The Pragmatic Programmers

Rails 5

Rails 5敏捷开发

Agile Web Development with Rails 5

荣获Jolt技术图书大奖

[美] Sam Ruby
Dave Thomas
David Heinemeier Hansson 著

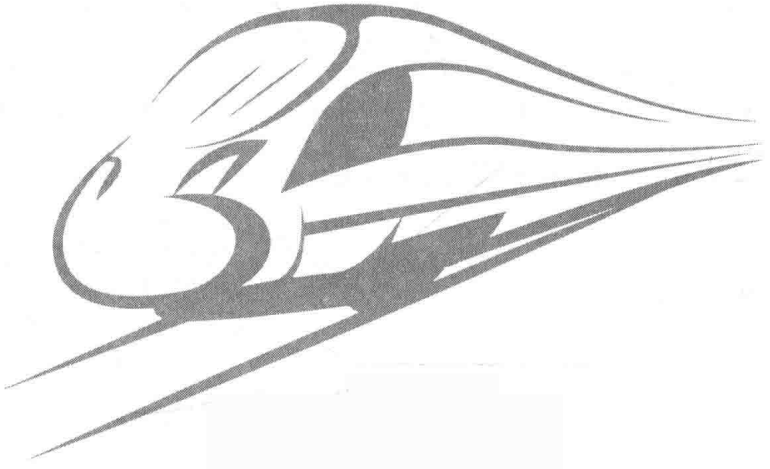
安道 叶炜 译

大疆Ruby技术团队 审校

華中科技大學出版社

Rails 5敏捷开发

Agile Web Development with Rails 5



[美] Sam Ruby
Dave Thomas
David Heinemeier Hansson 著

华中科技大学出版社
中国 · 武汉

内 容 简 介

本书以讲解“购书网站”案例为主线，逐步介绍 Rails 的内置功能。全书分为三部分，第一部分介绍 Rails 的安装、应用程序验证、Rails 框架的体系结构，以及 Ruby 语言知识；第二部分采用迭代方式构建应用程序，然后依据敏捷开发模式开展测试，最后使用 Capistrano 完成部署；第三部分补充日常实用的开发知识。本书既有直观的示例，又有深入的分析，同时涵盖了 Web 开发各方面的知识，堪称一部内容全面而又深入浅出的佳作。第 5 版增加了关于 Rails 5、Ruby 2.2 新特性和最佳实践的内容。

湖北省版权局著作权合同登记　图字：17-2017-396 号

图书在版编目(CIP)数据

Rails 5 敏捷开发/(美) 山姆·鲁比，(美) 戴夫·托马斯，(美) 大卫·海尼梅尔·汉森著；安道，叶炜译. —武汉：华中科技大学出版社，2018.1

ISBN 978-7-5680-3659-7

Ⅰ.①R… Ⅱ.①山… ②戴… ③大… ④安… ⑤叶… 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(2017)第 323429 号

Rails 5 **敏捷开发**

Rails 5 Minjie Kaifa

[美]Sam Ruby, Dave Thomas, David Heinemeier Hansson　著

安道　叶炜　译

大疆 Ruby 技术团队　审校

策划编辑：徐定翔

责任编辑：陈元玉

责任监印：周治超

出版发行：华中科技大学出版社(中国·武汉)　电话：(027)81321913

武汉市东湖新技术开发区华工科技园　邮编：430223

录　　排：华中科技大学惠友文印中心

印　　刷：湖北新华印务有限公司

开　　本：787mm×960mm　1/16

印　　张：30

字　　数：566 千字

版　　次：2018 年 1 月第 1 版第 1 次印刷

定　　价：115.00 元

本书若有印装质量问题，请向出版社营销中心调换

全国免费服务热线：400-6679-118　竭诚为您服务

读者对本书的赞誉

Early praise for Agile Web Development with Rails 5

《Rails 5 敏捷开发》是快速掌握 Rails 开发的最佳资源。尽管已推出多年，这本书仍然很有价值。

➤ Prathamesh Sonpatki

BigBinary 公司总监，Rails 问题审核团队成员

本书的内容组织得非常出色。前两部分介绍如何构建 Rails 应用，演示项目简单易懂，全面展示了 Rails 为开发者创造的价值。第三部分的很多议题也很有价值。总而言之，这是一本好书，值得继续向 Rails 新手推荐！

➤ Jeff Holland

Ackmann & Dickenson 公司高级软件工程师

不管使用哪种语言进行 Web 开发，本书都值得一看！

➤ Charles Stran

The Blaze 公司产品工程与设计总监

这本书的新版依然很棒，在 Rails 开发时我反复参考。它是关于 Rails 开发的最佳图书之一。

➤ Stephen Orr

Siftware 公司高级开发工程师

译者序

如果说 Rails 是世界上开发效率最高的 Web 开发框架，恐怕没有人会有异议。Rails 以其“约定胜于配置”的先进设计理念和对 Ruby 语言元编程能力的娴熟应用，创造了 Web 开发框架历史上的一个奇迹。从开始流行至今，Rails 一直都是其他语言开发框架的模仿对象，例如 PHP 语言的 Yii 框架、Python 语言的 Django 框架，等等。一直被模仿，从未被超越，这句话放在 Rails 身上真是恰如其分。

从商业应用的角度看，互联网从兴起到现在已经经历了 Web 1.0、Web 2.0 和移动互联网时代，正在进入人工智能和物联网时代。而移动互联网本身经历了 2G、3G 和 4G 时代，即将进入 5G 时代。未来，移动互联网仍将是最重要的基础设施和商业竞争的主战场，因此 Web 应用不仅不会走向夕阳薄暮，反而会迎来一个更加蓬勃的春天。在这样一个背景下，Rails 将继续成为程序员手中的利器，帮助创业者和商业公司在时代潮头横刀立马。

作为 Ruby China 社区（`https://ruby-china.org`）的用户，译者一直为国内 Ruby 社区融洽的氛围和高质量的讨论内容而感到庆幸不已，同时也想尽自己的绵薄之力回馈社区。对于 Rails 程序员来说，入门和提高都不是一件易事，这不仅因为 Ruby 语言表达能力极强、灵活多变，更因为 Rails 本身功能完备、包罗万象。可以说，Rails 开发就像演奏音乐，既可以行云流水，也可以凝滞生涩，强者以一当十、游刃有余，弱者步履蹒跚、漏洞百出，两者高下立判。因此，学习 Rails 尤其需要名师和秘籍，不仅要领新手入门，扶上马、送一程，更要能帮助开发者掌握要领、理清思路、拓宽视野，为继续修行提高指明方向、注入

动力。《Rails 5 敏捷开发》正是这样一本好书，入门提高皆宜。我们也为能有机会翻译这样一本好书而不胜欣喜。

在策划和翻译本书的过程中，华中科技大学出版社的徐定翔老师和 Ruby China 社区的各位同仁给予了热情鼓励和实际帮助，在此一并表示感谢。同时也要感谢家人的理解和包容，正是在你们的支持下，长达数月的翻译过程才能最终成为走向胜利的长征，让一切的艰辛和付出都有了回报。

本书承蒙大疆 Ruby 技术团队的审校，他们为本书译稿提供了众多宝贵意见，特此感谢！当然，书中若仍有不当之处，所有责任都在译者自身。

希望译者的工作成果能够为大家学习 Rails 助一臂之力。最后，以一句结语和大家共勉：学习 Rails，永远在路上！

译　者

2017 年 9 月 10 日

序

Foreword to the Rails 5 Edition

学习 Ruby on Rails 是一个正确的决定。这门语言、这个框架，以及围绕它们而生的社区从未像今天这样完美，而且融入这一社区也从未像今天这样容易。早期的荒野已不复存在，随之而去的还有一些兴奋的拓荒者，现今这片土地生机勃勃、欣欣向荣，实用主义至上。

希望你在阅读本书的过程中能窥见这一历程的成果。Ruby on Rails 接手了大多数开发者多数时候所要做的大量工作。对 Web 开发而言，这个量是相当多的。有时，甚至有些过火。

但是，不要畏缩，在着手开发之前你无须了解细枝末节。Ruby on Rails 的设计原则是尽量平缓学习曲线，让你不断进步。

你不可能在一夜之间就精通全栈 Web 开发。不是说有了 Ruby on Rails 就不用掌握各方面的知识，你要知晓 HTTP、数据库、JavaScript、面向对象最佳实践和测试方法论。总有一天你会被逼着去精通这些知识，到那时既不用担心，也不要奢望“21 天之后”（或者某些出版商试图麻痹你而营销的其他时间）就能做到。

有所获益固然让人欣喜，但是你所踏入的社区才是重点。Ruby on Rails 社区关注的焦点是如何为 Web 编写优秀的软件。初看起来可能让人不解，把 `if` 语句放在条件的开头还是在末尾使用 `unless` 语句真的这么重要吗？是的，很重要，让更多的程序员注重这种细节是我们的重要使命之一。

这是因为 Ruby on Rails 不单单是让你快速完成任务，完成任务只是其使命的一部分，而且是很小的一部分。我们的更大愿景是让编写 Web 软件变得有趣、有益、给人启迪，让你渴望学习各种技艺，把学习过程变成一次神奇的冒险。

Rails 的每个新版本都会扩大处理问题的范围（不置可否，它不是极简主义者心目中的框架）。Rails 5.0 也不例外，这个最新的大版本进入了一个全新的领域：实时 Web，并提供一种真正可行的解决方案。

但是，不要操之过急。要学的内容很多，而我迫不及待地想看到你的作品了。过去 13 年我一直使用 Ruby 编程并开发 Rails，每当看到有新开发者发现这门语言和这个框架的美妙之处，我便充满前行的动力。从某种程度上说，我甚至是嫉妒的。

欢迎进入 Ruby on Rails 的世界！

David Heinemeier Hansson

前言

Preface to the Rails 5 Edition

Rails 1.0 发布于 2005 年。起初，Rails 是不太为人所知的前沿工具，经过 10 多年的发展，如今已变得成熟、稳定，集成了大量相关的库，并成为其他框架做基准测试的比较对象。

你即将阅读的这本书自 Rails 出现伊始就问世了，而且随着 Rails 的发展而变迁。本书的早期版本是一个小型框架的完整参考指南，那时在线文档不仅缺乏，而且一致性不高。而如今，本书介绍的是整个 Rails 生态系统，书中充满各种信息，远比你所需、所想的要多。

本书不仅随着 Rails 而发展，Rails 也随着本书一起发展。本书的撰写征询了 Rails 核心团队的意见，不仅使用 Rails 的各个版本测试书中的代码，反过来，Rails 自身也使用书中的代码来进行测试，倘若测试失败，便不会发布新版本。

所以，请放心，书中给出的方案不仅肯定可用，而且从 Rails 开发者的角度说明了使用 Rails 的最佳方式。希望你在阅读本书的过程中能像我们撰写时一样愉快。

本书涵盖 Rails 5.0。虽然很多命令换成了新的，但是背后的开发模型不变。与 Rails 4.0 相比，新增的重要功能（例如 Web 控制台和 Action Cable）可以通过 gem 添加到使用旧版 Rails 的应用中。Rails 的这个版本是为了进化，而不是革命。

目前，Rails 5 最重要的新功能是 Action Cable。这一功能不仅增加了代码行数，也产生了一定影响，但本书并不会着重说明，这是为什么呢？

这是因为难以实现的功能未必要在一本书中得到更多关注（显然体现在页数上）。虽然重要性是因素之一，但绝非唯一因素。

对于本书应该涵盖 Rails 5 的哪些新功能和重要变化，下面说明我们的思考过程。先说看似着墨不多的三个重要功能，然后介绍对本书而言更加重要的三个次要功能，最后评述为什么这么做。即使你刚接触 Rails，不能完全理解这里提到的所有内容也没关系，了解我们确定本书所涵盖功能的思考过程有助于你弄清本书想要解决的问题。

Action Cable

Action Cable 是一项工程壮举，既需要新标准的支持，也需要新标准的普及，还需要修改多个组件——为此，Rails 团队甚至把 Web 服务器由 WEBRick 换成了 Puma。

最初，我计划说明如何使用 Action Cable 构建一个聊天应用，但是聊天应用极难正确实现。与 David Heinemeier Hansson 讨论之后，他建议我从小处着手，多介绍 Rails 提供的功能，别花太多篇幅说明如何构建某一种类型的应用。

最终，我们决定实现一个相对简单的功能：动态更新价格。这个功能涉及所有基础知识（定义频道、创建订阅，以及发送消息）。掌握基础之后，就没有什么能限制你去构建各种应用了。

Rails API

如果你是 Backbone.js、AngularJS、React 等框架的拥趸，你会发现这个功能使用很便利。Rails API 是 Rails 的子集，专为这类应用定制。

这就涉及两个问题。其一，这个 API 本质上是 Rails 的子集，因此新东西不多。其二，也是更为重要的，无法选择一个标准的通用 JavaScript 框架。

如果没有第二个问题，我们本可以接着第 24.4 节讲下去。但可悲的是，JavaScript 框架太多了，而且每个框架的学习曲线都很陡峭。因此我们还在观望，目前这还是高级话题，超出了本书的范畴。我们相信需要这一功能的人能在网上找到所需的资料，稍后将详述这一点。

Turbolinks 3

Turbolinks 3 是 Rails 应用默认依赖的一个包（严格来说是一个单独的包），因此可以算得上是核心组件。那么，为什么不深入说明呢？

主要是因为它太好用了，悄无声息，我们根本感觉不到它的存在。不闻不问，它就能加快速度。

本书之前的版本曾说明在遇到问题时如何禁用 Turbolinks，但在本书中我们不会遇到类似的问题，因此 Turbolinks 的相关内容会少一些。

那么，哪些变化介绍得相对多一些呢？

- 目前，影响最大的变化是所有 `rake` 任务都换成了 `rails` 命令。例如，`rake test` 变成了 `rails test`。大多数情况下，这只是一个全局搜索替换的问题。但要注意，本书并未专门介绍这一变化，毕竟这是一本针对 Rails 新手的书，过多讨论之前的行为对新手没有太大帮助。
- 控制器测试现在变成了集成测试，因此也不再属于单元测试。这也意味着 Rails 把 `assigns()`等方法移出了核心（可以通过 `rails-controller-testing` gem 添加回来）。这一变化可不是简单地搜索替换就能解决的，而要完全重写很多测试。同样，本书对这一变化也没有着墨太多，因为本书关注的是目前的运作方式。
- 同步邮件发送换成了异步邮件发送。应用基本上无需改动，但测试要做一些修改。

此外还有其他改动，例如，`belongs_to` 现在默认必须指定，因此，要想更新现有关系，就必须明确将其标记为可选的。但是前述几项功能影响的页数更多。

可是，这些对身为读者的你有什么影响呢？

要知道，本书针对的是当今的 Web。本书第一版出版时，很多今天可用的资源还未出现，如 Stack Overflow、Hacker News、GitHub，等等。如今，只要知道

怎么问问题，知道怎么识别正确答案，你就能走得更远。

本书已充分考虑到这个现实。我们发现 Rails 相关的问题在网上大都能找到答案。我们知道大家等待 Rails 5 发布的时间比以往要长（Rails 4.2 发布于 2014 年 12 月，Rails 4.0 发布于 2013 年 6 月），因此网上有大量关于 Rails 5 新特性的文档和问答（有些质量不高，多数很好，有些极棒）。

要知道，过时的控制器测试示例可能比 Action Cable 早期测试版的文档更容易误导人。

因此，如果本书能为你打下基础，让你知道如何正确描述问题，如何甄别网上有冲突的答案，那么本书的任务就达成了。

在开始阅读之前，我还要再提醒一下。一般来说，Ruby 语言，尤其是 Rails Web 框架特别吸引专己守残的人。由于某些原因，这一点在测试库上体现得尤为明显。这没什么，世界之大，包罗万象。但是要记住，本书采用的方式首先考虑的是 Rails 核心团队的选择。在此之余，我们才会提供其他选择，并告诉你如何选择适合自己的。

说不定有一天你会发现自己更喜欢别的方案。当那一天来临之际，你便会成为我们的一员，也变得像我们一样固执己见。那时，本书的任务也就完成了。

Sam Ruby
rubys@intertwingly.net
2016 年 9 月

致谢

Acknowledgments

Rails在不断进化，本书也不例外。Depot应用的很多部分重写了多次，书中的文字和代码也做了更新。为了避开已经弃用的功能，本书的结构做了多次调整，因为一些热极一时的功能如今已光辉不再。

所以，如果没有Ruby和Rails社区的大力协助，本书就无法付梓。这一版的草稿得到了众多人员的审校，他们是：

Sefik Kanat Bolazar	Kosmas Chatzimichalis	Sage Hamblin
Michael Hansen	Rod Hilton	Jeff Holland
Bruce Jackon	Askarbek Karasaev	Nigel Lowry
Graham Menhennitt	Nathan Ruehs	Prathamesh Sonpatki
Charles Stran	Masaki Suketa	David Wilbur
Stephen Orr		

本书的每一版都先发布测试版，这些早期版本以 PDF 电子书的形式发布，读者可以在线发表评论。读者针对这一版发表的评论中，建议和缺陷报告超过 50 条。众人的智慧融合一处，大大提升了本书的价值。感谢大家对测试版的支持，感谢大家提供了这么多有价值的反馈。尤其感谢 Kosmas Chatzimichalis，感谢他不辞辛苦。

最后，感谢 Rails 核心团队，他们给予了巨大帮助，为我们解答疑问、检查代码片段、修正缺陷。在 Rails 发布的过程中，甚至有一步是确认新版本不会破坏本书中的示例。

Sam Ruby

引言
Introduction

Ruby on Rails 是一个框架，一个使 Web 应用的开发、部署和维护变得更容易的框架。10 多年间，Rails 从无人知晓的玩具变成世界瞩目的杰出工具；更重要的是，它已经成为实现各种应用的首选框架。

这是为什么呢？

Rails 就是趁手
Rails Simply Feels Right

很多开发者厌倦了他们一直用来创建 Web 应用的技术。不管用的是 Java、PHP 还是.NET，开发者越发觉得它们太难用了。恰逢此时，Rails 横空出世，人们发现 Rails 要简单得多。

光是简单还不够。这些人毕竟是编写真实网站的专业开发者，他们希望自己开发出来的应用能经受住时间的考验，所以在设计和实现时总是选择先进的专业技术。这些开发者深入研究 Rails 之后发现，Rails 可不只是一个让人快速构建网站的工具而已。

例如，所有 Rails 应用都使用模型-视图-控制器（Model-View-Controller，MVC）架构实现。Java 也有供开发者使用的 MVC 框架，如 Tapestry 和 Struts。但是 Rails 更进一步，使用 Rails 开发时，什么代码应该放在什么位置都有规定，而且应用的各部分代码之间通过标准的方式交互。

专业的程序员都编写测试代码，Rails 同样提供了这方面的支持，所有 Rails 应用都内建对测试的支持。新增功能时，Rails 会自动为那个功能创建测试样板。使用 Rails 开发的应用更易于测试，因此 Rails 应用更能得到充分测试。

Rails 应用使用 Ruby 编写，这是一门现代的面向对象的脚本语言。Ruby 简洁明了，却又不至于晦涩难懂，使用 Ruby 代码能以自然而简明的方式把想法表达出来。因此，Ruby 程序很容易编写，而且几个月之后也很容易读懂——这是非常重要的。

Rails 充分利用了 Ruby 的特性，又以新颖的方式做了扩展，解放了程序员，把程序变得更简短、更易于阅读。Rails 还把以往在外部配置文件中执行的任务放到了代码基中，这样更易于理清来龙去脉。下述代码定义了项目中的一个模型类，现在先别担心细节，我们关注的是，这么几行代码就能表达如此多的信息：

```
class Project < ApplicationRecord
  belongs_to  :portfolio
  has_one     :project_manager
  has_many    :milestones
  has_many    :deliverables, through: milestones
  validates   :name, :description, presence: true
  validates   :non_disclosure_agreement, acceptance: true
  validates   :short_name, uniqueness: true
end
```

让 Rails 代码保持短小、可读的思想还有两个：DRY 和约定胜于配置。DRY 是“don’t repeat yourself”（不要自我重复）的缩写，含义是系统中的每项知识只应该在一个地方描述。Rails 借助 Ruby 实现了这个思想。Rails 应用中很少有重复；一件事只需说一遍，只要在 MVC 架构约定的地方说一遍，以后就无需再重复了。习惯使用其他 Web 框架的程序员大多有过这样的经历：对模式稍作修改就要修改好几处代码。对他们而言，这可真是一大福音。

约定胜于配置也是一个重要思想。Rails 对应用各方面的拟合有一套默认的设定，遵守这一约定写出的 Rails 应用比使用 XML 配置的 Java Web 应用往往需要更少的代码。如果需要覆盖约定，Rails 也提供了简单的方法。

转用 Rails 的开发者还会发现别的不同之处。Rails 不只是紧跟 Web 标准的步伐，而是在定义标准。使用 Rails 还能轻易集成 Ajax、REST 式接口和 WebSockets，因为这些功能是内置的（如果你不熟悉 Ajax、REST 式接口或 WebSockets，别怕，第 11 章和第 20.1.1 节会说明）。

部署也是令开发者头痛的问题。但是对于 Rails 来说，使用一个命令就可以把

应用的连续多个版本部署到任意多台服务器中（如果发现某一个版本不够完善，还能轻易回滚）。

Rails 是从一个真实的商业应用中抽取出来的。创造一个框架最好的方法或许就是先找出一类特定应用的核心使用场景，然后逐渐从中抽取通用的基础代码。使用 Rails 开发应用时，你会发现在动手编写代码之前，手上已经有一个出色的半成品应用了。

此外，Rails 还有一些特性，一些难以言说的特性。不管怎么样，你就是觉得它趁手。当然，只有自己动手编写几个 Rails 应用之后，你才会发现我们所言不假（别急，再等 45 分钟左右）。这正是本书的目的。

Rails 敏捷
Rails Is Agile

本书的标题虽然是“Rails 5 敏捷开发”，但是你会发现我们并没有专门讲解如何在开发 Rails 应用的过程中运用某个敏捷实践法则。其实，本书不涉及多少敏捷实践法则，例如 Scrum 过程。

Rails 出现之后的这些年，敏捷的发展经历了几个阶段。一开始，它是一个鲜为人知的术语，后来变成热炒的对象，被视为正规的实践法则，而后又受到不少抨击，说某些实践法则根本不应该当成真理，最后人们纷纷回归本源。

但是敏捷不是这么容易就能说清楚的。这其中的原因既简单又微妙。敏捷深植于 Rails 的骨髓之中。

“敏捷宣言”[1]（Dave Thomas 是这份宣言的 17 位起草人之一）描述的价值观可以概括为以下四个偏好：

- 个体和互动胜于过程和工具。
- 能运行的软件胜于详尽的文档。
- 与客户协作胜于合同谈判。
- 应对变化胜于固守计划。

Rails 非常注重个体和互动。不涉及繁重的工具、复杂的配置和冗长的过程，

[1] http://agilemanifesto.org/.

有的只是开发者小组、最受欢迎的编辑器和 Ruby 代码。这是一个透明的过程，开发者做了什么事情，顾客立马就能看到，这才是真正的交互过程。

Rails 开发过程不是由文档驱动的，在 Rails 项目中，你找不到一份 500 页的说明书。但是你会发现一群用户和开发者聚在一起分析需求，设法找出实现需求的方式。随着开发者和用户对问题了解的深入，解决问题的方案也会随之改变。你会发现，Rails 框架能在开发循环中尽早交付可用的软件，虽然此时软件可能很粗糙，但是却能让用户亲身体验你将要交付的产品。

Rails 以此鼓励与客户协作。看到 Rails 项目能迅速应对变化之后，客户会慢慢相信开发团队能交付自己真正需要的产品，而不只是自己说什么，开发团队就做什么。客户与开发团队将不再对抗，而是进行建设性对话。

说到底，这都是应对变化带来的好处。Rails 强烈要求，甚至可以说是强迫遵循 DRY 原则。这意味着，一旦需要变化，Rails 应用中受影响的代码要比使用其他框架开发的应用少得多。而且，由于 Rails 应用是用 Ruby 编写的，程序概念能准确、简练地表述出来，因此，变化更容易限制在小范围内，更容易通过代码表达。

对单元测试和功能测试的特别重视，以及对测试固件（fixture）和桩件（stub）的支持，又给开发者提供了安全保障，让他们放心修改代码。有完善的测试作保障，开发者们将更有勇气面对变化。

综上所述，我们觉得，与其在开发 Rails 应用的过程中穿插讲解敏捷实践法则，还不如分析 Rails 框架自身是怎么做的。阅读教学那几章时，请想象自己正在使用这种方式开发 Web 应用：你跟客户坐在一起工作，一同商讨问题的优先级和解决方案。读到第三部分的高级话题时，再考虑 Rails 的底层结构能如何帮助你更快地满足用户的需求，减少繁文缛节。

最后，对敏捷和 Rails 还有一点要说，这么说可能显得不专业，但我还是要说：运用得当，编程将是一件乐事！

本书的读者群
Who This Book Is For

本书针对想构建和部署 Web 应用的程序员，包括新接触 Rails（可能也是新接