场景和不同数据规模的大数据计算的需要。

本书首先通过代码实战的方式对学习Spark前必须掌握的Scala内容进行讲解并结合Spark源码的阅读来帮助读者快速学习Scala函数式编程与面向对象完美结合的编程艺术，接着对Hadoop和Spark集群安装部署以及Spark在不同集成开发环境的开发实战作出了详细的讲解，然后基于大量的实战案例来讲解Spark核心RDD编程并深度解密RDD的密码，并且通过实战的方式详解了TopN在Spark RDD中的实现，为了让读者彻底了解Spark，本书用了大量的篇幅详细解密了Spark的高可用性、内核架构、运行机制等内容。

Spark零基础实战这本书定位于零基础的学员，也可以作为有一定大数据Hadoop经验的从业者以及对大数据非常感兴趣的学生的第一本Spark入门书籍。

**图书在版编目（CIP）数据**

Spark零基础实战/王家林等编著. —北京：化学工业出版社，2016.10
ISBN 978-7-122-28017-6

Ⅰ.①S… Ⅱ.①王… Ⅲ.①数据处理软件 Ⅳ.①TP274

中国版本图书馆CIP数据核字（2016）第215244号

责任编辑：王淑燕 宋湘玲　　装帧设计：关 飞
责任校对：宋 玮

出版发行：化学工业出版社（北京市东城区青年湖南街13号 邮政编码100011）
印　　装：大厂聚鑫印刷有限责任公司
787mm×1092mm 1/16 印张20 字数503千字　　2016年11月北京第1版第1次印刷

购书咨询：010-64518888（传真：010-64519686） 售后服务：010-64518899
网　　址：http://www.cip.com.cn
凡购买本书，如有缺损质量问题，本社销售中心负责调换。

定　　价：68.00元

# 前言

大数据已经成为公众流行词多年，不管在业界还是在其他领域都紧随时代发展的潮流，人类社会的发展已经进入到大数据时代。我们生活的今天大到互联网公司，小到每一个个体或者每一台移动设备其每天都会产生海量的新数据，那么对于这些海量数据的处理就面临着巨大的考验，而在此过程中为了满足业务需要，各类技术如雨后春笋般出现并得到IT企业的实践应用和发展，就应对海量数据的处理框架而言，于2006年诞生的Hadoop，使业界掀起一股热潮，它改变了企业对数据的存储、处理和分析的过程，加速了大数据的发展，形成了自己的极其火爆的技术生态圈，并受到非常广泛的应用。而Spark在2009年最初来源于伯克利大学的研究性项目，于美国加州大学伯克利分校的AMPLab实验室诞生，2010年实现开源并在2013年成为Apache的基金孵化器项目并在不到一年的时间成为其的顶级项目，在短短几年的时间内获得极速发展并被各大互联网公司应用于实际项目中以实现海量数据的处理，可以毫不夸张地讲Spark是大数据时代发展的必然产物，势必会成为最好的大数据处理框架之一。

根据Stackoverflow调查显示Spark是2016年IT从业者获得薪水最高的技术之一，从事Spark开发的IT人员年薪达到125000美元，从事Scala开发的IT人员年薪同从事Spark的IT人员保持一致的水平，可见Spark已经成为开发人员在大数据领域收入最好的技术之一。了解Spark或者读过Spark源码的人都知道Spark主要是Scala语言开发的，而Scala语言是一门面向对象与函数式编程完美结合的语言。因此本书主要以零基础实战掌握Spark运行机制为导向详细对Scala的语法和重要知识点进行实战讲解，通过源码对Spark的内核架构进行剖析并赋予实战案例来引导读者能够在掌握Scala的同时快速进行Spark的深入学习。

Spark基于RDD（弹性分布式数据集）实现了一体化、多元化的大数据处理体系，是目前最热门最高效的大数据领域的计算平台。Spark框架完美融合了Spark SQL、Spark Streaming、MLLib、GraphX子框架，使得各子框架之间实现数据共享和操作，强大的计算能力和集成化使得Spark在大数据计算领域具有得天独厚的优势，因此国际上很多大型互联网公司均使用Spark实现海量数据的处理，如国内的BAT等，有超过千台节点组成的集群高效快速地处理每日生成的海量数据。

Spark在大数据处理领域的迅猛发展，给了很多互联网公司高效处理海量数据的方案，但是Spark人才的稀缺使得很多公司心有余而力不足，以至于不能将企业的生产力量化提高成了很多企业面临的主要问题，大数据Spark工程师的缺少直接制约了很多公司的转型和发展，在此情况下本书以零基础实战为主导，由基础部分细致地带领初学者从零基础入门直到深入学习Spark。本书主要面向的对象是预从事大数据领域的初学者、高校学生以及有一定

大数据从事经验的工作人员等。

本书以零基础实战 Spark 为主导，首先实战讲解 Scala 基础语法与定义、Scala 面向对象编程、Scala 函数式编程、Scala 类型系统模式匹配、Scala 因式转换以及 Scala 并发编程等，基本包含了 Scala 所有重要内容并且每一部分在实战的同时配合 Scala 在 Spark 源码中的应用带领读者彻底理解 Scala 语言的艺术。其次对 Spark 源码在不同方式下的编译进行演示，对 Hadoop 不同模式的集群搭建、Spark 集群的搭建以及 Spark 在 IDE、IntelliJ IDEA 不同工具下的实战和源码导入均作了细致讲解，相信通过源码的学习和不同工具下对 Spark 程序的开发实战可以帮助读者对 Spark 有一个全面的理解和认识，并能快速投入到实际开发中。然后对 Spark 中最为重要的核心组件之一 RDD（弹性分布式数据集）进行了详细地解析，并介绍 Spark Master HA 的 4 种策略，解密如何通过 ZOOKEEPER 这种企业经常使用的策略来保证 Spark Master HA。本书最后一部分综合讲解了 Spark 内核架构以及实战解析 Spark 在不同模式下的运行原理。希望本书可以引领读者细致高效地学习 Spark 框架，并成为企业渴求的 Spark 高端人才。

参与本书编写的有王家林、孔祥瑞等。本书能顺利出版，离不开化学工业出版社的大力支持与帮助，包括进度把控、技术服务、排版等各个方面，在此表示诚挚地感谢。

在本书阅读过程中，如发现任何问题或有任何疑问，可以加入本书的阅读群（QQ：302306504）提出讨论，会有专人帮忙答疑。同时，该群中也会提供本书所用案例代码。

如果读者想要了解或者学习更多大数据的相关技术，可以关注 DT 大数据梦工厂微信公众号 DT _ Spark 及 QQ 群 437123764，或者扫描下方二维码咨询，也可以通过 YY 客户端登录 68917580 永久频道直接体验。王家林老师的新浪微博是 http://weibo.com/ilovepains/ 欢迎大家在微博上进行互动。

由于时间仓促，书中难免存在不妥之处，请读者谅解，并提出宝贵意见。


**王家林 2016. 8. 13 于北京**

# 目 录

# 第1章 Scala光速入门

Spark 以其极快的发展速度和大规模数据集优秀的处理计算能力而被业界所推崇，很多人对 Spark 优秀的处理能力产生了极强的兴趣，很多公司为了更好地处理大规模数据的业务选择了 Spark 进行开发。要学习 Spark 就需要学习 Scala，因为 Spark 主要是由 Scala 语言编写的，如果想更好地学习 Spark 的内核架构和底层实现，就需要熟练掌握 Scala 编程语言。本章通过 Scala 基础与语法以及 Array、Map 等数据结构的丰富的代码实战带领大家快速掌握 Scala 入门需要的知识。

## 1.1 Scala 基础与语法入门实战

### 1.1.1 Scala 基本数据类型

Scala 包括 8 种常用数据类型：Byte、Char、Short、Int、Long、Float、Double 和 Boolean。基本数据类型取值范围及示例如表 1-1 所示。

**表 1-1 基本数据类型**

| 基本类型 | 取值范围或示例 |
| --- | --- |
| Byte | 范围在－128～127 |
| Char | 范围 U+0000～U+FFFF |
| Short | 范围在－32768～32767 |
| Int | 范围－2147483648～2147483647 |
| Long | 范围－9223372036854775808～9223372036854775807 |
| Float | 单精度浮点数例如：0.0f 或者 0.0F 若没有 f 或者 F 后缀则是 Double 类型 |
| Double | 双精度浮点数例如：0.11 |
| Boolean | 布尔类型表示为 true 和 false |

String 也是 Scala 基本数据类，String 属于 java.lang 包，其余的所有基本类型都是 Scala 的包成员，例如 Int 是 scala.Int。Scala 中基本类型包和 java.lang 包是默认导入的，

可以直接使用。

### 1.1.2 Scala 变量声明

Scala 通过 var 和 val 来声明变量。

【例 1-1】 定义 val 类型变量

val 类型变量定义后不可以重新赋值。val 类型变量相当于 Java 中的 final 修饰的变量。

```
scala> val i = 10
i: Int = 10                                    //Scala根据类型推断,推断出i的类型为Int并把10赋值给i
scala> println(i)
10

scala> i = 11                                  //i定义为val类型,是不可变的。重新给i赋值会报如下错误
<console>:8: error: reassignment to val
       i = 11
         ^
scala> val i,j = 10                            //Scala允许一次定义多个变量
i: Int = 10
j: Int = 10

scala> val a = 1f                              //定义Float类型的数据,同样适用了类型推断
a: Float = 1.0

scala> val i :Int = 10                         //显示指定数据类型
i: Int = 10

scala> val name:String = "zhangsan"   //显示指定定义的字符串类型
name: String = zhangsan
```

【例 1-2】 定义 var 类型变量

var 类型变量确定以后值可以修改。

```
scala> var x = 5                    //定义可变变量x,类型推断出是整数
x: Int = 5

scala> x = 10                       //重新赋值变量x
x: Int = 10

scala> var  sex = "male"      //定义字符串类型变量
sex: String = male

scala> sex = "famale"         //给sex重新赋值
sex: String = famale

scala> val name:String = "zhangsan"       //var类型变量定义时显示指定变量类型
name: String = zhangsan
```

### 1.1.3 算术操作符介绍

Scala 有丰富的内置运算符，Scala 的操作符也是函数，可以通过“对象．运算符（）”

方式使用。下面分别介绍不同类型的运算符，如表 1-2 所示。

表 1-2　算术运算符

| 运算符 | 描述 | 运算符 | 描述 |
| --- | --- | --- | --- |
| + | 两个数相加 | / | 两个数相除 |
| − | 两个数相减 | % | 模运算，整数除法后的余数 |
| * | 两个数相乘 | | |

【例 1-3】　算术运算符实战

Scala 是一种面向函数和面向对象相结合的语言。其算术操作符是函数的另一种展现。如表 1-3 所示。

```
1.  scala> 1 + 2                    //+ 在scala中也是对象的函数，这里可以写成1.+(2),下面- * /
                                     等都和+ 一样
2.
3.  res0: Int = 3
4.  scala> 5- 1                     //两个数相减
5.  res2: Int = 4
6.  scala> 3 * 5                    //两个数相乘
7.  res3: Int = 15
8.  scala> 6 / 3                    //两个数相除
9.  res4: Int = 2
10. scala> 5 % 2                    //求余
11. res5: Int = 1
```

表 1-3　关系运算符

| 运算符 | 描　　述 |
| --- | --- |
| == | 判断两个数的值是否相等，如果是的话那么条件为真，否则为假 |
| ! = | 判断两个数的值是否相等，如果值不相等，则条件为真，否则为假 |
| > | 判断左边数的值是否大于右边数的值，如果是的话那么条件为真，否则为假 |
| < | 判断左边数的值是否小于右边数的值，如果是的话那么条件为真，否则为假 |
| >= | 判断左边数的值是否大于或等于右边数的值，如果是的话那么条件为真，否则为假 |
| <= | 判断左边数的值是否小于或等于右边数的值，如果是的话那么条件为真，否则为假 |

【例 1-4】　关系运算符实战

所有的关系判断返回值都为 Boolean 类型。如表 1-4 所示。

```
1.  scala> 3 == 2                   //判断3是否等于2，返回false
2.
3.  res6: Boolean = false
4.  scala> 3 != 2                   //判断3是否不等于2
5.  res7: Boolean = true
6.  scala> 3 > 1                    //判断3是否大于1
7.  res8: Boolean = true
8.  scala> 3 < 1                    //判断3是否小于1
9.  res9: Boolean = false
```

表 1-4　逻辑运算符

| 运算符 | 描　述 | 示　例 |
| --- | --- | --- |
| &&. | 所谓逻辑与操作。如果两个操作数为非零则条件为真 | (A &&. B)为 false |
| \|\| | 所谓的逻辑或操作。如果任何两个操作数是非零则条件变为真 | (A \|\| B)为 true |
| ! | 所谓逻辑非运算符。使用反转操作数的逻辑状态。如果条件为真,那么逻辑非操作符作出结果为假 | ! (A &&. B)为 true |

**【例 1-5】** 逻辑运算符实战

逻辑运算符返回结果为 Boolean 类型。

```
scala> true && false                    //逻辑运算符的两侧是结果为Boolean类型的表达式
res11: Boolean = false
scala> true && false
res11: Boolean = false
scala> 3 > 1 && 2 != 1                  //如果左侧表达式为假，则不再判断右侧表达式
res13: Boolean = true
scala> 3 > 1 || 2 != 1                  //如果左侧表达式为真，则不再判断右侧表达式
res14: Boolean = true
scala> !(3 > 1)                         //! 否定表达式的结果
res15: Boolean = false
```

位运算符适用于位和位操作，主要包括三种：&.，|，和^，其真值表如表 1-5 所示。

表 1-5　位运算符

| p | q | p &. q | p \| q | p ^ q |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |

**【例 1-6】** 位运算符实战

位运算是指按照计算数据在计算机中的二进制进行 0、1 的按位“与”、“或”等的操作。如表 1-6 所示。

```
scala> 0 & 0                            // 按位与运算
res2: Int = 0
scala> 0 | 0                            // 按位或运算
res3: Int = 0
scala> 0 ^ 0                            // 按位异或运算
res4: Int = 0
scala> 0 & 1
res5: Int = 0
scala> 0 | 1
res6: Int = 1
scala> 0 ^ 1
res7: Int = 1
scala> 1 & 1
res8: Int = 1
scala> 1 | 1
res9: Int = 1
scala> 1 ^ 1
res10: Int = 0
```

表 1-6　赋值运算符

| 运算符 | 描　述 | 示　例 |
|---|---|---|
| = | 简单地将右侧的值赋于左侧 | C=A+B 将分配 A+B 的值到 C |
| += | 先执行加法再赋值 | C+=A 相当于 C=C+A |
| -= | 先执行减法再赋值 | C-=A 相当于 C=C-A |
| *= | 先执行乘法再赋值 | C *=A 相当于 C=C *A |
| /= | 先执行除法再赋值 | C/=A 相当于 C=C/A |
| %= | 先执行模量去余再赋值 | C%=A 相当于 C=C%A |
| <<= | 左移位并赋值 | C<<=2 等同于 C=C<<2 |
| >>= | 向右移位并赋值 | C>>=2 等同于 C=C>>2 |
| &= | 按位与赋值 | C&=2 等同于 C=C&2 |
| ^= | 按位异或赋值 | C^=2 等同于 C=C^2 |
| \|= | 按位或赋值 | C\|=2 等同于 C=C\|2 |

【例 1-7】 赋值运算符实战

```
1.  scala> var a = 3 + 8                    //3 + 8的结果值赋值给a
2.  a: Int = 11
3.  scala> a += 3                           //a加3，Scala中没有a++表达式，该表达式可以写成 a = a+ 3
4.  scala> println(a)                       //打印出当前a的值
5.  14
6.  scala> a -= 5                           //a减5，Scala中没有a- -表达式,该表达式可以写成 a = a - 5
7.  scala> println(a)
8.  9
9.  scala> a *= 2                           //a乘以2，该表达式可以写成a = a * 2
10. scala> println(a)
11. 22
```

## 1.1.4　条件语句

条件语句语法结构的两种方式如下。

方式一：写在同一行。

```
if（布尔表达式） x else y
```

方式二：写在不同行并且有多条语句组成的结构体。

```
if（布尔表达式）{
        ……
} else {
        ……
}
```

Scala 中条件语句是有返回值的。可以将其赋值给某个变量，类似于 Java 中的三目运算符。

【例 1-8】 if 语句实战

if 语句通过判断条件是否为真来执行后面的结构体。

```
1.  scala> val i =  if (3 > 1) 3 else 1
2.  i: Int = 3                              //3赋值给了变量i
```

【例 1-9】 if 语句中变量的应用

定义一个变量 x，并将这个变量应用到 if 语句的条件判断中。

```
1. scala> var x = 10                          //定义一个中间变量
2. x: Int = 10
3. scala> if (x < 20 ){                       //如果x<20为true，则执行大括号
                                               中的语句
4. |     println("This is if test")
5. |}
6. This is if test
```

【例 1-10】 将 if 语句赋值给一个变量

Scala 中 if 语句是有返回值的，可以将 if 语句的返回结果赋值给一个变量。

```
1. scala> val result = if(x == 10 ){ 10 }      //将if语句的结果赋值给result
2. result: AnyVal = 10
3. scala> println(result)
4. 10
```

【例 1-11】 if 与 else 结合使用

将 if-else 的返回结果赋值给 result。

```
1. scala> val x = 10
2. x: Int = 10
3. scala> val result = if(x > 5) x else 5     //if-else类似于java中的三目运算符
4. result: Int = 10
5. scala> println(result)
6. 10
7. scala>
```

Scala 中的条件语句在其他的一些结构中被称为守卫。例如：for 循环中、match 模式匹配中等。

### 1.1.5 循环

Scala 中循环结构有 for、while、do-while 循环等。

#### (1) for 循环

for 循环语句可以重复执行某条语句，直到某个条件得到满足，才退出循环。

for 语法：

```
for（变量<－集合）
    {
     循环体
  }
```

【例 1-12】 利用 for 循环打印出 1 到 10

当满足后面的循环条件时执行 for 循环体，示例代码如下：

```
1. scala> for(i <- 1 to 10) print(i + " ")      // 1 to 10 生成Range(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
2. 1 2 3 4 5 6 7 8 9 10
3.
```

【例 1-13】 利用 for 循环打印出 1 到 9，to 包含尾部，until 不包含尾部