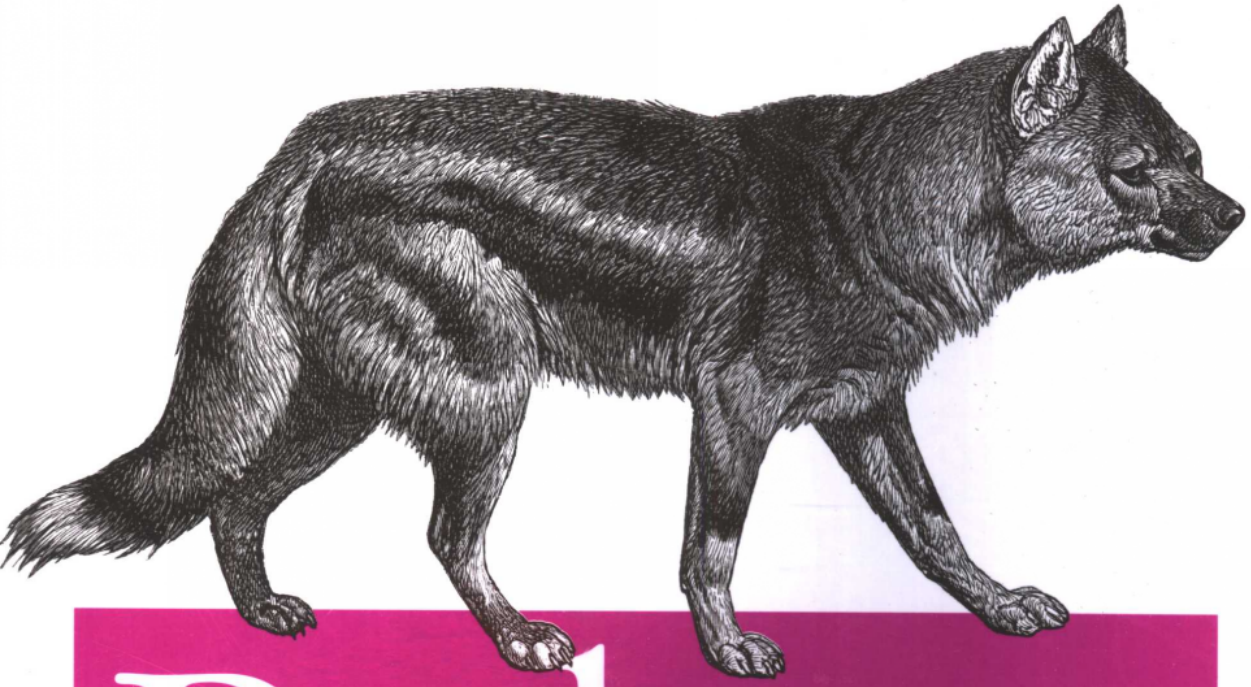
Ruby Cookbook（影印版）



Ruby Cookbook™

O'REILLY®
東南大學出版社

Lucas Carlson & Leonard Richardson 著

O'REILLY®

Ruby Cookbook（影印版）

你想探究Ruby的极致吗？Ruby Cookbook就是关于这一当今最热门编程语言的最全面的问题求解指南。本书使用清晰的阐述和数千行可以在你的项目中使用的源代码，来为在实际应用中可能碰到的数百个问题提供解决方法。从数据结构到集成前沿技术的算法，Ruby Cookbook为每一位编程人员都准备了一些专题。初学者和资深的Ruby专家都可以从本书学习到下列专题中有关Ruby编程的知识和技巧：

- 字符串和数字
- 数组和哈希表
- 类、模块和命名空间
- 反射机制和元编程
- 多任务
- 图形界面和终端界面
- 数据库
- 图像
- 互联网服务，如电子邮件、SSH和BitTorrent
- XML和HTML处理
- Ruby on Rails（包括Ajax集成）

如果你需要一个网络应用程序，本书将向你展示如何使用Rails来开始进行开发。假设你需要重命名数以千计的文件，本书将告诉你如何使用Ruby来完成诸如此类的日常任务。你还可以学习如何使用Ruby读写微软Excel表格文件，如何使用贝叶斯（Bayesian）过滤器进行文本归类以及如何创建PDF文件。甚至本书还将介绍一些冒傻气的把戏，比如如何让你的键盘灯闪烁不停。

总而言之，在目前已写成的关于Ruby的书中，Ruby Cookbook是最有用的一本。如果你需要解决一个Ruby问题，大可不必进行重复劳动，就直接在本书中寻找现成答案吧。

Lucas Carlson 是一位专注于使用Rails进行网络开发的专业Ruby程序员。他是6个Ruby库程序的作者，也对其他很多库程序有所贡献，其中包括Rails以及RedCloth。

Leonard Richardson 负责为多种编程语言开发和维护库程序，其中包括Rubyful Soup。

"程序员并不是仅仅靠语法知识就可以生存的，而是要靠他们所编写的每一行实际的代码。本书充满了实用的方法、技巧、知识和智慧。我希望它将读者的Ruby编程带入更高境界。"

– Yukihiro（Matz）Matsumoto，Ruby的作者

"你认为书中各个主题不过是些传统的日常菜谱，错，其实它们全都是被涂上了粘稠的荷兰奶酪酱的美味。哦，伙计，这就是实例：杂凑的版本系统，几个BitTorrent的客户端以及你走失的恐龙的寻物启事。有关各种协议的这一章好有趣。好像大快朵颐后浓稠香甜的酱汁将从你的嘴边流下来！对了，还有些大块的咸肉。"

– why the lucky stiff

ISBN 7-5641-0596-8

9 787564 105969 >

ISBN 7-5641-0596-8

定价：98.00元

Visit O'Reilly on the Web at
www.oreilly.com

O'Reilly Media, Inc.授权东南大学出版社出版

Ruby Cookbook(影印版)

Ruby Cookbook

Lucas Carlson & Leonard Richardson

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

Ruby 经典实例 = Ruby Cookbook /（美）卡尔森（Carlson, L.），（美）理查德森（Richardson, i ）著. —影印本. — 南京：东南大学出版社，2006.11

书名原文：Ruby Cookbook

ISBN 7-5641-0596-8

Ⅰ. R... Ⅱ. ①卡... ②理... Ⅲ. 软件开发－英文 Ⅳ. TP311.52

中国版本图书馆 CIP 数据核字（2006）第 128239 号

江苏省版权局著作权合同登记

图字：10-2006-259 号

书　　名 / Ruby Cookbook（影印版）

书　　号 / ISBN 7-5641-0596-8

责任编辑 / 张烨

封面设计 / Karen Montgomery，张健

出版发行 / 东南大学出版社（press.seu.edu.cn）

地　　址 / 南京四牌楼 2 号（邮政编码 210096）

印　　刷 / 扬中市印刷有限公司

开　　本 / 787 毫米 × 980 毫米　16 开本　56.75 印张

版　　次 / 2006 年 11 月第 1 版　2006 年 11 月第 1 次印刷

印　　数 / 0001-2500 册

定　　价 / 98.00 元（册）

O'Reilly Media, Inc.介绍

O'Reilly Media, Inc.是世界上在 UNIX、X、Internet 和其他开放系统图书领域具有领导地位的出版公司，同时是联机出版的先锋。

从最畅销的《The Whole Internet User's Guide & Catalog》（被纽约公共图书馆评为二十世纪最重要的50本书之一）到GNN（最早的Internet门户和商业网站），再到WebSite（第一个桌面PC的Web服务器软件），O'Reilly Media, Inc.一直处于Internet发展的最前沿。

许多书店的反馈表明，O'Reilly Media, Inc.是最稳定的计算机图书出版商 —— 每一本书都一版再版。与大多数计算机图书出版商相比，O'Reilly Media, Inc.具有深厚的计算机专业背景，这使得O'Reilly Media, Inc.形成了一个非常不同于其他出版商的出版方针。O'Reilly Media, Inc.所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。O'Reilly Media, Inc.还有许多固定的作者群体 —— 他们本身是相关领域的技术专家、咨询专家，而现在编写著作，O'Reilly Media, Inc.依靠他们及时地推出图书。因为O'Reilly Media, Inc.紧密地与计算机业界联系着，所以O'Reilly Media, Inc.知道市场上真正需要什么图书。

出版说明

随着计算机技术的成熟和广泛应用，人类正在步入一个技术迅猛发展的新时期。计算机技术的发展给人们的工业生产、商业活动和日常生活都带来了巨大的影响。然而，计算机领域的技术更新速度之快也是众所周知的，为了帮助国内技术人员在第一时间了解国外最新的技术，东南大学出版社和美国 O'Reilly Meida, Inc.达成协议，将陆续引进该公司的代表前沿技术或者在某专项领域享有盛名的著作，以影印版或者简体中文版的形式呈献给读者。其中，影印版书籍力求与国外图书“同步”出版，并且“原汁原味”展现给读者。

我们真诚地希望，所引进的书籍能对国内相关行业的技术人员、科研机构的研究人员和高校师生的学习和工作有所帮助，对国内计算机技术的发展有所促进。也衷心期望读者提出宝贵的意见和建议。

最新出版的一批影印版图书，包括：

- 《深入浅出 Ajax》（影印版）
- 《Ajax Hacks》（影印版）
- 《深入理解 Linux 网络内幕》（影印版）
- 《Web 设计技术手册 第三版》（影印版）
- 《软件预构艺术》（影印版）
- 《Ruby on Rails: Up and Running》（影印版）
- 《Ruby Cookbook》（影印版）
- 《Python 编程 第三版》（影印版）
- 《Python 技术手册 第二版》（影印版）
- 《Ajax 设计模式》（影印版）
- 《实用软件项目管理》（影印版）
- 《用户界面设计模式》（影印版）

Preface

Life Is Short

This is a book of recipes: solutions to common problems, copy-and-paste code snippets, explanations, examples, and short tutorials.

This book is meant to save you time. Time, as they say, is money, but a span of time is also a piece of your life. Our lives are better spent creating new things than fighting our own errors, or trying to solve problems that have already been solved. We present this book in the hope that the time it saves, distributed across all its readers, will greatly outweigh the time we spent creating it.

The Ruby programming language is itself a wonderful time-saving tool. It makes you more productive than other programming languages because you spend more time making the computer do what you want, and less wrestling with the language. But there are many ways for a Ruby programmer to spend time without accomplishing anything, and we've encountered them all:

- Time spent writing Ruby implementations of common algorithms.
- Time spent *debugging* Ruby implementations of common algorithms.
- Time spent discovering and working around Ruby-specific pitfalls.
- Time spent on repetitive tasks (including repetitive programming tasks!) that could be automated.
- Time spent duplicating work that someone else has already made publicly available.
- Time spent searching for a library that does X.
- Time spent evaluating and deciding between the many libraries that do X.
- Time spent learning how to use a library because of poor or outdated documentation.
- Time lost staying away from a useful technology because it seems intimidating.

We, and the many contributors to this book, recall vividly our own wasted hours and days. We've distilled our experiences into this book so that you don't waste your time—or at least so you enjoyably waste it on *more* interesting problems.

Our other goal is to expand your interests. If you come to this book wanting to generate algorithmic music with Ruby then, yes, Recipe 12.14 will save you time over starting from scratch. It's more likely that you'd never considered the possibility until now. Every recipe in this book was developed and written with these two goals in mind: to save you time, and to keep your brain active with new ideas.

Audience

This cookbook is aimed at people who know at least a little bit of Ruby, *or* who know a fair amount about programming in general. This isn't a Ruby tutorial (see the Resources section below for some real tutorials), but if you're already familiar with a few other programming languages, you should be able to pick up Ruby by reading through the first 10 chapters of this book and typing in the code listings as you go.

We've included recipes suitable for all skill levels, from those who are just starting out with Ruby, to experts who need an occasional reference. We focus mainly on generic programming techniques, but we also cover specific application frameworks (like Ruby on Rails and GUI libraries) and best practices (like unit testing).

Even if you just plan to use this book as a reference, we recommend that you skim through it once to get a picture of the problems we solve. This is a big book but it doesn't solve every problem. If you pick it up and you can't find a solution to your problem, or one that nudges you in the right direction, then you've *lost* time.

If you skim through this book once beforehand, you'll get a fair idea of the problems we cover in this book, and you'll get a better hit rate. You'll know when this book can help you; and when you should consult other books, do a web search, ask a friend, or get help some other way.

The Structure of This Book

Each of this book's 23 chapters focuses on a kind of programming or a particular data type. This overview of the chapters should give you a picture of how we divided up the recipes. Each chapter also has its own, somewhat lengthier introduction, which gives a more detailed view of its recipes. At the very least, we recommend you skim the chapter introductions and the table of contents.

We start with six chapters covering Ruby's built-in data structures.

- Chapter 1, *Strings*, contains recipes for building, processing, and manipulating strings of text. We devote a few recipes specifically to regular expressions (Recipes 1.17 through 1.19), but our focus is on Ruby-specific issues, and regular

expressions are a very general tool. If you haven't encountered them yet, or just find them intimidating, we recommend you go through an online tutorial or *Mastering Regular Expressions* by Jeffrey Friedl (O'Reilly).

- Chapter 2, *Numbers*, covers the representation of different types of numbers: real numbers, complex numbers, arbitrary-precision decimals, and so on. It also includes Ruby implementations of common mathematical and statistical algorithms, and explains some Ruby quirks you'll run into if you create your own numeric types (Recipes 2.13 and 2.14).
- Chapter 3, *Date and Time*, covers Ruby's two interfaces for dealing with time: the one based on the C time library, which may be familiar to you from other programming languages, and the one implemented in pure Ruby, which is more idiomatic.
- Chapter 4, *Arrays*, introduces the array, Ruby's simplest compound data type. Many of an array's methods are actually methods of the `Enumerable` mixin; this means you can apply many of these recipes to hashes and other data types. Some features of `Enumerable` are covered in this chapter (Recipes 4.4 and 4.6), and some are covered in Chapter 7.
- Chapter 5, *Hashes*, covers the hash, Ruby's other basic compound data type. Hashes make it easy to associate objects with names and find them later (hashes are sometimes called "lookup tables" or "dictionaries," two telling names). It's easy to use hashes along with arrays to build deep and complex data structures.
- Chapter 6, *Files and Directories*, covers techniques for reading, writing, and manipulating files. Ruby's file access interface is based on the standard C file libraries, so it may look familiar to you. This chapter also covers Ruby's standard libraries for searching and manipulating the filesystem; many of these recipes show up again in Chapter 23.

The first six chapters deal with specific algorithmic problems. The next four are more abstract: they're about Ruby idiom and philosophy. If you can't get the Ruby language itself to do what you want, or you're having trouble writing Ruby code that looks the way Ruby "should" look, the recipes in these chapters may help.

- Chapter 7, *Code Blocks and Iteration*, contains recipes that explore the possibilities of Ruby's code blocks (also known as *closures*).
- Chapter 8, *Objects and Classes*, covers Ruby's take on object-oriented programming. It contains recipes for writing different types of classes and methods, and a few recipes that demonstrate capabilities of all Ruby objects (such as freezing and cloning).
- Chapter 9, *Modules and Namespaces*, covers Ruby's modules. These constructs are used to "mix" new behavior into existing classes and to segregate functionality into different namespaces.
- Chapter 10, *Reflection and Metaprogramming*, covers techniques for programatically exploring and modifying Ruby class definitions.

Chapter 6 covers basic file access, but doesn't touch much on specific file formats. We devote three chapters to popular ways of storing data.

- Chapter 11, *XML and HTML*, shows how to handle the most popular data interchange formats. The chapter deals mostly with parsing other people's XML documents and web pages (but see Recipe 11.9).
- Chapter 12, *Graphics and Other File Formats*, covers data interchange formats other than XML and HTML, with a special focus on generating and manipulating graphics.
- Chapter 13, *Databases and Persistence*, covers the best Ruby interfaces to data storage formats, whether you're serializing Ruby objects to disk, or storing structured data in a database. This chapter demonstrates everything from different ways of serializing data and indexing text, to the Ruby client libraries for popular SQL databases, to full-blown abstraction layers like ActiveRecord that save you from having to write SQL at all.

Currently the most popular use of Ruby is in network applications (mostly through Ruby on Rails). We devote three chapters to different types of applications:

- Chapter 14, *Internet Services*, kicks off our networking coverage by illustrating a wide variety of clients and servers written with Ruby libraries.
- Chapter 15, *Web Development: Ruby on Rails*, covers the web application framework that's been driving so much of Ruby's recent popularity.
- Chapter 16, *Web Services and Distributed Programming*, covers two techniques for sharing information between computers during a Ruby program. In order to use a web service, you make an HTTP request of a program on some other computer, usually one you don't control. Ruby's DRb library lets you share Ruby data structures between programs running on a set of computers, all of which you control.

We then have three chapters on the auxilliary tasks that surround the main programming work of a project.

- Chapter 17, *Testing, Debugging, Optimizing, and Documenting*, focuses mainly on handling exception conditions and creating unit tests for your code. There are also several recipes on the processes of debugging and optimization.
- Chapter 18, *Packaging and Distributing Software*, mainly deals with Ruby's Gem packaging system and the RubyForge server that hosts many gem files. Many recipes in other chapters require that you install a particular gem, so if you're not familiar with gems, we recommend you read Recipe 18.2 in particular. The chapter also shows you how to create and distribute gems for your own projects.
- Chapter 19, *Automating Tasks with Rake*, covers the most popular Ruby build tool. With Rake, you can script common tasks like running unit tests or packaging your code as a gem. Though it's usually used in Ruby projects, it's a general-purpose build language that you can use wherever you might use Make.

We close the book with four chapters on miscellaneous topics.

- Chapter 20, *Multitasking and Multithreading*, shows how to use threads to do more than one thing at once, and how to use Unix subprocesses to run external commands.
- Chapter 21, *User Interface*, covers user interfaces (apart from the web interface, which was covered in Chapter 15). We discuss the command-line interface, character-based GUIs with Curses and HighLine, GUI toolkits for various platforms, and more obscure kinds of user interface (Recipe 21.11).
- Chapter 22, *Extending Ruby with Other Languages*, focuses on hooking up Ruby to other languages, either for performance or to get access to more libraries. Most of the chapter focuses on getting access to C libraries, but there is one recipe about JRuby, the Ruby implementation that runs on the Java Virtual Machine (Recipe 22.5).
- Chapter 23, *System Administration*, is full of self-contained programs for doing administrative tasks, usually using techniques from other chapters. The recipes have a heavy focus on Unix administration, but there are some resources for Windows users (including Recipe 23.2), and some cross-platform scripts.

How the Code Listings Work

Learning from a cookbook means performing the recipes. Some of our recipes define big chunks of Ruby code that you can simply plop into your program and use without really understanding them (Recipe 19.8 is a good example). But most of the recipes demonstrate techniques, and the best way to learn a technique is to practice it.

We wrote the recipes, and their code listings, with this in mind. Most of our listings act like unit tests for the concepts described in the recipe: they poke at objects and show you the results.

Now, a Ruby installation comes with an interactive interpreter called `irb`. Within an `irb` session, you can type in lines of Ruby code and see the output immediately. You don't have to create a Ruby program file and run it through the interpreter.

Most of our recipes are presented in a form that you can type or copy/paste directly into an `irb` session. To study a recipe in depth, we recommend that you start an `irb` session and run through the code listings as you read it. You'll have a deeper understanding of the concept if you do it yourself than if you just read about it. Once you're done, you can experiment further with the objects you defined while running the code listings.

Sometimes we want to draw your attention to the expected result of a Ruby expression. We do this with a Ruby comment containing an ASCII arrow that points to the expected value of the expression. This is the same arrow `irb` uses to tell you the value of every expression you type.

We also use textual comments to explain some pieces of code. Here's a fragment of Ruby code that I've formatted with comments as I would in a recipe:

```
1 + 2                       # => 3

# On a long line, the expected value goes on a new line:
Math.sqrt(1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10)
# => 7.41619848709566
```

To display the expected *output* of a Ruby expression, we use a comment that has no ASCII arrow, and that always goes on a new line:

```
puts "This string is self-referential."
# This string is self-referential.
```

If you type these two snippets of code into `irb`, ignoring the comments, you can check back against the text and verify that you got the same results we did:

```
$ irb
irb(main):001:0> 1 + 2
=> 3
irb(main):002:0> Math.sqrt(1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10)
=> 7.41619848709566
irb(main):003:0> puts "This string is self-referential."
This string is self-referential.
=> nil
```

If you're reading this book in electronic form, you can copy and paste the code fragments into `irb`. The Ruby interpreter will ignore the comments, but you can use them to make sure your answers match ours, without having to look back at the text. (But you should know that typing in the code yourself, at least the first time, is better for comprehension.)

```
$ irb
irb(main):001:0> 1 + 2      # => 3
=> 3
irb(main):002:0>
irb(main):003:0* # On a long line, the expected value goes on a new line:
irb(main):004:0* Math.sqrt(1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10)
=> 7.41619848709566
irb(main):005:0> # => 7.41619848709566
irb(main):006:0*
irb(main):007:0* puts "This string is self-referential."
This string is self-referential.
=> nil
irb(main):008:0> # This string is self-referential.
```

We don't cut corners. Most of our recipes demonstrate a complete `irb` session from start to finish, and they include any imports or initialization necessary to illustrate the point we're trying to make. If you run the code exactly as it is in the recipe, you should get the same results we did.* This fits in with our philosophy that code samples should

* When a program's behavior depends on the current time, the random number generator, or the presence of certain files on disk, you might not get the exact same results we did, but it should be similar.

be unit tests for the underlying concepts. In fact, we tested our code samples like unit tests, with a Ruby script that parses recipe texts and runs the code listings.

The `irb` session technique doesn't always work. Rails recipes have to run within Rails. Curses recipes take over the screen and don't play well with `irb`. So sometimes we show you standalone files. We present them in the following format:

```
#!/usr/bin/ruby -w
# sample_ruby_file.rb: A sample file

1 + 2
Math.sqrt(1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10)
puts "This string is self-referential."
```

Whenever possible, we'll also show what you'll get when you run this program: maybe a screenshot of a GUI program, or a record of the program's output when run from the Unix command line:

```
$ ruby sample_ruby_file.rb
This string is self-referential.
```

Note that the output of `sample_ruby_file.rb` looks different from the same code entered into `irb`. Here, there's no trace of the addition and the square root operations, because they produce no output.

Installing the Software

Ruby comes preinstalled on Mac OS X and most Linux installations. Windows doesn't come with Ruby, but it's easy to get it with the One-Click Installer: see *http://rubyforge.org/projects/rubyinstaller/*.

If you're on a Unix/Linux system and you don't have Ruby installed (or you want to upgrade), your distribution's package system may make a Ruby package available. On Debian GNU/Linux, it's available as the package `ruby-[version]`: for instance, `ruby-1.8` or `ruby-1.9`. Red Hat Linux calls it ruby; so does the DarwinParts system on Mac OS X.

If all else fails, download the Ruby source code and compile it yourself. You can get the Ruby source code through FTP or HTTP by visiting *http://www.ruby-lang.org/*.

Many of the recipes in this book require that you install third-party libraries in the form of Ruby gems. In general, we prefer standalone solutions (using only the Ruby standard library) to solutions that use gems, and gem-based solutions to ones that require other kinds of third-party software.

If you're not familiar with gems, consult Chapter 18 as needed. To get started, all you need to know is that you first download the Rubygems library from *http://rubyforge.org/projects/rubygems/* (choose the latest release from that page). Unpack

the tarball or ZIP file, change into the `rubygems-[version]` directory, and run this command as the superuser:

```
$ ruby setup.rb
```

The Rubygems library is included in the Windows One-Click Installer, so you don't have to worry about this step on Windows.

Once you've got the Rubygems library installed, it's easy to install many other pieces of Ruby code. When a recipe says something like "Ruby on Rails is available as the `rails` gem," you can issue the following command from the command line (again, as the superuser):

```
$ gem install rails --include-dependencies
```

The RubyGems library will download the `rails` gem (and any other gems on which it depends) and automatically install them. You should then be able to run the code in the recipe, exactly as it appears.

The three most useful gems for new Ruby installations are `rails` (if you intend to create Rails applications) and the two gems provided by the Ruby Facets project: `facets_core` and `facets_more`. The Facets Core library extends the classes of the Ruby standard library with generally useful methods. The Facets More library adds entirely new classes and modules. The Ruby Facets homepage (*http://facets.rubyforge.org/*) has a complete reference.

Some Ruby libraries (especially older ones) are not packaged as gems. Most of the nongem libraries mentioned in this book have entries in the Ruby Application Archive (*http://raa.ruby-lang.org/*), a directory of Ruby programs and libraries. In most cases you can download a tarball or ZIP file from the RAA, and install it with the technique described in Recipe 18.8.

Platform Differences, Version Differences, and Other Headaches

Except where noted, the recipes describe cross-platform concepts, and the code itself should run the same way on Windows, Linux, and Mac OS X. Most of the platform differences and platform-specific recipes show up in the final chapters: Chapter 20, Chapter 21, and Chapter 23 (but see the introduction to Chapter 6 for a note about Windows filenames).

We wrote and tested the recipes using Ruby version 1.8.4 and Rails version 1.1.2, the latest stable versions as of the time of writing. In a couple of places we mention code changes you should make if you're running Ruby 1.9 (the latest unstable version as of the time of writing) or 2.0.

Despite our best efforts, this book may contain unflagged platform-specific code, not to mention plain old bugs. We apologize for these in advance of their discovery. If you have problems with a recipe, check out the eratta for this book (see below).

In several recipes in this book, we modify standard Ruby classes like `Array` to add new methods (see, for instance, Recipe 1.10, which defines a new method called `String#capitalize_first_letter`). These methods are then available to every instance of that class in your program. This is a fairly common technique in Ruby: both Rails and the Facets Core library mentioned above do it. It's somewhat controversial, though, and it can cause problems (see Recipe 8.4 for an in-depth discussion), so we felt we should mention it here in the Preface, even though it might be too technical for people who are new to Ruby.

If you don't want to modify the standard classes, you can put the methods we demonstrate into a subclass, or define them in the `Kernel` namespace: that is, define `capitalize_first_letter_of_string` instead of reopening `String` and defining `capitalize_first_letter` inside it.

Other Resources

If you need to learn Ruby, the standard reference is *Programming Ruby: The Pragmatic Programmer's Guide* by Dave Thomas, Chad Fowler, and Andy Hunt (Pragmatic Programmers). The first edition is available online in HTML format (*http://www.rubycentral.com/book/*), but it's out of date. The second edition is much better and is available as a printed book or as PDF (*http://www.pragmaticprogrammer.com/titles/ruby/*). It's a much better idea to buy the second edition and use the first edition as a handy reference than to try to read the first edition.

"Why's (Poignant) Guide to Ruby," by "why the lucky stiff," teaches Ruby with stories, like an English primer. Excellent for creative beginners (*http://poignantguide.net/ruby/*).

For Rails, the standard book is *Agile Web Development with Rails* by Dave Thomas, David Hansson, Leon Breedt, and Mike Clark (Pragmatic Programmers). There are also two books like this one that focus exclusively on Rails: *Rails Cookbook* by Rob Orsini (O'Reilly) and *Rails Recipes* by Chad Fowler (Pragmatic Programmers).

Some common Ruby pitfalls are explained in the Ruby FAQ (*http://www.rubycentral.com/faq/*, starting in Section 4) and in "Things That Newcomers to Ruby Should Know" (*http://www.glue.umd.edu/~billtj/ruby.html*).

Many people come to Ruby already knowing one or more programming languages. You might find it frustrating to learn Ruby with a big book that thinks it has to teach you programming *and* Ruby. For such people, we recommend Ruby creator Yukihiro Matsumoto's "Ruby User's Guide" (*http://www.ruby-doc.org/docs/UsersGuide/rg/*). It's a short read, and it focuses on what makes Ruby different from other programming languages. Its terminology is a little out of date, and it presents its code samples

through the obsolete `eval.rb` program (use `irb` instead), but it's the best short introduction we know of.

There are a few articles especially for Java programmers who want to learn Ruby: Jim Weirich's "10 Things Every Java Programmer Should Know About Ruby" (*http://onestepback.org/articles/10things/*), Francis Hwang's blog entry "Coming to Ruby from Java" (*http://fhwang.net/blog/40.html*), and Chris Williams's collection of links, "From Java to Ruby (With Love)" (*http://cwilliams.textdriven.com/pages/java_to_ruby*) Despite the names, C++ programmers will also benefit from much of what's in these pieces.

The Ruby Bookshelf (*http://books.rubyveil.com/books/Bookshelf/Introduction/Bookshelf*) has produced a number of free books on Ruby, including many of the ones mentioned above, in an easy-to-read HTML format.

Finally, Ruby's built-in modules, classes, and methods come with excellent documentation (much of it originally written for *Programming Ruby*). You can read this documentation online at *http://www.ruby-doc.org/core/* and *http://www.ruby-doc.org/stdlib/*. You can also look it up on your own Ruby installation by using the `ri` command. Pass in the name of a class or method, and `ri` will give you the corresponding documentation. Here are a few examples:

```
$ ri Array                    # A class
$ ri Array.new                # A class method
$ ri Array#compact            # An instance method
```

Conventions Used in This Book

The following typographical conventions are used in this book:

Plain text
: Indicates menu titles, menu options, menu buttons, and keyboard accelerators (such as Alt and Ctrl).

Italic
: Indicates new terms, URLs, email addresses, and Unix utilities.

`Constant width`
: Indicates commands, options, switches, variables, attributes, keys, functions, types, classes, namespaces, methods, modules, properties, parameters, values, objects, events, event handlers, XML tags, HTML tags, macros, programs, libraries, filenames, pathnames, directories, the contents of files, or the output from commands.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book 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: "*Ruby Cookbook*, by Lucas Carlson and Leonard Richardson. Copyright 2006 O'Reilly Media, Inc., 0-596-52369-6."

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*.

Comments and Questions

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://www.oreilly.com/catalog/rubyckbk

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our web site at:

http://www.oreilly.com

Acknowledgments

First we'd like to thank our editor, Michael Loukides, for his help and for acquiescing to our use of his name in recipe code samples, even when we turned him into a talking frog. The production editor, Colleen Gorman, was also very helpful.