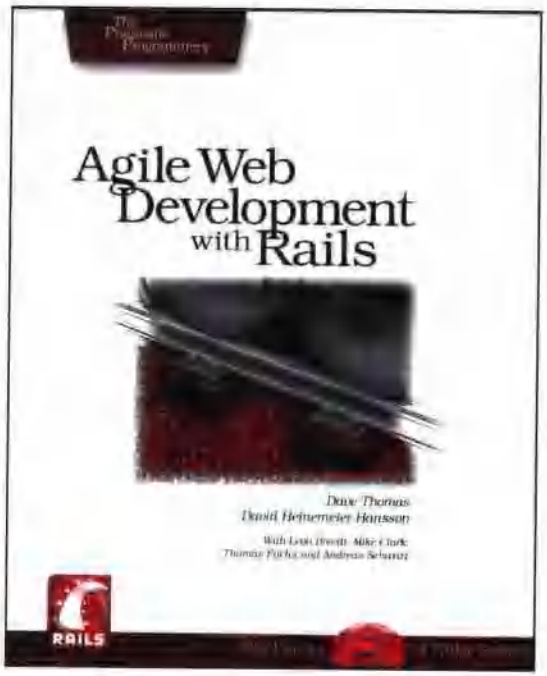
The Pragmatic Programmers

Broadview
WWW.BROADVIEW.COM.CN

应用Rails进行敏捷Web开发

Agile Web Development with Rails



[美] Dave Thomas
David Heinemeier Hansson 著

林芷薰 译
透 明 审校

RAILS

The Facets of Ruby Series

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

应用 Rails 进行敏捷 Web 开发

Agile Web Development with Rails

[美] Dave Thomas
David Heinemeier Hansson 著

林芷薰 译

透 明 审校

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

这是**第一本**关于 *Ruby on Rails* 的著作！

2006 年 3 月，它荣获 Jolt 大奖的“**最佳技术类图书**”奖！

本书作者 David Heinemeier Hansson 于 2005 年 8 月被“全球开源大会”评选为“**年度最佳黑客**”！

全书主要内容分为两大部分。在“构建应用程序”部分中，读者将看到一个完整的“在线购书网站”示例。在演示的过程中，作者真实地再现了一个完整的迭代式开发过程，让读者亲身体验实际应用开发中遇到的各种问题，以及 Rails 如何有效解决这些问题。在随后的“Rails 框架”部分中，作者深入介绍了 Rails 框架的各个组成部分。尤为值得一提的是本部分的后几章：作者先后介绍了 Web 2.0、Web Service 等流行技术在 Rails 中的支持，然后又凭借丰富的实践经验介绍了 Rails 在安全性、伸缩性、部署等方面的常见问题和解决方案。

除了上述两部分之外，对 Rails 缺乏了解的读者应该首先阅读“起步”部分，通过一个最简单的示例应用感性了解这个时下热门的 web 框架。不熟悉 Ruby 的读者应该阅读“附录”部分中的“Ruby 简介”，以便了解 Ruby 的基本语法与常见用法。整体而言，全书既有直观的实例，又有深入的分析，同时还涵盖了 web 应用开发中各方面的相关知识，堪称一部深入浅出的佳作。

0-9766940-0-X Agile Web Development with Rails by Dave Thomas, David Heinemeier Hansson.

All rights reserved. Authorized translation from the English language edition published by The Pragmatic Programmer, LLC..

本书简体中文专有翻译出版权由 The Pragmatic Programmer, LLC. 授予电子工业出版社未经许可，不得以任何方式复制或抄袭本书的任何部分。

版权贸易合同登记号：图字：01-2006-2240

图书在版编目（CIP）数据

应用 Rails 进行敏捷 Web 开发 /（美）托马斯（Thomas,D.），（美）汉松（Hansson,D.H.）著；林花薰译. —北京：电子工业出版社，2006.7
书名原文：Agile Web Development with Rails
ISBN 7-121-02872-7

Ⅰ.应... Ⅱ.①托...②汉...③林... 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（2006）第 076089 号

责任编辑：周 筠 陈元玉
印 刷：北京市天竺颖华印刷厂
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
经 销：各地新华书店
开 本：787×980 1/16 印张：36.25 字数：650 千字
印 次：2006 年 7 月第 1 次印刷
定 价：65.00 元

凡购买电子工业出版社的图书，如有缺损问题，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。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。

代译序

就在手边。这就是“知识”，Rails 的开发者们已经把他们开发 web 应用的经验融入到了这个框架之中。

另一方面——在我看来是更加重要的——原因就在本书的标题中：agile。由于一种常见的误解，有必要在这里强调：agile 这个词的准确翻译应该是“敏捷”而非“快速”。“快速”仅仅是指速度而言；“敏捷”则不仅意味着开发速度快，而且还意味着应用程序具有能够随时应对变化的灵活性、让修改既有代码与添加新功能易如反掌的优雅性、以及在快速迭代中反复折腾也不会散架的高质量。现代企业（尤其是从事互联网业务的企业）随时面对着全球化经济的机遇与挑战，飞速变化的商业环境和业务使他们对 IT 提出了更高的要求：他们不仅要快速，更要求敏捷。

Rails 正是一个具备了敏捷特性的 web 开发框架。除了框架本身的设计之外，它也得益于 Ruby 语言本身：这种语言比之 Java/C#等语言更具动态性，它的语法能够随着不同的应用场景而进化演变，这就使得开发者能够在 Ruby 基础上创造出形形色色的 DSL——简单地说，也就是让 Ruby 程序看起来更像是在描述问题领域，而不是“编写计算机程序”。实际上，Rails 框架本身就是针对 web 应用的 DSL，其中的 ActiveRecord 则是针对数据库的 DSL。此外，Rails 还内建了对于测试驱动、自动构建等敏捷实践的支持。语言、框架、开发过程的三位一体，让 Rails 具备了敏捷 web 开发的全部要素。在阅读本书的过程中，读者就可以亲身感受到这种敏捷的体验。

作为新技术最热心的尝试者与敏捷方法最忠实的推行者，ThoughtWorks 已经在 Ruby/Rails 方面积累了相当丰富的经验，并且已经用 Rails 进行了好几个真实项目的开发。从芷薰开始翻译本书起，ThoughtWorks 中国公司就与他建立了紧密的联系，并全程承担了对译本的审阅工作。如果你在阅读本书之后需要更多关于 Ruby、关于 Rails、关于敏捷方法的知识，也许 ThoughtWorks 可以给你提供必要的帮助。更多关于 ThoughtWorks 的信息，请关注 ThoughtWorks 中文网站：www.ThoughtWorks.com.cn。

我想，亲爱的读者现在大概已经迫不及待地要翻开手上的书一探究竟了。那么，就请你不要犹豫，立刻随着 David Heinemeier Hansson 和芷薰一起进入 Rails 的世界吧。最后，祝你阅读愉快、编程愉快。

透　明

2006 年 6 月 11 日于北京

试读结束，需要全本PDF请购买 www.ertongbook.com

Contents
目录

试读结束，需要全本PDF请购买 www.ertongbook.com

好的符号让大脑得以从无关紧要的工作中脱身，专注于更高级的问题……

▶ Alfred North Whitehead

第 1 章
简介
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。

当然了，所有的 web 框架都有这套宣传词。那么，Rails 的不同之处在哪里？我们可以从几个方面来回答这个问题。

首先，我们可以从架构的角度来看。时至今日，大部分开发者在开发实际的 web 应用时都已经用上了“模型-视图-控制器”（Model-View-Controller，MVC）的架构。他们发现，MVC 可以帮助他们更加清晰地构造应用程序。（在下一章我们还会详细讨论 MVC。）Tapestry 和 Struts 等 Java 框架都建立在 MVC 的基础上，Rails 也是一个 MVC 框架。当我们使用 Rails 进行开发时，每块代码都有它应该放的地方，应用程序的各个部分以一种标准的方式进行交互。换句话说，打从一开始，Rails 就帮你把整个应用程序的骨架准备好了。

然后，编程语言也是一个重要的方面。Rails 应用程序采用 Ruby 语言编写，这是一种现代的面向对象脚本语言。Ruby 语言非常简练，而又不至于简洁得难以理解，你可以清晰而自然地用 Ruby 代码描述自己的想法。这也使得 Ruby 程序非常容易编写，而且——更要紧的是——放上几个月以后也很容易读懂。

Ruby 的编程风格对于 Lisp 程序员来说非常亲切，不过对于其他人可能有点陌生。这种语言允许开发者很轻松地创建一些特殊的方法，这些方法的行为就好像是对语法进行扩展一样。有些人将这种能力称为“元编程”（meta-programming），不过我们对这些“大词”不感兴趣，我们只是认为它很有用：这种功能让程序变得更短、更易读，并且让我们能够在代码中完成一些通常须

用上外部配置文件才能完成的任务。这样一来，我们可以更轻松地看懂其中的逻辑。譬如说，下面的代码定义了一个项目中的模型类。现在你不必操心其中的细节，只要注意在这短短几行代码中描述了多少信息。

```
class Project < ActiveRecord::Base
   belongs_to                    :portfolio
   has_one                       :project_manager
   has_many                      :milestones
   has_and_belongs_to_many       :categories
   validates_presence_of         :name, :description
   validates_acceptance_of       :non_disclosure_agreement
   validates_uniqueness_of       :key
end
```

我们还可以从哲学的角度来看。Rails 的设计始终遵循两个核心原则：**DRY** 和**惯例重于配置**（convention over configuration）。DRY 也就是“**不要重复你自己**”（Don't Repeat Yourself）的缩写：系统中的每项知识只应该在一个地方描述。借助 Ruby 的强大威力，Rails 实现了这一目标。在 Rails 应用程序中，你几乎不会看到重复的代码，每件事情都只需要说一遍——你只要在符合 MVC 架构惯例的某个地方说一遍，以后就不必再重复了。

惯例重于配置也同样重要。对于“如何将应用程序组装起来”这件事，Rails 自有一套默认的规则——相当有道理的一套规则。只要遵循命名惯例，编写一个 Rails 应用程序所需的代码量比起典型的、使用 XML 配置的 Java web 应用要少得多。如果你不想遵循这些惯例，在 Rails 中也很简单。

而且，也别忘了 Rails 带来的所有那些酷玩意，包括对 web service、邮件接收、AJAX（用于高交互性的 web 应用）的整合支持，一个完善的单元测试框架（透明地支持 mock 对象），以及开发、测试、生产环境的隔离。

同样值得一提的还有 Rails 的代码生成能力（.NET 也提供了类似的能力）。它们可以创建 Ruby 代码骨架，让你专心填充应用程序的逻辑。

最后，Rails 之所以与众不同，还由于它的出身——Rails 是从一个真实的商用程序中抽取而成的。要创造一个框架，最好的办法也许就是：首先找出一类特定应用的核心场景，然后逐渐从中抽取出通用的代码基础。其结果是，在开发 Rails 应用程序时，你会发现：在你开始动手编写任何一行代码之前，一个出色的应用程序已经有一半在你手上了。

当然，Rails 还有别的好处——有些甚至很难言传。总之，Rails 就是让人感觉很爽。当然了，正所谓百闻不如一见，听我们说得再多，也不如让你自己动手写一点 Rails 的应用程序（这大概是下一个 45 分钟的任务……）。这也就是我们这本书的目标所在。

Dave 喜欢 Rails 的 10 大理由

1. 将敏捷带到了 web 开发中。
2. 我也可以——像那些酷小孩们一样——创建优美的 web 页面。
3. 它让我专注于“如何创建应用程序”，而不是“如何玩一个框架”。
4. 尽管应用程序不断成长，也会一直保持可维护性。
5. 对于客户不断提出的新要求，我可以更多地说“Yes”。
6. 测试是内建的（并且很简单），因此我们可以更频繁地进行测试。
7. 即时的反馈：编辑代码，点击“刷新”按钮，就可以在浏览器里看到变化。
8. 元编程使我可以在很高的层面上编程。
9. 代码生成让我能够更快地开始。
10. 没有 XML。

1.1 Rails Is Agile
Rails 是敏捷的

既然本书的名字叫作《*Agile Web Development with Rails*》，你可能会感到奇怪：为什么书里没有关于“在 Rails 中运用某某敏捷实践”这样的章节。

原因很简单：敏捷是 Rails 的基础所在。

我们来看看“敏捷宣言”[1]所描述的价值观，这段简短的文本描述出了敏捷开发者的选择。

- 人和交互重于过程和工具。
- 可以工作的软件重于求全责备的文档。
- 与客户合作重于合同谈判。
- 随时应对变化重于循规蹈矩。

Rails 非常强调人和交互。这里没有繁重的工具，没有复杂的配置，没有冗长的过程。这里只有开发者组成的小组、他们最爱的编辑器，以及 Ruby 代码。于是，开发的透明度更高：开发者所做的工作能够立即让客户看到。这是一个天生的交互式过程。

[1] http://www.agilemanifesto.org/。Dave Thomas 是这份文本的 17 位作者之一。

Rails 并不打算废弃所有文档，而是使你可以毫不费劲地为所有代码生成 HTML 格式的文档。但 Rails 的开发过程并不由文档驱动。在一个 Rails 项目的核心地带，你不会找到一份 500 页的规约说明书，只会看见一组用户和开发者共同发掘需求、寻找实现需求的办法。你会发现，随着开发者和用户对试图解决的问题越来越了解，解决方案也会不断变化。你会发现，这个团队在开发循环的初期就开始交付可以工作的软件。这个软件的细节可能很粗糙，但它让用户可以亲身体验你所交付的东西。

因此，Rails 也鼓励着用户与开发团队合作。一旦看到 Rails 项目能够以如此之快的速度响应变化，客户就会开始相信开发团队能够交付自己真正需要的东西，而不仅仅是自己所要求的东西。客户与开发团队之间的对抗将被建设性的讨论取代。

说到底，这些都要归结到“响应变化”的问题上。Rails 强烈要求——甚至可以说是强迫——遵循 DRY 原则，这就意味着一旦变化来临，Rails 应用需要修改的代码量比用其他框架开发的应用要少得多。而且，由于 Rails 应用是用 Ruby 编写的，而 Ruby 又能够准确、简练地描述程序概念，因此变化也更加容易被限制在一个小模块内部，并且代码修改也更容易。对单元测试和功能测试的强烈重视，以及对测试套件和 mock 对象的支持，又给了开发者一张可靠的安全网，这是进行修改时不可或缺的。有了一组完善的测试作为保障，开发者们将更有勇气面对变化。

所以，我们觉得，与其想方设法地把 Rails 应用的开发过程跟敏捷原则扯上关系，还不如让 Rails 框架自己来讲述这些原则。在阅读本书的“实例教学”部分内容时，请想象你自己正在用这种方式开发 web 应用：跟客户坐在一起工作，共同决定每个问题的优先级，然后共同为每个问题找到解决办法。然后，在读到后面的“深入参考”部分内容时，再考虑 Rails 的结构能够怎样帮助你更快地满足用户需求。

最后一点关于敏捷和 Rails 的提示：虽然这听起来有点不太专业，不过，请留意在 Rails 中编写代码有多么愉快。

1.2 Finding Your Way Around 读你所需

现在看看，本书的厚度已经超出了我们的预期。在一股激情的推动下，我们其实写出了两本书：一本 Rails 的实例教程，以及一本详细指南。

本书的前两部分将介绍 Rails 背后的概念，并提供一个相当大的范例——我们将一起构造一个简单的在线商店系统。如果你希望亲身体验一下 Rails 编程的感觉，这是一个不错的起点。实际上，大多数读者似乎乐于一边读书一边亲手构造这个示例应用。如果你懒得敲键盘，也可以直接下载源代码。[2]

本书的第 3 部分（从原书第 181 页开始）则会详细介绍 Rails 的诸多功能。如果你想弄清一个组件怎么用、如何高效而安全地部署 Rails 应用，就请阅读这一部分。

在阅读的过程中，你会看到下列约定形式：

真实代码

本书中展示的代码片段大多来自真实运行的示例应用，你可以下载完整的应用程序。为了帮助读者理解，如果一段代码能够在下载的应用中找到，在页边上就会有一个标记指明它所在的文件，就像这样：

depot/app/controllers/say_controller.rb

```
class SayController < ApplicationController
end
```

翻到本书的“交叉引用”部分（从原书第 527 页开始），寻找对应的编号，你就能找到包含上述代码的文件名。如果你阅读的是本书的 PDF 版本，而且你的 PDF 阅读器又支持超链接的话，你可以直接点击页面上的标记，代码就应该会出现在浏览器窗口中。某些浏览器（例如 Safari）会错误地将 rhtml 模板解释为 HTML 页面，如果发生这种情况，只需浏览页面的源代码即可看到真正的源码。

Ruby 贴士

没错，你需要懂 Ruby 才能写 Rails 应用程序。不过我们明白，很多人在读这本书的时候其实是同时在学习 Ruby 和 Rails。本书的附录 A（原书第 481 页）对 Ruby 语言做了一个非常简单的介绍。当书中第一次用到某种 Ruby 特有的语言构造时，我们会为它做一个指向该附录相关内容的交叉引用。譬如说，这段内容如果用到了 :name 这个 Ruby 符号，在页边上就会有一个指示：该符号在原书第 483 页处有解释。另外，如果你不懂 Ruby，或者想要

:name
➡page 483

[2] 下载地址：http://www.pragmaticprogrammer.com/titles/rails/code.html.

快速刷新一下自己的记忆，你可以首先翻到原书第 481 页，阅读附录 A。书中有很多 Ruby 代码，所以要是对 Ruby 一窍不通的话……

David 说……

你会不时看到“*David* 说……”这样的边框，其中的内容是 David Heinemeier Hansson 想要与你分享的、关于 Rails 的独特见解——原理、技巧、推荐，凡此种种。David 是 Rails 的创始人之一，所以如果你想成为 Rails 专家的话，这些内容是不容错过的。

Joe 问……

Joe 是一个虚构的开发者形象，他常常会针对我们在书中讲解的内容提些问题，而我们则会试着回答这些问题。

这不是一本 Rails 参考手册，我们将展示大部分模块和方法，可能是通过示例，也可能是通过文字介绍，但我们不会列出上百页的 API 列表。这么做的原因是，只要你装上 Rails，就已经得到了完整的 API 文档，而且肯定比本书的内容更新。如果你通过 RubyGems 安装了 Rails（这也是我们推荐的安装方式），只要启动 Gem 文档服务器（使用 `gem_server` 命令），再用浏览器访问 http://localhost:8808，你就可以访问所有的 Rails API 文档。

Rails Versions
Rails 的版本

本书所介绍的是 Rails 1.0 版，其发布时间是 2005 年中期。不过，直到 2005 年 6 月本书第一次印刷之时，Rails 1.0 版还没有正式发布，因此我们尚无法在 Rails 1.0 环境下运行所有的示例代码。为了尽量与时俱进，本书中介绍的 API 都是 Rails 1.0 版本的，但代码只在 Rails 0.13 版本下测试通过——那也是 Rails 1.0 之前最后的一个版本。

1.3 Acknowledgments
致谢

本书实在算不得短小精干，如果没有来自 Ruby 和 Rails 社群不胜枚举的大力支持，恐怕我们也很难将它完成。要列出每个人的名字着实不易，所以如果您曾经给过我们帮助、而下列名单中又少了您的大名，请原谅我们的疏忽失察。

本书很幸运地拥有一群出色的审阅者——他们竟然为这本书制造出超过 *6M* 的评注。所以，我们要衷心感谢：

试读结束，需要全本PDF请购买 www.ertongbook.com