PEARSON

本书是对“如何编写更易维护、更易管理、更讨人喜爱，以及功能更为强大的Ruby应用程序”的全面指导。

PRACTICAL OBJECT-ORIENTED DESIGN IN RUBY

AN AGILE PRIMER

面向对象设计实践指南
Ruby语言描述

[美] Sandi Metz 著　　张雪平 彭晓东 译

人民邮电出版社
POSTS & TELECOM PRESS

PRACTICAL
OBJECT-ORIENTED
DESIGN IN RUBY
AN AGILE PRIMER

面向对象设计实践指南
Ruby语言描述

[美] Sandi Metz 著　　张雪平 彭晓东 译

人民邮电出版社
北京

图书在版编目（CIP）数据

面向对象设计实践指南 : Ruby语言描述 / (美) 梅茨 (Metz,S.) 著 ; 张雪平, 彭晓东译. -- 北京 : 人民邮电出版社, 2014.1
书名原文: Practical object-oriented design in Ruby:an agile primer
ISBN 978-7-115-33245-5

Ⅰ. ①面… Ⅱ. ①梅… ②张… ③彭… Ⅲ. ①计算机网络－程序设计 Ⅳ. ①TP393.09

中国版本图书馆CIP数据核字(2013)第230826号

内 容 提 要

本书是对“如何编写更易维护、更易管理、更讨人喜爱且功能更为强大的 Ruby 应用程序”的全面指导。为帮助读者解决 Ruby 代码难以更改和不易扩展的问题，作者在书中运用了多种功能强大和实用的面向对象设计技术，并借助大量简单实用的 Ruby 示例对这些技术进行全面解释。

全书共 9 章，主要包含的内容有：如何使用面向对象编程技术编写更易于维护和扩展的 Ruby 代码，单个 Ruby 类所应包含的内容，避免将应该保持独立的对象交织在一起，在多个对象之间定义灵活的接口，利用鸭子类型减少编程间接成本，合理运用继承，通过组合构建对象，设计出最划算的测试，解决不良设计的 Ruby 代码所导致的常见问题等。

本书适合所有对面向对象设计和 Ruby 编程语言感兴趣的程序员阅读参考。

◆ 著 [美] Sandi Metz
译 张雪平 彭晓东
责任编辑 杨海玲
责任印制 程彦红 焦志炜

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：14.25
字数：310 千字 2014 年 1 月第 1 版
印数：1－3 000 册 2014 年 1 月北京第 1 次印刷
著作权合同登记号 图字：01-2013-1131 号

定价：55.00 元

读者服务热线：(010)81055410 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

版权声明

译者序

让编写的应用程序具有可维护性和可扩展性，并在面对不断变化的需求时，它们也能泰然处之，这是每一位有责任心的程序员都在不懈追求的目标。但出于各种原因，并不是所有人都敢承认自己在这一点上已做得很出色。

本书是 Sandi Metz 对自己三十多年开发经验以及演讲的深度总结。在原书中，她以浅显易懂的文字和细腻的表达方式，对软件设计过程中常会遇到的问题做了深入的分析和细致的讲解。其中，既有对常见实用设计原则（如 SOLID 原则、DRY 原则等）和设计模式（如工厂模式、组合模式、模板方法模式等）的介绍，也有对面向对象设计原则的详细分析和讲解，并且还通过大量的示例进行说明。全书重点突出、详略得当，可谓恰到好处。在阅读那些形象生动的示例时，你会不由自主地认同作者的观点。她认为："具有透明性、合理性、可用性和典范性这四种特点（简写为 TRUE）的代码，不仅能满足今天的需求，而且也易于更改，以满足未来的需求。"透过她的分析和介绍，你定会茅塞顿开；同时，为曾经可能疑惑的问题找到答案，并对问题的本质形成清晰的认识。

全书讲述的重点是实用面向对象设计技术，演示代码借用的是 Ruby 语言。尽管如此，Ruby 语言像 Smalltalk 语言一样，也是一种完全面向对象的语言，而且它还是一种动态语言。因此，就编程语言本身而言，它并不会比其他语言更难。在阅读书中的示例时，只要你对其他任何一种面向对象语言（如 C++、Java、C#等）有所了解，那么语法上的差异并不会成为你阅读和理解这些示例的障碍。

本书由我和彭晓东共同翻译完成。正如原书前言里所描述的："本书与面向对象软件设计有关。它不是一本学术巨著，它讲述的是程序员如何编写代码的故事。"希望大家在阅读本书后，能从中获得关于"如何编写能带给你快乐的代码"的实用方法。

在翻译过程中，除得到身边许多朋友及家人的支持外，还得到了人民邮电出版社的杨海玲女士和原书作者 Sandi Metz 给予的无私帮助。在此，向所有为本书的翻译及出版付出努力的人们表示衷心的感谢。鉴于个人能力所限，错漏之处在所难免，敬请大家批评指正；同时，对可能造成的误解深表歉意！

张雪平

2013 年 8 月于上海

序

在软件开发过程中存在着一个不争的事实，即随着代码的增长和对正构建中的那个系统的需求发生变化，那些在当前系统里还不存在的附加逻辑将会被添加进去。几乎在所有的情况里，代码的可维护性在其整个生命周期中始终比优化其现有状态更加重要。

使用面向对象（OO）设计能保证你的代码比不使用它更易于维护和向前发展。如果你不了解编程，那么要如何才能解开那些使用 OO 实现可维护性的秘密呢？事实是，我们中的许多人在编写完全面向对象的代码方面从未进行过系统的训练。多数情况下，他们的技术很大程度上都是经由身边的同事、大量旧书籍和在线资源的潜移默化而获得的。如果说在学校期间我们还能在 OO 方面打下一定的基础的话，那么通常是使用类似 Java 或 C++那样的语言实现的（幸运的话，也可能遇到有人使用 Smalltalk 教学）。

Sandi Metz 所编写的这本书涵盖了使用 Ruby 语言来进行 OO 设计的基本内容。也就是说，它完全适合于无数想要成长为像熟练程序员那样具有专业开发技能的 Ruby 和 Rails 新人。

Ruby 自身很像 Smalltalk，也是一种完全面向对象的语言。在其内部，甚至包括很多基础数据结构（如字符串和数字），都是使用带有行为的对象来表示的。当在 Ruby 中编写应用程序时，你也可以通过自己编写对象来达到这一目的：只需各自封装好某个状态，并定义好自己的行为即可。如果你对 OO 还没有足够的经验，那么可能会感到有些气馁：不知道该如何开始才好。本书将一步一步给你指导：从“往一个类里放置什么内容”这样基础的问题，到“单一职责原则”这样基本的概念，再到“如何在继承和组合之间获得平衡”，以及弄清楚“如何测试隔离环境里的对象”。

不过，最精彩的部分还是 Sandi 所传达出的信息。她拥有丰富的经验，是你在社区能遇见的最优秀成员之一（我认为她透过自己的写作充分地体现了这一点）。虽与 Sandi 相识多年，不过我曾经还是对她产生过怀疑：她的手稿是否真的能带给我们快乐，如同现实生活中认识的 Sandi 所带给我们的快乐一样。我很高兴地宣布：这点毋庸置疑，它的确带来了快乐。这也正是我非常欢迎她成为我们最新一位专业 Ruby 丛书作者的原因。

——Obie Fernandez

“Addison Wesley 专业 Ruby 丛书”丛书编辑

前 言

我们既要竭尽全力，也要做得有意义。与其他所有情形一样，我们要享受整个过程。

那些从事软件编写工作的人们真的很幸运。构建软件是一件让人感到快乐无比的事情，因为我们可以用创造力把事情做好。我们照这种方式编排的生活可以“鱼和熊掌，两者兼得”：既可以享受代码编写所带来的乐趣，也可以非常确定地认为我们所编写的代码能够派上用场。我们做的是有意义的事情。我们是现代工匠，正在建造构成当今现状的各式建筑，而且参与人数一点儿不会比泥瓦匠或桥梁建造者少。我们完全有理由对自己取得的成就引以为傲。

从热情澎湃的新手到厌世嫉俗的前辈，无论他是工作在最轻量级的互联网创业领域，还是呆在固步自封、历史悠久的企业里，所有的程序员都具有这样一个共同特点：我们既要竭尽全力，也要做得有意义，同时还要享受整个过程。

于是，当软件出现问题时特别让人感到不安。糟糕的软件会阻碍我们实现目标，扰乱我们的幸福生活。曾经我们还感觉生产力十足，可现在却屡遭挫败。它曾经的运行快速如飞，可现在却慢如蜗牛。我们曾经还是心平气和，可现在却心灰意冷。

当要投入更多成本才能把事情做好时，我们便会产生这样的挫折感。我们内部的计算器总是在不断地运作，不停地将总的完成量与总的投入精力进行对比。当完成工作的成本超过其价值时，便会认为我们的努力将白费。如果编程能带来快乐，那是因为它让我们变得很有用。当这种事情变成一种痛苦时，便会出现这样的迹象：我们相信自己还可以，而且也应该做更多的工作。我们的快乐要紧跟工作的步伐。

本书与面向对象软件设计有关。它不是一本学术巨著，它讲述的是程序员如何编写代码的故事。它教授了如何编排软件以便让今天更富有成效，同时也能让下月以及明年继续保持住这种效果。它展示了如何编写应用程序，让它不只是在当前能获得成功，同时还要适应未来的变化。它能让你在应用程序的整个生命周期里提高生产率，减少成本。

本书相信你会竭尽全力，而且也为你提供了最合适的工具。它完全注重实用，其本质上完全是一本关于“如何编写能带给你快乐的代码”的书。

读者对象

本书假设你曾经至少尝试过编写面向对象的软件。你是否觉得已获得成功，这一点无关紧

要，只要你在某种面向对象语言方面做过尝试即可。第 1 章是简要概述了面向对象编程（OOP），其目的在于要定义一些常见的术语，而不是想对编程进行教学。

如果你想学习面向对象设计（OOD），但尚未在此方面做过任何编程，那么在阅读本书之前至少需要找点教程学习一下。OOD 可以解决很多问题，而饱受这些问题所带来的苦楚，恰好是完全理解那些解决方案的一个先决条件。有经验的程序员可以跳过这一步，但对大多数读者来说，如果他们在开始翻阅本书之前能编写一些 OO 代码，则一定会感到更轻松。

本书使用 Ruby 语言来讲解 OOD，但为理解本书里的概念，也不需要你对 Ruby 有过多的了解。书中有很多代码示例，但都很简单。如果你对某种 OO 语言有过编程经验，那么很快便会发现 Ruby 其实很容易理解。

如果你对某种静态类型的 OO 语言（如 Java 或 C++）比较熟悉，那么你已经有了阅读本书所需要的背景知识。事实上，Ruby 是门动态语言，这将大大简化示例的语法；同时，也会让设计思想直接体现其本质。但本书中的每一个概念都可以直接转换到静态类型 OO 语言。

本书组织

第 1 章首先概述了 OO 设计的原因和时机，其次概述了面向对象编程。这一章是相对独立的。你可以先阅读它，后阅读它，或者完全跳过它。尽管你可能正在开发某款应用程序，但如果苦于缺乏设计，你会发现这章还是很有意义的。

如果你已拥有编写面向对象应用程序的经验，想要跳跃式前进，那么完全可以从第 2 章开始。如果这样做，那么你偶尔会遇到某个陌生词汇。那时，你再回过头来浏览第 1 章对面向对象编程部分的介绍。这一章主要介绍和定义了本书用到的一些常见的面向对象术语。

第 2 章到第 9 章渐进式地对面向对象设计做了解释。第 2 章涉及的是如何决定单个类所包含的内容。第 3 章说明了对象之间为何彼此纠缠不清，同时也展示了如何将它们分离。这两章关注的重点是对象而非消息。

第 4 章强调的重点从以对象为中心的设计开始转向以消息为中心的设计。第 4 章与接口定义有关，它关心的是对象之间如何进行对话。第 5 章与鸭子类型（Duck Typing）技术有关，并引入了不同类的对象可以扮演公共角色的思想。

第 6 章讲解了与经典继承相关的技术。接着，这项技术在第 7 章里被用于创建鸭子类型的角色。第 8 章对通过组合构建对象的技术进行了解释，并对组合、继承和动态类型角色共享这三者如何进行选择提供了指导。第 9 章专注于测试设计，它使用了本书前面章节里的代码来进行说明。

每一章都建立在前一章的概念之上，并且都包含了很多代码，所以最好是能按顺序阅读。

使用方法

在具有不同背景的读者看来，本书所提供的内容肯定存在差异。对于那些已熟悉 OOD 的人来说，他们会在本书里找到值得思考的事情，可能还会见到一些新的观点，也可能会对其中提出的某些建议持不同的意见。因为就 OOD 而言，没有绝对的权威；对原则（以及本作者）进行挑战，反而能够全面加深理解。最后，是你在主导自己的设计，因此需要你来提问、测试和选择。

虽然本书可能会引起各层次读者的兴趣，但其编写目标更偏向于新手。如果你是一位新手，那么这部分的介绍便是特别为你而准备的。请记住这一点：面向对象设计不是“黑魔法”。它是件很容易做到的事情，只是你还不太了解而已。事实上，当你读到这里的时候，已表明你很关心设计。这种对学习的渴望是能够从本书受益的唯一先决条件。

第 2 章到第 9 章对 OOD 原则进行了解释，并提供了直观的编程规则。这些规则所具有的意义，对专家和新手来说可能不尽相同。如果你是一位新手，那么如有必要，直接按照这些规则开始即可。这种早期的顺从可以避免很多麻烦，在你有了足够经验之后再自行决定。在这些规则真正发挥作用时，你一定已拥有了足够的经验来组成自己的规则，而你作为一名设计师的职业生涯也会就此开始。

致 谢

本书能够诞生简直就是一个奇迹。事实上，它融入了太多人的付出与鼓励。

在整个漫长的编写过程中，Lori Evans 和 TJ Stankus 对每一章都提供了前期的反馈。他们住在北卡罗来纳州的达勒姆，因此无法躲开我。尽管如此，我对他们的感激之情丝毫不减，感谢他们所给予的帮助。

在本书的编写中途，创作花费的时间差不多是最初估计时间的两倍。在这一情况发生之后，Mike Dalessio 和 Gregory Brown 阅读了草稿，并给予了宝贵的反馈和支持。他们的鼓励和热情让这项工程在黑暗的日子里得以存活下去。

在接近完工时，Steve Klabnik、Desi McAdam 和 Seth Wax 审阅了此书。他们就像是你（绅士般的读者）的替身，对审阅工作一丝不苟。他们的印象和建议所引起的变化让所有后来的人都受益无穷。

迟到的草稿交由 Katrina Owen、Avdi Grimm 和 Rebecca Wirfs-Brock 进行了仔细、全面的阅读。本书也根据他们宽容、全面的反馈进行了大量修订。在 Katrina、Avdi 和 Rebecca 卷入进来之前，我对他们还完全不了解。非常感谢他们的参与，也被他们的慷慨所感动。如果你发现本书能带来帮助的话，那么当你下次遇到他们时，请记得当面向他们表示感谢。

我很感激 Gotham Ruby 小组，也同样感激在 GoRuCo 2009 和 2011 由我所做的设计讨论会上，每一位对此表达出赞赏之情的人们。在 GoRuCo，人们把握住了了解未知情况的机会；同时，也给了我一个可以表达这些思想的论坛，本书便是从那里开始的。Ian McFarland 和 Brian Ford 记录下了这些谈话，而且他们对此项目所拥有持续的热情。这种热情既具有感染力，也非常令人信服。

在本书的创作过程中，还从 Pearson Education 的 Michael Thurston 那里获得了极大的帮助。他就像一艘平静和组织有序的远洋渡轮，驶过被我的捣蛋作品搞得波浪翻滚的大海。我希望你也能看到他所面对过的问题。带着无尽的耐心和仁慈，他坚持认为写作应该以一种方便阅读的结构进行编排。我相信他的努力得到了回报，也希望你能认同这一点。

同时，我还要感谢 Debra Williams Cauley。她是我在 Addison-Wesley 的编辑。在 2006 年，于芝加哥举办的第一届 Ruby on Rails 大会上，她无意中听到走廊里传来一段刺耳的责骂。于是，她发起了这个运动，并最终导致了本书的出现。尽管我尽了最大努力，她也不肯接受任何否定的答案。她巧妙地从一个参数转移到下一个，直到最终找到了足以让她信服的那个。这点恰恰

反映出了她的执着与奉献精神。

我欠整个面向对象设计社区一笔债。在这本书里，我没有对那些思想进行补充，我只是一名翻译，我站在了巨人的肩膀上。不用多说，对那些思想的所有赞誉都应该归于其他人——翻译的失败则由我一个人承担。

最后，本书的诞生还要归功于我的合伙人 Amy Germuth。在此项目的开始之初，我根本无法想象要去编写一本书。在她看来，人们都在做这样的事情，这让创作变成了很自然的事情。你手里拿着的这本书正是表达了对她无尽耐心和不断支持的敬意。

感谢所有人！

目 录

第 1 章

面向对象设计

世界是过程式的。时间不停在向前流动，而事件也一个接一个地逝去。你每天早上的过程或许就是：起床、刷牙、煮咖啡、穿衣，然后上班。这些活动都可以使用过程软件来建模。因为了解事件的顺序，所以你可以编写代码来完成每一件事情，然后仔细地将这些事情一个接一个地串在一起。

世界也是面向对象的。与你互动的对象可能包括有你的老伴和猫，或者是车库里的旧汽车和一大堆的自行车零件，又或者是你的那颗扑通跳动的心脏，以及用来保持健康的锻炼计划。在这些对象中，每一个都拥有它们自己的行为，而且它们之间某些的交互可能还是可预测的。例如，你的老伴意外地把猫给踩了一下，从而引起一个激烈的反应，大家的心跳频率都迅速升高，同时也为你增添了一种新的锻炼方式，这是完全可能的。

在由对象构成的世界里，新的行为编排在很自然的情况下便会出现。你不用显式地为“老伴踩猫”这一流程编写代码，而你所需要的是这样两个对象：一个会抬脚的老伴和一只不喜欢被踩的猫。把这两个对象一起放到一个房间里，行为的意外组合便会出现。

本书与面向对象软件的设计有关，它将世界看作是对象之间一系列的自然交互。面向对象设计（OOD）要求你从认为世界是由一组预定义过程构成的观念，转变至认为世界是由多个对象之间的一系列消息传递构成。OOD 的失败看起来很像是编码技术的失败，但它们实际上是视角的失败。学习如何进行面向对象设计的第一个要求便是要让自己沉浸在对象之中。一旦你采用面向对象的视角，那么其余部分也会水到渠成。

本书将通过沉浸式的过程给你指导。本章一开始会对 OOD 进行一般性的讨论。它首先是讨论设计的情况，接着会描述何时进行设计以及如何对它进行评判。在本章的结尾是对面向对象编程的概述，它定义了全书所用的术语。