

O'REILLY®

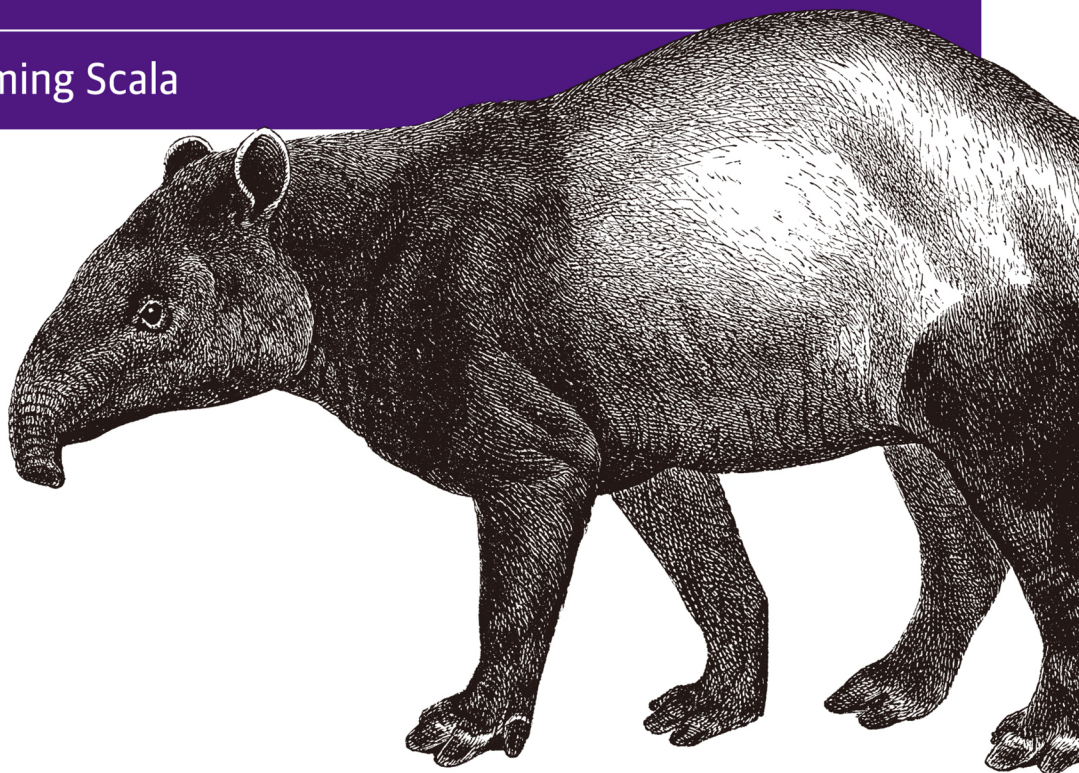
TURING

图灵程序设计丛书

第2版

Scala 程序设计

Programming Scala



[美] Dean Wampler Alex Payne 著
王渊 陈明 译



中国工信出版集团



人民邮电出版社
POSTS & TELECOM PRESS



图灵程序设计丛书

Scala程序设计（第2版）

Programming Scala

[美] Dean Wampler Alex Payne 著
王渊 陈明 译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc. 授权人民邮电出版社出版

人民邮电出版社

北 京

图书在版编目 (C I P) 数据

Scala程序设计 : 第2版 / (美) 万普勒
(Wampler, D.), (美) 佩恩 (Payne, A.) 著 ; 王渊, 陈
明译. — 北京 : 人民邮电出版社, 2016. 3
(图灵程序设计丛书)
ISBN 978-7-115-41681-0

I. ①S… II. ①万… ②佩… ③王… ④陈… III. ①
JAVA语言—程序设计 IV. ①TP312

中国版本图书馆CIP数据核字(2016)第023309号

内 容 提 要

本书通过大量的代码示例, 全面介绍 Scala 这门针对 JVM 的编程语言, 向读者展示了如何高效地利用 Scala 语言及其生态系统, 同时解释了为何 Scala 是开发高扩展性、以数据为中心的应用程序的理想语言。

本书既适合 Scala 初学者入门, 也适合经验丰富的 Scala 开发者参考。

-
- ◆ 著 [美] Dean Wampler Alex Payne
译 王 渊 陈 明
责任编辑 岳新欣
执行编辑 刘 敏
责任印制 杨林杰
 - ◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京 印刷
 - ◆ 开本: 800×1000 1/16
印张: 31.25
字数: 761千字 2016年3月第1版
印数: 1-3 000册 2016年3月北京第1次印刷
著作权合同登记号 图字: 01-2015-6359号
-

定价: 109.00元

读者服务热线: (010)51095186转600 印装质量热线: (010)81055316

反盗版热线: (010)81055315

广告经营许可证: 京崇工商广字第 0021 号

O'Reilly Media, Inc.介绍

O'Reilly Media 通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自 1978 年开始，O'Reilly 一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly 的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly 为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了 Make 杂志，从而成为 DIY 革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly 的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly 现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版、在线服务或者面授课程，每一项 O'Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar 博客有口皆碑。”

——*Wired*

“O'Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——*Business 2.0*

“O'Reilly Conference 是聚集关键思想领袖的绝对典范。”

——*CRN*

“一本 O'Reilly 的书就代表一个有用、有前途、需要学习的主题。”

——*Irish Times*

“Tim 是位特立独行的商人，他不光放眼于最长远、最广阔的视野，并且切实地按照 Yogi Berra 的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去，Tim 似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——*Linux Journal*

目录

序	xv
前言	xvii
第 1 章 零到六十：Scala 简介	1
1.1 为什么选择 Scala	1
1.1.1 富有魅力的 Scala	2
1.1.2 关于 Java 8	3
1.2 安装 Scala	3
1.2.1 使用 SBT	5
1.2.2 执行 Scala 命令行工具	6
1.2.3 在 IDE 中运行 Scala REPL	8
1.3 使用 Scala	8
1.4 并发	17
1.5 本章回顾与下一章提要	27
第 2 章 更简洁，更强大	28
2.1 分号	28
2.2 变量声明	29
2.3 Range	31
2.4 偏函数	32
2.5 方法声明	33
2.5.1 方法默认值和命名参数列表	33
2.5.2 方法具有多个参数列表	34
2.5.3 Future 简介	35

2.5.4	嵌套方法的定义与递归	38
2.6	推断类型信息	40
2.7	保留字	44
2.8	字面量	46
2.8.1	整数字面量	46
2.8.2	浮点数字面量	47
2.8.3	布尔型字面量	48
2.8.4	字符字面量	48
2.8.5	字符串字面量	48
2.8.6	符号字面量	50
2.8.7	函数字面量	50
2.8.8	元组字面量	50
2.9	Option、Some 和 None：避免使用 null	52
2.10	封闭类的继承	53
2.11	用文件和名空间组织代码	54
2.12	导入类型及其成员	55
2.12.1	导入是相对的	56
2.12.2	包对象	57
2.13	抽象类型与参数化类型	57
2.14	本章回顾与下一章提要	59
第 3 章	要点详解	60
3.1	操作符重载？	60
3.2	无参数方法	63
3.3	优先级规则	64
3.4	领域特定语言	65
3.5	Scala 中的 if 语句	66
3.6	Scala 中的 for 推导式	67
3.6.1	for 循环	67
3.6.2	生成器表达式	67
3.6.3	保护式：筛选元素	67
3.6.4	Yielding	68
3.6.5	扩展作用域与值定义	69
3.7	其他循环结构	70
3.7.1	Scala 的 while 循环	71
3.7.2	Scala 中的 do-while 循环	71
3.8	条件操作符	71
3.9	使用 try、catch 和 final 子句	72
3.10	名字调用和值调用	75

3.11 惰性赋值	78
3.12 枚举	79
3.13 可插入字符串	81
3.14 Trait: Scala 语言的接口和“混入”	83
3.15 本章回顾与下一章提要	85
第 4 章 模式匹配	86
4.1 简单匹配	86
4.2 match 中的值、变量和类型	87
4.3 序列的匹配	90
4.4 元组的匹配	94
4.5 case 中的 guard 语句	94
4.6 case 类的匹配	95
4.6.1 unapply 方法	96
4.6.2 unapplySeq 方法	100
4.7 可变参数列表的匹配	101
4.8 正则表达式的匹配	103
4.9 再谈 case 语句的变量绑定	104
4.10 再谈类型匹配	104
4.11 封闭继承层级与全覆盖匹配	105
4.12 模式匹配的其他用法	107
4.13 总结关于模式匹配的评价	111
4.14 本章回顾与下一章提要	111
第 5 章 隐式详解	112
5.1 隐式参数	112
5.2 隐式参数适用的场景	115
5.2.1 执行上下文	115
5.2.2 功能控制	115
5.2.3 限定可用实例	116
5.2.4 隐式证据	120
5.2.5 绕开类型擦除带来的限制	122
5.2.6 改善报错信息	124
5.2.7 虚类型	124
5.2.8 隐式参数遵循的规则	127
5.3 隐式转换	128
5.3.1 构建独有的字符串插入器	132
5.3.2 表达式问题	134
5.4 类型类模式	135

5.5	隐式所导致的技术问题	137
5.6	隐式解析规则	139
5.7	Scala 内置的各种隐式	139
5.8	合理使用隐式	146
5.9	本章回顾与下一章提要	146
第 6 章	Scala 函数式编程	147
6.1	什么是函数式编程	148
6.1.1	数学中的函数	148
6.1.2	不可变变量	149
6.2	Scala 中的函数式编程	151
6.2.1	匿名函数、Lambda 与闭包	152
6.2.2	内部与外部的纯粹性	154
6.3	递归	154
6.4	尾部调用和尾部调用优化	155
6.5	偏应用函数与偏函数	157
6.6	Curry 化与函数的其他转换	158
6.7	函数式编程的数据结构	162
6.7.1	序列	162
6.7.2	映射表	166
6.7.3	集合	168
6.8	遍历、映射、过滤、折叠与归约	168
6.8.1	遍历	169
6.8.2	映射	170
6.8.3	扁平映射	172
6.8.4	过滤	173
6.8.5	折叠与归约	174
6.9	向左遍历与向右遍历	178
6.10	组合器：软件最佳组件抽象	183
6.11	关于复制	186
6.12	本章回顾与下一章提要	188
第 7 章	深入学习 for 推导式	189
7.1	内容回顾：for 推导式组成元素	189
7.2	for 推导式：内部机制	192
7.3	for 推导式的转化规则	194
7.4	Option 以及其他的一些容器类型	197
7.4.1	Option 容器	197
7.4.2	Either: Option 类型的逻辑扩展	200

7.4.3 Try 类型	205
7.4.4 Scalaz 提供的 Validation 类	206
7.5 本章回顾与下一章提要	209
第 8 章 Scala 面向对象编程	210
8.1 类与对象初步	211
8.2 引用与值类型	213
8.3 价值类	214
8.4 父类	217
8.5 Scala 的构造器	217
8.6 类的字段	221
8.6.1 统一访问原则	223
8.6.2 一元方法	224
8.7 验证输入	224
8.8 调用父类构造器（与良好的面向对象设计）	226
8.9 嵌套类型	230
8.10 本章回顾与下一章提要	232
第 9 章 特征	233
9.1 Java 8 中的接口	233
9.2 混入 trait	234
9.3 可堆叠的特征	238
9.4 构造 trait	243
9.5 选择类还是 trait	244
9.6 本章回顾与下一章提要	245
第 10 章 Scala 对象系统 (I)	246
10.1 参数化类型：继承转化	246
10.1.1 Hood 下的函数	247
10.1.2 可变类型的变异	250
10.1.3 Scala 和 Java 中的变异	252
10.2 Scala 的类型层次结构	253
10.3 闲话 Nothing（以及 Null）	254
10.4 Product、case 类和元组	258
10.5 Predef 对象	260
10.5.1 隐式转换	260
10.5.2 类型定义	262
10.5.3 条件检查方法	263
10.5.4 输入输出方法	263
10.5.5 杂项方法	265

10.6	对象的相等	265
10.6.1	equals 方法	266
10.6.2	== 和 != 方法	266
10.6.3	eq 和 ne 方法	267
10.6.4	数组相等和 sameElements 方法	267
10.7	本章回顾与下一章提要	268
第 11 章	Scala 对象系统 (II)	269
11.1	覆写类成员和 trait 成员	269
11.2	尝试覆写 final 声明	272
11.3	覆写抽象方法和具体方法	272
11.4	覆写抽象字段和具体字段	274
11.5	覆写抽象类型	280
11.6	无须区分访问方法和字段：统一访问原则	280
11.7	对象层次结构的线性化算法	282
11.8	本章回顾与下一章提要	287
第 12 章	Scala 集合库	288
12.1	通用、可变、不可变、并发以及并行集合	288
12.1.1	scala.collection 包	289
12.1.2	collection.concurrent 包	290
12.1.3	collection.convert 包	291
12.1.4	collection.generic 包	291
12.1.5	collection.immutable 包	291
12.1.6	scala.collection.mutable 包	292
12.1.7	scala.collection.parallel 包	294
12.2	选择集合	295
12.3	集合库的设计惯例	296
12.3.1	Builder	296
12.3.2	CanBuildFrom	297
12.3.3	Like 特征	298
12.4	值类型的特化	298
12.5	本章回顾与下一章提要	300
第 13 章	可见性规则	301
13.1	默认可见性：公有可见性	301
13.2	可见性关键字	302
13.3	Public 可见性	303
13.4	Protected 可见性	304
13.5	Private 可见性	305

欢迎访问：电子书学习和下载网站 (<https://www.shgis.com>)

文档名称：《Scala程序设计 第2版》Dean Wampler和Alex Payne 著.pdf

请登录 <https://shgis.com/post/4111.html> 下载完整文档。

手机端请扫码查看：

