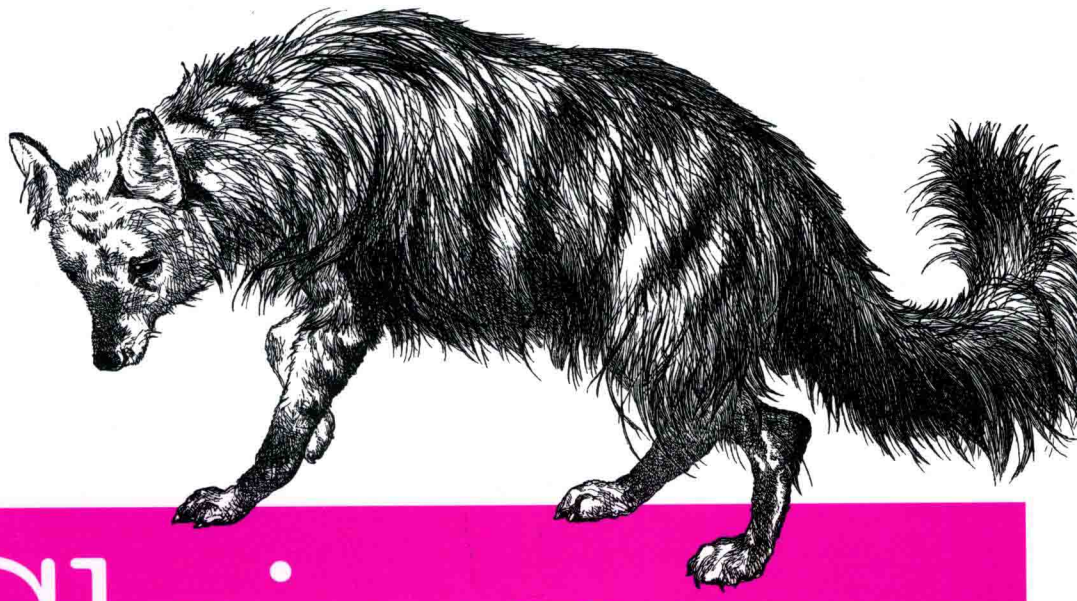
O'REILLY®

TURING 图灵程序设计丛书



Clojure 经典实例

Clojure Cookbook

[美] Luke VanderHart [加] Ryan Neufeld 著
王海鹏 徐宏宁 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

Clojure经典实例

Clojure Cookbook
Recipes for Functional Programming

[美] Luke VanderHart [加] Ryan Neufeld 著
王海鹏 徐宏宁 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

Clojure经典实例 / (美) 范德哈特(VanderHart,L.)，(加) 诺伊费尔德 (Neufeld,R.) 著；王海鹏，徐宏宁译. -- 北京 : 人民邮电出版社，2015.8
（图灵程序设计丛书）
ISBN 978-7-115-39594-8

Ⅰ. ①C… Ⅱ. ①范… ②诺… ③王… ④徐… Ⅲ. ①JAVA语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2015)第131402号

内 容 提 要

本书以具体实例的形式讲解了Clojure语言在不同领域的应用，不仅介绍如何运用Clojure，而且还展示了很多常见库。书中给出了添加了注释的示例代码，详细分析并解释了数百个真实世界的编程任务。读者既可通过本书深入了解Clojure的精髓，也可将本书用作参考指南，解决具体问题。

本书适合各层次Clojure开发人员阅读。

◆ 著 [美] Luke VanderHart [加] Ryan Neufeld
译 王海鹏 徐宏宁
责任编辑 岳新欣
执行编辑 张 曼
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：26.5
字数：626千字 2015年8月第1版
印数：1－3 500册 2015年8月北京第1次印刷
著作权合同登记号 图字：01-2014-6469号

定价：95.00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

版权声明

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*

译者序

编程语言习得

“熟悉与优雅正交。”
——Rich Hickey

约二十年前，我买过一本高等教育出版社出版的《LISP 语言》，作者是马希文、宋柔。可惜当年没有老师指导，自己水平不够，未能深入下去，只留下了一点模糊的印象：LISP 语言适用于人工智能，括号很多。

几年前，图灵公司的朋友送我一本《黑客与画家》，我连夜看完，重新燃起了对 LISP 的兴趣。我在书评中写道：“读完之后有一种想去学习 LISP 语言的冲动。一个不懂 LISP 的 Java 程序员，不是一个好的 C++ 程序员。”

现在，我终于找到了机会，开始学习 Clojure 这种运行在 JVM 上的 LISP 方言。经过一段时间的学习，我完全被它迷住了！

首先吸引我的是它的函数式编程特性。作为一个学习 C++ 和 Java 多年的程序员，我已习惯在程序中使用各种名词抽象，也就是领域术语，希望在程序中体现领域专家的思想和认识水平。而在 Clojure 编程中，虽然它也很适合领域抽象，但它的抽象程度更高，它希望达到数学家认识世界的水平。问题的开头通常是“给定一个无限序列……”，而常见的例子是如何实现斐波那契数列。

Leslie Lamport 说过，要将事情描述得清晰准确，人类发明的最好语言就是数学。这种对“表达的经济性”的追求，对于中国人是不陌生的。中国是诗歌的国度，而且古人对言简意赅的追求也有许多例子，比如“逸马杀犬于道”的故事。所以我觉得，LISP/Clojure 在精神上与有追求的中国程序员是契合的。

其次，它特别适合开发领域特定语言（DSL）。在 LISP 社区中流传着一个笑话，可以说明这一点：任何足够大的软件，最后都会实现一个半调子 LISP 解析器。LISP 的底层抽象极其简单，允许程序员设计更多的抽象，来描述这个世界。

学习一门新的语言，会改变学习者的思维方式。在面向对象编程时，我们更多关注单个对象。在函数式编程中，我们更多关注函数和集合。在工作中，不一定马上有机会使用 Clojure，但其中学到的思维方式，将对编程产生立竿见影的影响。在翻译本书时，我同时在用 Lua 开发项目，学习了 Clojure，让我能写出更简洁、更优雅的 Lua 代码。

学习新语言有这样一些原则：（1）专注于与你相关的内容；（2）从学习这门语言的第一天起，就把它当作你的交流方式；（3）当你听得懂别人在说什么时，就会不知不觉慢慢习得这门语言；（4）语言不是大量的知识积累，而更像一种生理训练；（5）心理状态和生理状态都很重要，要愉快和放松。对于模棱两可要有一定的容忍性，对于细枝末节不要过于纠结，因为那会把你逼疯。

本书提供了大量的例子，覆盖了日常编程领域的方方面面，正是学习 Clojure 的好读物。在翻译本书的过程中，我学到了很多，在此郑重推荐给大家。不足之处，还望大家指正。

王海鹏

2014 年秋

前言

本书的首要目标是提供中等长度的 Clojure 代码示例，超越基本知识，关注真实世界的日常应用程序（而不是概念或学术问题）。

与此前的许多其他 Clojure 书籍不同，本书的主题不是语言本身，或它的功能和能力。本书关注开发者面对的具体任务（不论他们使用哪种编程语言），展示如何用 Clojure 来解决这些具体问题。

因此，本书确实不是也做不到包罗万象，因为可能的问题示例有无限多个。但是，我们希望记录大多数程序员会经常遇到的一些比较常见的问题。通过归纳，读者将能够学到一些常见的模式、方法和技术，有助于他们为自己面对的问题设计解决方案。

本书如何写成

关于本书，你要了解一件重要的事情：它首先是团队协作的成果。它不是由一两个人写成的，甚至不是一个确定好的团队的成果。相反，它是 60 多个最优秀的 Clojure 程序员协作的结果，他们来自世界各地、各行各业。这些作者每天都在真实的场景中使用 Clojure：从航空航天到社交媒体，从银行业到机器人，从 AI 研究到电子商务。

因此，你会在提供的实例中看到许多差异。有些快速而简要，有些则内容更为丰富，针对 Clojure 的基本原理和实现提供了易于理解的深刻洞见。

我们希望兴趣各异的读者都能从本书中有所获。我们相信，它的用处不仅在于查找具体问题的解决方案，也在于考察 Clojure 能够提供的各种表达能力。在编辑提交的内容时，我们非常吃惊地发现很多概念和技术对我们来说也是新的，希望对读者来说也是新的。

我们在写作和编辑时还发现，要确定我们想介绍的内容的范围是一件很难的事情。每个实例都很棒，可以无限细分，进而涉及多个话题，而每个话题又值得写一个实例、一章甚至

一本书。但每个实例也需要保持独立。每个实例应该提供一些有用的、有价值的信息，让读者可以理解并消化。

我们真诚地希望自己很好地平衡了这些目标，也希望你觉得这本书有用而不乏味，内容深刻而不是艰深难懂。

读者对象

我们希望所有使用 Clojure 的人都能从本书中学到一些东西。有许多实例介绍的是真正基础的内容，初学者会觉得有用，但还有许多实例探讨的是专业话题，高级开发者会觉得有用，有助于他们开始实践。

但如果你是 Clojure 新手，这可能不是你要看的第一本书，至少不要只看这本书。本书介绍了许多有用的话题，但不像优秀的入门教材那样系统或完整。下面列出了一般的 Clojure 书籍，将它们作为前导教材或补充教材会很有帮助。

其他资源

本书内容并不全面，也永远不可能全面。有许多内容要讲，并且由于采用了面向任务的实例，自然就排除了有条理、叙述式地解释整个语言的特点和能力。

要更线性、彻底地了解 Clojure 及其特点，我们推荐下面的书。

- 《Clojure 编程：Java 世界的 Lisp 实践》（O'Reilly，2012），作者是 Chas Emerick、Brian Carper 和 Christophe Grand。这是一本全面的、用于一般目的的 Clojure 好书，关注语言和常见任务，面向 Clojure 的初学者。
- 《Clojure 程序设计（第 2 版）》（Pragmatic Bookshelf，2012），作者是 Stuart Halloway 和 Aaron Bedra。这是第一本关于 Clojure 的书，为 Clojure 语言提供了清晰全面的介绍和指导。
- *Practical Clojure*（Apress，2010），作者是 Luke VanderHart 和 Stuart Sierra。它简明扼要地解释了 Clojure 是什么，它的特点是什么。
- 《Clojure 编程乐趣》（Manning，2011），作者是 Michael Fogus 和 Chris Houser。这是一本比较高级的教材，真正深入到 Clojure 的主题和原理。
- *ClojureScript: Up and Running*（O'Reilly，2012），作者是 Stuart Sierra 和 Luke Vander Hart。虽然本书和这里列出的其他 Clojure 书籍主要或全部在探讨 Clojure 本身，但 ClojureScript（一种 Clojure 方言，能编译成 JavaScript）已经得到了相当的发展。这本书介绍了 ClojureScript，以及如何使用它，并探讨了 ClojureScript 和 Clojure 之间的相似与不同。

最后，你应该看看本书的源代码，它们可以从 GitHub 自由下载（https://github.com/clojure-cookbook/clojure-cookbook）。网上选择的实例比印刷版本更多，我们仍在接受新实例的“拉取请求”（pull request），也许某天会加入本书的下一版。

本书结构

本书的章节主要是依据主题对实例进行分组，而不是严格的分类。一个实例完全有可能适用于不止一个章节，在这种情况下，我们试着根据我们的猜测，将它放在大部分读者首先会去寻找的地方。

实例包含三个主要部分和一个次要部分：问题、解决方案、讨论和参阅。实例的问题陈述提出了任务或要克服的障碍。它的解决方案解决了问题，展示了特定的技术或库，能够高效地完成该任务。讨论完善了相关知识，探讨了解决方案和相关注意事项。最后，参阅部分向读者指出了一些附加的资源或相关实例，帮助你采用描述的解决方案。

各章简介

本书由以下几章构成。

- 第 1 章“原生数据”和第 2 章“复合数据”介绍了 Clojure 内建的原生和复合数据结构，解释了许多常见的（以及不太常见的）使用方式。
- 第 3 章“广义计算”包含了一些有用的主题，广泛适用于许多不同的应用领域和项目，从协议这样的 Clojure 特征，到可选的编程范式，例如用 `core.logic` 实现逻辑编程，或用 `core.async` 实现异步协作。
- 第 4 章“本地 I/O”包含了程序在运行时与本地计算交互的所有方式。这包括读写标准输入输出流，创建并操作文件，序列化和反序列化文件等。
- 第 5 章“网络 I/O 和 Web 服务”包含了类似第 4 章的主题，但探讨的是通过网络的远程通信。它包括的实例涉及各种网络通信协议和库。
- 第 6 章“数据库”展示了连接和使用各种数据库的技术和工具。特别关注了 Datomic 数据库，它共享了 Clojure 背后关于值、状态和标识的哲学，并扩展到了持久存储的领域。
- 第 7 章“Web 应用”深入探讨了 Clojure 最常见的应用领域：构建和维护动态网站。它全面介绍了 Ring(Clojure 中最流行的 HTTP 服务器库),以及 HTML 模板和渲染的工具。
- 第 8 章“性能与开发效率”解释了拥有 Clojure 程序之后还需要做些什么，介绍了打包、分发、性能剖析、日志的常见模式，将正在进行的任务与应用的生命周期关联起来。
- 第 9 章“分布式计算”关注云计算以及在重量级分布式数据处理中使用 Clojure。特别关注了 Cascalog，它是一个声明式 Clojure 接口，面向 Hadoop MapReduce 框架。
- 最后但同样重要的是第 10 章“测试”介绍了各种技术，来确保代码和数据的完整性和正确性：从传统的单元测试和集成测试，到更全面的产生式测试和模拟测试，甚至还有

可选的编译时验证，利用 core.typed 实现静态类型。

软件获取

若要按照本书的实例操作，你需要正确安装 Java 开发工具（JDK）和 Clojure 事实上的构建工具 Leiningen。我们推荐第 7 版 JDK，但至少需要第 6 版。对于 Leiningen，至少是第 2.2 版。

如果你还没安装 Java（或者希望升级），请访问 Java 下载页面（http://www.oracle.com/technetwork/java/javase/downloads/index.html），按提示下载并安装 Java JDK。

要安装 Leiningen，请遵循 Leiningen 网站（http://leiningen.org/）的安装指南。如果已经安装了 Leiningen，通过执行 `lein upgrade` 命令来取得最新版本。如果不熟悉 Leiningen，请访问使用指南（https://github.com/technomancy/leiningen/blob/stable/doc/TUTORIAL.md），了解更多信息。

你不需要手工安装的就是 Clojure 本身，因为 Leiningen 将随时根据需要，替你安装。要验证安装，就运行 `lein repl` 来检查 Clojure 的版本：

```
$ lein repl
# ...
user=> *clojure-version*
{:major 1, :minor 5, :incremental 1, :qualifier nil}
```

某些实例在 GitHub 上提供了一些在线材料。如果你的系统上没有安装 Git，请按照安装指南（https://help.github.com/articles/set-up-git）操作，以便能将 GitHub 代码库签出到本地。

某些实例（如数据库实例），需要进一步安装软件。在这种情况下，实例将包含安装工具的额外信息。

本书约定

在这本全是解决方案的书中，你会发现其中有不少代码。Clojure 源代码使用等宽字体，像这样：

```
(defn add
  [x y]
  (+ x y))
```

如果 Clojure 表达式被求值并返回，该值将以注释的形式给出，跟在一个箭头后面，就像它出现在命令行中一样：

```
(add 1 2)
```

```
;; -> 3
```

在合适的时候，代码示例可能略去或省略返回值注释。最常见的两种情况就是在定义函数/var 时和缩短较长的输出时：

```
;; 这会返回 #'user/one，但你真的关心吗？
(def one 1)

(into [] (range 1 20))
;; -> [1 2 ... 20]
```

如果表达式产生输出到 STDOUT 或 STDERR，就会有注释说明（分别用 *out* 或 *error*），跟着就是每行输出的注释：

```
(do (println "Hello!")
    (println "Goodbye!"))
;; -> nil
;; *out*
;; Hello!
;; Goodbye!
```

REPL会话

看到 REPL 驱动开发目前正在流行，因此本书就成为了 REPL 驱动的。REPL（读取、求值、打印、循环）是交互式的提示符，对表达式求值并打印出结果。Bash 提示符、irb 和 python 提示符都是 REPL 的例子。本书中几乎每个实例，都是为在 Clojure REPL 中运行而设计的。

虽然 Clojure REPL 传统上显示为 user=> ...，但本书希望读者能够复制粘贴实例中所有的例子，并看到标示的结果。因此，例子中省略了 user=> 并以注释的方式给出了输出，让事情变得更容易。如果你在计算机旁，这就特别有帮助：只要复制粘贴代码示例，不用担心遇到不能执行的代码。

如果例子只适用于 REPL 的环境，我们将保留传统的 REPL 风格（带上 user=>）。下面两个例子分别是只适用于 REPL 的例子和它的简化版本。

只适用于 REPL：

```
user=> (+ 1 2)
3
user=> (println "Hello!")
Hello!
nil
```

简化版本：

```
(+ 1 2)
```

```
;; -> 3

(println "Hello!")
;; *out*
;; Hello!
```

控制台/终端会话

控制台会话（例如，shell 命令）用等宽字体表示，行开始的美元符号（$）表示 shell 提示符。输出打印前面没有（$）：

```
$ lein version
Leiningen 2.0.0-preview10 on Java 1.6.0_29 Java HotSpot(TM) 64-Bit Server VM
```

命令行末的反斜杠（\）告诉控制台，命令将在下一行继续。

我们的金童 lein-try

Clojure 不以它的扩展标准库而闻名。不像 Perl 或 Ruby 这样的语言，Clojure 的标准库相对比较小。Clojure 选择了简单和强大。因此 Clojure 是一种有许多库的语言，但不是内建的库（好吧，Java 除外）。

因为本书中这么多解决方案都依赖于第三方库，所以我们开发了 lein-try（https://github.com/rkneufeld/lein-try）。Leiningen 是 Clojure 事实上的项目工具，lein-try 是 Leiningen（http://leiningen.org/）的一个小插件，让你快速而容易地尝试各种 Clojure 库。

要使用 lein-try，请确保安装了 Leiningen，然后将你的用户特性描述文件（~/.lein/profiles.clj）编辑成下面的样子：

```
{:user {:plugins [[lein-try "0.4.1"]]}}
```

现在，在项目内外都可以使用 **lein try** 命令来启动 REPL，访问任何你喜欢的库：

```
$ lein try clj-time
#...
user=>
```

长话短说：只要可能，在用第三方库的实例中，你会看到要求执行 lein-try 命令。在 3.4 节中，有一个用 lein-try 尝试实例的例子。

如果实例不能通过 lein-try 运行，我们会努力提供足够的指令，说明如何在你的机器上运行该实例。

排版约定

本书使用下面的字体约定。

- 楷体
  新术语第一次出现时使用楷体，目的是强调。
- 等宽字体
  用于函数名、方法名和参数，用于数据类型、类和命名空间，在例子中表示输入和输出，在正则文本中表示字面代码。
- 等宽黑体
  用于表示命令，你应该在命令行中照样输入。
- < 可取代的值 >
  路径、命令、函数名中的元素，应该由用户提供的值来取代尖括号及其中的内容。

库的名称符合两种惯例之一：具有适当名称的库用普通字体（如 Hiccup 或 Swing），而名称与代码符号相似的库用等宽字体（如 `core.async` 或 `clj-commons-exec`）。

这个符号表示提示或建议。

这个符号表示一般注释。

这个符号表示警告或注意。

使用代码示例

补充材料（代码示例、练习等）可以在 https://github.com/clojure-cookbook/clojure-cookbook 下载。

本书的目的是帮助你完成工作。一般来说，如果示例代码出现在本书中，就可以在你的程序和文档中使用它，不需要联系我们获得许可，除非你打算复制大量的代码。例如，写一个程序，用到本书中的几段代码，不需要获得许可。而销售或分发 O'Reilly 图书的示例

CD-ROM，确实需要许可。回答问题时摘录本书并引用示例代码，不需要获得许可。在你的产品文档中包含本书的大量示例代码，则确实需要许可。

我们感谢你说明来源，但这不是必须的。来源说明通常包括标题、作者、出版商和ISBN。例如："Luke VanderHart 和 Ryan Neufeld 所著 *Clojure Cookbook* (O'Reilly). Copyright 2014 Cognitect, Inc., 978-1-449-36617-9."

如果你觉得你对代码的使用超出了合理的范围或上述允许的情况，可随时联系我们：permissions@oreilly.com.

Safari® Books Online

Safari Books Online（http://www.safaribooksonline.com）是应运而生的数字图书馆。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。技术专家、软件开发人员、Web设计师、商务人士和创意专家等，在开展调研、解决问题、学习和认证培训时，都将Safari Books Online视作获取资料的首选渠道。

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

请把对本书的意见和疑问发送给出版社。

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://oreil.ly/clojure-ckbk

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

致谢

如果没有 Clojure 社区中许多人的无私奉献，本书不可能写成。超过 65 个 Clojure 开发者响应号召，提交实例，审读，并为本书的方向提供了建议。归根到底，这是一本属于社区的书，我们只是很荣幸能够将内容组织到一起。这些贡献者是：

- Adam Bard，adambard on GitHub
- Alan Busby，thebusby on GitHub
- Alex Miller，puredanger on GitHub
- Alex Petrov，ifesdjeen on GitHub
- Alex Robbins，alexrobbins on GitHub
- Alex Vzorov，0rca on GitHub
- Ambrose Bonnaire-Sergeant，frenchy64 on GitHub
- arosequist
- Chris Allen，bitemyapp on GitHub
- Chris Ford，ctford on GitHub
- Chris Frisz，cjfrisz on GitHub
- Clinton Begin，cbegin on GitHub
- Clinton Dreisbach，cndreisbach on GitHub
- Colin Jones，trptcolin on GitHub
- Craig McDaniel，cpmcdaniel on GitHub
- Daemian Mack，daemianmack on GitHub
- Dan Allen，mojavelinux on GitHub
- Daniel Gregoire，semperos on GitHub
- Dmitri Sotnikov，yogthos on GitHub

- Edmund Jackson，ejackson on GitHub
- Eric Normand，ericnormand on GitHub
- Federico Ramirez，gosukiwi on GitHub
- Filippo Diotalevi，fdiotalevi on GitHub
- fredericksgary
- Gabriel Horner，cldwalker on GitHub
- Gerrit，gerritjvv on GitHub
- Guewen Baconnier，guewen on GitHub
- Hoàng Minh Thắng，myguidingstar on GitHub
- Jason Webb，bigjason on GitHub
- Jason Wolfe，w01fe on GitHub
- Jean Niklas L'orange，hyPiRion on GitHub
- Joey Yang，joeyyang on GitHub
- John Cromartie，jcromartie on GitHub
- John Jacobsen，eigenhombre on GitHub
- John Touron，jwtouron on GitHub
- Joseph Wilk，josephwilk on GitHub
- jungziege
- jwhitlark
- Kevin Burnett，burnettk on GitHub
- Kevin Lynagh，lynaghk on GitHub
- Lake Denman，ldenman on GitHub
- Leonardo Borges，leonardoborges on GitHub
- Mark Whelan，mrwhelan on GitHub
- Martin Janiczek，Janiczek on GitHub
- Matthew Maravillas，maravillas on GitHub
- Michael Fogus，fogus on GitHub
- Michael Klishin，michaelklishin on GitHub
- Michael Mullis，mmullis on GitHub
- Michael O'Church，michaelochurch on GitHub
- Mosciatti S.，siscia on GitHub
- nbessi
- Neil Laurance，toolkit on GitHub
- Nurullah Akkaya，nakkaya on GitHub
- Osbert Feng，osbert on GitHub
- Prathamesh Sonpatki，prathamesh-sonpatki on GitHub
- R. T. Lechow，rtlechow on GitHub