

目录	1
----	---

目录

1 松本行弘的程序世界	1
2 内容提要	2
3 推荐序	2
4 中文版序	3
5 DONE 前言 <2016-08-30 Tue 14:10>	4
6 DONE 第一章我为什么开发 Ruby	5

1 松本行弘的程序世界

本书为“Ruby”之父经典力作，展现了大师级的程序思考方式。作者凭借对编程本质的深刻认识和对各种技术优缺点的掌握，阐述了 Ruby 的设计理念，并由此延伸，带领读者了解编程的本质，一窥程序设计的奥秘。本书不是为了介绍某种特定的技术，而是从宏观的角度讨论与编程相关的各种技术。书中第 1 章介绍了作者对编程问题的新思考和新看法，剩下的内容出处《日经 Linux》杂志于 2005 年 5 月到 2009 年 4 月连载的“松本编程模式讲坛”，其中真正涉及“模式”的内容并不多，大量篇幅都用于介绍技术内幕和背景分析等内容，使读者真正了解相关技术的立足点。另外，书中还包含许多以 Ruby、Lisp、Smalltalk、Erlang、JavaScript 等动态语言所写成的范例。

- Ruby 之父佳作，进入不同凡响的程序世界
- 深入剖析程序设计的道与术
- 举一反三，触类旁通

2 内容提要

本书是探索程序设计思想和方法的经典之作。作者从全局的角度，利用大量的程序示例及图表，深刻阐述了 Ruby 编程语言的设计理念，并以独特的视角考察了与编程相关的各种技术。阅读本书不仅可以深入了解编程领域各个要素之间的关系，而且能够学到大师的思考方法。本书面向各层次程序设计人员和编程爱好者，也可以供相关技术人员参考。

3 推荐序

在流行的编程语言中，Ruby 比较另类，这是因为大多数编程语言的首要着眼点在于为解决特定的问题领域而设计语言，而 Ruby 的首要着眼点在于“人性化”，让程序员充分享受编程的乐趣。由于组织国内的 Ruby 会议，我曾两次邀请松本行弘来中国。他是一位性格平和、对生活充满热爱的人，在演讲中也一再传递 code for fun 的宗旨，即编程语言不应该是冷冰冰地给机器阅读的执行的指令，而应该让程序员编程的工作过程变成一种充满乐趣和享受的过程。而且，松本先生发明 Ruby 语言也是因为他对创造一种人性化的面向对象脚本语言的热爱。

程序员社区经常拿另外一个主流的面向对象脚本语言 Python 来和 Ruby 做对比。从全球范围来看，Python 的社区更大，应用更广泛，但 Ruby 的语法相对 Python 来说更强大和宽松，给程序员发挥的自由度更大，可以基于 Ruby 创建各个领域的 DSL，比方说 Ruby on Rails 就是一个基于 Ruby 的 Web 快速开发领域的 DSL。

总之，Ruby 语言的这种“人性化”以及给程序员很大编程自由度的气质奠定了整个 Ruby 社区的气质：热爱生活的程序员，所求编程的自由度，带点非主流的极客色彩。也正因为如此，Ruby 和基于 Ruby 的 Rails 得到了硅谷许许多多创业公司青睐，有名者如 Twitter、Groupon、Hulu、github 等。而这种气质也鲜明地体现在 Rails 框架的创建者 David Heinemeier Hansson 及其所在的 37signals 公司身上。37signals 的 20 多位员工遍全球，每天只上班四天，David Heinemeier Hansson 本人不是一位保时捷车队的职业赛车手。

当然, Ruby 并非只在非主流程序员社区中流行, 随着全球 IT 产业进行云计算时代, Ruby 也发挥着越来越大的作用。著名的 SAAS 厂商 salesforce 在 2010 年底以 2.1 亿美元收购了 PAAS 厂商 Heroku, 并且在 2011 年 7 月聘请松本行担任 Heroku 首席架构师, 开拓 Ruby 在云计算领域的应用。Heroku 本身就是一个完全采用 Ruby 架构的 PAAS 平台, 同样支持 Ruby 的 PAAS 厂商还有 EngineYard、VMware 等。随着这些云计算厂商的努力, Ruby 必然在未来得到越来越广泛的应用。我之前阅读了本书的部分章节, 这本书实际上是松本行弘从一个编程语言设计者的角度去看待各种各样的流行编程语言, 分析它们有哪些特点, 以及 Ruby 编程语言是如何取舍的。Ruby 语言的设计本身大量参考了一个更古老而著名的面向对象编程方法的开山之作 Smalltalk, 而且从函数式编程语言鼻祖 Lisp“偷师学艺”了不少好东西。程序员社区有个著名的说法: 任何现代编程语言都脱胎于 Smalltalk 和 Lisp, 都与这两个编程语言有着似曾相识的特性, 自 Smalltalk 和 Lisp 诞生以来, 编程语言领域可谓大势已定了。因此, 集这两种编程语言诸多特点于一身的 Ruby 语言很值得编程爱好者去学习, 而看看 Ruby 设计师是怎么设计 Ruby 语言的, 则可以让高屋建瓴地理解一些主流的编程语言。

范凯

4 中文版序

从年轻的时候开始, 我就对编程语言有着极为浓厚的兴趣。比起“使用计算机干什么”这一问题, 我总是一门心思想着“如何将自己的意图传达给计算机”。从这个意义上说, 我认为自己是个“怪人”。但是, 想选择一个能让自己工作变得轻松的编程语言, 想编写一种让人用起来感到快乐的编程语言, 一直是我梦寐以求的, 这种迫切的心情怕不输于任何人。虽说是有点自卖自夸, 但是 Ruby 确实给我带来了“快乐”, 这一结果让我感到很满足。

让我感到惊奇的是, 有很多人, 包括那些没有我这么“怪”的人, 都对这种快乐有着共鸣。Ruby 自 1995 年在互联网上公布以来, 着实让世界各地的程序员都认识了它, 共享着这种快乐, 提高了软件开发的生产力。完全出乎我意料的是, 世界各地的人, 不管是东方还是西方, 都极为欣赏 Ruby。在刚开始开发 Ruby 的时候, 我想都没有想到过有这样的结果, 程序员的感觉

会超越人种、国籍、文化，有如此之多的共通之处。

现在，为世界各地的程序员所广泛接受的 Ruby，正带来一种新的文化。已经有越来越多的开发人员，在实践中果敢地施行着 Ruby 语言及其社区所追求的“对高生产力的追求”、“富有柔性的软件开发”、“对程序员人性的尊重”、“鼓起勇气挑战新技术”等原则。在 Ruby 以前，这些想法也都很好，却一直实践不起来。我相信，Ruby 的卓越之处，不仅在于语言能力，而且更重要的是引领了这种文化的践行。

本书在解说编程中的技术与原则时，不局限于表面现象，而是努力挖掘其历史根源，提示其本质。虽然很多章节都以 Ruby 为题材，但这些原则对于 Ruby 以外的语言也行之有效。衷心希望大家能够实践本书中所讲述的各项原则，成为一个更好的开发人员。

松本行弘

2011 年 4 月 18 日

5 **DONE** 前言 <2016-08-30 Tue 14:10>

本书的目的不是深入讲解哪种特定的技术，也没有全面讨论我所开发的编程语言 Ruby，而是从全局角度考察了与编程相关的各种技术。读者千万不要以为拿着这本书，就可以按图索骥地解决实际问题了。实际上，最好反它看成是一幅粗略勾勒出了编程世界诸要素之间关系的“世界地图”。

每种技术、思想都有其特定的目的、渊源和发展进步的过程。本书试图换一个角度重新考察各种技术。如果你看过后能够感觉到“啊，原来是这样的呀!”或者“噢，原来这个技术的立足点在这里呀!”那么我就深感心慰了。我的愿望就是这些知识能够激发读者学习新技术的求知欲。

本书的第 2 章到第 14 章，是在《日经 Linux》杂志于 2005 年 5 月到 2009 年 4 月连载的“松本编程模式讲坛”基础上编辑修改而成的。但实际上连载与最初的设想并不一致，真正涉及“模式”的内容并不多，倒是技术内幕、背景分析等内容占了主流。现在想来，大方向并没有错。

除了连载的内容之外，本书还记录了我对编程问题的新思考和新看法。特别是第 1 章“我为什么开发 Ruby”，针对“为什么是 Ruby”这一点，比其他杂志做了更加深入的解说。另外，在每章的末尾增加了一个小专栏。

对于连载的内容，因为要出成一本书，除修改了明显的错误和不合时代的部分内容之外，我力求每一章都独成一体、内容完整，同时也保留了连载时的风貌。通读全书，读者也许会感到有些话题或讲解是重复的，这一点敬请原谅。

我的本职工作是程序员，不能集中大段时间去写书，不过无论如何最后总算是赶出来了。非常感谢我的家人，她们在这么长时间里宽容着我这个情绪不稳的丈夫和父亲。

稿子写完了，书也出来了，想着总算告一段落了吧，而《日经 Linux》又要开始连载“松本行弘技术剖析”了，恐怕还要继续让家里人劳心。

松本行弘

2009 年 4 月于樱花季节过后的松江

6 **DONE** 第一章我为什么开发 Ruby

1.1 我为什么开发 Ruby

Ruby 是起源于日本的编程语言。近年来，特别是因为其在 Web 开发方面的效率很高，Ruby 引起了全世界的关注，它的应用范围也扩展到了很多企业领域。

作为一门编程语言，Ruby 正在被越来越多的人所了解，而作为一个工程师的我，松本行弘，刚开始的时候并没有想过“让全世界的人都来用它”或者“这下子可以大赚一笔了”，一个仅仅是从兴趣开始的项目却在不知不觉中发展成了如今的样子。

当然了，那时开发 Ruby 并不是我的本职工作，纯属个人兴趣，我是把它作为一个自由软件来开发的。但是世事弄人，现在开发 Ruby 竟然变成了我的本职工作了，想想也有些不可思议。

“你为什么开发 Ruby？”每当有人这样问我的时候，我认为最合适的回答应该就像 Linux 的开发者 Linus Torvalds 对“为什么开发 Linux”的回答一样吧——

“因为它给我带来了快乐。”

当我还是一个高中生，刚刚开始学习编程的时候，不知何故，就对编程语言产生了兴趣。

周围很多喜欢计算机的人，¹有的是“想开发游戏”，有的是“想用它来做计算”，等等，都是“想用计算机来做些什么”。而我呢，则想弄明白“要用什么编程语言来开发”、“用什么语言开发更快乐”。

高中的时候，我自己并不具备开发一种编程语言所必需的技术知识，而且当时也没有计算机。但是，我看了很多编程语言类的书籍和杂志，知道了“还有像 Lisp 这样优秀的编程语言”、“Smalltalk 是做面向对象设计的”，等等，在这些方面我很着迷。上大学时就自然而然地选修了计算机语言专业。10 年后，我通过开发 Ruby 实现了自己的梦想。

从 1993 年开发 Ruby 到现在已经过去 16 年了。在这么久的时间里，我从未因为设计 Ruby 而感到厌烦。开发编程语言真是一件非常有意思的事情。

1.1.1 编程语言的重要性

为什么会喜欢编程语言？我自己也说不清。至少，我知道编程语言是非常重要的。

最根本的理由是：语言体现了人类思考的本质。在地球上，没有任何超越人类智慧的生物，也只有人类能够使用语言。所以，正是因为语言，才造成了人类和别的生物的区别；正是因为语言，人之间才能传递知识和交流思想，才能做深入的思考。如果没有了语言人类和别的动物也就不会有太大的区别了。

在语言领域里，有一个 Sapir-Whorf 假说，认为语言可以影响说话者的思想。也就是说，语言的不同，造成了思想的不同。人类的自然语言是不是像这个假说一样，我不是很清楚，但是我觉得计算机语言很符合这个假说。也就是说，程序员由于使用的编程语言不同，他的思考方法和编写出来的代码都会受到编程语言的很大影响。

也可以这么说，如果我们选择了好的编程语言，那么成为好程序员的可能性就会大很多。

20 年来一直被奉为名著的《人月神话》的作者 Frederick. Brooks 说过：一个程序员，不管他使用什么编程语言，他在一定时间里编写的程序行数是一定的。如果真是这样，一个程序员一天可以写 500 行程序，那么不论他用

¹当时喜欢计算机的人当然还是少数。

汇编、C, 不是 Ruby, 他一天都应该可以写 500 行程序。

但是, 汇编的 500 行程序和 Ruby 的 500 行程序所能做的事情是有天壤之别的。程序员根据所选择编程语言的不同, 他的开发效率就会有十倍、百倍甚至上千倍的差别。

由于价格降低、性能提高, 计算机已经很普及了。现在基本上各个领域都使用了计算机, 但如果没有软件, 那么计算机这个盒子恐怕一点用都没有了。而软件开发, 就是求能够用更少的成本、更短的时间, 开发出更多的软件。

需要开发的软件越来越多, 开发成本却有限, 所以对于开发效率的要求就很高。编程语言就成了解决这个矛盾的重要工具。

1.1.2 Ruby 的原则

Ruby 本来是我因兴趣开发的。因为对多种编程语言都很感兴趣, 我广泛对比了各种编程语言, 哪些特性好, 哪些特性没什么用, 等等, 通过一一进行比较、选择, 最终把一些好的特性吸纳进了 Ruby 编程语言之中。

如果什么特性都不假思索地吸纳, 那么这种编程语言只会变成以往编程语言的翻版, 从而失去了它作为一种新编程语言的存在价值。

编程语言的设计是很困难的, 需要仔细斟酌。值得高兴的是, Ruby 的设计很成功, 很多人都对 Ruby 给出了很好的评价。

那么, Ruby 编程语言的设计原则是什么呢?

Ruby 编程语言的设计目标是, 让作为语言设计者的我能够轻松编程, 进而提高开发效率。

根据这个目标, 我制订了以下 3 个设计原则。

- 简洁性
- 扩展性
- 稳定性

关于这些原则, 下面分别加以说明。

1.1.3 简洁性

以 Lisp 编程语言为基础而开发的商业软件 Viaweb 被 Yahoo 收购后, Viaweb 的作者 PaulGraham 也成了大富豪。最近他又成了知名的技术专栏作家, 写了一篇文章就叫“简洁就是力量”。²

他还撰写了很多倡导 Lisp 编程语言的文章。在这些文章中他提到, 编程语言在这半个世纪以来是向着简洁化的方向发展的, 从程序的简洁程度就可以看出一门编程语言本身的能力。上面提到的 Brooks 也持同样的观点。

随着编程语言的演进, 程序员已经可以更简单、更抽象地编程了, 这是很大的进步。另外随着计算机性能的提高, 以前在编程语言里实现不了的功能, 现在也可以做到了。

面向对象编程就是这样的例子。面向对象的思想只是把数据和方法看作一个整体, 当作对象来处理, 并没有解决以前解决不了的问题。

用面向对象记述的算法也一定可以用非面向对象的方法来实现。而且, 面向对象的方法并没有实现任何新的东西, 却要在运行时判定要调用的方法, 倾向于增大程序的运行开销。即使是实现同样的算法, 面向对象和程序往往更慢, 过去计算机的执行速度不够快, 很难请允许我像这样的“浪费”。

而现在, 由于计算机性能大大提高, 只要可以提高软件开发效率, 浪费一些计算机资源也无所谓了。

再举一些例子。比如内存管理, 不用的内存现在可用垃圾收集器自动释放, 而不用程序员自己去释放了。变量和表达式的类型检查, 在执行时已经可以自动检查, 而不用在编译时检查了。

我们看一个关于斐那契 (Fibonacci) 数的例子。图 1-1 所示为用 Java 程序来计算斐波那契数。算法有很多种, 我们最常用的递归算法来实现。

图 1-2 所示为完全一样的实现方法, 它是用 Ruby 编程语言写的, 算法完全一样。和 Java 程序相比, 可以看到构造完全一样, 但是程序更简洁。Ruby 不进行明确的数据类型定义, 不必要的声明都可以省略。所以, 程序就非常简洁了。

```
class Sample {
```

²Paul Graham 目前是世界知名的天使投资人, 其公司 Y Combinator 投资了很多极有前途的创业项目。Paul Graham 曾出版过两本 Lisp 专著, 最新著作《黑客与画家》已经由人民邮电出版社出版。——编者注


```
private static int fib (int n){
    if (n<2){
        return n;
    }
    else {
        return fib (n-2) +fib (n-1);
    }
}

public static void main(String [] argv){
    System.out.println ("fib(6)="+fib (6));
}
}
```

图 1-1 计算斐波那契数的 Java 程序

```
def fib(n)
  if n<2
    n
  else
    fib(n-2) +fib(n-1)
  end
end
print "fib(6)=", fib(6), "\n"
```

图 1-2 计算斐波那契数的 Ruby 程序