Clojure Programming

Java世界的Lisp实践

Clojure 编程

Chas Emerick, Brian Carper & Christophe Grand 著

徐明明 杨寿勋 译

O'REILLY®

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Clojure编程

Clojure Programming

Chas Emerick, Brian Carper
& Christophe Grand 著

徐明明 杨寿勋 译

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内容简介

Clojure 是一种实用的通用语言，它是传奇语言 Lisp 的方言，可与 Ruby、Python 等动态语言相媲美，更以无缝 Java 库、服务，以及拥有 JVM 系统得天独厚的资源优势而胜出。本书既可以用来熟悉 Clojure 基础知识与常见例子，也可了解其相关的实践领域与话题，更可以看到这一 JVM 平台上的 Lisp 如何帮助消除不必要的复杂性，为大家在编程实践中解决最具挑战性的问题开辟新的选择——更具灵活性、更适合于 Web 编程和操作数据库，可以应付更为苛刻的应用程序安全要求，更有效的并发性和并行处理、数据分析能力，以及在未来云环境下的更大的发展潜力。

版权贸易合同登记号　图字：01-2013-1664

图书在版编目（CIP）数据

Clojure 编程 /（美）埃默里克（Emerick,C.），（美）布卡珀（Carper,B.），（美）格兰德（Grand,C.）著；徐明明，杨寿勋译. —北京：电子工业出版社，2013.4
书名原文：Clojure programming
ISBN 978-7-121-19718-5

Ⅰ. ①C… Ⅱ. ①埃… ②布… ③格… ④徐… ⑤杨… Ⅲ. ①程序语言－语言设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2013）第 040832 号

策划编辑：张春雨
责任编辑：刘　舫
封面设计：Karen Montgomery　张　健
印　　刷：北京中新伟业印刷有限公司
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
开　　本：787×980　1/16　　印张：39.25　　字数：911 千字
印　　次：2013 年 4 月第 1 次印刷
印　　数：3000 册　　定价：99.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

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

还记得第一次接触 Clojure 是在 JavaEye 论坛上，那是庄晓丹同学的一篇名为《几行代码解决淘宝面试题之 Clojure 版》[注1] 的帖子，给我留下了很深的印象。淘宝那道面试题目是：

> “有一个很大的整数链表，需要求这个链表中所有整数的和，写一个可以充分利用多核 CPU 的代码来计算结果。”

用 Java 来解答这道题需要七八十行代码[注2]：程序员要自己手动把这个大链表切割成一个个子链表，自己手动创建多个线程去分别计算切割出来的子链表，然后再等待所有线程做完之后对结果进行归约以计算出最后的结果。确实是一道很难的面试题，涉及的知识点包括线程、线程池、线程同步等，一般人还真不一定能把它写出来。而在 Clojure 看来完全不能作为面试题，因为利用 Clojure 来解决太简单了，稍微学过一点 Clojure，掌握了 map、reduce、pmap 这几个基本函数的同学都能搞定。这个对比是很震撼的，Clojure 解决起问题来是那么简洁、优雅、自然，这算是我对 Clojure 的初体验吧。

在 Clojure 之前我也尝试过几种函数式语言，比如“最纯的函数式语言”Haskell、“天生擅长高并发的”Erlang，这些都是很棒的语言，但是学过一段时间之后就忘了，因为工作、生活中实在是用不到。而 Clojure 对我则有独特的吸引力，首先，因为它是 Lisp——一门富有传奇色彩的语言，一直希望有机会可以学习一门 Lisp 的方言；其次，Clojure 是一门接地气的语言，它运行在 JVM 这个最成功、应用最广泛的平台之上，能够跟 Java 代码无缝互操作，JVM 上所有的资源都可以为 Clojure 所用。

Clojure 是这样的有潜力、接地气，那么如果你要选择一门新语言来玩玩，不选它选什么？

对 Clojure 有了初次接触之后就有了深入学习的想法，在网络上到处寻找资料，找到的资料几乎都是英文的，中文资料少之又少。最终发现一篇比较简单而又相对完整的教程[注3]，当时就想如果把这个教程翻译成中文既可以加深自己对于 Clojure 概念的理解，也可以丰富 Clojure 的中文资料，便于以后对这门语言感兴趣的同学查阅，于是边学边把这篇教程翻译了一下。[注4] 目前用 Google 搜索 Clojure 中文资料，这篇文章还是排在第一位的，深感欣慰。

注 1：参见 *http://www.iteye.com/topic/713259*。
注 2：参见 *http://www.iteye.com/topic/711162*。
注 3：参见 *http://java.ociweb.com/mark/clojure/article.html*。
注 4：参见 *http://xumingming.sinaapp.com/302/clojure-functional-programming-for-the-jvm-clojure-tutorial/*。

“空有一身的 Clojure 本事，没地方施展啊！”学了一段时间 Clojure 之后有了这样的感觉，相信这也是很多利用业余时间学习新语言的同学的困惑。这时恰逢 Twitter 开源了他们的实时计算框架 Storm，我对实时计算很感兴趣而它又是用 Clojure 开发的。太好了，正愁没有项目练手呢，于是去读它的源代码。看到真正在生产环境中使用的 Clojure 代码才发现自己还有很多东西是不会的，对于 Clojure 只是学会了形，没有学到神。比如，Clojure 对于函数作为头等公民的强调、Clojure 对于值语义的注重，这些对于从 Java 世界过来的程序员来说，不是那么容易转换思维的（本书有专门一节强调要转换思维，讲解如何转换思维，读了之后获益良多，建议大家重点看一下）。而通过学习 Storm 源码，学到了很多这方面活生生的例子，对自己的水平有很大的提高，这里也建议每个程序员都应该参与到开源项目中去，会收益良多的。

2012 年 5 月，张春雨同学在 Clojure 中文邮件列表中发帖，要给本书找译者，因为自己之前也看过一些，而且也算有一定的 Clojure 编码经验、有一定的翻译经验，于是就把这个活应了下来。这本书大体可以分为两部分，前半部分重点讲解 Clojure 的基本概念、原理；后半部分则介绍用 Clojure 来解决实际编程中的各种问题的实例。你既可以从最基本原理开始，“自底向上”地去读；也可以从实例入手，去看看怎么用 Clojure 去解决你感兴趣的问题，遇到不懂的概念再去前半部分找对应的章节深入阅读。

翻译这本书的过程也是一个学习的过程，之前零散地学了很多 Clojure 知识，但都不是很系统。《Clojure 编程》这本书给我最大的帮助是帮我把这些零碎的知识点串了起来，形成了一个系统。它从最基本的原理讲起，在讲解各个知识点的同时由浅入深地把背后的原理一点点娓娓道来。翻译的过程中不时地发出这样的感叹：“原来那么实现是因为这个原因啊”，每天都会发现自己又多懂了那么一点点，这算是翻译过程中最大的乐趣吧，相信大家去读这本书的时候也会有类似的体会。

翻译的过程中得到了很多人的帮助，首先感谢我的老爸老妈，是你们把我带到了这个世界，才有机会去探索这神奇的计算机世界；感谢我自己，这几个月你格外辛苦，没想到你的名字也会出现在一本书的封面上，干得不错；感谢小废同学，如果没有你的捣乱，这本书可能去年就已经面市了；感谢我的合译者杨寿勋同学，没有你的通力合作，这本书不会这么快上市；感谢我们的审稿同学 haungz、孙宁、庄晓丹，是你们使得这本书的质量提高了一个档次，帮我挑出了那么多技术上的、语法上的错误；感谢策划编辑张春雨同学，没有你，这本书的译者可能就是别人了；感谢责任编辑刘舫，虽然从来没正式地打过交道，但是，从一次次反馈中感受到了你编辑工作的严谨；感谢所有帮助过我的人。

{:author “徐明明”
:date: “2013 年 3 月 6 日”
:city: ”杭州”}

前言

Clojure 是一种动态的、强类型的、寄居在 Java 虚拟机（JVM）上的语言，如今已经是它的第 5 个年头了。现今它已经被各种背景的程序员热情地应用到几乎所有的领域。Clojure 提供了一些很引人注目的特性来帮助我们解决现代编程所面临的各种挑战：

- 函数式编程的基础，包括一套性能可以和典型的可变数据结构媲美的持久性数据结构。
- 由 JVM 所提供的成熟、高效的运行时环境。
- 跟 JVM/Java 的互操作能力使得很多架构、运维方面的需求可以得到满足。
- 一套提供并发、并行语义的机制。
- 是一种 Lisp 方言，因此提供了非常灵活、强大的元编程能力。

Clojure 为那些受编程语言和环境限制而痛苦的程序员提供了一个实际可行的新选择。我们将会通过展示 Clojure 与大家每天都会使用的现有技术、类库或者服务的无缝交互来证明这一点。我们会逐步介绍 Clojure 的基础知识，从大家最熟悉的一些方面开始而不是从一些枯燥的原理开始。

这本书的目标读者是谁？

我们在写这本书的时候，脑子里面是有一些目标读者的，希望你是其中之一。

Clojure 跟你现在最喜欢的语言在表达力、简洁性和灵活性方面可以匹敌——通常还有所超越，同时还使你可以不费任何力气就能获得由 JVM 所提供的性能、类库、社区以及运维稳定性。这也使得 Clojure 很自然地成为 Java 程序员（以及使用解释型、不是那么快的非 Java 语言的 JVM 程序员）的“下一代”语言，因为它们不想因为脱离 JVM 而导致写出来的程序性能下降，或者不想因此而放弃之前在 JVM 平台上的投资。而对于那

些不想放弃语言强大表达力，但是又希望获得一个更可靠、更高效的平台以及更多可以选择的高质量类库的 Ruby、Python 程序员来说，Clojure 也是很自然的选择。

职业 Java 程序员

这个世界上有成千上万的 Java 程序员，但是有一小部分 Java 程序员工作在极限条件下解决十分重大的问题，通常是一些领域相关的问题。如果你就是这样的一个 Java 程序员，那么你可能一直在寻找着更好的工具、技术或者实践方法来提高工作效率以贡献更多的价值给你的团队、公司或者社区。而且你可能对于 Java 跟其他语言相比所表现出来的种种局限而苦恼，但是你舍不得离开 JVM 这个生态系统：它的流程成熟度、大量的第三方库、软件供应商的支持服务以及大量熟悉 Java 的程序员资源，即使其他语言的特性是那么诱人。

你会发现 Clojure 是一个很不错的选择。它以优异的性能运行在 JVM 之上，能够跟现在你已经有的所有的类库、工具以及应用进行交互，而且它比 Java 简单很多，同时又更富表达力、更简洁。

Ruby、Python 以及其他程序员

从任何一个方面来讲，Ruby 和 Python 都已经不算什么新语言了，但是最近这些年获得了极大的关注（应该可以算是“主流语言”了吧）。不难看出其中的缘由：它们都是极富表达力的动态语言、有很活跃的社区、在很多领域都鼓励最大化程序员的效率。

Clojure 将是你理想的“下一代语言”。作为一个 Ruby 或者 Python 程序员，你肯定不愿意放弃这些语言所提供的优秀特性，但是又想让你的程序在一个更强劲的平台上运行，可以有更好的运行时性能，更多的类库可供选择。高效寄居于 JVM 之上的 Clojure 完全满足这些条件，通常在程序简洁性以及工作效率方面更会超出你的期望。

在书中我们会经常把 Clojure 跟 Java、Ruby 以及 Python 进行比较来把一些概念“翻译”成你的专业领域中对应的概念。在这些比较中，我们比较的总是这些语言的权威实现：

- Ruby MRI（也称为 CRuby）
- CPython
- Java 6/7

如何阅读这本书

在规划整本书结构的时候，我们准备给大家介绍两方面的内容：一是有关 Clojure 语言的细节；另外就是一些实际的、大家都会碰到的实际问题的例子。但是我们努力避免那些不是很成功的规划办法。具体一点来说就是，我们避免以一个例子贯穿整本书，这种

方式的结果是，很多时候例子跟要叙述的概念往往不是那么相关，而且这一个例子可能对于读者来说不是那么熟悉。

有了这个概念之后，我们把这本书分成了两部分，从最基本的概念开始介绍 Clojure 的语言细节，这部分大概占全书的三分之二；从第 4 部分开始介绍一些具体的、实际生活中的例子。这种清晰的分隔符合了“duplex book”的要求（这个概念最早是 Martin Fowler 在 *http://martinfowler.com/bliki/DuplexBook.html* 里面提到的）。因此，你可以以两种方式来阅读这本书。

从 Clojure 的实际应用例子开始

学习一门语言最好的办法通常就是看怎么用它来解决实际的问题。如果你喜欢这种方式，那么希望在本书第 2 部分举的例子有跟你日常做的事情相关的例子，这样你可以自己做个比较：用你自己熟悉的语言怎么做这件事情，再看看书里介绍的用 Clojure 如何解决这个问题。在这个过程中你会遇到很多未知的概念以及语言结构，每当遇到这些问题的时候，你可以以当前的上下文作为线索去本书的第 1 部分查找对应的章节阅读。

自底向上——从 Clojure 最基本的概念开始

有时候弄懂一个东西的唯一办法是从基本的概念开始学习，如果你习惯这种方式的话，那么从第 1 章的第 1 页开始读是最好的办法了。我们尝试以渐进的方式阐述 Clojure 所有基本的原则以及结构，这样做的结果是你很少需要先阅读书的后面章节内容才能理解前面章节的内容。而当你准备自己动手实践的时候，你可以到后面的实践部分挑一个你最感兴趣、跟你要做的事情最相关的例子进行阅读。

我们是谁？

我们是通过不同途径认识并且喜欢上 Clojure 的三个程序员。在写这本书的时候，我们试图把我们对于为什么使用以及如何使用 Clojure 的经验告诉大家以让你也可以很好地使用它。

Chas Emerick

Chas 从 2008 年的早期开始就一直活跃于 Clojure 社区。他给 Clojure 语言核心贡献过代码，参与过十多个 Clojure 开源项目，并且经常会撰写或者演讲一些有关 Clojure 或者软件开发方面的东西。

Chas 维护了 Clojure Atlas（*http://clojureatlas.com*），这是一个以交互式的、可视化的方法帮助大家学习 Clojure 及其标准库的一个网站。

Chas 创建了 Snowtide 公司（*http://snowtide.com*），它坐落于马萨诸塞州西部，Chas 的主要领域在于非结构化数据的提取，并且特别擅长与 PDF 相关的数据提取。他经常在他的博客（*http://cemerick.com*）上发表一些有关 Clojure、软件开发、企业管理等方面的文章。

Brian Carper

Brian 是一个从 Ruby 转过来的 Clojure 语言爱好者。他从 2008 年开始就使用 Clojure 了，不管是自己业余的小项目还是工作中的项目，涉及的领域从 Web 开发到数据分析再到桌面应用编程等。

Brian 是 Gaka（*https://github.com/briancarper/gaka*）的作者，这是一个把 Clojure 代码转换成 CSS 的编译器；他还是 Oyako（*https://github.com/briancarper/oyako*）的作者，这是一个对象关系映射的类库。他在他的博客 *http://briancarper.net* 上发表了 Clojure 以及其他主题的一些文章。

Christophe Grand

Christophe 是一个迷失在 Java 世界的函数式编程爱好者，直到 2008 年遇到 Clojure，一见钟情！他是好几个软件的作者：Enlive（*http://github.com/cgrand/enlive*）是一个 HTML/XML 转换、提取以及模板库；Parsley（*http://github.com/cgrand/parsley*）是一个递增式的 parser 生成器；Moustache（*http://github.com/cgrand/moustache*）是一个为 Ring 设计的路由及中间件应用的 DSL。

作为一个独立咨询者，他开发并且提供 Clojure 培训。他在他的博客 *http://clj-me.cgrand.net* 上撰写了有关 Clojure 的文章。

致谢

本书跟其他任何图书一样，如果没有那么多人不知疲倦的帮助的话是不会有这本书的。

首先要感谢 Clojure 语言的作者 Rich Hickey。在短短的几年里，他设计、实现并且把 Clojure 引入了这个世界。对于很多人来说这并不只是另外一个工具而已，它重新激起了我们对编程的热爱。除此之外，他个人也教给了我们很多东西——当然是有关编程的，不过同时也学到了他的耐心、品格和眼光。多谢！ Rich。

Dave Fayram 和 Mike Loukides 对于本书雏形的形成起了至关重要的作用。当然，如果没有我们的编辑 Julie Steele 以及所有 O'Reilly 那些好人，大家现在也看不到这本书，他们帮我们搞定了出版过程中的一切事宜。

如果没有审校人员的把关，这本书不会有这么高的质量。审校人员有：Sam Aaron、Antoni Batchelli、Tom Faulhaber、Chris Granger、Anthony Grimes、Phil Hagelberg、

Tom Hicks、Alex Miller、William Morgan、Laurent Petit 以及 Dean Wampler。同时我们也想感谢那些通过 O'Reilly 论坛、E-mail、Twitter 等途径给本书早期预览版提出反馈和建议的读者们。

Michael Fogus 和 Chris Houser 在很多方面给了我们灵感。其中之一就是他们在 *The Joy of Clojure* 一书中对于跟 REPL 交互内容的显示格式，我们直接照搬过来了。

如果这里少提了任何人，那么请接受我们默默的感谢以及诚挚的歉意，最后我们很高兴能把本书带给大家！

最后，当然感谢得还远远不够

Clojure 社区已经作为我的另外一个家好多年了。这个社区里面的成员都很热情，给我以帮助，是我的榜样。特别是经常在 `#clojure` 频道里出现的人们，不但跟他们成了好朋友，也从他们那里学到了从别的地方学不到的东西。

感谢我的合作者，Christophe 和 Brian：跟你们一起撰写本书对我来说是莫大的荣耀。没有你们的话，我是绝对完不成这本书的。

感谢我的父母，Charley 和 Darleen：我对于事物运作原理无法抑制的好奇心、我对于语言以及修辞的热爱、我对于商业的兴趣跟你们早年对我的影响是分不开的。没有你们影响的话，我是不会走上现在的路的：创建自己的软件公司以及撰写这本书。

最后，感谢我的妻子 Krissy：为了让我能追求自己的理想你做出了那么多牺牲。此时说什么都显得那么苍白，我只说一句：我爱你。

—Chas Emerick，2012 年 2 月

感谢所有帮助创建 Clojure 这门语言的社区成员：感谢你们不知疲倦的辛苦工作，你们的工作使得我的职业以及个人编程生活变得那么有趣，使得我的眼界更加开阔。

感谢我的合作者 Christophe 和 Chas：我从来没有跟比你们聪明的人工作过。跟你们一起工作我很荣幸。

感谢我的妻子 Nicole：我每天晚上打字使得你不能入眠。

—Brian Carper，2012 年 2 月

感谢 Rich Hickey，你创建了 Clojure 并且培育了这么一个友好的社区。

感谢 Clojure 社区，你们帮助我提高了我的水平。

感谢我的合作者，Brian 和 Chas：跟你们合作我很荣幸。

A mon professeur Daniel Goffinet, et à ses exercices improbables, qui a radicalement changé mon approche de la programmation et de l'informatique—sur ces sujets je lui suis plus redevable qu'à nul autre.

（感谢 Daniel Goffinet 教授，他从根本上改变了我对于编程以及计算的看法——在这些主题上我最感谢他。）

A mes parents pour votre amour bien sûr mais aussi pour tout le temps à s'inquiéter que je passais trop de temps sur l'Amstrad.

（感谢我的父母：感谢你们对我的爱以及帮我买的那个 8-bit 的计算机，你们还担心我在上面花太多时间呢。）

A ma compagne Emilie, et mon fils Gaël, merci d'être là et de m'avoir supporté pendant l'écriture de ce livre.

（感谢我的妻子 Emilie 和我的孩子 Gaël：感谢你们在我写作过程中对我的支持。）

—Christophe Grand，2012 年 2 月

本书使用的一些格式约定

我们在本书中使用如下一些约定：

斜体（*Italic*）

表示一个新词组、URL、邮件地址、文件名或者文件扩展名。

等宽字体（`Constant width`）

用来表示程序代码，也会用在一个普通段落中用来引用变量或者函数名、数据库、数据类型、环境变量、表达式以及关键字。

以分号（;）开头的行

表示这行文字是由在 REPL 中运行的代码打印出来的（也就是说，打印到标准输出或者标准错误）。

以分号 + 等号（;=）开头的行

用来表示这是 REPL 中运行的代码的返回值。

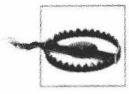
加粗的等宽字体（**`Constant width bold`**）

表示命令或者其他一些用户应该原封不动地输入的文本。

斜体的等宽字体（Constant width italic）

表示这个文本应该被替换成用户提供的值或者它的值是由上下文确定的。

这个图标表示提示、建议或者一般的标注。



这个图标表示警告或提示。

中文版书中切口以“▭”表示原书页码，便于读者与原英文版图书对照阅读，本书的索引中所列的页码为原英文版页码。

示例使用

本书的宗旨就是帮助你完成工作。一般而言，你可以在自己的程序和文档中随意使用书中的代码。除非你原样引用大量代码。否则无须联系我们获得授权。例如，在编写程序时引用了本书若干代码片段是无须授权的，然而销售或分发 O'Reilly 图书示例光盘则是需要授权许可的。通过引用本书内容以及代码的方式来答疑解难是不必有授权的。而将书中的代码大量加入到你的产品以及文档中，则需要授权。

如果你在引用书中内容时注明出处，我们将不胜感激，但是这不是必需的。引用声明通常是包含了标题、作者、出版商和 ISBN。例如：“*Clojure Programming* by Chas Emerick, Brian Carper, and Caristophe Grand (O'Reilly). Copyright 2012 Chas Emerick, Brian Carper, and Christophe Grand, 978-1-449-39470-7.”

如果你发现自己对书中代码的使用有失公允，或是违反了前述条款，敬请通过 *permissions@oreilly.com* 与我们联系。

Safari® Books Online

Safari Books Online 是一个按需索取的电子图书馆，让你能够轻松地在超过 7500 本技术及创新参考书和视频中进行检索，快速找到你需要的答案。

订阅之后，你可以在我们的在线图书馆中阅读图书并观看视频；在你的手机或移动设备上读书；在印刷前看到新标题，独家阅读撰写中的原稿，向作者进行反馈；复制粘贴代码示例，管理你的收藏，下载章节内容，在关键部分加上书签，添加笔记，打印页面，并从大量其他省时的特性中获益。

O'Reilly Media 已将本书英文版上传至 Safari Books Online。要想访问本书或其他来自 O'Reilly 和其他出版商的类似主题，可以在 *http://my.safaribooksonline.com* 进行免费注册。

如何联系我们

请将对本书的评价和存在的问题通过如下地址告知出版者：

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站，你可以在那找到关于本书的相关信息，包括勘误列表、示例代码以及其他的信息。本书的网站地址是：

http://shop.oreilly.com/product/0636920013754.do

对于本书的评论和技术性的问题，请发送电子邮件到：

bookquestions@oreilly.com

关于本书的更多信息、会议、资料中心和 O'Reilly 网络，请访问以下网站：

http://www.oreilly.com/
http://www.oreilly.com.cn/

关于作者

Chas Emerick 从 2008 年年初开始一直活跃于 Clojure 社区。他曾为 Clojure 核心语言作出贡献，参与了数十个 Clojure 开源项目，经常作关于 Clojure 和软件开发方面的写作和演讲。

Chas 维护着 Clojure Atlas（*http://clojureatlas.com*），这是 Clojure 语言和标准函数库的一个交互式可视化和辅助学习工具。

作为 Snowtide（*http://snowtide.com*，在马萨诸塞州西部的一个小软件公司）的创始人，Chas 的主要领域是无结构数据提取，特别专长于 PDF 文档相关方面。Chas 在 *http://cemerick.com* 上的写作涉及 Clojure、软件开发、企业活动及其他爱好。

Brian Carper 是从 Ruby 程序员转变成一名 Clojure 爱好者的。他从 2008 年开始用 Clojure 编程，在家里和工作上使用 Clojure 编写从 Web 开发到数据分析，到 GUI 应用程序的一切东西。

Brian 是 Gaka（*https://github.com/briancarper/gaka*，一种 Clojure 到 CSS 的编译器）和 Oyako（*http://github.com/briancarper/oyako*，一种对象－关系映射函数库）的作者。他在 *http://briancarper.net* 的写作涉及 Clojure 和其他的话题。

Christophe Grand 是函数式编程的长期爱好者，失落在 Java 王国，在 2008 年年初接触到 Clojure 时真是一见钟情！他创作了 Enlive（*https://github.com/cgrand/enlive*，一种 HTML/XML 转换、提取、模板函数库）、Parsley（*htts://github.com/cgrand/parsley*，一种渐进式解析器生成器）和 Moustache（*https://github.com/cgrand/moustache*，一种用于 Ring 的路由和中间件应用程序 DSL）。

作为独立咨询师，他开发 Clojure、辅导并提供 Clojure 方面的培训。他也在 *http://clj-me.cgrand.net* 写作一些关于 Clojure 的文章。

封面介绍

本书的封面动物是彩鹬。彩鹬（彩鹬科）有三种：大彩鹬、澳洲彩鹬和南美洲彩鹬。

这些沼泽鸟类与真正的鹬科不同，如它们的名称所暗示的，彩鹬科的毛色更为鲜亮。它们与水雉或矶鹞亲缘关系更近。彩鹬生活在沼泽和各种湿地，取食植物种子、大米、小米、昆虫、蜗牛和甲壳动物。它们除了繁殖季节，终日独居，躲藏起来，所以难得一见。

大彩鹮生活在非洲、印度和东南亚。澳洲彩鹮长期被认为是一个亚种，发现于澳大利亚，被认定为濒危物种。这两种彩鹮表现出不寻常的性二形性，雌鸟比雄鸟更大、毛色更鲜亮，而雄鸟负责孵化并养育幼鸟。

南美洲彩鹮分布在南美洲南部。它们与其他彩鹮的区别是趾上有蹼。南美洲彩鹮只与一只配偶交配，没有表现出大彩鹮和澳洲彩鹮那样的性二形性。在智利和阿根廷，人们捕猎它们为食。

封面图片来自 *Riverside Natural History* 一书。

目录