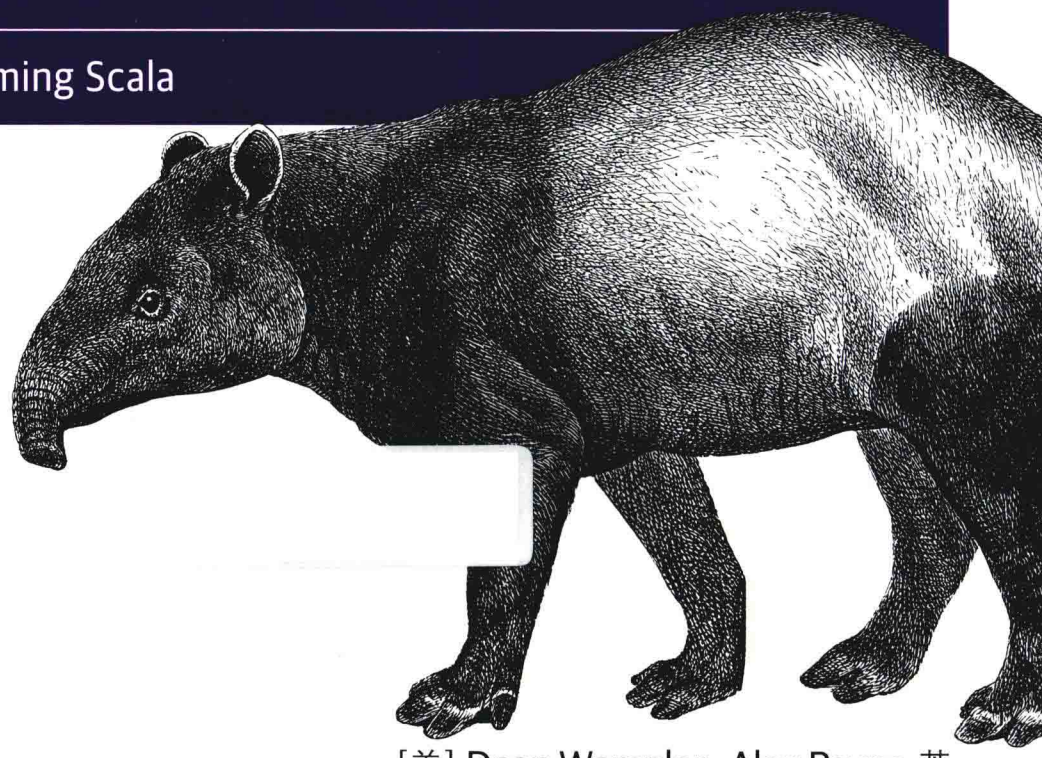
O'REILLY®

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

TURING 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（CIP）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. ①JAVA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
北京鑫正大印刷有限公司印刷

◆ 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：京东工商广字第 8052 号

序

作为一名程序员，我的职业生涯中一直贯穿着这样的主题：寻求更好的抽象和更好的工具来编写更好的软件。经过了这些年，我认为可组合性（composability）是一项比其他特征更重要的特征。如果我们编写的代码具有很好的可组合性，这通常意味着这些代码同样具备软件工程师所看重的其他特征，如正交性（orthogonality）、松耦合性以及高聚合性（high cohesion）。这些都是互通的。

几年前，当我发现 Scala 语言时，它的可组合性便给我带来了很大的震撼。

Martin Odersky 创造 Scala 时，运用了一些简洁的设计方法以及源于面向对象和函数式编程的一些看似简单却很强大的抽象，这使得 Scala 具备高聚合性，而具备了正交性的高度抽象则给这门语言带来可用于软件设计各个方面的可组合性。Scala 是一门真正具备了可扩展性的语言，我们既能使用它编写各种脚本语言，也能使用它实现大规模企业应用和中间件。

Scala 起源于学术界，却已经成长为了一门注重实用性的语言，对于那些真实生产环境中的应用场景，Scala 已经完全准备好了。

《Scala 程序设计》一书的实用性让我感到兴奋。Dean 干得太棒了，除了使用有趣的讨论和示例对 Scala 这门语言进行讲解之外，还将这些内容套到真实世界的应用场景中。这本书是为那些希望能够解决实际问题的程序员所编写的。

几年前，我们还都是面向切面编程委员会的成员时，我认识了 Dean。我很庆幸能够认识他。Dean 拥有一个少见的混合型大脑，他既能思考高深的学术问题，也能想到如何运用实际的方法解决问题。

通过阅读这本书，你将学到如何使用 mixin 和函数组合编写可重用组件；如何运用 Akka 库编写响应式（reactive）应用；如何高效地使用 Scala 提供的一些高级特征，如宏、higher kinded 类型；如何通过 Scala 的丰富、灵活而又富有表现力的语法构造领域特定语言；如何有效地测试你的 Scala 代码；如何通过 Scala 简化大数据问题，等等。

读者们，请好好享受阅读这本书的时光，正如我所做的那样。

——Jonas Bonér

Typesafe 公司联合创始人兼技术总监，2014 年 8 月

前言

《Scala 程序设计》向读者介绍了一门既令人振奋又功能强大的语言，该门语言集合了现代对象模型、函数式编程以及先进类型系统的所有优点，同时又能应用获得产业界大量投资的 Java 虚拟机（JVM）。这本书通过大量的代码示例，向读者全面阐述了如何使用 Scala 迅速编写代码，解释了为什么 Scala 是编写可扩展、分布式、基于组件且支持并发和分布的应用程序的最完美语言。Scala 运行在先进的 JVM 平台之上，通过阅读本书，读者还能了解到 Scala 是如何发挥 JVM 平台优势的。

如果你想了解更多内容，请访问 http://programming-scala.org 或查阅本书目录（http://shop.oreilly.com/product/0636920033073.do）。

欢迎阅读《Scala程序设计（第2版）》

本书第 1 版出版于 2009 年秋，是当时市面上第三本讲述 Scala 的图书，仅仅因为耽误了几个月，未能成为第二本。Scala 当时的官方版本号为 2.7.5，而 2.8.0 版则接近完工。

从那时起，Scala 世界发生了巨大的变化。编写本书时，Scala 的版本号为 2.11.2。为了进一步提升 Scala 语言以及相关工具，Scala 的创建者 Martin Odersky 与基于 actor 模型的并发框架 Akka 的作者 Jonas Bonér 一同创立了 Typesafe（http://typesafe.com）公司。

现在已经出版了很多 Scala 的图书，我们真的有必要再推出第 2 版吗？市场上现存不少适合初学者的 Scala 指南，也出现了一些供高级学习者使用的图书，由 Artima 出版 Odersky 等人撰写的《Scala 编程（第 2 版）》仍被视为 Scala 语言的百科全书。

然而，本书第 2 版非常完整地描述了 Scala 语言及其生态系统，既为初学者成为 Scala 高级用户提供了所需要的指导，又关注了开发人员所面对的实用性问题，这是本书独一无二的地方，也是第 1 版广受欢迎的原因所在。

与 2009 年相比，现在有更多的机构选用了 Scala，大多数 Java 开发者也都听说过这门语言。同时，也出现了一些针对 Scala 语言的持续质疑。Scala 是不是太复杂了？既然 Java 8 已经引入了一些 Scala 特性，那还有必要使用 Scala 吗？

对于真实世界中的种种质疑，我将一一解答。我常常说，Scala 的一切，包括它的不足之处，都让我为之着迷。我希望读者阅读完本书也会有同感。

如何阅读本书

本书论述全面，初级读者无须阅读所有内容便可以使用 Scala 进行编程。本书的前 3 章“零到六十：Scala 简介”“更简洁，更强大”和“要点详解”，简要概括了 Scala 的核心语言特性。第 4 章“模式匹配”和第 5 章“隐式详解”描述了使用 Scala 编程时每天都会用到的两类基本工具，通过对这两类工具的描述将读者引领到更深的领域里。

函数式编程（FP）是一种重要的软件开发方案，假如你之前从未接触过 FP，那么阅读第 6 章能通过 Scala 学习函数式编程。紧接着第 7 章，将说明 Scala 对 `for` 循环的扩展，以及如何使用该扩展提供的简洁语法实现复杂而又符合规范的函数式代码。

之后，第 8 章将介绍 Scala 是如何支持面向对象编程（OOP）的。为了强调 FP 对于解决软件开发问题的重要性，我将 FP 相关章节放到 OOP 章节之前。如果将 Scala 当作“更好的面向对象的 Java”，那会较容易上手，但这样会丢掉这门语言最有力的工具。第 8 章的大多数内容在概念上很容易理解，读者将学到在 Scala 中如何定义类、构造函数等与 Java 相似的概念。

第 9 章将继续探索 Scala 的功能——使用 trait 对行为进行封装。Java 8 受到了 Scala trait 机制的影响，通过对接口进行扩展，新增了部分 trait 功能。对于这部分内容而言，即便是有经验的 Java 程序员也需要花时间理解。

接下来的 4 章，从第 10 章到第 13 章，“Scala 对象系统（I）”“Scala 对象系统（II）”“Scala 集合库”以及“可见性规则”，详细地讲解了 Scala 的对象模型和库类型。由于第 10 章包含了一些必须要尽早掌握的基本知识，因此阅读时要务必仔细。第 11 章讲述了如何正确地实现普通类型层次，你可以在第一遍阅读本书时略过这一章。第 12 章讨论了集合设计问题并提供了合理使用集合的相关信息。再重申一遍，假如你初次接触 Scala，那么请先略过此章，当你试图掌握集合类 API 的详细内容时，再回来学习。最后，第 13 章详细解释了 Scala 是如何对 Java 的 public、protected 以及 private 可见性概念进行细粒度扩展的。可以快速阅览此章。

从第 14 章开始，我们将进入更高级的主题：Scala 复杂类型。这部分内容划分为两章：第 14 章包含了 Scala 新手相对容易理解的概念，而第 15 章则讲述了更高级的内容，你可以选择以后再进行阅读。

类似地，第 16 章“高级函数式编程”讲述的内容中包括了更多高级的理论概念，例如，Monad 和仿函式（Functor）这些起源于范畴论的概念。一般水平的 Scala 开发者在最初并不需要掌握这些内容。

第 17 章“并发工具”有助于开发大型服务的程序员实现并发性的可伸缩性和可扩展性。这一章既论述了 Akka 这一基于 actor 的富并发模型，又讲述了像 `Future` 这类有助于编写异步代码的库类型。

第 18 章“Scala 与大数据”，通常而言，在大数据以及其他以数据为中心的计算领域里，应用 Scala 和函数式编程能够构造杀手级应用。

第 19 章“Scala 动态调用”和第 20 章“Scala 的领域特定语言”是较为高级的专题，探讨了可用于构建富领域特定语言的一些工具。

第 21 章“Scala 工具和库”讨论了一些 IDE 和第三方库。假如你是 Scala 新手，那么请阅读 IDE 和编辑器支持的相关小节，同时阅读关于 Scala 公认的项目构建工具：SBT 的相关小节。本章最后列出了可以引用的库列表。第 22 章“与 Java 的互操作”对于那些需要互用 Java 和 Scala 代码的团队而言很有帮助。

第 23 章“应用程序设计”是为架构师和软件组长而写的。我在这一章分享了自己在应用设计方面的一些观点。传统模式使用了相对较大的 JAR 文件，而这些 JAR 文件又包含了复杂的对象图谱。因此我认为这种模式是一种不良模式，需要进行变更。

最后，第 24 章“元编程：宏与反射”介绍了本书最高级的主题。当然，如果你是初学者，也可以略过这一章。

本书在附录 A 中总结了一些资料，供读者进一步阅读。

本书未涉及的内容

模块化库是 Scala 最新的 2.11 版的一大焦点，它将库文件分解成更小的 JAR 文件，这样一来，在将系统部署到空间受限的环境时（如手机设备），便能很容易移除不需要的代码。除此之外，新版移除了库中一些原本被标示为“过时”（deprecated）的包和类型，还将其他的一些包和类型标示为 deprecated，这通常是因为 Scala 不再维护这些包和类型，而且有更好的第三方的替代品。

因此，我们不会在本书中讨论那些在 2.11 版本中被标示为 deprecated 的包，具体如下。

- `scala.actors`（http://www.scala-lang.org/api/current/scala-actors/#scala.actors.package）
  一套 actor 库。请使用 Akka actor 库。（我们将在 17.3 节对该库进行描述。）
- `scala.collection.script`（http://www.scala-lang.org/api/current/#scala.collection.script.package）
  该库用于编写监控集合以及更新集合相关“脚本”。
- `scala.text`（http://www.scala-lang.org/api/current/#scala.text.package）
  一套用于“格式化打印”（pretty-printing）的库。

下面列举了在 Scala 2.10 中标示为 deprecated 并已从 2.11 版移除的包。

- `scala.util.automata`（http://www.scala-lang.org/api/2.10.4/#scala.util.automata.package）
  使用正则表达式构建确定有限自动机（DFA）。
- `scala.util.grammar`（http://www.scala-lang.org/api/2.10.4/#scala.util.grammar.package）
  属于 parsing 库。

- `scala.util.logging`（http://www.scala-lang.org/api/2.10.4/#scala.util.logging.package）
  推荐使用某一 JVM 平台上活跃的第三方日志库。
- `scala.util.regexp`（http://www.scala-lang.org/api/2.10.4/#scala.util.regexp.package）
  对正则表达式进行句式分析。`scala.util.matching` 包同样支持正则表达式，请使用功能更为强大的 `scala.util.matching` 包。
- .NET 编译器后台
  Scala 团队曾在 .NET 运行的环境之上搭建编译器后台及库。不过由于大家对这次迁移的兴趣不断衰减，因此这项工作已经暂停。

我们不会对 Scala 库中每个包和类型都进行讨论。由于篇幅和其他原因，下面这些包并不会在本书中提及。

- `scala.swing`（http://www.scala-lang.org/api/current/scala-swing/#scala.swing.package）
  对 Java Swing 库进行封装。尽管仍然有人维护该库，但已很少有人使用它。
- `scala.util.continuations`（http://www.scala-lang.org/files/archive/api/current/scala-continuations-library/#scala.util.continuations.package）
  编译器插件，用于生成连续传递格式（continuation-passing style，CPS）的代码。这是一个特殊的工具，目前很少有人使用它。
- `App`（http://www.scala-lang.org/api/current/#scala.App）和 `DelayedInit`（http://www.scala-lang.org/api/current/#scala.DelayedInit）特征
  使用这两个类型能很方便地实现 `main` 类型（入口类型），它们也是 Java 类中 `static main` 方法的同义词。不过由于它们有时候会导致奇怪的行为，因此我并不推荐使用它们。我会使用通用的、符合规范的 Scala 方法编写 `main` 方法。
- `scala.ref`（http://www.scala-lang.org/api/current/#scala.ref.package）
  对某些 Java 类型进行了封装，如 `WeakReference`，这是 `java.lang.ref.WeakReference` 的封装类。
- `scala.runtime`（http://www.scala-lang.org/api/current/#scala.runtime.package）
  用于实现类库的类型。
- `scala.util.hashing`（http://www.scala-lang.org/api/current/#scala.util.hashing.package）
  提供了多种散列算法。

欢迎阅读《Scala程序设计（第1版）》

一门编程语言能够流行起来是有一定原因的。有时候，某一平台的程序员会青睐于某一特定语言或平台提供商所建议的语言。大多数 Mac OS 开发者习惯使用 Objective-C，大多数 Windows 平台开发者使用 C++ 和 .NET 语言，而嵌入式系统开发者则使用 C 和 C++。

有时，语言能流行起来归功于其技术上的优势，这一优势能够使其变得时尚、让人着迷。C++、Java 和 Ruby 便曾引起程序员的狂热崇拜。

有时，语言会因为适应时代的需要而流行起来。Java 最初被视为一门能够用于编写基于浏览器的富客户端应用的完美语言。当面向对象编程变得主流时，Smalltalk 抓住了这一机遇。

现在，并发、异构型、永不停止的服务以及不断缩短的开发计划，使得业内对函数式编程越来越感兴趣。似乎面向对象编程的统治地位即将结束，而混合式编程范式将流行起来，甚至会变得不可或缺。

如今我们构建的应用大多是可靠、高性能、高并发的互联网或企业级应用程序，我们也希望会有一门通用编程语言适应这一要求，Scala 能将我们从其他语言的阵营中吸引过来，便是因为它具备了许多最理想的特性。

Scala 是一门多范式语言，同时支持面向对象和函数式编程。Scala 具有可扩展性，从小脚本到基于组件的大规模应用程序，Scala 均可胜任。Scala 是深奥的，它从全世界的计算机科学系中吸收了先进的思想。Scala 又是实用的，它的创建者 Martin Odesky 参与了多年的 Java 开发，能理解专业开发人员的需求。

Scala 简洁、优雅而又富有表现力的语法，以及提供的众多工具让我们为之着迷。本书力图阐明为什么这些特性会使 Scala 引人注目、不可或缺。

假如你是一位有经验的开发者，希望能快速全面地了解 Scala，那么这本书很适合你。你也许正在思考是否改用 Scala 或将其作为另一门补充语言；抑或你已经决定使用 Scala，需要学习并很好地掌握 Scala 的特性。无论是哪种情况，我们都希望能以一种平易近人的方式阐明这门强大的语言。

我们假设你已经很好地掌握了面向对象的编程，但并没有接触过函数式编程。我们认为你熟悉一门或多门其他的语言。我们对比了 Java、C#、Ruby 等语言的特性，如果你熟悉任何一种语言，会了解 Scala 中的相似特性，以及一些全新特性。

无论你是否具有面向对象或函数式编程的背景，都会了解到 Scala 如何优雅地融合这两种编范式，展示了它们的互补性。基于众多示例，你还能明白针对不同的设计问题如何以及何时应用 OOP 和 FP 技术。

最后，我们希望你也会为 Scala 着迷。即便 Scala 未能成为你日常使用的语言，无论你使用什么语言，我们也希望你能从 Scala 中洞察到些许知识。

排版约定

本书使用以下排版约定。

- 楷体
  表示新术语。

- 等宽字体（`constant width`）
  表示程序片段，以及正文中出现的变量、函数名、数据库、数据类型、环境变量、语句和关键字等。
- 加粗等宽字体（**`constant width bold`**）
  表示应该由用户输入的命令或其他文本。
- 倾斜的等宽字体（*`constant width italic`*）
  表示应该由用户输入的值或根据上下文决定的值替换的文本。

该图标表示提示或建议。

该图标表示一般注记。

该图标表示警告或警示。

使用代码示例

本书就是要帮读者解决实际问题的。也许你需要在自己的程序或文档中用到本书中的代码。除非大段大段地使用，否则不必与我们联系取得授权。因此，使用本书中的几段代码写成一个程序不用向我们申请许可。但是，销售或者分发 O'Reilly 图书随附的代码光盘则必须事先获得授权。引用书中的代码来回答问题也无需我们授权。将大段的示例代码整合到你自己的产品文档中则必须经过许可。

使用我们的代码时，希望你能标明它的出处。出处一般要包含书名、作者、出版商和书号，例如："*Programming Scala, Second Edition* by Dean Wampler and Alex Payne. Copyright 2015 Dean Wampler and Alex Payne, 978-1-491-94985-6."

如果还有其他使用代码的情形需要与我们沟通，可以随时通过 permissions@oreilly.com 与我们联系。

获得示例代码

读者可以从 GitHub（https://github.com/deanwampler/prog-scala-2nd-ed-code-examples）下载代码示例，并把下载后的文件解压到指定位置。请阅读随示例发布的 README 文件，了解如何构建和使用这些示例。（第 1 章会对相关指令进行归纳说明。）

一些示例文件可以作为脚本，使用 scala 命令运行，而另外一些则必须编译成 class 文件，还有一些文件本身就包含了故意植入的错误，无法通过编译。为了表明文件类型，我采用了某种特定的文件命名方式。实际上，在学习 Scala 的过程中，你也能从文件内容中发现文件类型。在大多数情况下，本书示例文件遵循下列命名规范。

- *.scala
  这是 Scala 文件的标准文件扩展名，不过你无法从该扩展名中分辨该文件是必须使用 scalac 进行编译的源文件，还是可以直接使用 scala 运行的脚本文件，或者是本书特意植入了错误的无效代码文件。因此，本书示例代码中使用了 .scala 扩展名的文件必须单独经过编译才能使用，编译过程与编译 Java 代码相似。

- *.sc
  以 .sc 后缀结尾的文件可以作为脚本文件，使用 scala 命令运行。例如：scala foo.sc 命令会执行 foo.sc 脚本。你还可以在解释模式下启动 scala，并通过 :load 命令加载任意脚本文件。请注意，使用 .sc 对脚本进行命名并不是 Scala 社区的命名标准，不过由于 SBT 构建项目时会忽略 .sc 文件，因此我们在此处用其对脚本进行命名。与此同时，IDE 提供的 worksheet 新功能将 worksheet 文件命名为 .sc 文件，我们会在第 1 章中讨论这一功能。所以使用 .sc 对脚本命名是一个可以接受的、便利的命名方法。再次申明，通常情况下我们使用 .scala 扩展名为脚本文件和代码命名。

- *.scalaX 及 *.scX
  某些示例文件中特意植入了某些导致编译异常的错误。为了避免导致编译出错，这些文件使用了 .scalaX 或 .scX 扩展名。.scalaX 表示代码文件，而 .scX 则表示脚本文件。再次重申，.scalaX 和 .scX 扩展名并不是业内使用的扩展名。这些文件中也嵌入了一些注释，用于说明这些文件无法执行的原因。

Safari® Books Online

Safari Books Online（http://www.safaribooksonline.com）是应需而变的数字图书馆。它同时以图书和视频的形式出版世界顶级技术和商务作家的专业作品。

Safari Books Online 是技术专家、软件开发人员、Web 设计师、商务人士和创意人士开展调研、解决问题、学习和认证培训的第一手资料。

对于组织团体、政府机构和个人，Safari Books Online 提供各种产品组合和灵活的定价策略。用户可通过一个功能完备的数据库检索系统访问 O'Reilly Media、Prentice

Hall Professional、Addison-Wesley Professional、Microsoft Press、Sams、Que、Peachpit Press、Focal Press、Cisco Press、John Wiley & Sons、Syngress、Morgan Kaufmann、IBM Redbooks、Packt、Adobe Press、FT Press、Apress、Manning、New Riders、McGraw-Hill、Jones & Bartlett、Course Technology 以及其他几十家出版社的上千种图书、培训视频和正式出版之前的书稿。要了解 Safari Books Online 的更多信息，我们网上见。

联系我们

请把对本书的评价和发现的问题发给出版社。

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）
奥莱利技术咨询（北京）有限公司

O'Reilly 的每一本书都有专属网页，你可以在那儿找到本书的相关信息，包括勘误表、示例代码以及其他信息。本书的网站地址是：

http://shop.oreilly.com/product/0636920033073.do

对于本书的评论和技术性问题，请发送电子邮件到：

bookquestions@oreilly.com

要了解更多 O'Reilly 图书、培训课程、会议和新闻的信息，请访问以下网站：

http://www.oreilly.com

我们在 Facebook 的地址如下：http://facebook.com/oreilly

请关注我们的 Twitter 动态：http://twitter.com/oreillymedia

我们的 YouTube 视频地址如下：http://www.youtube.com/oreillymedia

第2版致谢

我，Dean Wampler，在编写这一版书的过程中得到了 Typesafe 公司许多同事的指导和反馈。除此之外，一些审核了早期版本的人也给我提供了有价值的反馈，非常感谢他们。我要特别感谢 Ramnivas Laddad、Kevin Kilroy、Lutz Huehnken 和 Thomas Lockney，他们帮我审阅了本书的底稿。感谢我的老同事、老朋友 Jonas Bonér，感谢他为本书作序。

特别感谢 Ann，允许我将大量的个人时间花在本书的编写工作中，感谢你长期以来对我的包容，我爱你！

第1版致谢

我们在编写这本书的时候，一些朋友阅读了早期版本，并对本书的内容提出了大量好的意见，我们在此对这些朋友表示感谢。特别要感谢 Steve Jensen、Ramnivas Laddad、Marcel Molina、Bill Venners 和 Jonas Bonér，他们给本书提供了大量的反馈。

我们在 Safari 发布初稿以及在 http://programmingscala.com 网站上提供在线版本时，收到了大量的反馈。在此列举了提供反馈的读者（排名不分先后），对他们表示感谢。他们 是 Iulian Dragos、Nikolaj Lindberg、Matt Hellige、David Vydra、Ricky Clarkson、Alex Cruise、Josh Cronemeyer、Tyler Jennings、Alan Supynuk、Tony Hillerson、Roger Vaughn、Arbi Sookazian、Bruce Leidl、Daniel Sobral、Eder Andres Avila、Marek Kubica、Henrik Huttunen、Bhaskar Maddala、Ged Byrne、Derek Mahar、Geoffrey Wiseman、Peter Rawsthorne、Geoffrey Wiseman、Joe Bowbeer、Alexander Battisti、Rob Dickens、Tim MacEachern、Jason Harris、Steven Grady、Bob Follek、Ariel Ortiz、Parth Malwankar、Reid Hochstedler、Jason Zaugg、Jon Hanson、Mario Gleichmann、David Gates、Zef Hemel、Michael Yee、Marius Kreis、Martin Süsskraut、Javier Vegas、Tobias Hauth、Francesco Bochicchio、Stephen Duncan Jr.、Patrik Dudits、Jan Niehusmann、Bill Burdick、David Holbrook、Shalom Deitch、Jesper Nordenberg、Esa Laine、Gleb Frank、Simon Andersson、Patrik Dudits、Chris Lewis、Julian Howarth、Dirk Kuzemczak、Henri Gerrits、John Heintz、Stuart Roebuck 以及 Jungho Kim。还有很多读者也为本书提供了反馈，不过我们只知道他们的用户名，在此我们要向 Zack、JoshG、ewilligers、abcoates、brad、teto、pjcj、mkleint、dandoyon、Arek、rue、acangiano、vkelman、bryanl、Jeff、mbaxter、pjb3、kxen、hipertracker、ctran、Ram R.、cody、Nolan、Joshua、Ajay、Joe 表示感谢。除此之外，我们还要向不知道名字的贡献者表示感谢。如果名单中漏掉了谁，我们在此表示歉意！

Mike Loukides 是我们的编辑，他深知应该如何以温和的方式催促进度。他在我们写书的过程中为我提供了巨大的帮助。O'Reilly 出版社的其他员工也总能回答我们的问题，并帮助我们继续工作。

感谢 Jonas Bonér 为本书作序。Jonas 是我在“面向切面编程”（AOP，Aspect-Oriented Programming）委员会的老朋友、老同事。这些年来，他为 Java 社区做了很多前沿性的工作。现在他将精力投入到了改进 Scala 和发展 Scala 社区的工作中。

Bill Venners 很友善地评论了本书，我们将其置于封底处。他与 Martin Odersky、Lex Spoon 一起编写了第一本 Scala 图书《Scala 编程》。对于 Scala 开发人员而言，他是不可或缺的人物。Bill 同时还创造了令人惊叹的 ScalaTest 库。

除了 Jonas 和 Bill 之外，我们还从世界各地的开发人员处学到了很多，他们是 Debasish Ghosh、James Iry、Daniel Spiewak、David Pollack、Paul Snively、Ola Bini、Daniel Sobral、Josh Suereth、Robey Pointer、Nathan Hamblen、Jorge Ortiz，以及一些通过发表博文、参与论坛讨论以及私人会话给我提供帮助的朋友。

Dean 要向 Object Mentor 公司的同事表示感谢，他同时也要感谢许多客户端开发人员。他们激发了许多编程语言、软件设计以及业内实际问题的讨论。芝加哥地区 Scala 狂热者团

体（Chicago Area Scala Enthusiasts，CASE）也为本书提供了很有价值的反馈及激励。

Alex 要向他在 Twitter 的同事表示感谢，他们对 Alex 的工作给予了鼓励，并在实际工作中很好地演示了 Scala 语言的能力。Alex 同时要对湾区 Scala 爱好者（Bay Area Scala Enthusiasts，BASE）表示感谢，他们的激情以及这个社团本身为他提供了帮助。

我们要特别感谢 Martin Odersky 和他的团队，感谢他们创造了 Scala。

目录