HZ BOOKS

PEARSON Addison Wesley

经典的设计模式，灵活的语言实现

Ruby 设计模式

Design Patterns in Ruby

(美) Russ Olsen 著

谈熠 陈熙 译

Ruby和Rails技术系列

机械工业出版社
China Machine Press

Ruby 设计模式

Design Patterns in Ruby

(美) Russ Olsen 著
谈熠 陈熙 译

机械工业出版社
China Machine Press

本书是一本关于设计模式方面的重点书籍。本书以通俗易懂的方式介绍了 Ruby 设计模式，主要包括 Ruby 概述、使用模板方法变换算法、使用策略替换算法、通过观察器保持协调、通过迭代器遍历集合、使用命令模式完成任务、使用适配器填补空隙、使用装饰器改善对象、单例、使用工厂模式挑选正确的类、通过生成器简化对象创建和使用解释器组建系统等内容。

本书适合程序员阅读，也可以作为 Ruby 语言的参考书。

图书在版编目（CIP）数据

Ruby 设计模式 /（美）奥尔森（Olsen，R.）著；谈熠，陈熙译．—北京：机械工业出版社，2009.1

（Ruby 和 Rails 技术系列）

书名原文：Design Patterns in Ruby

ISBN 978-7-111-25120-0

Ⅰ.R…　Ⅱ.①奥…　②谈…　③陈…　Ⅲ.计算机网络—程序设计　Ⅳ.TP393.09

中国版本图书馆 CIP 数据核字（2008）第 144655 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）

责任编辑：王　玉

北京京北印刷有限公司印刷 · 新华书店北京发行所发行

2009 年 1 月第 1 版第 1 次印刷

186mm × 240mm · 17.25 印张

标准书号：ISBN 978-7-111-25120-0

定价：45.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

本书的赞誉

本书记录了用于解决 Ruby 开发者常见问题的聪明方法。Russ Olsen 在这方面做得棒极了。他不仅收录了经典的设计模式，还扩充了只与 Ruby 相关的新模式。他清楚地介绍每个模式，使 Ruby 开发者能在日常工作中获得宝贵的经验。”

——Steve Metsker，Dominion Digital，Inc. 管理咨询顾问

“此书为‘四人组（GoF）’所提出的一些重要设计模式提供了极佳的演示，而没有过多地诉诸技术性解释。作者以精准而易懂的风格进行了完整的描述。即使之前没有接触过设计模式的开发者，通过阅读此书，也能很快地、自信地在使用 Ruby 时运用设计模式。Olsen 干得很出色，他将一本关于经典的‘枯燥’主题的书写得有声有色并不乏幽默。”

——Peter Cooper

“在我关注设计模式 10 年之后，本书使我对这个主题重新燃起了兴趣。Russ 在四人组的基础上，挑选了 Ruby 中最有用的设计模式，并以一种直接和逻辑的方式介绍它们。本书提升了我使用 Ruby 的能力，并且鼓励我拂去覆盖在四人组的那本书上的灰尘。”

——Mike Stok

“对于来自静态类型的面向对象语言的程序员来说，本书给出了一个很好的途径，以学习如何在更加动态、灵活的语言（比如 Ruby）中运用设计模式。”

——Rob Sanheim，Ruby Ninja，Relevance

译者序

随着计算机程序设计语言的不断发展和进步，动态语言已经得到开发领域的认可，并在软件行业的各个领域中获得长足的发展。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 Ruby，开展着各种激动人心的工作。这个社群无需豪言壮语，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 语言的强大和优雅。使用这门语言也是莫大的乐趣。

自 1994 年，四人组（GoF）的名著《Design Patterns：Elements of Reusable Object-Oriented Software》[⊖]问世以来，设计模式便成为软件工业的圭臬。四人组所提出的设计模式总结归纳了传统软件开发的经验，为开发中所遇到的各种常见问题提供了解决之道。Ruby 语言的动态特征恰恰为这些设计模式的实现提供了一个轻盈而高效的平台。此外，Ruby 本身具有传统的静态类型语言所不具备的动态优越性，这些高级特性为 Ruby 语言创造了域指定语言、元编程和惯例优于配置等设计模式。传统的软件开发思路是：好的设计会产生好的软件。因此在实际开发之前，值得花时间做一个全面而细致的设计。而在阅读本书之后，你会发现 Ruby 使你的设计灵活地贯穿于开发之中。

本书由谈熠和陈熙合力翻译，在翻译中得到了陈理人、陈慧勤、陈少科和金爱珍等朋友的指导和帮助。借此机会，特别感谢华章公司引进此书，并将如此重要的一部书交托给我们来翻译，希望译文能不辱使命。我们还要感谢华章的编辑不辞辛劳的工作。

Ruby 和基于 Ruby 的著名程序框架 Rails 在国内的应用开始步入快速发展阶段。我们虽然有数年的使用 Ruby 的工作经验，然而 Ruby 和 Rails 社区中如泉水般喷涌的新技术始终鞭策着我们不断地学习和埋头奋进。限于译者水平，译稿中若存有这样那样的错误，恳请你不吝指正。在 Ruby on Rails 中文网站的用户社区 http：//rubyonrailscn. org/community/中，有不少国内使用 Ruby 和 Rails 的顶尖高手。如果你遇到了技术方面的问题，你可以在那里最快地找到解决之道。此外，也可以通过电子邮件联系我们：yi@ rubyonrailscn. org 。

译者

2008 年 8 月

⊖ 此书由机械工业出版社出版。——编辑注

序

《*Design Patterns: Elements of Reusable Object-Oriented Software*》的作者被读者亲切地称为“四人组（GoF）”。该书是第一本作为设计模式参考的主流书籍。从1995年至今，该书已经售出了50多万册。毫无疑问，它影响了全球数以百万计的程序员的思路和代码。我仍然清楚地记得在20世纪90年代末我第一次买那本书时的情景。部分热情源自向我推荐该书的同辈人。我将它视为我迈向一个成熟程序员的一步。我在几天内便翻遍了那本书，并急切地构想着每一个模式的实际应用。

大家普遍认为设计模式的最大用处在于，当程序员们进行关于开发的讨论时，为他们提供一个能够清楚描绘设计决定的词汇库。特别是在结对编程（极限编程和其他敏捷开发的基础）的情况下，设计是一个进行态的共享步骤。你能够轻松地跟与你结对的同事说：“我觉得需要在这里使用些策略”、或者“让我们给这个功能加一个观察器。”

在一些公司里，考察对于设计模式知识的把握程度已经成为一种快速检阅程序员应聘者的途径。在那里通常会听到：

“你最喜欢的设计模式是什么?”

“嗯……工厂模式吧?”

“谢谢你前来面试，门在那里。”

上面说到的“最喜欢”的模式是不是听起来有点奇怪？我们钟爱的设计模式应当是对于给定的情况能对症下药的那个。缺乏经验的程序员在学习设计模式时容易犯的一个典型错误是，把一种模式当作目标去实现，而不是当作一种手段。为什么会有人为了好玩而去实现设计模式时又找不着头绪?

至少在静态类型的世界里，实现设计模式需要解决一些技术难点。最好的情况是，你使用一些忍者技巧显示你超群的编程能力。而在最糟糕的情况下，你最后得到的是一堆从模板文件里七拼八凑而成的废物。所以，设计模式至少对于像我这样的奇客（Geek）来说是一个有趣的题目。

那么，用Ruby实现四人组所提出的设计模式困难吗？不一定。对于入门者来说，没有了静态类型的Ruby大大减少了实现程序的代码总量。Ruby标准库也以单行包含的方式提供一些最常见的模式。而另外的一些也构建在Ruby语言自身中。比如，四人组所说的Command对象本质上就是对一段用来做特殊事情的代码的包装，即在某个时刻运行一段特定的代码。当然，这是对Ruby中的block对象（或Proc）的一个相当准确的描述。

Russ自2002年起使用Ruby进行工作。他知道最有经验的Ruby用户早就通晓设计模式，并且知道如何运用。所以，他在本书中遇到的最大挑战，就我所知，是要以一个既能和专业Ruby程序员相关相通，又能帮助初学者走近我们所热爱的这门语言的角度来撰写此书。我认为他成功了，你也会这样认为。再用Command对象举例子：它最基本的实现是简单地采用一个block。当加入了状

态和一些行为后，实现就变得一点都不简单了。Russ 在此为我们提供了专用于 Ruby 的、被证明过的建议，而且“药到病除”。

本书的另一个优点就是，加入了 Russ 定义的专用于 Ruby 的新设计模式。Russ 对它们进行了详细的说明，其中包括我最喜欢的内部域指定语言。就像一个 Interpreter（解释器）模型的进化，我相信他对这个主题的论述，使本书出版界在本领域中的第一部有益的著作。

最后，我觉得本书会给刚开始在专业工作中使用 Ruby 的人和来自其他并不十分强调面向对象设计和模式的语言（比如 PHP）背景的人带来巨大的福音。在描述这些设计模式的过程中，Russ 捕捉了我们用来解决每天使用 Ruby 编程时都会遇到的障碍的精要——对于新手来说，这可谓是无价之宝。本书具有如此多的优点，我早就将它列入送给我的新同事和朋友的礼物清单中。

——Obie Fernandez 专业 Ruby 丛书编辑

前 言

我曾经共事过的一个同事一直对我说，那些厚重的，关于设计模式的书足以证明编程语言的匮乏。他的意思是，因为设计模式是代码中的约定惯例，所以一个好的编程语言应当全面地整合这些设计模式，使它们消失在代码中。

举一个极端的例子，在20世纪80年代，我曾参与的一个项目是用C语言编写面向对象代码。是的，是C语言，而不是C++。我们所采用的技巧是将每个“对象”映射到一个函数指针列表。然后通过查找表中的指针来调用相应函数从而操作我们的“对象”，从而模拟对于对象中方法的调用。这种方法比较笨拙而且烦琐，但是能够解决问题。如果对这个方法进行归纳总结，可能会称其为“面向对象”设计模式。当然，后来有了先进的C++，以及后来的Java，我们的面向对象模式便消失了。今天，我们不太会将面向对象看作一种设计模式——因为今天的它实在太简单。

但很多情况仍未变得简单。名副其实的“四人组”之书（《*Design Patterns*：*Elements of Reusable Object-Oriented Software*》，Gamma，Helm，Johnson和Vlissides著）依旧是每个程序员的必修读物。然而对于该书所说的许多模型，在今天各类流行语言（Java和C++，和（或）C#）中的具体实现看上去多少有点像我那个20世纪80年代的过时手工对象系统。它太痛苦、太冗长、太容易出错。

Ruby编程语言带着我们向我那位同事的理想迈进了一步。用Ruby实现设计模式是如此简单，以致有时候设计模式淡出到程序背后。由于以下原因，用Ruby构建模式变得更简单：

- Ruby是一种动态类型语言。免去了静态类型，Ruby极大地减少了大多数程序的代码量，包括用来实现设计模式的代码。
- Ruby内建代码闭包。闭包允许我们传递一段代码并指定其执行访问而不必辛苦地构建空而无用的整个类和对象。
- Ruby的类是真实的对象。因为类在Ruby中即另一个对象，我们可以像对待任何其他对象一样对Ruby类进行常规的运行时操作：我们可以创建全新的类。我们可以通过增加或删除方法来修改已存在的类。我们甚至可以克隆一个类，修改它的副本，而保持原类不变。
- Ruby具备一个优雅的代码重用系统。除了支持普通的类继承以外，Ruby还允许我们定义混合（mixin）。那是一种简单但灵活的方法，使代码可以被若干个类共享。

以上这些因素使得Ruby代码变得可压缩。在Ruby中，像在Java和C++中一样可以实现复杂的想法。不过只有通过Ruby才可能把实现的具体细节更有效地隐藏起来。正如你将在以后见到的那样，许多设计模式在传统静态语言中需要大量的模板代码，而在Ruby中只需要一两行即可。你可以简单地使用include Singleton命令将类变成一个单例（singleton）。你可以像实现继承那样简单地实现委托（delegate）。因为Ruby使每行代码能做更多有趣的事情，最后，只需要更少的代码来

完成工作。

这并不是简单地在键盘上偷懒，而是对 DRY（不重复你自己）原则的应用 。我不认为现在还会有人对旧 C 语言中的面向对象模式的消亡而感到惋惜。它的确曾为我解决了问题，但我也不得不花时间为它工作。Ruby 提供了一个通向更高境界的阶梯，只需要工作一次，便可以将模式从代码中压缩出来。简而言之，Ruby 使你把注意力放在解决实际问题上而不是其他工作中上。我希望本书能够帮助你了解如何达到那个境界。

本书的目标读者

简单地说，本书写给那些想知道如何用 Ruby 有效地开发软件的人。假设你熟悉面向对象编程，但对设计模式没有任何了解，你可以在阅读本书的过程中学会。

你也不需要掌握很多 Ruby 知识来阅读本书。你会在第 2 章中找到对于这门语言的快速入门。同时，我会对每个 Ruby 语言特定的环节进行详细解释。

本书内容的组织结构

本书分为三个部分。第一部分从整个设计模式发展的历史和背景的最简单纲要开始，涵盖重点的 Ruby 语言的快速入门。

第二部分是本书的主体，通过 Ruby 的角度来讲解四人组提供的设计模式。这个模式要解决怎样的问题？这个模式的传统实现方法（四人组给出的方法）在 Ruby 中是怎样的？传统实现方法在 Ruby 中是否有效？Ruby 是否提供了其他选择让实现变得更简单？

第三部分将讲述随着 Ruby 语言的推广和发展而出现的 3 种新设计模式。

有言在先

在重复那句我多年来一直默念的箴言之前，我无法在一本关于设计模式的书上签下我的名字：设计模式是一小盒封装好的用来解决常见编程问题的方案。在理想的情况下，当遇到一个相应的问题时，你应当激活对应的设计模式，然后就会解决问题。但第一步（等待相应问题的到来）经常困扰一些工程师。只有当你面对应当使用设计模式来解决的问题之后，才能说你正确地应用了设计模式。

在一些圈子中，对于设计模式的滥用已经玷污了设计模式的名声。我可能会争辩，用 Ruby 能比用其他语言更简单地写一个使用工厂方法的转换器，然后用它获得一个生成器的代理器。那个生成器随后创建一个命令去进行一个 2 加 2 的操作。Ruby 的确能让这些步骤的实现变得简单，即使在 Ruby 中，这样的设计也没有任何意义。

你也不能将程序的构建看作一个对于现有设计模式简单拼接的组合。任何有趣的程序都有特殊之处，一些只用来完美地解决特定问题的代码。设计模式是要帮助你认识和解决在软件创建过程中重复产生的一般问题。使用设计模式的优势在于，让你能够迅速克服一些别人已经面对过的问题，

从而使你能调整难点——那些只有在你的程序中面对的问题。设计模式不是万能的灵丹妙药，无法帮你解决所有的设计问题。它们仅是一个技巧，一个帮助你构造程序的非常有用的技巧。

关于本书中的代码风格

用 Ruby 编程变得如此有趣的原因之一是，这门语言尽量不影响你进行程序设计——如果有多种方法可以用来描述事物的话，Ruby 通常支持所有这些方法：

```
# 方法之一

if (divisor == 0)
  puts 'Division by zero'
end

# 另一种

puts 'Division by zero' if (divisor == 0)

# 第三种

(divisor == 0) && puts 'Division by zero'
```

出于语法上的考虑，Ruby 也尽量不强调语法。只要表意清晰，Ruby 尽可能地让你省略不必要的东西。比如，在调用一个方法的时候，你可以省略参数列表两旁的括号：

```
puts('A fine way to call puts')
puts 'Another fine way to call puts'
```

当你定义一个函数的参数列表时，以及编写 if 语句的条件时，你甚至都可以省略括号。

```
def method_with_3_args a, b, c
  puts "Method 1 called with #{a} #{b} #{c}"
  if a == 0
    puts 'a is zero'
  end
end
```

所有这些快捷方式能够在编写真正的 Ruby 程序时带来方便。然而，当它们被大量使用的时候，通常会给初学者带来困惑。所以许多刚用 Ruby 的程序员可能对以下格式感到难以适应：

```
if file.eof?
  puts( 'Reached end of file' )
end
```

或者

```
puts 'Reached end of file' if file.eof?
```

也可能

```
file.eof? || puts('Reached end of file')
```

由于本书主要讨论 Ruby 的深层力量和优雅，而不是纠缠语言的语法细节上，所以，我尽量保持平衡，一方面尽量让示例代码看上去像真正的 Ruby 代码，另一方面，确保它们对初学者来说是友善的。在具体的实践中，我采用了一些显而易见的快捷方式，同时避免使用更极端的窍门。这并非意味我不了解，或不赞同 Ruby 的语法快捷功能。我只是更想对使用 Ruby 的新手们尽量保持概念的优雅。当你忘我地投入这门语言之后，你会有大量的时间学习语法快捷功能。

致　谢

我一直以来都觉得引用名言“无人是孤岛”多少会让人感觉呆板。大多数岛屿都是水下山脉的顶峰，你所见到的那一小块绿色的部分是由水下看不见的连绵山体所支撑的。同样，在无人帮助的情况下，没人能完成任何一件事情。所以，每件事所得到的赞誉应当由我们的朋友、家人和同事们共享。我们都是跳出水面的岛屿，背后由帮助过我们，且不知名的人所支撑。离开了慷慨相助的人们，这本书绝不会开始，也不可能完成。

我要特别感谢我的好朋友 Bob Kiel。他至少告诉了我 17 827 次我应当写本书。同时也感谢 Xandy Johnson，谢谢他在我写作过程中的鼓励和大力帮助。

我还要向所有校阅过本书的人说声“谢谢”，他们包括 Mike Abner、Geoff Adams、Peter Cooper、Tom Corbin、Bill Higgins、Jason Long、Steve Metsker、Glenn Pruitt、Rob Sanheim、Mike Stok 和 Gary Winklosky。特别感谢 Andy Lynn 和 Arild Shirazi 对本书的早期书稿进行的大量审阅工作，也特别感谢 Rob Cross 查出“最后一个”错别字。

同样感谢 Heli Roosild，他是一个专业作者。他花时间阅读了我写的一些东西并说：“可以，就这样。”

还要感谢 Lara Wysong、Raina Chrobak 和 Christopher Guzikowski，以及 Addison-Wesley 的全体工作人员，特别是 Chris，他审读了我写的 900 字的博客文章和整本书。还要感谢 Jill Hobbs 用精准的眼光和更为精准的笔尖为本书进行了校对。

我想感谢 FGM 的朋友们，他们营造了一个求知的环境让写作本书成为可能。

还要感谢 Steve McMaster 和他在 SamSix 的同伴，感谢他们的支持和鼓励。

然后从个人角度，感谢我的妻子 Karen，谢谢她的鼓励和建议，并感谢她在数月中让我租用厨房餐桌的一角。同时也感谢我的儿子 Jackson，让我在书中不时地能讲讲他的故事。感谢 Diana Greenberg，一个和我们患难与共的朋友。

感谢我的兄弟 Charles，他树立了一个勇敢和坚定的榜样，使我每天都努力向他看齐。

最后，感谢我的姐姐 Dolores，她培养了我对阅读的兴趣。我清楚地记得那天她把我从借来的那些快速读物中拉开，拖着我走进图书馆，带我到严肃读物的书架前。她拿下一本说：“这才是你应该读的书。”我还能记起那本书的样子。那本书书名是《*The Rise and Fall of the Third Reich*》（第三帝国的兴衰），William L. Shirer 所著，总共 1 264 页。我想我当时应该是 7 岁。

目　录

第三部分 Ruby 的设计模式

第一部分

设计模式和Ruby