The Pragmatic Programmers

Broadview
WWW.BROADVIEW.COM.CN

Programming Ruby 中文版 第2版

Programming Ruby: The Pragmatic Programmers' Guide, Second Edition

[美] Dave Thomas
Chad Fowler
Andy Hunt
著

孙勇
姚延栋
张海峰
译

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

Second Edition
Includes Ruby 1.8

Programming Ruby 中文版，第2版

Programming Ruby: The Pragmatic Programmers' Guide, Second Edition

[美] Dave Thomas
Chad Fowler
Andy Hunt 著

孙 勇 姚延栋 张海峰 译

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

Ruby 是一种跨平台、面向对象的动态类型编程语言。Ruby 体现了表达的一致性和简单性，它不仅是一门编程语言，更是表达想法的一种简练方式。它不仅受到广大程序员的欢迎，无数的软件大师亦为其倾倒。*Programming Ruby* 是关于 Ruby 语言的一本权威著作，也被称为 PickAxe Book（镐头书，由封面上的工具得名）。本书是它的第 2 版，其中包括超过 200 页的新内容，以及对原有内容的修订，涵盖了 Ruby 1.8 中新的和改进的特性以及标准库模块。它不仅是您学习 Ruby 语言及其丰富特性的一本优秀教程，也可以作为日常编程时类和模块的参考手册。

本书适合各种程度的 Ruby 程序员，无论新手还是老兵，都会从中得到巨大的帮助。

0-9745140-5-5 Programming Ruby: The Pragmatic Programmers' Guide, Second Edition by Dave Thomas, with Chad Fowler, Andy Hunt

版权贸易合同登记号　图字：01-2006-2241

图书在版编目（CIP）数据

Programming Ruby 中文版：第 2 版 /（美）托马斯（Thomas,D.），（美）弗沃尔（Fowler,C.），（美）亨特（Hunt,A.）著；孙勇，姚延栋，张海峰译. —北京：电子工业出版社，2007.3
书名原文: Programming Ruby: The Pragmatic Programmers' Guide, Second Edition
ISBN 978-7-121-03815-0
I. P… II.①托…②孙…③姚…④张… III.计算机网络－程序设计 IV.TP393.092

中国版本图书馆 CIP 数据核字（2007）第 010837 号

策划编辑：方　舟
责任编辑：陈元玉
印　　刷：北京市天竺颖华印刷厂
装　　订：三河市金马印装有限公司
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：54.75　　字数：1 000 千字
印　　次：2007 年 3 月第 1 次印刷
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，联系电话：（010）68279077；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。

世界各地的开发者对 *Programming Ruby* 和 Ruby 语言的评论

“Ruby 是一门非常强大而有用的语言，无论何时我都用它工作，而这本书也总在我身边。”

▶ Martin Fowler，首席科学家，ThoughtWorks

“如果你的世界像我一样以 Java 为主，那么你需要这本杰出的书来学习所有你错过的精彩内容。只要看一眼，你的 Java 世界就会被撼动。实际上，读了仅仅几页 *Programming Ruby* 之后，再使用 Ruby 之外的语言编程感觉就像做无用功（push rope）。”

▶ Mike Clark，作家、顾问

Ruby 是巧妙、优雅而有趣的语言，值得写一本巧妙、优雅而有趣的书。*Programming Ruby* 的第 1 版就是这样一本书，而第 2 版则更为出色。

▶ James Britt，管理员，http://ruby-doc.org

“学习一门新的编程语言的原因是学习以不同的方式思考。学习以 Ruby 的方式去思考的最好方式是读 *Programming Ruby*。几年前我就是用本书的第 1 版这样做的。从那时起，我拥有了持续的**令人愉快**的 Ruby 编程体验。这源自学习此语言时所用的高质量的资源。据我所知，我并不是唯一一个声称每种语言都需要像这样一本高质量的书的人。”

▶ Chad Fowler，Ruby Central 公司董事

“这本书使我开始学习 Ruby。它至今仍旧是我遇到问题时转而求助的第一本书。”

▶ Ryan Davis，Seattle.rb 的创立者

“这本书改变了我的生活。听起来有点陈词滥调，但这是事实。使用了 6 年 Java，写了 300 000 行代码后，我需要一个转变了。当我读到本书的第 1 版时开始了这种转变。借助可靠的社区支持以及持续增长的强大类库基础，我创建了一个主要以应用 Ruby 解决现实问题来盈利的公司。Ruby 已经准备好步入黄金时段，而这本书的最新版将会向等待它的世界展示 Ruby 是一块多么瑰丽的宝石。”

▶ Rich Kilmer, InfoEther 有限责任公司首席执行官

“本书的第 1 版已经是我多年的桌旁伴侣。第 2 版将是它急切等待的替代者。”

▶ Tom Enebo，JRuby 开发者

"*Programming Ruby* 的第 1 版在日本之外得到了大规模的介绍，逐渐成为发布语言参考的事实标准，而且成了清晰、高效的科技写作再三引用的典型。扩充的第 2 版的出现令世界各地的 Ruby 程序员倍感兴奋，而且必定会引起学习这一优雅而又强大的语言的新浪潮。"

▶ David A. Black，博士，Ruby Central 公司董事

"对我来说，Ruby 绝对是解决脚本和原型问题的首选，这本书将会帮助你发现它的用途和优雅之处。除此之外，阅读本书也是一种乐趣。"

▶ Robert Klemme

"当本书的第 1 版发布时我买了一本，用它来学习 Ruby 极其出色。于是我买了第二本放在家中。但是从那时起 Ruby 发生了很多变化。我很高兴看到 *Programming Ruby* 的第 2 版面世，它将再一次帮助程序员来学习这个奇妙而漂亮的语言。这不仅对 Ruby 新手来说是好消息，像我这样的 Ruby 开发者也想要一本（不，是两本），以便 Ruby 的所有最新细节都变得唾手可得。"

▶ Glenn Vanderburg，Countrywide Financial 软件架构师

Ruby 是一门可以花一个下午来学习使用而花多年（可能是一生）来深入掌握的伟大语言之一。在 C 中，我总是要解决语言的一些局限；而在 Ruby 中，我总是能发现解决问题的更简洁、更优雅的方式。*Programming Ruby* 是阐释 Ruby 语言最精彩的读物。它不仅教你语法，还教你语言的精神和对它的感觉。

▶ Ben Giddings

孔子说，"你听到，会忘记"。他还说，"你做了，会理解"。但实际上"做"并不容易，除非使用支持快速简洁原型设计的强大语言。对我来说，这种语言就是 Ruby！谢谢！"

▶ Michael Neumann

推荐序一

孟 岩

如果你想掌握 Ruby，这本书是最好的起点。如果你想运用 Ruby，这本书也是案头必备。所以，如果你已经决定要走入 Ruby 的世界，那么这本书是必经之路，而本不需要一篇“推荐序”。

问题在于，我们为什么还要学习一种新的语言？特别是当 Ruby 整体上仍然是一个没有完全成熟的“小语种”的时候，为什么要把宝贵的精力投入到 Ruby 中？这才是我想讨论的问题。

跟很多人一样，我学习程序设计是从 Basic 语言开始的。然而在初步了解了程序设计的基本概念之后，我便迅速地转向了 C 语言，并且在上面下了一番苦功夫。是 C 语言帮助我逐步理解了计算机系统以及算法、数据结构等基础知识，从而迈入程序设计的大门之中的。C 语言出色地实现了真实计算机系统的抽象，从而表现出极佳的适应性和强大的系统操控能力。直到今天，我仍认为认真理解和实践 C 语言是深入理解计算机系统的捷径。然而，当我尝试着用 C 语言来干点实事的时候，立刻发现一些问题：C 的抽象机制过于简单，容易犯错，开发效率低下，质量难以稳定。这时，有一位老师向我介绍了 Visual Basic，他盛赞 VB 是“最高级的开发语言”，代表未来的发展方向。正好当时有一个学校的项目摆在我面前，我几乎没有学习，凭着自己那点 Quick Basic 的底子，借助在线帮助，迅速地将项目完成，并且获得了好评。

通过这些初级的实践，我体会到 VB 在开发应用程序方面所具有的惊人的高效率，进而意识到，C 与 VB 是两类设计目标完全不同的语言。以 C 语言为代表的系统编程语言的优势在于能够充分发挥机器的性能，而像 VB 这样的语言的优势则是充分提高人的效率。事实上，执行性能与开发效率是软件开发中的一对矛盾，所有的程序设计语言都必须面对这个矛盾，作出自己的选择。

在当时，大多数新语言的选择是上下通吃。它们一方面提供了丰富多彩的高级抽象，另一方面又提供了强有力的底层操作能力，希望由此实现高性能与高效率的统一。C++、Java、C#和 Delphi 都是走的这条路线，甚至 VB 从 5.0 开始也强化了底层操作机制，并提供了编译模型，不落人后。

我当时选择了 C++。对于熟悉 C 语言的我来说，这是一个自然而然的选择。深入学习 C++是一个漫长而艰苦的过程，不但要理解伴随面向对象和泛型而来的大量概念，牢记各种奇怪的语法规则，还要了解实践应用中大量存在的陷阱，掌握一系列“模式”、“惯用法”和“条例”。在克服这种种困难的过程中，我对程序设计的认识确实得到了强有力的提升，但是 C++真的实现了执行性能和开发效率的双丰收了吗？很遗憾，答案是“否”。我自己的体会是，使用 C++要花费大量精力来进行试探性设计，还可能要投入巨大精力来消除代码缺陷，因此开发效率不高。一些权威的调查显示，在新项目的执行中，C++的开发效率甚至低于 C，这不得不发人深思。C++提供了大量的语言机制，又存在一些陷阱，想要实现良好的运行性能，就必须遵守一系列清规戒律，这就带来了思想上的不自由，其结果往往是效率的低下。

随后流行的 Java 和 C#等语言，有效地消除了 C++中存在的一些陷阱和缺陷，并且提供了很好的基础设施和开发环境，大大提高了开发效率。但是它们都设定了严整的结构和规则，进行严格的编译期检查，强调规范、纪律和计划。这些语言的拥护者们骨子里都认为，构造大规模软件是一个严肃的工程，必须施加强有力的约束和规范，时时刻刻预防和纠正开发者可能犯下的错误。当然，这样必然会导致对开发者思维的束缚和思考效率的限制，从而导致宏观上的低效。但他们认为，这是构造大型高质量软件所必须付出的代价。

然而，世界上另外有一些人的想法完全不同。他们认为，一种“可上九天揽月，可下五洋捉鳖”的终极语言即使能够实现，也注定是低效的。既然这个世界本身就是丰富多彩的，为什么不保留编程语言的多样性，各取所需，各得其所，彼此协同合作，不亦乐乎？既然 C 语言在充分发挥机器性能方面已经登峰造极，那么就应该尽力创造能够充分发挥人脑效能的程序设计语言。当人的效率充分提高的时候，所有的问题都会迎刃而解，或者至少大为简化。

1998 年，我读到一本薄薄的小书，叫做《Perl 入门》。这本书介绍了一种完全陌生的语言，不但看上去稀奇古怪，而且骨子里透露出来一股与 C++、Java 等完全不同的气质：狂放、不羁、乖张、散漫，无法无天。对于当时的我来说，这一切令人诧异，难以接受。而 Perl 的拥护者似乎也懒得挑战“主流意识形态”，他们自嘲地说，Perl 就是骆驼，样子不好看，气味也不好闻，但就是能干活。随后进入视野的 Python 外表温文尔雅，简朴工整，但其实质与 Perl 一样，也是崇尚自由灵活，追求简单直接。伴随着 Web 走来的 JavaScript 和 PHP，虽然外表上差别很大，但是总体上看，与 Perl、Python 一样，都是把自由放在结构之上，把人放在机器之上的语言。人们称它们为“动态语言”或者“脚本语言”。这些语言的出现和流行，强有力地挑战了过去 20 年来人们深信不疑的传

统观念。传统认为编译阶段的类型检查至关重要，可是动态语言却把这类检查推迟到执行阶段的最后一刻才进行；传统认为严整的结构和严肃的开发过程是必不可少的，可是动态语言却能够用一种看上去随意自由的风格去“堆砌”系统；传统把精致的模型和多层次的设计视为最佳实践，而动态语言却往往是蛮不讲理地直来直去；传统把频繁的变化和修改视为不良过程的标志，而动态语言却将此视为自然而然的工作方式；传统认为必须在机器性能与人的效率之间取得折中，动态语言却偏执地强调人的效率，而把机器性能这档子事情抛到九霄云外。动态语言离经叛道，却又大获成功，不由分说地把人们好不容易搭建起来的那个严谨的、精致的圣殿冲击得摇摇欲坠。人们怀着复杂的心情观察着动态语言的发展，猜度着它的方向和影响。

这时候我发现了 Ruby。

很多转向 Ruby 的人，在谈论这种语言的时候，都提到“乐趣”这个字眼，我也不例外。使用 Ruby 编程，你会体验到一种难以名状的趣味，这种感觉，就好像是突然掌握了十倍于从前的力量，同时又挣脱了长久以来一直束缚在身上的枷锁一般。“白日放歌须纵酒，青春作伴好还乡”，当年临摹“Hello World”时的淳朴热情仿佛又回到了身上。Ruby 实现了最纯粹意义上的面向对象，让 Smalltalk、Perl 和 Lisp 的灵魂在新的躯壳里高歌。相比于 Python，Ruby 的思想更加清晰一致，形式更加灵活；相比于 Perl，Ruby 更简单质朴，绝少光怪陆离之举；相比于 Smalltalk 和 Lisp，Ruby 更富有现代感和实干气质；相比于庙堂之上的“工业语言”，Ruby 自由挥洒、轻快锐利；而相比于 JavaScript 和 PHP，Ruby 从 Smalltalk 继承而来的深厚底蕴又大占优势。面对执行性能与开发效率的谜题，Ruby 毫不犹豫地选择了开发效率，选择了对人脑的友好。Ruby 的基本思想非常简单淳朴，对于基本原则的坚持非常彻底，毫不打折扣，而在具体应用中又集各家所长，实现了巧妙的平衡。从我自己的体验来讲，使用 Ruby 写程序，与使用 C++等主流语言感觉完全不同，没有战战兢兢的规划和设计，没有紧绷的神经，没有一大堆清规戒律，有的是一种闲庭信步的悠然，有的是时不时灵光一闪的洋洋自得，有的是抓住问题要害之后猛冲猛打的快感。我的水平还不高，还不能够设计更精妙的框架和 DSL（领域语言），但我相信我已经初步体会到了 Ruby 的乐趣。

自从这本书的第 1 版出版以来，Ruby 的发展越来越快。最初人们用 Ruby 来完成一些临时的任务，然后就迅速被它迷住了，越来越多有用和有趣的东西被开发出来，越来越多有才华的人加入了 Ruby 的阵营，从而形成了一股潜流。这潜流在 2003 年以后就已经出现了，只是还没有引起外部世界的注意。然而自 2005 年以后，潜流变成了潮流，越来越多的人被卷进来。整个社群围绕着 Ruby 的质朴与自由，创造了大量的珍宝（gems）。有人用 200 行 Ruby 代码写了一个飞快的全文搜索引擎，有人把(X)HTML 的

解析变成了 CSS Selector 的有趣复用，有人用 13 行 Ruby 代码完成了一个 P2P 文件共享程序，有人把 Google 引以为豪的 MapReduce 算法轻巧地实现在一个小小的 Ruby 程序库中。当然，最为人津津乐道的还是 Ruby on Rails 在 Web 开发领域掀起的风暴。所有这一切，都令人对 Ruby 报以越来越热烈的掌声。

今天，Ruby 已经成为世界上发展最快的程序设计语言之一，一个充满热情和创造力的社群围绕着它，开展着种种激动人心的工作。在这里没有什么豪言壮语，但是所有的工作都在扎扎实实地推进，人们被自己内心的力量驱动着，而这种力量来自于 Ruby 质朴和自由的乐趣，它是近于纯粹的。

我深深地知道，Ruby 今天还不是“主流”，其前景究竟如何，也不是没有争论。在今天这个时代，选择技术路线是一件关乎生计和名利的事情，是不纯粹也不能纯粹的事情。在各种场合，年轻的或是年长的程序员们头头是道地分析着 IT 大局，掰着指头计算着技术的“钱”途，这些都无可厚非，甚至非常必要。但是，作为一个把程序设计当成自己的事业，或者至少是职业的人，总应该在自己的内心中留下那么一小片空间，在那里抛开一切功利，回归质朴，追求纯粹的编程的乐趣。如果你跟我一样，希望体会这样一种内心的乐趣，我热情地邀请你翻开这本书，加入 Ruby 社群。也请你相信，当越来越多的人为着自己的乐趣而共同努力的时候，我们就能创造历史。

孟　岩

《程序员》杂志技术主编

2006 年 12 月于北京

推荐序二

熊　节

根据我的观察，习惯于 Java 或者 C#的程序员在初初接触 Ruby 时，最能打动他们的往往就是像本文标题这样的一句代码：原本熟悉的字符串或者整数突然摇身一变，有了很多新的行为，甚至让整个 Ruby 语言都似乎变了个样。尽管“改变标准库的行为”并不总是值得推荐的做法，但如果使用得当，你能够在 Ruby 的基础上创造出一种贴近项目需求、易写易读的方言——也有人把这些方言叫做“领域专用语言”（DSL，Domain Specific Language）。更多的程序员是因为 Rails 这个框架才开始对 Ruby 语言产生兴趣，而 Rails 在很大程度上正是一种针对 Web 应用开发的 DSL。

能够创造 DSL，这是 Ruby 语言最大的魅力之一。但仅仅这一点并不足以解释为何有那么多优秀的程序员如此盛赞 Ruby 语言，更不足以解释为何它会突然间红透半边天——毕竟，在元编程方面更具实力的 LISP 和 Smalltalk 并没有像如今的 Ruby 这样流行。作为一个 Java 程序员的 Mike Clark 给了我们一个有趣的比喻：推绳子——他说“读了仅仅几页 Programming Ruby 之后，再使用 Ruby 之外的语言编程感觉就像是在‘推绳子’（push rope）。”把一根软绵绵的绳子往前推，那种有劲使不上的感觉，正是用惯 Ruby 之后再回到 Java/C#时的真实感受。

灵活、优雅、巧妙、便利……这些溢美之词我们已经听得太多了。但在我看来，Ruby 最大的特点就一个字：快。这不仅意味着你能够很快地为自己的问题找到现成的解决办法，更意味着你能够直观地描述自己心中的想法，并且在改变想法时能够很快地调整你的程序。这种能力对于今天的软件开发者而言显得尤为重要，因为世界在飞快地改变，软件项目的需求在飞快地改变。对于今天的软件客户来说，尽快得到可以工作的软件、尽快反馈、尽快看到调整的效果，比一个完美但尚未实现的设计要有价值得多。而 Ruby 这种“快速实现想法”的能力，正是众多开发者对之青睐有加的根源所在。

Ruby 能够帮你描述心中所想——这句话，在某种意义上，也意味着你需要熟悉 Ruby 的思考方式。尽管自称是面向对象的脚本语言，Ruby 的精神仍然与函数式编程（functional programming）一脉相承。这种精神不仅体现在语法层面上，还体现在构建系统的思路上。Ruby 社群很少会一开始就把要实现的目标想得清清楚楚，或是首先制定种

种规范标准；相反，他们会充分利用 Ruby 的灵活与简洁、优雅与巧妙，从一个简单的、能够工作的软件开始，逐步增加更多的功能，并通过不断重构和优化让良好的设计逐渐浮现。

是以，跨进 Ruby 的世界，也许你首先需要学会的是这种“渐进式”的思维方式——不仅仅是编写软件，就连“学习 Ruby 语言”本身也一样。你无须读 18 本书或者参加半年培训来学会 Ruby 编程——另一方面即便你这么做了也未必就能学会，如果你没有使用 Ruby 来编写真正有用的程序的话。所以，如果你对 Ruby 产生了兴趣，稍微了解一下，然后就开始写吧：把编写 shell 脚本的首选语言从 Perl 改为 Ruby，用 Rake 来构建你的项目，或者——像大多数人那样——用 Rails 来开发一个小网站。你会遇到无数的问题，解决这些问题的过程就是对你的技术进行重构的过程。

但你至少还需要通过某种途径来“稍微了解”Ruby 语言，而且在遇到问题时也需要一本手册来帮你排疑解难。在你手上的这本 *Programming Ruby* 正是为此而出的一本书。书中的精彩内容无须我在这里赘述，你大可以自己去发掘。我唯一想要告诉你的是：如果你想要开采最瑰丽的“红宝石”宝藏，这本书就是你不可或缺的“镐头”。锻造这柄镐头的是两位大名鼎鼎的“实用主义程序员”Dave Thomas 和 Andy Hunt，这两位撰写过一系列 C++/Java/.NET 技术图书的开发者最终选择用 Ruby on Rails 来开发他们自己的网站（PragmaticProgrammer.com），本身就已经证明了 Ruby 的价值，同时也让我们对本书的实用性更有信心。

所以，你还在犹豫什么呢？既然已经拿起了这本书，既然已经对 Ruby 产生了兴趣，就不要再浪费时间了。翻开书，跟着这两位讲求实效的作者一道，现在就开始你的 Ruby 编程之旅吧。Ruby 已经向你说过“hello”了，你将会如何回应它呢？

最后，和以往一样，祝你在 Ruby 的世界里，编程快乐！

熊　节
ThoughtWorks 咨询师
2006 年 12 月于西安

译 序

关于 Ruby 语言及相关技术，非常感谢孟岩兄和熊节兄在前面所做的精辟入里的分析与推荐，这里就不再赘述了。相信您读了之后，一定是已经怦然心动而跃跃欲试了。

借此机会，我们还要感谢许多人。

首先要感谢博文视点公司引进此书，并将如此重要的一部书交托给我们来翻译，希望能幸不辱命。也要感谢本书的两位编辑方舟和陈元玉，没有二位认真、辛苦的工作，本书不可能有现在的翻译质量。我们也从他们的技术和文字校订中学到了很多。

另外译者也要相互感谢一下。在本书翻译的四个月期间，姚延栋刚做了父亲，张海峰也是家有幼子，并开始了新的职业征途，但两位都竭力保证了翻译的进度和质量。孙勇则不辞辛苦地对全书进行了统稿。当然，一定要感谢家人对我们的宽容和理解。

由于译者对 Ruby 语言也是新学，水平有限，难免在译稿中存有这样那样的错误。如果您有技术或文字方面的问题，欢迎致信 progruby.cn@gmail.com，我们会尽力帮您解答。

译 者

2006 年 12 月于北京

第 1 版序

Foreword to the First Edition

人受创造之驱动；我清楚自己非常喜欢创造。尽管我不善于油画、素描或者音乐，但是我可以写软件。

在接触计算机后不久，我就对编程语言产生了浓厚的兴趣。我坚信一定可以实现一种理想的编程语言，而我希望能成为其设计者。后来在积累了一些经验之后，我意识到设计这种理想的、通用的语言比我想象的要难得多，但是我依旧希望能设计一种能解决我日常遇到的大多数问题的语言，这是我学生时代的梦想。

多年后，我和同事谈起了脚本语言以及它们的处理能力和潜力。作为一个超过 15 年的面向对象的爱好者，我认为面向对象编程也非常适合脚本语言。我在网络上做了一段时期的调查，但是我找到的候选，例如 Perl 和 Python，都不是我真正需要的。我需要一个比 Perl 更强大、比 Python 更面向对象的语言。

后来，我想起了以前的梦想，于是决定自己设计一种语言。起初我只是在工作时用来自娱。但是，逐渐地它成长为一个足以替代 Perl 的工具。我命名为 Ruby —— 先前的名字叫 Red Stone —— 并在 1995 年公开发布。

从那时起，越来越多的人开始对 Ruby 感兴趣。不管你相信与否，现在 Ruby 在日本实际上已经比 Python 更流行了。我希望最终它能在全世界被广泛地接受。

我认为生活的目的是快乐，至少部分是快乐。基于这种信念，Ruby 被设计成一种使编程不但容易而且有趣的语言。它能让你面对更少的压力，集中精力于程序设计中的有创造性的一面。如果你不相信我，请阅读此书并试用一下 Ruby。我确信你自己会体会到这一点。

我非常感谢那些参加 Ruby 社区的人；他们给了我莫大的帮助。我总是觉得 Ruby 是我的一个孩子，但实际上它是许多人共同努力的结晶。没有他们的帮助，Ruby 不可能发展成现在这样。

我要特别感谢此书的作者，Dave Thomas 和 Andy Hunt。Ruby 一直不是一个文档齐全的语言。因为我喜欢写程序胜于写文档，所以 Ruby 手册没有应有的那样完善。你必须通过阅读源代码来获悉语言的准确行为。但是现在 Dave 和 Andy 为你做了这一工作。

他们对来自远东的一个不怎么出名的语言产生了兴趣。他们研究它，阅读了成千上万行源代码，编写了不计其数的测试脚本和电子邮件，澄清了语言的某些不明确的行为，发现了许多 bug（甚至修正了其中的许多），并最终编写了这本出色的书。毫无疑问，Ruby 因此已经成了一个文档完善的语言。

他们在本书上花的精力绝不是微不足道的。当他们在写此书时，我对语言本身进行了一些修改。我们一起为这些更新而工作，并使此书尽量准确。

我希望 Ruby 和本书能使你编程更容易和有趣。工作得开心！

Yukihiro Matsumoto, a.k.a. "*Matz*"

まつもとゆきひろ

2000 年 10 月于日本

第 2 版序

Foreword to the Second Edition

在 1993 年，没有人会相信一个由日本业余语言设计者创造的面向对象语言能最终在世界范围内被广泛使用并变得几乎像 Perl 那样流行。这种想法被认为是极其愚蠢的，我承认这一点，我也不相信。

但是这确实发生了，它大大超出了我的期望。这是——至少部分是——由本书的第 1 版引起的。著名的 *Pragmatic Programmers* 选择了一个在日本之外没有名气的动态语言，并写了一本关于它的好书。奇迹就这样发生了。

那都是过去的事情了，未来将从现在开始。我们有了 *Programming Ruby* 的第 2 版，它比第 1 版更好。这已不再是奇迹。这次，持续增长的 Ruby 社区帮助编写了此书。而我只须参与社区一起工作。

我真的很感谢 Pragmatic Programmers，Dave Thomas 和 Andy Hunt 以及来自社区的其他帮助编写此书的人（抱歉不能一个个列出）。我非常热爱和睦的 Ruby 社区。这是我所见过的最好的软件社区之一。我也非常感谢全世界使用 Ruby 的每一位程序员。

宝石已经开始旋转，它的光芒将闪耀整个地球。

Yukihiro Matsumoto, a.k.a. "*Matz*"

まつもとゆきひろ

2004 年 8 月于日本

前言

Preface

本书是 *Programming Ruby* 的第 2 版，*Programming Ruby* 已广为 Ruby 爱好者所知。它是 Ruby 编程语言的教程和参考文献。如果你有本书的第 1 版，你会发现这一版有了重大变化。

当 Andy 和我写第 1 版的时候，我们不得不介绍 Ruby 的背景和吸引力。其中我们写道："当发现 Ruby 时，我们意识到找到了一直在寻找的东西。和我们曾经用过的任何语言相比，Ruby **都称得上是罕见的**。你可以集中精力于解决手头上的问题而不是与编译和语言本身周旋。这就是它使你成为一位更好的程序员的方法：通过让你将时间花在为用户创建解决方案上而不是编译上。"

今天这种信念变得更强了。4 年后，Ruby 仍旧是我们的选择：我用它来开发客户端程序，用它来运行我们的出版商务系统，还用它来做所有使系统运行更流畅的小程序。

在这 4 年中，Ruby 发展迅速。大量的方法被加入内建的类和模块，标准库（那些包含在 Ruby 发行版中的库）也有了重要发展。现在社区有了标准的文档系统（RDoc），而 RubyGems 也成了为打包发行 Ruby 代码的系统选择。

这些改变是令人激动的，但是这也使得本书第 1 版变得有点过时。第 2 版弥补了这一缺陷：像第 1 版一样，它是基于 Ruby 的最新版而编写的。

Ruby 版本

Ruby Versions

本书阐述的是 Ruby 1.8（特别涵盖了集成入 Ruby 1.8.2 的改变）。[1]

[1] Ruby 版本号遵循其他许多开源项目的模式。子版本号是偶数的版本（1.6, 1.8 等），也是稳定的公开发布版。这些版本的 Ruby 被预先打好包并放在各种各样的 Ruby 网站上以供下载。软件开发版的子版本号是奇数，例如 1.7 和 1.9。对于这种版本，你必须用本书第 4 页介绍的方法来下载并自己编译。

用来写本书的 Ruby 版本到底是什么？让我们来问 Ruby 吧。

```
% ruby -v
ruby 1.8.2 (2004-12-30) [powerpc-darwin7.7.0]
```

这演示了重要的一点。你在本书中看到的大多数代码例子都在我排版本书时被实际执行过。当你看到一个程序的输出时，该输出是通过实际运行代码并将结果插入到本书而成的。

本书的变动
Changes in the Book

和第 1 版相比，除了更新到 Ruby 1.8，你还会发现一些其他的变化。

在本书的前半部分，我新加了 6 章。与第 1 版相比，**入门**一章对设置和运行 Ruby 做了更完整的介绍。第二个新加入的**单元测试**章反映了 Ruby 爱好者对测试越来越重视。第三个新的章节涵盖了 Ruby 程序员使用的三个工具：用来体验 Ruby 的 *irb*，用来给代码写文档的 *RDoc* 以及用来打包发布代码的 *RubyGems*。新添的最后一章介绍了 *duck typing*，这种编程哲学和 Ruby 背后的思想非常一致。

新加入的还不止这些。你还会发现线程一章大大扩展了对同步的讨论，而编写 Ruby 扩展一章的大部分也被重写了。关于网页编程一章讨论了一个可选的模板系统并加入了 SOAP 一节。语言参考一章也被大大扩展了（特别是关于 block、procs、break 和 return 新规则的部分）。

本书的后半部分对内建的类和模块进行了介绍，该部分包含的重要改动超过了 250 处。其中很多是方法的更新，一些是更新过时的老方法，而另一些是具有新行为的方法。你还会发现加入了许多对新模块和类的介绍。

最后，本书用了一节来介绍标准库。自 Ruby 1.6 以来标准库有了极大的发展，现在它的内容如此之多，以至于没有上千页篇幅根本就不可能对之进行较为详细的介绍。同时，Ruby Documentation 项目正在忙于为库代码添加 RDoc 文档（第 199 页的第 16 章介绍了 RDoc）。这意味着你可以使用 Ruby 发行版自带的 ri 工具来获得关于库模块的精确