O'REILLY®

Clojure Cookbook

Clojure经典实例（影印版）

東南大學出版社

Luke VanderHart, Ryan Neufeld 著

Clojure经典实例（影印版）

Clojure Cookbook

Luke VanderHart, Ryan Neufeld 著

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo O'REILLY®

O'Reilly Media, Inc.授权东南大学出版社出版

南京 东南大学出版社

图书在版编目(CIP)数据

Clojure 经典实例：英文/(英)范德哈特(Vander Hart, L.),(英)诺伊费尔德(Neufeld, R.)著. 一影印本. 一南京：东南大学出版社，2015.2

书名原文：Clojure Cookbook

ISBN 978-7-5641-5346-5

Ⅰ.①C… Ⅱ.①范… ②诺… Ⅲ.①程序语言—语言设计—英文 Ⅳ.①TP312

中国版本图书馆 CIP 数据核字(2014)第 273737 号

江苏省版权局著作权合同登记
图字：10-2014-149 号

Clojure 经典实例(影印版)

出版发行：东南大学出版社
地　　址：南京四牌楼 2 号　　邮编：210096
出 版 人：江建中
网　　址：http://www.seupress.com
电子邮件：press@seupress.com
印　　刷：常州市武进第三印刷有限公司
开　　本：787 毫米×980 毫米　16 开本
印　　张：29.75
字　　数：583 千字
版　　次：2015 年 2 月第 1 版
印　　次：2015 年 2 月第 1 次印刷
书　　号：ISBN 978-7-5641-5346-5
定　　价：84.00 元

本社图书若有印装质量问题，请直接与营销部联系。电话(传真)：025-83791830

Preface

The primary goal of this book is to provide mid-length examples of Clojure code that go beyond the basics, with a focus on real-world, everyday applications (as opposed to more conceptual or academic issues).

Unlike many of the other books on Clojure written to date, the organizing theme of this book is not the language itself, or its features and capabilities. Instead, it focuses on specific *tasks* that developers face (regardless of what language they're using) and shows an example of how to use Clojure to solve each of those specific problems.

As such, this book is not and cannot be truly comprehensive; there are infinite possible example problems. However, we do hope we've documented some of the more common ones that most programmers encounter frequently, and that by induction readers will be able to learn some common patterns, approaches, and techniques that will serve them well as they design solutions for their own unique problems.

How This Book Was Written

An important thing you should understand about this book is that it is, first and foremost, a group effort. It is not authored by one or two people. It isn't even the work of a single, well-defined group. Instead, it is the collaborative product of more than 60 of the best Clojurists from all over the world, from all backgrounds. These authors use Clojure every day on real applications, ranging from aerospace to social media, banking to robotics, AI research to e-commerce.

As such, you will see a lot of diversity in the recipes presented. Some are quick and to the point. Others are more deliberate, presenting digestible yet penetrating insights into the philosophy and implementation of certain aspects of Clojure.

We hope that there is something in this book for readers of diverse interests. We believe that it will be useful not only as a reference for looking up solutions to specific problems, but also as a worthwhile survey of the variety and expressivity that Clojure is capable of. As we edited submissions, we were astonished by the number of concepts and techniques that were new to us, and will hopefully be new to our readers as well.

Something else that we discovered while writing and editing was how difficult it was to draw a circumference around what we wanted to cover. Every single recipe is a beautiful, endless fractal, touching multiple topics, each of which deserves a recipe, a chapter, or a book of its own. But each recipe also needs to stand on its own. Each one should provide some useful nugget of information that readers can understand and take away with them.

We sincerely hope that we have balanced these goals appropriately, and that you find this book useful without being tedious, and insightful without being pedantic.

Audience

Anyone who uses Clojure will, we hope, be able to get something out of this book. There are a lot of recipes on truly basic things that beginners will find useful, but there are also many recipes on more specialized topics that advanced users should find useful for getting a head start on implementation.

That said, if you're completely new to Clojure, this probably isn't the book to start with —at least, not by itself. It covers a great many useful topics, but not as methodically or as thoroughly as a good introductory text. See the following section for a list of general Clojure books you may find useful as prior or supplemental texts.

Other Resources

One thing that this book is not, and could never be, is complete. There is too much to cover, and by presenting information in a task-oriented recipe format we have inherently precluded ourselves from methodical, narrative explanation of the features and capabilities of the whole language.

For a more linear, thorough explanation of Clojure and its features, we recommend one of the following books:

- *Clojure Programming* (O'Reilly, 2012), by Chas Emerick, Brian Carper, and Christophe Grand. A good, comprehensive, general-purpose Clojure book focusing on the language and common tasks, oriented toward beginner Clojure programmers.
- *Programming Clojure*, 2nd ed. (Pragmatic Bookshelf, 2012), by Stuart Halloway and Aaron Bedra. The first book on Clojure, this is a clear, comprehensive introductory tutorial on the Clojure language.
- *Practical Clojure* (Apress, 2010), by Luke VanderHart and Stuart Sierra. This is a terse, no-nonsense explanation of what Clojure is and what its features do.

- *The Joy of Clojure* (Manning, 2011), by Michael Fogus and Chris Houser. This is a slightly more advanced text that really digs into the themes and philosophies of Clojure.
- *ClojureScript: Up and Running* (O'Reilly, 2012), by Stuart Sierra and Luke Vander-Hart. While *Clojure Cookbook* and the other Clojure books listed here focus mainly or entirely on Clojure itself, ClojureScript (a dialect of Clojure that compiles to JavaScript) has gained considerable uptake. This book introduces ClojureScript and how to get started with it, and covers the similarities and differences between ClojureScript and Clojure.

Finally, you should look at the source code for this book itself, which is freely available on GitHub (*http://bit.ly/clj-ckbk*). The selection of recipes available online is larger than that in the print version, and we are still accepting pull requests for new recipes that might someday make it into a future edition of this book.

Structure

The chapters in this book are for the most part groupings of recipes by theme, rather than strictly categorical. It is entirely possible for a recipe to be applicable to more than one chapter—in these cases, we have simply tried to place it where we think the majority of readers will likely look first.

A *recipe* consists of three primary parts and one secondary: problem, solution, discussion, and "see also." A recipe's problem statement lays out a task or obstacle to be overcome. Its solution tackles the problem head-on, illustrating a particular technique or library that effectively accomplishes the task. The discussion rounds everything out, exploring the solution and any caveats that may come with it. Finally, we tie off each recipe with a "see also" section, pointing you, the reader, to any additional resources or recipes that will assist you in enacting the described solution.

Chapter Listing

The book is composed of the following chapters:

- Chapter 1, *Primitive Data*, and Chapter 2, *Composite Data*, cover Clojure's built-in primitive and composite data structures, and explain many common (and less common) ways one might want to use them.
- Chapter 3, *General Computing*, is a grab bag of useful topics that are generally applicable in many different application areas and project domains, from Clojure features such as Protocols to alternate programming paradigms such as logic programming with `core.logic` or asynchronous coordination with `core.async`.

- Chapter 4, *Local I/O*, deals with all the ways in which your program can interact with the local computer upon which it is running. This includes reading fromand writing to standard input and output streams, creating and manipulating files, serializing and deserializing files, etc.
- Chapter 5, *Network I/O and Web Services*, contains recipes with similar themes to Chapter 4, *Local I/O*, but instead deals with *remote* communication over a network. It includes recipes on a variety of network communication protocols and libraries.
- Chapter 6, *Databases*, demonstrates techniques and tools for connecting to and using a variety of databases. Special attention is given to Datomic, a datastore that shares and extends much of Clojure's underlying philosophy of value, state, and identity to persistent storage.
- Chapter 7, *Web Applications*, dives in-depth into one of the most common applications for Clojure: building and maintaining dynamic websites. It provides comprehensive treatment of Ring (the most popular HTTP server library for Clojure), as well as tools for HTML templating and rendering.
- Chapter 8, *Performance and Production*, explains what to do with a Clojure program once you have one, going over common patterns for packaging, distributing, profiling, logging, and associated ongoing tasks over the lifetime of an application.
- Chapter 9, *Distributed Computation*, focuses on cloud computing and using Clojure for heavyweight distributed data crunching. Special attention is given to Cascalog, a declarative Clojure interface to the Hadoop MapReduce framework.
- Last but not least, Chapter 10, *Testing*, covers a variety of techniques for ensuring the integrity and correctness of your code and data, ranging from traditional unit and integration tests to more comprehensive generative and simulation testing, and even optional compile-time validations using static typing with `core.typed`.

Software Prerequisites

To follow along with the recipes in this book you will need valid installations of the Java Development Kit (JDK) and Clojure's de facto build tool, Leiningen. We recommend version 7 of the JDK, but a minimum of 6 will do. For Leiningen, you should have at least version 2.2.

If you don't have Java installed (or would like to upgrade), visit the Java Download Page (*http://bit.ly/java-download*) for instructions on downloading and installing the Java JDK.

To install Leiningen, follow the installation instructions on Leiningen's website (*http://leiningen.org/*). If you already have Leiningen installed, get the latest version by executing the command **lein upgrade**. If you aren't familiar with Leiningen, visit the tutorial (*http://bit.ly/lein-tutorial*) to learn more.

The one thing you *won't* need to manually install is Clojure itself; Leiningen will do this for you on an ad hoc basis. To verify your installation, run **lein repl** and check your Clojure version:

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

Some recipes have accompanying online materials available on GitHub. If you do not have Git installed on your system, follow the setup instructions (*https://help.github.com/articles/set-up-git*) to enable you to check out a GitHub repository locally.

Some recipes—such as the database recipes—require further software installations. Where this is the case, recipes will include additional information on installing those tools.

Conventions Used in This Book

Being a book full of solutions, you'll find no shortage of Clojure source code in this book. Clojure source code appears in a monospace font, like this:

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

When a Clojure expression is evaluated for a return value, that value is denoted with a comment followed by an arrow, much like it would appear on the command line:

```
(add 1 2)
;; -> 3
```

Where appropriate, code samples may omit or ellipsize return value comments. The two most common places you'll see this are when defining a function/var or shortening lengthy output:

```
;; This would return #'user/one, but do you really care?
(def one 1)

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

When an expression produces output to `STDOUT` or `STDERR`, it is denoted by a comment (`*out*` or `*error*`, respectively), followed by a comment with each line of output:

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

REPL Sessions

Seeing that *REPL-driven development* is in vogue at present, it follows that this be a REPL-driven book. REPLs (read-eval-print loops) are interactive prompts that evaluate expressions and print their results. The Bash prompt, `irb`, and the `python` prompt are examples of REPLs. Nearly every recipe in this book is designed to be run at a Clojure REPL.

While Clojure REPLs are traditionally displayed as `user=> ...`, this book aims for readers to be able to copy and paste all of the examples in a recipe and see the indicated results. As such, samples omit `user=>` and comment out any output to make things easier. This is especially helpful if you're following along on a computer: you can blindly copy and paste code samples without worrying about trying to run noncode.

When an example is *only* relevant in the context of a REPL, we will retain the traditional REPL style (with `user=>`). What follows is an example of each, a REPL-only sample and its simplified version.

REPL-only:

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

Simplified:

```
(+ 1 2)
;; -> 3

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

Console/Terminal Sessions

Console sessions (e.g., shell commands) are denoted by monospace font, with lines beginning with a dollar sign (`$`) indicating a shell prompt. Output is printed without a leading `$`:

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

A backslash (\) at the end of a command indicates to the console that the command continues on the next line.

Our Golden Boy, lein-try

Clojure is not known for its extensive standard library. Unlike languages like Perl or Ruby, Clojure's standard library is comparatively small; Clojure chose *simplicity* and *power* instead. As such, Clojure is a language full of libraries, not built-ins (well, except for Java).

Since so many of the solutions in this book rely on third-party libraries, we developed `lein-try` (*https://github.com/rkneufeld/lein-try*). `lein-try` is a small plug-in for Leiningen (*http://leiningen.org/*), Clojure's de facto project tool, that lets you quickly and easily try out various Clojure libraries.

To use `lein-try`, ensure you have Leiningen installed, then edit your user profile (*~/.lein/profiles.clj*) as follows:

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

Now, inside of a project or out, you can use the **`lein try`** command to launch a REPL with access to whichever library you please:

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

Long story short: where possible, you'll see instructions on which `lein-try` command to execute above recipes that use third-party libraries. You'll find an example of trying recipes with `lein-try` in Recipe 3.4, "Trying a Library Without Explicit Dependencies" on page 128.

If a recipe *cannot* be run via `lein-try`, we have made efforts to include adequate instructions on how to run that recipe on your local machine.

Typesetting Conventions

The following typographic conventions are used in this book:

Italic
: Used for URLs, filenames, pathnames, and file extensions. New terms are also italicized when they first appear in the text, and italics are used for emphasis.

`Constant width`
: Used for function and method names and their arguments; for data types, classes, and namespaces; in examples to show both input and output; and in regular text to show literal code.

`Constant width bold`
: Used to indicate commands that you should enter literally at the command line.

<replaceable-value>
: Elements of pathnames, commands, function names, etc. that should be replaced with user-supplied values are shown in angle brackets.

The names of libraries follow one of two conventions: libraries with proper names are displayed in plain text (e.g., "Hiccup" or "Swing"), while libraries with names meant to mimic code symbols are displayed in constant-width text (e.g., `core.async` or `clj-commons-exec`).

This element signifies a tip or suggestion.

This element signifies a general note.

This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *http://bit.ly/clj-ckbk*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Clojure Cookbook* by Luke VanderHart and Ryan Neufeld (O'Reilly). Copyright 2014 Cognitect, Inc., 978-1-449-36617-9."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of product mixes and pricing programs for organizations, government agencies, and individuals. Subscribers have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like 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, and dozens more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://oreil.ly/clojure-ckbk*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia* or *https://twitter.com/clojurecook book*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

This book would not have been possible without the selfless contributions of many within the Clojure community. Over 65 Clojurists rose to the occasion, submitting recipes, proofreading, and offering their input on the direction of the book. Ultimately, this is the community's book—we're just honored to have been able to help put it together. Those contributors are:

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

- Eric Normand, ericnormand (*https://github.com/ericnormand*) on GitHub
- Federico Ramirez, gosukiwi (*https://github.com/gosukiwi*) on GitHub
- Filippo Diotalevi, fdiotalevi (*https://github.com/fdiotalevi*) on GitHub
- fredericksgary (*https://github.com/fredericksgary*)
- Gabriel Horner, cldwalker (*https://github.com/cldwalker*) on GitHub
- Gerrit, gerritjvv (*https://github.com/gerritjvv*) on GitHub
- Guewen Baconnier, guewen (*https://github.com/guewen*) on GitHub
- Hoàng Minh Thắng, myguidingstar (*https://github.com/myguidingstar*) on GitHub
- Jason Webb, bigjason (*https://github.com/bigjason*) on GitHub
- Jason Wolfe, w01fe (*https://github.com/w01fe*) on GitHub
- Jean Niklas L'orange, hyPiRion (*https://github.com/hyPiRion*) on GitHub
- Joey Yang, joeyyang (*https://github.com/joeyyang*) on GitHub
- John Cromartie, jcromartie (*https://github.com/jcromartie*) on GitHub
- John Jacobsen, eigenhombre (*https://github.com/eigenhombre*) on GitHub
- John Touron, jwtouron (*https://github.com/jwtouron*) on GitHub
- Joseph Wilk, josephwilk (*https://github.com/josephwilk*) on GitHub
- jungziege (*https://github.com/jungziege*)
- jwhitlark (*https://github.com/jwhitlark*)
- Kevin Burnett, burnettk (*https://github.com/burnettk*) on GitHub
- Kevin Lynagh, lynaghk (*https://github.com/lynaghk*) on GitHub
- Lake Denman, ldenman (*https://github.com/ldenman*) on GitHub
- Leonardo Borges, leonardoborges (*https://github.com/leonardoborges*) on GitHub
- Mark Whelan, mrwhelan (*https://github.com/mrwhelan*) on GitHub
- Martin Janiczek, Janiczek (*https://github.com/Janiczek*) on GitHub
- Matthew Maravillas, maravillas (*https://github.com/maravillas*) on GitHub
- Michael Fogus, fogus (*https://github.com/fogus*) on GitHub
- Michael Klishin, michaelklishin (*https://github.com/michaelklishin*) on GitHub
- Michael Mullis, mmullis (*https://github.com/mmullis*) on GitHub
- Michael O'Church, michaelochurch (*https://github.com/michaelochurch*) on GitHub
- Mosciatti S., siscia (*https://github.com/siscia*) on GitHub
- nbessi (*https://github.com/nbessi*)

- Neil Laurance, toolkit (*https://github.com/toolkit*) on GitHub
- Nurullah Akkaya, nakkaya (*https://github.com/nakkaya*) on GitHub
- Osbert Feng, osbert (*https://github.com/osbert*) on GitHub
- Prathamesh Sonpatki, prathamesh-sonpatki (*https://github.com/prathamesh*) on GitHub
- R.T. Lechow, rtlechow (*https://github.com/rtlechow*) on GitHub
- Ravindra R. Jaju, jaju (*https://github.com/jaju*) on GitHub
- Robert Stuttaford, robert-stuttaford (*https://github.com/robert-stuttaford*) on GitHub
- Russ Olsen, russolsen (*https://github.com/russolsen*) on GitHub
- Ryan Senior, senior (*https://github.com/senior*) on GitHub
- Sam Umbach, sumbach (*https://github.com/sumbach*) on GitHub
- Sandeep Nangia, nangia (*https://github.com/nangia*) on GitHub
- Steve Miner, miner (*https://github.com/miner*) on GitHub
- Steven Proctor, stevenproctor (*https://github.com/stevenproctor*) on GitHub
- temacube (*https://github.com/temacube*)
- Tobias Bayer, codebrickie (*https://github.com/codebrickie*) on GitHub
- Tom White, dribnet (*https://github.com/dribnet*) on GitHub
- Travis Vachon, travis (*https://github.com/travis*) on GitHub
- Stefan Karlsson, zclj (*https://github.com/zclj*) on GitHub

Our biggest contributors also deserve special thanks: Adam Bard, Alan Busby, Alex Robbins, Ambrose Bonnaire-Sergeant, Dmitri Sotnikov, John Cromartie, John Jacobsen, Robert Stuttaford, Stefan Karlsson, and Tom Hicks. All together, these outstanding individuals contributed almost a third of the book's recipes.

Thanks also to our technical reviewers, Alex Robbins, Travis Vachon, and Thomas Hicks. These fine gentlemen scoured the book for technical errors in record time, in the 11th hour no less. Where a regular technical reviewer would merely submit textual descriptions of problems, these folks went above and beyond, often submitting pull requests *fixing* the very errors they were reporting. All in all, they were a pleasure to work with and the book is much better because of their involvement.

Finally, thanks to our employer, Cognitect, for giving us time to work on the book, and to all of our colleagues who offered advice, feedback, and best of all, more recipes!

Ryan Neufeld

First, a huge thanks to Luke. It was Luke who originally pitched the idea for the book, and I'm very grateful that he extended an invitation for me to join him in authoring it. They say the best way to learn something is to write a book on it—this couldn't be any closer to the truth. Working on the book has really rounded out my Clojure skills and taken them to the next level.

And, most importantly, I have to thank my family for putting up with me through the process of writing the book. Getting this thing off the ground has been a Herculean task and I couldn't have done it without the love and support of my wife Jackie and daughter Elody. If it hadn't been for the hundreds upon hundreds of hours of evenings, weekends, and vacation time I usurped from them, I wouldn't have been able to write this book.

Luke VanderHart

Most of all, I'd like to thank my coauthor Ryan, who worked incredibly hard to make the book happen.

Also, all of my coworkers at Cognitect provided lots of thoughts and ideas, and most importantly were a sounding board for the many questions that arose during the writing and editing process. Many thanks for that, as well as for providing the opportunity to write code in Clojure all day, every day.