The Pragmatic Programmers

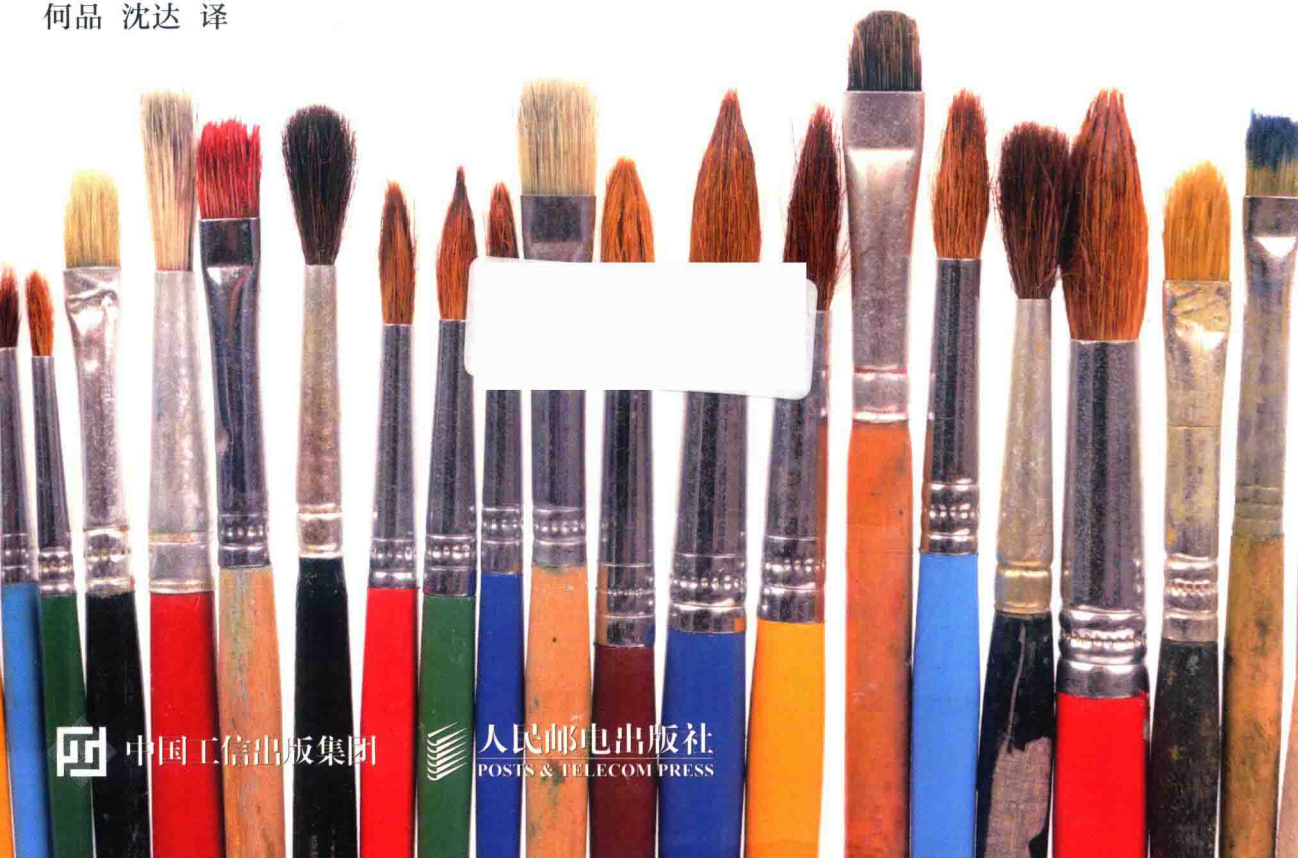
异步图书 www.epubit.com

Scala实用指南

Pragmatic Scala

Create Expressive, Concise, and Scalable Applications

[美] 文卡特·苏帕拉马尼亚姆（Venkat Subramaniam） 著
何品 沈达 译



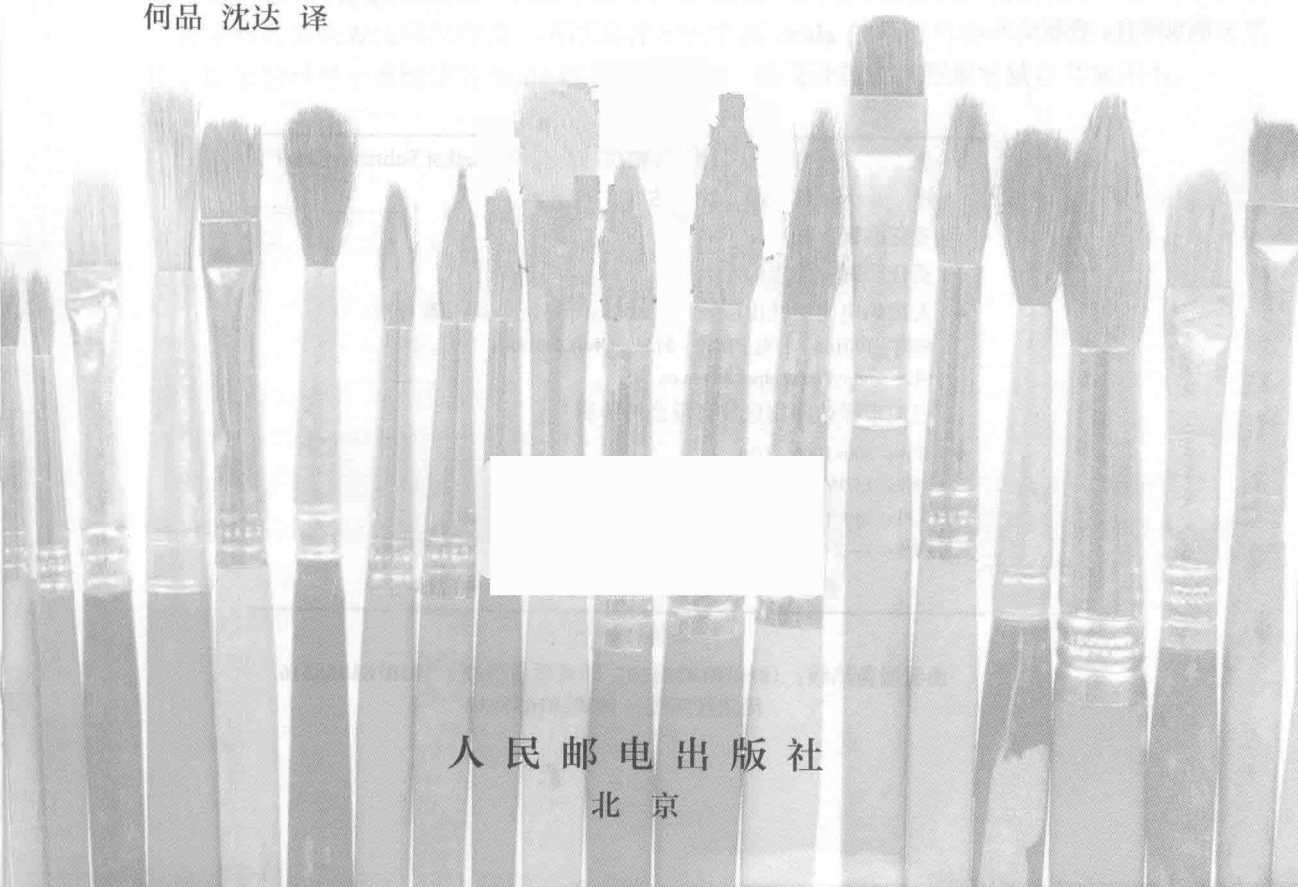
中国工信出版集团
人民邮电出版社 POSTS & TELECOM PRESS

Scala实用指南

Pragmatic Scala

Create Expressive, Concise, and Scalable Applications

[美] 文卡特·苏帕拉马尼亚姆（Venkat Subramaniam） 著
何品 沈达 译



人民邮电出版社
北京

图书在版编目（CIP）数据

Scala实用指南 / (美) 文卡特·苏帕拉马尼亚姆 (Venkat Subramaniam) 著 ; 何品, 沈达译. -- 北京 : 人民邮电出版社, 2018.7
书名原文: Pragmatic Scala: Create Expressive, Concise, and Scalable Applications
ISBN 978-7-115-48356-0

Ⅰ. ①S… Ⅱ. ①文… ②何… ③沈… Ⅲ. ①JAVA语言－程序设计－指南 Ⅳ. ①TP312.8-62

中国版本图书馆CIP数据核字(2018)第086260号

版权声明

◆ 著 [美] 文卡特·苏帕拉马尼亚姆（Venkat Subramaniam）
译 何 品 沈 达
责任编辑 杨海玲
责任印制 焦志炜

◆ 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
三河市祥达印刷包装有限公司印刷

◆ 开本：800×1000 1/16
印张：15.75
字数：330 千字 2018 年 7 月第 1 版
印数：1－2 400 册 2018 年 7 月河北第 1 次印刷

著作权合同登记号 图字：01-2016-1193 号

定价：69.00 元

读者服务热线：(010)81055410 印装质量热线：(010)81055316
反盗版热线：(010)81055315

内容提要

本书是为想要快速学习或者正在学习 Scala 编程语言的 Java 开发者写的，循序渐进地介绍了 Scala 编程语言的多个方面。

本书共分为 4 个部分：第一部分详细介绍 Scala 的一些基础知识，并和 Java 中的相关概念进行了参照，方便读者快速上手 Scala；第二部分进一步介绍 Scala 的一些中级知识，以及与 Java 的一些差异点，方便读者编写出更简洁的代码；第三部分介绍在 Scala 中如何进行并发编程，并务实地介绍 Akka 套件；第四部分通过实战练习对前面的知识进行综合应用，并系统地介绍如何与 Java 进行互操作。此外，附录部分还包括一些额外指引。

本书的目标读者是对 JVM 平台上的语言以及函数式编程感兴趣的程序员。阅读本书不需要读者熟悉 Scala 编程语言，但需要读者具备 Java、面向对象编程的背景知识。因为本书以一种非常务实的方式组织内容，所以读者无法学到 Scala 的所有内容，但是足以应付日常工作，如果想要更全面地学习 Scala 以及其背后的一些设计理念，则最好辅以其他图书。

对本书的赞誉

为 Java 程序员提供了节奏明朗、易于阅读和实用的 Scala 指南，涵盖了这一强大的多范式编程语言的多个重要方面，确保读者可以快速上手 Scala，并变得富有生产力。

——Ramnivas Laddad，*AspectJ in Action* 一书的作者，演说家和咨询师

在本书中，作者提供了坚实的基础，以帮助你在一本完整而简洁的书中学习 Scala，你可以（也应该）从头到尾地阅读。书中探讨了 Scala 中所有你需要熟悉的最重要主题，从 REPL 开始介绍，然后讲到用 Scala 进行函数式编程、使用 Actor 处理并发以及与 Java 的互操作能力。你一定会迫不及待地打开自己的编辑器或者 IDE，来探索本书中众多引人入胜的有趣例子！

——Scott Leberknight，Fortitude 科技公司软件架构师

在充满了 Twitter、博客和小视频的世界里，长篇大论依然还有一席之地。它给了老师足够的时间来引入具备挑战性且复杂的话题。而且，平心而论，Scala 本身就是一个具备挑战性且复杂的话题。请借此机会，让 Venkat 带你熟悉 Scala 编程语言、函数式编程、并发、测试策略以及更多的主题吧。

——Brian Sletten，Bosatsu 咨询公司总裁

我为我所有的 Scala 课程都推荐本书有两个原因：它轻松易读地涵盖了所有的 Scala 基础知识，并循序渐进地讨论了一些高级特性。对于任何学习 Scala 的人来说，这都是需要的一本书。

——Daniel Hinojosa，程序员、讲师、演说家、*Testing in Scala* 一书的作者

我最喜欢 Venkat 的一点是，他可以通过对话的风格来介绍复杂的概念或者未知的话题，从读者熟悉的概念开始，循序渐进。这本书也不例外。我强烈推荐给任何想要学习 Scala，尤其是有 Java 背景的读者。

——Ian Roughley，nToggle 公司工程总监

序

接到同事沈达请我为本书作序的邀请时很惊喜，惊的是我近半年没使用 Scala 进行编程了，担心自己已经丢失了对最新 Scala 特性的跟踪，喜的是速读本书后我发现，本书竟是如此熟悉和容易上手，完全刷新了我对 Scala 各个特性的记忆，并使我有了使用 Scala 写一个小工具的冲动。

Scala 是一门极简且高效的程序设计语言，也是一门复杂的语言，更是一门千人千面的语言。使用 Scala 可以进行面向对象的声明式编程，也可以进行函数式编程；可以进行业务代码的编制，也可以进行元程序的编制（定义程序的程序）；可以开发大规模的服务应用，亦可进行类似 shell 的脚本编程；可以使用共享变量的并发编程模式，当然也可以采用基于 Actor 的消息机制的高并发编程模式。一个新人进入 Scala 的世界，学习和训练的路径有很多，而本书更多的是将语言最好的实践经验总结出来并呈现给读者，让读者能够高效简洁地使用 Scala 语言来实现复杂的功能。

本书对 Scala 各种特性进行了循序渐进的讲解，从 Java 引入（有 Java 基础的程序员会更容易入手），到使用 Scala 进行面向对象编程，进而介绍函数式编程的概念，随后讲解 Scala 最擅长的并发领域编程使用，从不可变性引出基于消息的 Actor 编程模型，最后通过 Scala 的实战讲解了如何编写单元测试。书中穿插了丰富的示例，令我惊讶的是某些部分使用了形象的图形去解释程序结构，非常通俗易懂。

早些年我在翻译 *Functional Programming in Scala* 一书的过程中，体会到函数式编程的强大，也体会到函数式编程的复杂，Scala 标准库（如 Collection）的编写过程践行了函数式编程的大量核心抽象，读者如果感兴趣可以自行查看。本书使用通俗方式和示例介绍了 Scala 函数式编程方面最实用的实现（如高阶函数、柯里化等），这些特性都对编写简洁、高效的程序十分有帮助，相信很多读者在日常工作中都会用到。

最后，Scala 程序员是幸福的，因为 Scala 给了我们去编写优雅的代码的能力；Scala 程序员也是辛苦的，因为 Scala 的学习曲线是有的，而且如何权衡和使用语言强大丰富的特性需要更多的实践和思考。

曹宝，挖财大数据负责人

译者序一

我第一次接触 Scala，还是在 2012 年，因为项目中用到了 Play 框架，好奇心驱使我要看一看源代码，我“不自量力”地打开一看才发自己完全看不懂。这种挫败感在今日依然记忆犹新。自此开始了我学习 Scala 并进一步了解 Akka 的过程，毕竟，我的初衷只是为了看懂 Play 的源代码而已。

我个人学习 Scala 的路线比较曲折。为了学习 Scala，我又相继学习了 Clojure、Haskell 和 Elixir 等编程语言，虽然这些编程语言我都不算特别深入，但是的确对我学习 Scala 有莫大的帮助。因为 Scala 是一门多范式语言，并且非常灵活，其中的知识点也异常多，在看了多本相关图书，并做了不少动手练习之后，我才有点儿“初窥门径”的感觉。

在我熟悉了 Scala，学习了 Akka，并将它们应用到生产实践中之后，我才发现，实际上，很多“费脑”的代码在我们的日常业务开发中几乎不会用到，只是设计库的话可能会用得多一点，而这些认知，我当年并没有，以至于学习和练习了太多“无用”的技能，并极大地推迟了我体味 Scala 的“乐趣”的时间。随着编程经验的增多，以及在北京参加 Scala 线下聚会的讨论，我觉得，Scala 不是难，而是很难，难在缺乏一本浅近易学、循序渐进的图书。社区有时候弥漫的风气会让你觉得：代码写得太平实，就不能表现出真正的实力。因此，无论国内还是国外，大家热衷分享的都是一些第一眼看过去不知所云，第二眼看过去竟会让你不知所措的代码片段。而我要说的是，这并非日常，也并不值得推崇。

Scala 是一门简洁的高级编程语言，同时结合了面向对象编程（OOP）和函数式编程（FP）两种编程范式，Scala 强大的静态类型系统和编译器，让我们可以在编写高性能的复杂应用程序时，提前避免错误的发生，而 JVM、JavaScript 以及 Native 的运行时又让我们可以“一次投入，多平台受益”。得益于 Scala 和 Java 的良好互操作性，我们可以方便自然地使用和编写用于 JVM 生态的库。随着 Java 8 以及 Scala 2.12 的发布，这样的便捷性已得到了进一步的增强。

我本打算写一系列叫作“Scala 快车道”的书，让更多的人能够感受到使用 Scala 编程的高效和快乐。不过已经有了这本优秀、务实的入门图书，所以我就毛遂自荐来翻译了。我要特别感谢沈达，在翻译本书的过程中他比我付出得更多，并极大地提高了译稿的质量，当然

还要感谢陈涛、张江锞、林炜翔、宋坤和周逸之等对本书进行的审阅，他们都怀着极大的热情帮助我们进一步提高了本书的质量。

我们把书中的代码都放在了 GitHub 上（https://github.com/ReactivePlatform/Pragmatic-Scala），这样方便大家下载和使用，默认使用的是原书文件夹的方式，而且还有一个名为 sbt 的分支，以方便大家在 IDE 中直接使用。我希望本书可以方便大家快速入门，并在项目中实践起来，同时适度地利用 Scala 的自由性和灵活性，编写简洁、平实和富有表现力的代码，让 Scala 更容易在团队之间交流，让更多人受益于这种简洁和表现力。

当然，我最应该感谢的是我的爱人和女儿们，感谢她们的体谅和支持，她们是我一切动力的来源。

何品

2018 年 3 月于杭州

译者序二

我是在开始学习 Java 的同时开始接触 Scala 的，在此之前饶有兴致地学过 Scheme，也看过几章《Haskell 趣学指南》，因此对 Scala 中的一些函数式编程的概念并不陌生。我喜欢 Scheme 那种简洁之美，但是很遗憾，使用 Scheme 构建应用程序往往缺砖少瓦，困难重重。而 Haskell 给人一种繁复艰深的感觉，阅读和编写 Haskell 代码的心智负担比较大。Scala 是一门理想的语言，既满足了编程语言爱好者不灭的好奇心，又恰到好处地弥补了 Java 语言所缺失的简洁和表达力。得益于与 Java 良好的互操作，使用 Scala 可以站在 Java 庞大的生态之上，迅速构建出应用程序。

Scala 的美在于精巧的内核，Scala 的丑陋在于复杂的实现。作为程序员，我们不可能只品尝精巧的美而忽视复杂的丑陋。本书的长处在于克制，恰到好处地引导 Java 程序员进入 Scala 的世界，也指明了深入学习的路径。对已经熟悉 Scala 的程序员来说，本书也可以作为编写易读 Scala 代码的指南。Scala 是多范式的，从实际工作的角度，我个人比较推崇编写贴近 Java 风格的 Scala 代码，并适度地利用 Scala 的语言特性简化代码，我认为这也是本书一以贯之的主题。

因为 Java 语言表现力有限，所以我们需要使用各种设计模式提高代码的抽象能力，固化编码逻辑。Scala 这门语言在设计之初就借鉴了大量现存的语言特性，并吸取了许多设计模式中的精华，因此表现力非常强大。就我个人所了解的，Spark Catalyst 源代码中利用抽象语法树的模式匹配做执行计划的优化，直观明了，大大降低了 SQL 执行计划优化器开发的门槛。很难想象，使用 C 或者 C++，如何才能够编写出易于阅读、易于维护的等价实现。

Scala 太灵活了，在学习的过程中难以避免会遇到不少艰深的小技巧，也会遇到各种陷阱。因此，一方面我们编码需要克制，另一方面我们需要加深自己对 JVM 上代码运行机制的理解。王宏江的博客是不可多得的学习资料，能够帮助我们拨开语言特性的迷雾，直击代码运行的本质。本书与其如出一辙，也有不少深入 JVM 字节码的分析，模仿这种分析方式，结合 GitHub 上的 Scala 标准库源代码，我们能够提升自己诊断问题的能力，加深对这门语言的理解。

本书诚如其名——实用。在内容的编排上，本书除了对语言本身的提炼，也同时介绍了 Akka 和单元测试，这对工程实践来说有极大的帮助。对还没有参与过真正工程开发的读者来

说，掌握单元测试是必要的。在阅读大型开源项目的时候，从单元测试入手，可以窥得项目的设计轮廓和 API 完整的使用方法。越是优秀的开源项目，其单元测试越是完整、易读。在翻译本书的时候，个人还没有接触过 Akka，审阅合译者翻译的本书第 13 章之后，我理解了 Actor 模型中隔离的可变性。在最近的工作中，这些知识和 Akka 的文档，帮助我发现了一个使用 Akka 的开源软件中对 IO 操作和 Actor 模型误用而导致的性能问题。

Scala 官方也提供了 Gitter 的中文聊天室，贴代码比较方便，任何 Scala 相关的问题都可以在聊天室交流。我（@sadhen）和何品（@hepin1989）都在聊天室中。

最后，感谢在翻译过程中挖财诸位同事在工作上的帮助，也非常感谢我的领导曹宝开明的管理风格和一贯以来对技术好奇心和驱动力的鼓励。当然，也非常感谢合作译者何品大哥，何品大哥对技术的执着和热情、在开源社区的参与度、技术深度和流畅严谨的译笔，都深深地感染着我鼓励着我。

沈达

2018 年 3 月于杭州城西

致谢

当我签约写这本书的时候，我对要面对的挑战知之甚少。由于颈部受伤，日常的例行工作都变得难以为继。在花了好几个月的时间恢复健康之后，我决定退出本书的编写工作。Pragmatic Programmers 出版社没有以出版方的身份作出回应，而是以朋友和家人的身份来帮助我。我以前找的是出版社，现在我知道，我找到了真正的朋友。感谢 Susannah Pfalzer、Dave Thomas、Andy Hunt 以及所有其他帮助本书出版的团队成员。

我由衷地感谢本书的技术审稿人。感谢 Scott Leberknight——他每次审我的书，我都收获颇丰。感谢 Daniel Hinojosa、Rahul Kavale、Anand Krishnan、Ted Neward、Rebecca Parsons、Vahid Pazirandeh 和 Ian Roughley 在本书出版过程中付出的宝贵时间和投入——我真心地感谢你们所做的一切。本书中任何的纰漏都是我的。

感谢本书还在预览状态时就购买本书的每个人。感谢 David Dieulivol 和 Chris Searle 提交勘误。

Jackie Carter 的鼓励、支持、意见和建议使我获益匪浅。和她互动绝对轻松愉悦，并且极具指导力量。感谢你 Jackie，感谢你所做的一切。你使编写本书的过程非常愉快。

如果没有我的妻子 Kavitha 以及儿子 Karthik 和 Krupa 的支持，我就不能完成这一切，感谢你们。

前言

很高兴见到你对 Scala 感兴趣。感谢你选择本书来学习和练习这门编程语言，你将感受到在一种编程语言中融合面向对象和函数式编程这两种编程范式所带来的巨大优势。

Java 生态系统是目前用于开发和部署企业级应用最强大的平台之一。Java 平台几乎无所不在并且用途广泛；它类库丰富，可以在多种硬件上运行，并且衍生出了 200 多种基于此平台的编程语言。

我有幸学过并在工作中用过十几种编程语言，而且还为其中一些写过书。我觉得，编程语言就像各种型号的汽车——它们各执所长，帮助我们掌控平台的方向。现如今，程序员能够自由选择乃至混合使用多种编程语言完成应用程序，着实令人欣喜。

典型的企业级应用受困于各种问题——烦琐的代码难以维护，可变性增加了程序出错的可能，而共享的可变状态也让并发编程的乐趣变成了炼狱。我们一再深陷主流编程语言拙劣抽象能力的泥潭中。

Scala 是编译成 JVM 字节码的最强大的编程语言之一[①]。它是静态类型的，简洁且富有表现力，而且它已经被各种组织用于开发高性能、具有伸缩性、即时响应性和回弹性的应用程序。

这门编程语言引入了合理的特性并规避了一些陷阱。Scala 及其类库让我们能够更多地关注问题领域，而不是陷入各种底层基础设施（如多线程与同步）实现细节的泥沼之中。

Scala 被设计成用于创建需要高性能、迅速响应和更具回弹性的应用。大型企业和社交媒体需要对庞大的数据进行高频的处理，Scala 正是为了满足这些需求而创造的。

Scala 被用于在多个领域（包括电信、社交网络、语义网和数字资产管理）中构建应用程序。Apache Camel 利用 Scala 灵活的 DSL 创建路由规则。Play 和 Lift 是两个使用 Scala 构建的强大的 Web 开发框架。Akka 则是一个用 Scala 构建的卓越类库，用于创建具有高即时响应性、并发性的反应式应用程序。这些类库和框架都充分利用了 Scala 的特性，如简洁性、表现力、模式匹配和并发。

---

① JVM 上主流的编程语言是 Java、Scala、Clojure、Groovy 和 Kotlin。——译者注

Scala 是一门强大的编程语言，但我们需要专注于 Scala 中最有价值的关键部分，才能通过它来获得生产效率。本书旨在帮助你学习 Scala 的精粹，让你高效产出，完成工作，并创建实用的应用程序。

Scala 提供了两种不同的编程风格，以帮助你创建实用的应用程序。

Scala 的编程风格

Scala 并不拘泥于一种编程风格。我们可以面向对象编程，也可以使用函数式风格，甚至可以结合两者的优点将它们混合使用。

面向对象编程是 Java 程序员熟悉的舒适区。Scala 是面向对象和静态类型的，并在这两方面都比 Java 走得更远。对于初学 Scala 的我们，这是个好消息，因为我们在面向对象编程上多年的投入不会浪费，而是化作宝贵的经验红利。在创建传统的应用程序时，我们可以倾向于使用 Scala 提供的面向对象风格。我们可以像使用 Java 那样编写代码，利用抽象、封装、继承尤其是多态的能力。与此同时，当这些能力无法满足需求时，我们也并不受限于这种编程模型。

函数式编程风格越来越受关注，而 Scala 也已支持这种风格。使用 Scala，我们更容易趋向不可变性，创建纯函数，降低不可预期的复杂度，并且应用函数的组合和惰性求值（lazy evaluation）策略。在函数式风格的助益下，我们可以用 Scala 创建高性能的单线程和多线程应用程序。

Scala 和其他编程语言

Scala 从其他编程语言（尤其是 Erlang）中借鉴了许多特性。Scala 中基于 Actor 的并发模型就深受在 Erlang 中大行其道的并发模型启发。类似地，Scala 中的静态类型和类型推断（type inference）也是受到别的编程语言（如 Haskell）的影响。其函数式编程的能力也是借鉴了一些函数式编程的先导者们的长处。

Java 8 引入了 lambda 表达式和强大的 Stream API 后（可以参考 *Functional Programming in Java: Harnessing the Power of Java 8 Lambda Expressions*[Sub14]一书），使用 Java 也能编写函数式风格的代码了。这对 Scala 或者 JVM 上的其他编程语言并不会是威胁，反而会缩小这些编程语言之间的隔阂，使程序员接纳或者在这些编程语言间切换变得更加轻松。

Scala 能和 Java 生态系统无缝衔接，我们能够在 Scala 中使用 Java 类库了。我们可以完全使用 Scala 构建应用程序，也可以将其与 Java 以及 JVM 上的其他编程语言混合使用。于是，Scala 代码既可以像脚本一样小巧，也可以像成熟的企业级应用一样庞大。

谁应该阅读本书

本书的目标读者是有经验的 Java 程序员。我假定读者了解 Java 语言的语法和 API。我还假定读者有丰富的面向对象编程能力。这些假定能够保证读者可以快速习得 Scala 的精粹并将其运用于实际的应用程序之中。

已经熟悉其他编程语言的开发者也可以使用本书，但是最好辅以一些优秀的 Java 图书。

已经在一定程度上熟悉 Scala 的程序员可以使用本书学习那些他们还没有机会探索的语言特性。熟悉 Scala 的程序员可以使用本书在他们的组织中培训同事。

本书中包含什么

我写本书的目的是让读者能够在最短的时间内上手 Scala，并使用它写出具有伸缩性、即时响应性和回弹性的应用。为了做到这一点，读者需要学习很多知识，但也有很多知识读者并不需要了解。如果读者的目的是想学习关于 Scala 编程语言的所有知识，那么总是会有一些知识无法在本书中找到。有其他一些关于 Scala 的图书在深度上做得很出色。读者在本书中学到的是那些必须了解的关键概念，目的是为了快速开始使用 Scala。

我假定读者对 Java 相当熟悉。因此，读者无法在本书中学习到编程的基本概念。但是，我并没有假定读者已经了解了函数式编程或者 Scala 本身——这是读者将会在本书中学习的内容。

我写本书是为了那些忙碌的 Java 开发者，所以我的目的是让读者能够快速适应 Scala，并尽早使用 Scala 来构建自己应用程序的一部分。读者将会看到书中的概念介绍节奏相当快，但是会附带大量示例。

学习一门编程语言的方式有很多，但没有比尝试示例代码（多多益善）更好的方式了。在阅读本书的同时，请键入示例代码，运行并观察结果，按照自己的思路修改它们、做各种实验、拆解并拼装代码。这将是最有趣的学习方式。

本书所用的 Scala 版本

使用自动的脚本，本书中的代码示例使用下面的 Scala 版本运行过：

```
Welcome to Scala 2.12.6 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_172).
```

花几分钟时间为自己的系统下载合适版本的 Scala。这有助于运行本书中的代码示例（从而避免 Scala 版本导致的细节困扰）。

线上资源

读者可以从出版社网站[①]上本书的页面下载到所有示例代码。读者也可以提供反馈，直接提交勘误，或者在论坛上评论和提问。

下面是能够帮助读者开始阅读本书的若干网络资源：直接访问 Scala 的官方网站可以下载 Scala。读者可以在其文档页面找到 Scala 标准库的文档。

让我们攀登 Scala 这座高峰吧。

① https://www.epubit.com

资源与支持

本书由异步社区出品，社区（https://www.epubit.com/）为您提供相关资源和后续服务。

配套资源

本书提供如下资源：

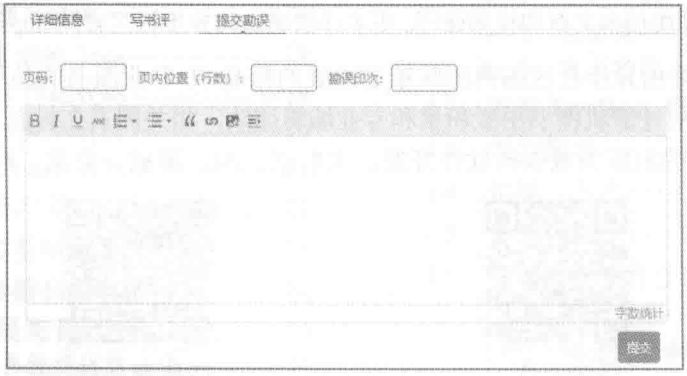
- 本书源代码。

要获得以上配套资源，请在异步社区本书页面中点击 配套资源 ，跳转到下载界面，按提示进行操作即可。注意：为保证购书读者的权益，该操作会给出相关提示，要求输入提取码进行验证。

提交勘误

作者和编辑尽最大努力来确保书中内容的准确性，但难免会存在疏漏。欢迎您将发现的问题反馈给我们，帮助我们提升图书的质量。

当您发现错误时，请登录异步社区，按书名搜索，进入本书页面，点击“提交勘误”，输入勘误信息，点击“提交”按钮即可。本书的作者和编辑会对您提交的勘误进行审核，确认并接受后，您将获赠异步社区的 100 积分。积分可用于在异步社区兑换优惠券、样书或奖品。



扫码关注本书

扫描下方二维码，您将会在异步社区微信服务号中看到本书信息及相关的服务提示。

与我们联系

我们的联系邮箱是 contact@epubit.com.cn。

如果您对本书有任何疑问或建议，请您发邮件给我们，并请在邮件标题中注明本书书名，以便我们更高效地做出反馈。

如果您有兴趣出版图书、录制教学视频，或者参与图书翻译、技术审校等工作，可以发邮件给我们；有意出版图书的作者也可以到异步社区在线提交投稿（直接访问 www.epubit.com/selfpublish/submission 即可）。

如果您是学校、培训机构或企业，想批量购买本书或异步社区出版的其他图书，也可以发邮件给我们。

如果您在网上发现有针对异步社区出品图书的各种形式的盗版行为，包括对图书全部或部分内容的非授权传播，请您将怀疑有侵权行为的链接发邮件给我们。您的这一举动是对作者权益的保护，也是我们持续为您提供有价值的内容的动力之源。

关于异步社区和异步图书

“异步社区”是人民邮电出版社旗下 IT 专业图书社区，致力于出版精品 IT 技术图书和相关学习产品，为作译者提供优质出版服务。异步社区创办于 2015 年 8 月，提供大量精品 IT 技术图书和电子书，以及高品质技术文章和视频课程。更多详情请访问异步社区官网 https://www.epubit.com。

“异步图书”是由异步社区编辑团队策划出版的精品 IT 专业图书的品牌，依托于人民邮电出版社近 30 年的计算机图书出版积累和专业编辑团队，相关图书在封面上印有异步图书的 LOGO。异步图书的出版领域包括软件开发、大数据、AI、测试、前端、网络技术等。

异步社区　　微信服务号