华章程序员书库

MANNING

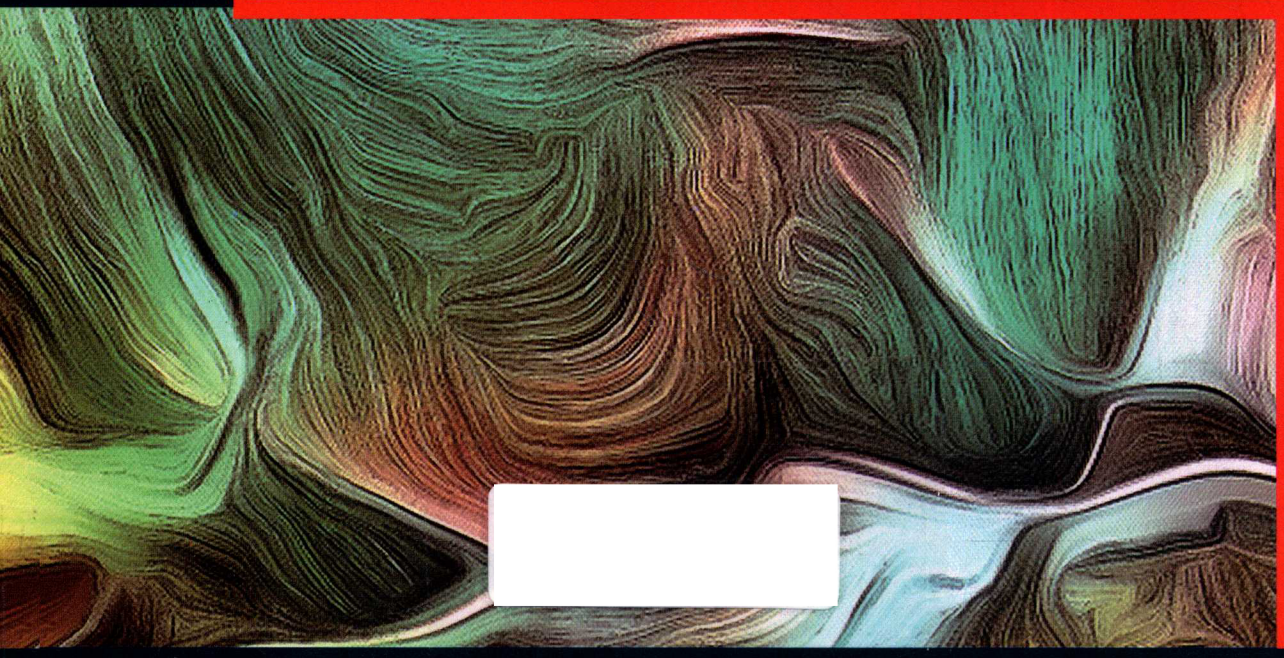
HZ BOOKS 华章IT

Clojure编程实战

（原书第2版）

Clojure in Action

Second Edition

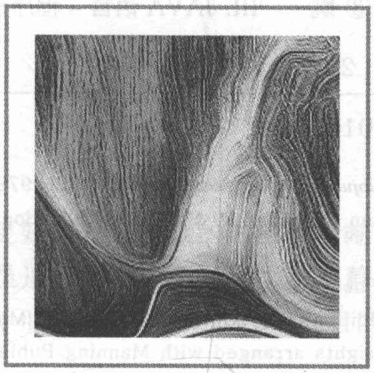


[美] 阿米特·拉索尔（Amit Rathore）
弗朗西斯·阿维拉（Francis Avila） 著

姚军 等译

机械工业出版社
China Machine Press

华章程序员书库



Clojure in Action

Second Edition

Clojure编程实战

（原书第2版）

[美] 阿米特·拉索尔（Amit Rathore）
弗朗西斯·阿维拉（Francis Avila） 著

姚军 等译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Clojure 编程实战（原书第 2 版）/（美）阿米特·拉索尔（Amit Rathore），（美）弗朗西斯·阿维拉（Francis Avila）著；姚军等译 . —北京：机械工业出版社，2018.9
（华章程序员书库）
书名原文：Clojure in Action, Second Edition

ISBN 978-7-111-60938-4

I. C… II. ① 阿… ② 弗… ③ 姚… III. JAVA 语言 - 程序设计 IV. TP312.8

中国版本图书馆 CIP 数据核字（2018）第 216470 号

本书版权登记号：图字 01-2016-3790

Clojure 编程实战（原书第 2 版）

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：张志铭 责任校对：张惠兰
印 刷：北京市兆成印刷有限责任公司 版 次：2018 年 10 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：18.5
书 号：ISBN 978-7-111-60938-4 定 价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

The Translator's Words 译者序

自编程语言出现以来，更好地管理复杂的抽象、清晰且简短的代码以及编程语言本身的可扩展性一直是无数程序员追求的目标，这催生了以静态类型、面向对象编程方法为基础的 Java 等企业级编程语言，以及以快速 Web 开发为目的的 JavaScript、Ruby、Python 等动态类型脚本语言。

但不管是上述的哪一种语言，在新的问题领域不断出现之时，它们都带来了越来越多的附带复杂性，那么如何控制这种复杂性，同时利用历史遗留下来的巨大生态系统及已有代码呢？ Clojure 就是这样的利器。它具备 Lisp 类语言的函数式编程风格，通过高阶函数、不可变数据对象等特性，生成更加清晰、一致且易于理解的代码，而且极好地解决了多线程条件下的编程难题，达到了其他语言难以企及的并发程度。Clojure 语言核心短小紧凑、语法简单明了，许多功能是通过其强大的宏系统实现的，这又使它具备了自扩展能力，可以轻松地增加语言特性甚至为非编程人员提供强大的领域特定语言。最难得的是，Clojure 植根于 JVM，并且提供了丰富的 Java 互操作功能，不仅可以轻松地共享 Java 语言长年积累下来的强大程序库和生态系统，利用原有代码，还可以让 Java 开发团队享受到新型语言的便利，可谓“鱼与熊掌兼得”。

本书的两位作者来自不同的开发领域，他们将企业编程和 Web 应用开发的知识熔于一炉，用既贴近现实应用又便于理解的实例，阐述了编程语言中这一后起之秀的方方面面。在不知不觉之中，读者可以体会到函数式编程的威力，熟悉原本令人望而生畏的独特语法，最终沉浸于新技术带来的快乐之中。详尽的解说、丰富的实例，也使本书成为 Clojure 语言的必备入门书籍。

本书的翻译工作主要由姚军完成，徐锋、刘建林、陈志勇、宁懿、白龙、陈美娜、谢志雄、方翊、林耀成、陈霞等人也为翻译工作做出了贡献，在此衷心感谢华章公司的编辑王春华老师和其他有关人员为本书所提的宝贵意见。

姚军

2018 年 7 月

第 1 版赞誉 *Praise for the first edition*

“一本容易理解的书，也是 Clojure 的快速入门途径。”

——Craig Smith, Suncorp

“广泛而全面地概述了这种激动人心的新语言的当前状态。”

——Tim Moore, Atlassian

“对构建实际应用程序所需的知识做了实用、全面的介绍。”

——Stuart Caborn, BNP Paribas

“我喜欢书中加入的测试和 Web 主题！”

——Chris Bailey, HotelTonight

“对 Clojure 及其在 JVM 语言系列中的独特地位有着深刻认识，是每位试图掌握 Clojure 的读者的必备读物。”

——Jason Rogers, MSCI Inc.

“不仅学习 Clojure——还学习如何用它来构建各种程序。”

——Baishampayan Ghose (BG), Qotd, Inc.

“用 Java 解释了函数式编程。”

——Doug Warren, Java Web Services

“这本书告诉你，结合 Java 程序库和一种务实的函数式语言能实现怎样的目标。”

——Federico Tomassetti, Politecnico di Torino

“非常容易理解的文字，出色的 Clojure 和 Lisp 入门书籍。”

——Kevin Butler, HandyApp, LLC

“介绍 Clojure 的各种特性，说明如何组合它们以实现多种工程解决方案。每种解决方案都极其简洁。强烈推荐这本书。”

——A. B.，Amazon 评论

第 2 版序言 *Preface to the second edition*

许多新接触 Clojure 的人都来自企业软件领域，包括本书的主要作者阿米特·拉索尔。在他们的世界里，刻板的静态类型、面向对象语言与由工具、框架和程序库组成的庞大生态系统联系在一起，这些工具、框架和程序库的设计主旨是降低组件和不断变化的业务需求之间的耦合度。这是包含依赖注入、Servlet 容器、XML 配置和代码生成的 Java 及 C# 世界。因为 Clojure 运行于 Java 之上，所以它成为试图摆脱这一领域复杂性同时又不想完全放弃熟悉且优秀工具的人的不二之选。对于 Clojure，企业软件开发人员害怕和不熟悉的特性是动态类型和一阶函数，但是 Clojure 吸引人的地方在于摆脱附带复杂性和静态类型，同时仍然可以在必要时使用旧代码。

我来自 Web 开发的“狂野西部”，那是 PHP、JavaScript、Python 和 Ruby 等动态类型编程语言的疯狂世界。这些语言中，有些在最初设计时很少（甚至没有）考虑过在大项目上的实用性，并且为了适应大项目而匆忙地发展出了新功能和变通方法。许多使用者（包括我）都没有经过计算机科学训练，职业生涯可能始于摆弄 HTML，为日常工作提供一个网站。他们的编程知识和所使用的语言一样，都是随着网站的成长而匆忙获得的。和企业软件领域不同，动态类型、自动类型强制和后期绑定是常规做法，第一类函数很常见，面向对象也不是基本前提。在这个领域里仍然有由框架和程序库组成的大型生态系统，但是它们不像企业软件开发中那样遵守规范和面向配置。对于 Web 开发人员来说，Clojure 最可怕的是其背后潜伏的企业软件“幽灵”——（简言之）Java。对于企业开发人员来说，Clojure 的 Java 传承是一个特性，而这对于 Web 开发人员则是一个 bug。

如果你来自 Web 开发者的世界，那么我可以告诉你：不用害怕 Java。许多企业软件的复杂性是体现在编译时的：静态类型、冗长的代码和许多 XML 配置。在流行的 Web 开发语言中，复杂性体现在运行时：弱类型和极端的动态性及易变性使程序难以推导。我正是在寻找这种附带复杂性的更好解决方案时发现了 Clojure，我对 Java 也持怀疑态度。我听说过 Java EE 的各种传说，看到过庞大的类文件和工厂接口。我感到疑惑：Clojure 是在 Java 这种死板、脆弱和复杂的软件栈基础上构建的，那么它怎么能够更好地控制软件复杂性？该如何平衡所有括号？

Clojure 处于混乱的 Web 开发领域和过于规格化的企业软件领域之间，前者的代码库难以安全地更改，后者的代码库则冗长且难以理解。Clojure 在我的程序上施加的限制比编写 PHP 时更严格，但是这种纪律性没有任何负面影响：你的代码仍然和往常一样简洁（也许还更简洁）；你可以轻松且毫无痛苦地利用 Java 生态系统的许多优势，例如健全的包管理和基于 JAR 的部署；由于 JVM，你的应用程序还可能运行得更快！

甚至在我成为专业的 Clojure 编码者之前，它就已经给我带来了好处。深入理解 Clojure 的简洁性和不可变性哲学，帮助我认识到在其他语言编程中遭遇的复杂性的根源，从而更好地控制复杂性。我现在以编写 Clojure（和 ClojureScript）为生，诚然，我的软件中仍然有许多附带复杂性，但是发现和控制它们变得更轻松了，我所编写的软件是使用 PHP 或者 Python 时做梦也想不到的。

本书的第 1 版帮助我走上 Clojure 的道路，那正是我现在的道路。我很荣幸能参与第 2 版的写作，希望它也能够帮助你“驯服”软件复杂性。不要害怕 Java 或者括号！它们实际上都相当“驯良”。

弗朗西斯·阿维拉

第 1 版序言 *Preface to the first edition*

我可以告诉你，我有多享受极客的生活；我可以告诉你，1985 年父亲给我展示穿孔卡时，我有多么着迷；我可以告诉你，我在 7 岁时是如何得到第一台计算机的；我还可以告诉你，我从 1989 年起就爱上了编程。我可以告诉你关于这一切的许多事情，但是不能肯定它们是否有趣。

作为替代，还是让我告诉你们我对答案的追求吧。多年以来，关于我们所在行业的一些问题一直困扰着我：为什么从来没有一个软件项目像它应有的那样简单？为什么没有一个项目能够按照时间和预算完成？为什么程序中总是有 bug？为什么软件从来不按照人们的意图工作？为什么对软件进行更改总是那么困难？为什么不管一个项目开始时有多么清晰的计划，却总会变成一个“大泥球”？

几乎每个人都会认识到这些问题，但他们似乎接受这种现状。行业中的大部分人通过在时间安排和预算上设置缓冲以及接受普通水平的软件来应对它们。难道没有更好的办法了吗？

本书不是答案，绝对不是，但它是我向前探索的一部分。我的想法是，工具越好，越能够帮助我们创建好的软件。

这就引出了显而易见的问题：什么是更好的工具？它们在哪方面表现更好？我的答案便是，好的工具是能够更好地帮助控制复杂性的工具。毕竟，复杂性是我们所在世界中一切状态的根源。确实，Fred Brooks 早在 1986 年就在一篇文章中提到了复杂性。他提出了基本复杂性和附带复杂性之间的区别。基本复杂性是问题领域中固有的，而附带复杂性是由问题领域之外的事物引入的。例如，在处理纳税申报的软件项目中，由复杂的纳税编码引入的复杂性是问题领域的一部分，因此其是基本复杂性。而由错综复杂的访问模式引起的任何复杂性则是附带复杂性。

那么，让我来改变一下措辞：好的工具能帮助我们减少附带复杂性。这些工具让我们尽可能好地完成工作，而且不会成为前进道路上的障碍。另外，出色的工具更不止于此，会为我们提供各种手段，提高设计人员和编程人员的效率，并且自身不会引入问题。Lisp 编程语言就是为了成为这样的工具而被设计的。Clojure 则是设计极其精巧的 Lisp。

每个遇上 Lisp 的程序员都有自己的故事，我的故事和许多人类似。我的职业生涯从 Java 开始，最终撞上了一堵自己建立的墙。于是我开始探索动态语言，它们看起来更有表现力和可塑性。我喜欢使用 Python 和 Ruby，并用它们编写了多个重要应用。当时我在一家名为 ThoughtWorks 的公司工作，有许多志同道合的同事。最终，其中一个人带领我转向 Common Lisp。对这种语言的理解越深，我就越强烈地意识到其他语言的粗糙。我在几个个人项目上使用了 Common Lisp，但是从未将其应用于重大项目。不过，它对我使用其他所有语言的编码工作都产生了深远的影响，我不断地寻找机会，试图将 Lisp 应用于现实世界的项目中。

我在 2008 年终于得到了机会。那时，我搬到加州湾区，加入了一家初创企业 Runa 的创建团队。按照真正的硅谷传统，我们的第一个办公室在创始人的车库里。我们希望凭借 Runa 公司的力量搅乱电子商务领域，想法是收集大量数据，用机器学习技术完全领会它们，然后提出个性化的交易，实时选择购物者。为了完成这些工作，我们必须攻克真正的技术难关。该系统必须每秒处理数千个请求，每天处理数太字节（TB）的数据；它还必须能够通过一组高级的陈述性领域特定语言（DSL）来编写脚本；它必须支持代码热交换，以便在运行中更新；它必须运行于云上，还必须完全由 API 驱动。我们不得不在缺乏资源的情况下完成该系统，因为设计团队只有三个人。

由于这些约束，我们需要一种能够提供“杠杆”的语言。因此，我们转向名为 Clojure 的新语言。这是一种运行于 JVM 之上的现代化函数式语言，它还承诺解决并发多线程代码中固有的问题。而且，它还是 Lisp 语言的一个变种！

我曾是这家初创企业的架构师，现在是工程副总裁。我将未来的成功押在这个名不见经传的人创建的新型编程语言（当时还在预发行阶段）上。但是我所读到的有关它的一切都与我产生了共鸣——一切都是那么合适。从那时起，我们使用 Clojure 取得了难以置信的成功。我们的团队在过去三年中成长了起来，但是仍然比类似公司的其他团队小一个数量级。我怀疑那些团队使用的是旧的 Java。过去的经验让我坚定了信念，在其他条件相同的情况下，工具非常重要，而有些工具远优于其他工具。

当我们开始着手工作时，我曾将 Clojure 当成秘密武器——但是 Clojure 的社区很强大，并且是相互扶持的，因此将其作为一个公开的秘密似乎是更好的想法。我启动了湾区的 Clojure 用户组，现在已经有数百名成员。已经有几十个人来参加过我们的会议，喜欢在会上听到的内容，并决定在自己的项目上使用 Clojure。

出于相同的信念，我编写了这本书，分享使用 Clojure 的经验，希望可以说服你们中的一些人，不仅看到这些“括号”，还能了解 Lisp 语言的总体能力和 Clojure 的具体特性。希望你们能觉得这本书实用且有趣。

阿米特·拉索尔

关于本书 *About this book*

从2011年本书第1版发行以来，软件工程领域已经有了许多变化。当时，Clojure刚刚发行了1.3版本，社区正在致力于1.4版本的开发。ThoughtWorks技术雷达已经将Clojure从“评估”推进到“试验”（https://www.thoughtworks.com/radar/languages-and-frameworks/clojure）。喜欢冒险的程序员和软件公司开始注意到这一点，但是用Clojure搭建重要项目的情况仍不多见。2015年年底，Clojure在编程领域已经有了一席之地。就连家喻户晓的沃尔玛和《消费者报告》杂志也将Clojure用于核心业务（http://cognitect.com/clojure#successstories）。Clojure现在已经站稳了脚跟，完全不再出现于ThoughtWorks的技术雷达上了。

即使在Clojure仍然被边缘化的领域，它的核心思路——不可变性和函数式编程——也已经声名远播并结出了硕果。受到Clojure启示的不可变数据库Datomic正在被越来越多的人采用。Java 8现在拥有了Lambda：用于高阶函数编程的匿名内联函数。在许多不同的编程语言中，现在可以找到多个不可变数据结构库。这些思路已经通过ClojureScript（2011年10月才发行的！）和Facebook React UI框架的协作，在JavaScript中掀起了革命。不可变性和函数式编程现在已经成为主流的思路。

为了应对文化中的这些变化，本书第2版缩小重点范围，面向更广的受众。越来越多Java生态系统之外的程序员听说了Clojure，并有兴趣学习它，所以我们扩充了入门章节的内容，假定读者拥有较少的Java知识，更加鲜明地强调Clojure的哲学思想，这些都可以在任何语言中实践，并为它们带来益处。随着受欢迎程度的大幅提高，用于常见编程任务的不同程序库和在线教程激增。因此，我们删除了处理数据库连接、构建Web服务等任务的实操章节。这些章节都已经随着程序库和替代方法的成长而变得老旧，如果我们用现代工具和技术重新编写它们，毫无疑问在出版之前它们就又过时了。幸好，在软件工程的任何子领域中，找到使用Clojure的最新文档都不再困难。

简而言之，我们已经不再需要像第1版中那样卖力地倡导Clojure。如果你打算阅读本书，可能已经知道它是受Lisp启发并在JVM基础上构建的一种强大的通用函数式语言。你已经听说了这样的故事：规模很小的Clojure团队构建强大的分布式系统所花的时间比使用

其他语言的大型团队还要短得多。你打算阅读本书，正因为你想要了解 Clojure 是如何实现这样的可能性的，也想知道如何达到同样的目标。

如何使用本书

学习 Clojure 对许多程序员来说是一个飞跃。完全不同的语法、从命令式编程到函数式编程、不可变性、宏系统……这一切都令人畏缩。本书采用一种缓慢而稳健的方法来帮读者学习这种语言和各种概念，并且假定读者之前没有任何 Lisp 或者函数式编程语言的经验。首先是必备的基础知识，然后慢慢地分层次介绍这种语言的不同特性，并以直观的方式将它们组合起来。本书对所有主题都采用基本原理方法，首先解释某件任务必须以某种方式完成的原因，然后再讨论 Clojure 方法。

一旦完成了基础知识的学习，本书将介绍由多位编程者编写的规模更大的“严肃”Clojure 程序所必需的特性、概念和技术。你将了解有效管理可变状态、大规模使用高阶函数编程、创建多态类型和抽象的同时平衡表达能力及性能、编写测试驱动 Clojure 程序、编写领域特定语言的方法。为了最大限度地利用本书，我们假定你熟悉某种面向对象（OO）语言，如 Java、C++、Ruby 或者 Python，但是不需要任何 Java、Lisp 或者 Clojure 背景。

路线图

本书共 11 章，下面描述每章的重点内容。

第 1 章简要介绍 Clojure 语言及其三大支柱：使用不可变数据结构的函数式编程、Lisp 语法以及与 Java 的互操作性。

第 2 章介绍 REPL（读取 - 求值 - 打印循环，这是 Clojure 的命令行解释程序），帮助你开始编写 Clojure 代码。本章包含对函数定义、流程控制和内建数据结构的纵览。

第 3 章讲解 Clojure 更独特的特性：元数据（为其他数据提供注解的数据）、异常处理、高阶函数（作为其他函数参数的函数）、两组作用域规则（词法和动态）、组织代码的命名空间、可以轻松简洁地将嵌套数据结构各部分放入变量的解构语法、为代码添加新字面语法的读取器字面量。本章中的许多方法可能与你所习惯的方法不同，但是在本章的最后，你将能够读写最简单的 Clojure 程序。

第 4 章讨论三种基本的多态性和使用多重方法时各种多态性在 Clojure 中的表现。如果你来自 Java/C++ 世界，那么这将是一种大不相同的方法。Clojure 的多重方法是实施多态行为的极度开放式方法，它们将方法分派的控制直接交给程序员。

第 5 章介绍 Clojure 与 JVM 的结合。没有一组强大的程序库，任何编程语言都不可能取得成功，Clojure 巧妙地回避了这个问题。你可以非常轻松地在程序中使用任何 Java 程序

库，从而可以立刻利用数千种久经考验的框架及程序库。Clojure 还可以利用 Java 技术栈。在这一章中，你将学习从 Clojure 中使用 Java 代码、从 Java 中使用 Clojure 代码以及编写定义或者扩展 Java 类的 Clojure 程序的方法。

第 6 章解释 Clojure 的状态管理和并发方法，以及四种基本的并发原语。同样，这是处理可变状态问题的一种新方法。Clojure 配备了极端高效的不可变数据结构，实现了类似数据库的 STM（软件事务内存）系统。这种组合使该语言提供了对正确、安全和无死锁并发的内置支持。这很重要！你的程序可以利用多个核心，而不会产生传统多线程代码的相关问题。

第 7 章关注 Clojure 不同于大部分其他编程语言的另一种特性，这就是宏系统（不要和 C 语言的宏以及类似概念混淆）。Clojure 本质上为代码生成提供了语言级支持。它的运行时系统中有一个钩子（hook），允许程序员以任何方式变换和生成代码。这是一种难以置信的强大功能，模糊了语言设计人员和应用编程人员之间的界限，使任何人都可以为该语言增加特性。

第 8 章深入介绍函数式编程范式以及第 3 章中涉及的高阶函数的使用方法。你将创建如下核心高阶函数的自有版本：`map`、`reduce` 和 `filter`。你还将全面理解函数的部分应用和局部套用。最后，你将在 Clojure 基础上构建自己的面向对象编程（OOP）系统，并忘掉 Clojure 与面向对象范式相关性的忧虑。实际上，你将不再用相同的方式去思考面向对象方法。

第 9 章讨论表达问题，其基础是第 4 章中研究的多态性。你将首先回顾这个老问题的概念，然后使用 Clojure 的多重方法以优雅的方式解决它。接着，在介绍其他 Clojure 特性（协议、记录和类型）之后，我们将告诉你一种有局限但性能更好的解决方案。

第 10 章说明如何将编写测试驱动代码的过程与第 2 章中介绍的 Clojure REPL 相结合，从而显著提升效率。本章还介绍模拟和打桩函数，以实现更好的单元测试策略。

第 11 章是本书的最后一章，重点是高级宏和 DSL，建立在第 7 章所学知识的基础上。本章将引领你完成整个周期：从寻找最小化附带复杂性的工具开始。Clojure 允许你通过宏系统根据自己的意愿改变这种编程语言，本章会深入介绍这一功能。你将设计一个内部 DSL，作为使用 DSL 驱动 Clojure 应用中核心业务逻辑的一个例子。

代码约定和下载

许多代码清单中有注释，以便强调重要的概念。在某些情况下，所用编号与代码清单后的解释关联。

下载和安装 Clojure 的指南参见附录。你可以在 manning.com/books/clojure-in-action-second-edition 网站上找到本书中所有例子的完整代码。

Acknowledgements 致　谢

我们要感谢 Manning Publications 中帮助本书出版的每个人，包括为我们提供机会撰写本书修订版的 Erin Twohey 和 Michael Stephens，在该过程中仔细审读手稿的 Karen Miller，提供专业性技术编辑的 Joseph Smith，以及为本书出版提供指导意见的 Kevin Sullivan、Jodie Allen、Linda Recktenwald 和 Mary Piergies。

我们还要感谢在该过程中多次阅读各个章节并提供宝贵反馈意见的评审人员：Bruno Sampaio Alessi、David Janke、Fernando Dobladez、Gary Trakhman、Geert Van Laethem、Gianluigi Spagnuolo、Jeff Smith、Jonathan Rioux、Joseph Smith、Justin Wiley、Palak Mathur、Rick Beerendonk、Scott M. Gardner、Sebastian Eckl 和 Victor Christensen。

还要感谢我们的 MEAP（Manning Early Access Program）读者，他们在“作者在线”论坛上发表了评论和修正意见。感谢他们对本书感兴趣并提供支持。

最后，我们要感谢提出不可变性的 Rich Hickey，感谢他创造了 Clojure 并鼓励我们更简洁地编程。

阿米特·拉索尔：

在一家初创公司工作（而且有了第一个孩子）的同时写一本书绝对不是放松的妙方！如果没有忍耐力超群的妻子 Deepthi 的支持，我绝对无法完成这两个版本。当我毫无进展的时候，只有她的鼓励能帮助我坚持下去。感谢你，亲爱的，没有你我绝对无法完成这个项目！

我还要感谢我的父母，在许多年之前，他们给了我走上这条道路的机会。我在印度长大，当时计算机还是幻想中的东西，大部分人都无法接触。父母贷款为我买了一台计算机，而不是购买他们的第一辆车，没有它就没有我的今天。因此，我要向父母说一百万次“谢谢”！

我还要感谢 Ravi Mohan，感谢他在 2001 年指引我了解 Lisp 并阅读 Paul Graham 的论文。感谢他为我展示了这条道路的魅力！我想，还要感谢 Paul Graham，他启发了我们中许

多人的灵感。

感谢 Runa 的伙伴们让我写这本书。公司的创始人 Ashok Narasimhan 对整个项目都极其支持。我的其他同事也很支持我。尤其要感谢 Kyle Oba 和 George Jahad 的反馈和鼓励。

最后，我要特别感谢 Siva Jagadeesan 以各种各样的方式支持这项工作，帮助我将这本书升级为第 2 版。

弗朗西斯·阿维拉：

首先，我要感谢阿米特·拉索尔，他编著的本书第 1 版对我进入 Clojure 世界有重要意义。我对本书所做的只是重新“装修”的工作，坚固的支柱都是阿米特建造的，他所写的那些内容仍然是本书的真正基础。

我还必须感谢妻子 Danielle，她鼓励我接受 Manning 的邀请，合作编著本书的第 2 版，在我写作时她常独自在长夜中陪伴新出生的女儿，并认真地为我审读稿件。感谢你的爱和支持，以及对我痴迷于那些奇怪的“括号”的宽容。

我还要感谢 Breeze EHR，这家小型初创企业促使我进入神奇的 Clojure 世界。特别感谢公司的创始人、核心 Tyler Tallman，他将我从 PHP 中带出来。每次他与我分享激动人心的新想法时，我总是表现出坏脾气，对此我深感抱歉。

Contents 目　录