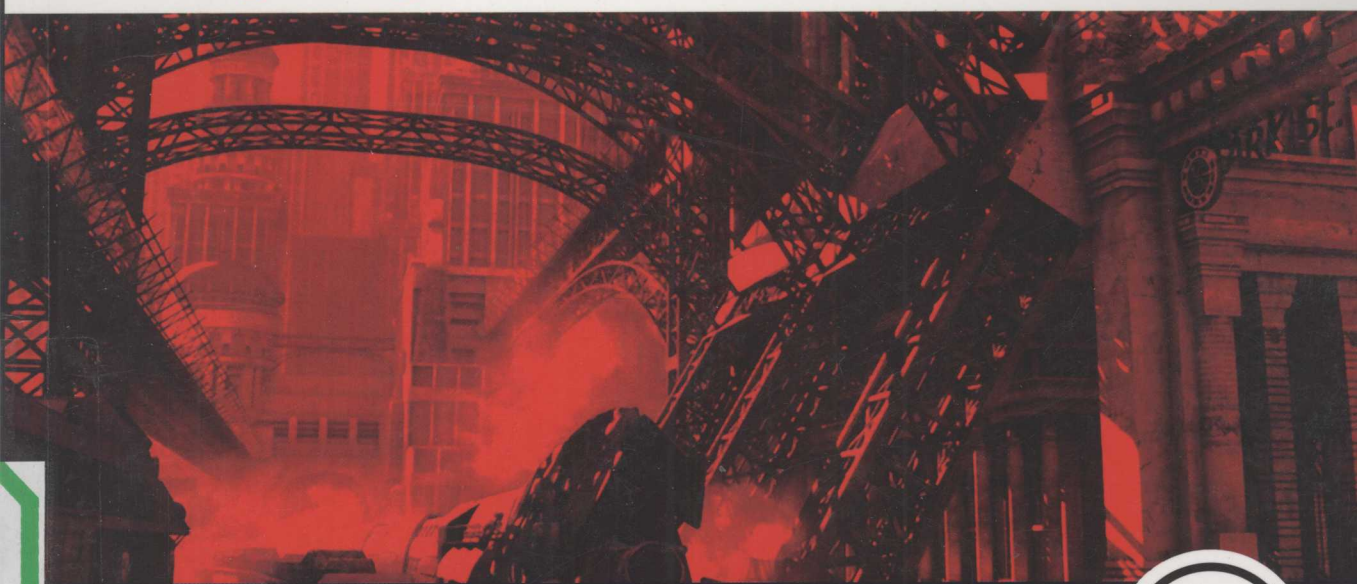
HZ BOOKS

Apress®

初学者学习Rails on Ruby的最佳指南

Rails 开发者指南

Beginning Rails: From Novice to Professional



（加） Jeffrey Allan Hardy 等著

刘 申 等译

Ruby和Rails技术系列

机械工业出版社
China Machine Press

Rails 开发者指南

Beginning Rails: From Novice to Professional

（加） Jeffrey Allan Hardy 等著
刘 申 等译

机械工业出版社
China Machine Press

Rails 是一种基于 Ruby 编程语言的 Web 应用框架，是 Ruby 语言的杀手级应用。本书深入浅出地讲解了 Rails 的相关知识，是一本非常实用的入门书籍。通过阅读本书你不但可以独立地开发新的 Web 应用，还可以真正地了解基于 Rails 的开发理念，并灵活地运用到开发过程中。本书共分为 11 章，从介绍 Rails 的基本框架入手，先后介绍了 Rails 中最重要的 3 个函数库：Active Record 、Action View 和 Action Controller，教你如何运用并掌握它们，从而创建出基于 Rails 的 Web 应用。

本书内容全面，实例丰富，可作为初学者的学习指南，也可作为网络应用开发人员的参考用书。

本书版权登记号：图字：01-2008-1861

图书在版编目（CIP）数据

Rails 开发者指南/（加）海德（Hardy，J. A. ）等著；刘申等译 . —北京：机械工业出版社，2009. 1

（Ruby 和 Rails 技术系列）

书名原文：Beginning Rails：From Novice to Professional

ISBN 978-7-111-25230-6

Ⅰ. R…　Ⅱ. ①海…　②刘…　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 154565 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：陈佳媛

北京瑞德印刷有限公司印刷 · 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 · 16. 25 印张

标准书号：ISBN 978-7-111-25230-6

定价：42. 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

译者序

Rails 近两年的发展势头很迅猛，受到越来越多企业以及开发者的关注，介绍它的书籍也越来越多。但是你可能会发现，在众多的技术类书籍中，很少有专门针对初学者入门的。大多数书籍都是以某一个实际开发为例，让你一步一步跟着它做，在这个过程中，为你讲解操作上的细节。这便引出了一个严重的问题：初学者跟着这样的书做完了以后，发现自己根本无法独立开发一个新的项目，在他的脑子里对 Rails 并没有一个完整的概念。

如果你想走进 Rails 的世界，并也遇到过上述的问题，那么本书就是你最好的选择了！

本书的目的不仅仅是带领你走进 Rails 的世界——可以独立开发出新的 Web 应用，而且让你真正了解 Rails Way（Rails 之道）。Rails 不仅仅是一个 Web 开发框架，它代表的是一种开发理念。只有真正理解这种思想，才能自如地运用它。对于开发者来说，如果不会语法，那很简单，我们可以去查 API，但是如果不了解 Rails Way，那说明你还是个 Rails 的门外汉。但是，如何把 Rails Way 传授给初学者呢？这便是本书的高明之处了。我想对于一个初学者来说，最难理解的地方可能就是 MVC 框架内部的逻辑关系了。而 MVC 框架的三个字母分别对应了 Rails 中三个最重要的函数库：Active Record、Action View 和 Action Controller。这三部分内容放在了本书中间部分去讲解。在它们之前，会先为你讲一些基础知识作为铺垫；在它们之后，会把这些知识进行巩固。

本书的翻译工作得到了很多人的帮助，在此对他们一并报以诚挚的谢意。其中，要特别感谢陈冀康编辑，谢谢他的辛勤工作与鼓励；感谢 InfoQ 中文站的两位同事霍泰稳、赖翥翔，感谢他们的理解与支持；还要感谢 FreeWheel 的 Bin Dong，是他把我带入了 Rails 世界；感谢 Ethos 的李剑、程序员杂志社的郑柯，是他们让我懂得了什么才是真正的“翻译”。

本书由刘申整体统筹、翻译。参与翻译的还有：宋薇、朱瑜敏、鲁奇、刘晓东。由于本书翻译的比较仓促，难免会有某些疏漏，欢迎读者朋友批评指正。

刘　申

2008 年 6 月于哈尔滨工业大学

致 谢

你在每本书的开头都会看到这部分内容，现在我们才真切地感受到它是多么必要：撰写一本书是一项任务量非常大的工作。虽然他们的名字不会出现在封面上，但是如果没有那些人的帮助、支持与鼓励，本书将永远不可能完成。

我们想感谢 Apress 的所有工作人员，是他们促成了此书的出版。感谢 Elizabeth Seymour 和 Sofia Marchant，是他们不厌其烦地督促我们保持进度，尽管我们自身也在尽力保持写作状态；感谢我们的编辑 Chris Mills，是他给予我们莫大的鼓励与支持；还有我们的文本编辑 Marilyn Smith，感谢她提出的众多建议与忠告；我们的产品编辑 Ellie Fountain，感谢她耐心地处理我们最后一分钟的修改。

所有基于开源项目的书籍都忽视了那个孕育它成长的社区。感谢 David Heinemeier Hansson、Rails 核心团队以及世界各地成百上千的贡献者们，在开发、支持并不断改进如此美妙的框架。

我们还想对家人和朋友表示衷心的谢意，你们以极大的耐心容忍了我们写书期间那一个个漫长的夜晚与神秘失踪。我们真心地期望能有更多的时间陪伴在你们左右。

Jeff 感谢 Unspace 允许他把时间投入在本书上，没有你的支持和鼓励，本书将不可能问世。同样要感谢 Hampton Catlin，他对书中的代码、文本，偶尔还有作者进行了仔细的检查。Jeff 要特别感谢他的妻子 Amy：她完全容忍了写书的这段时光，你的爱与支持永远是那么鼓舞人心，很幸运能遇见你。

Cloves 想对两位伟大的开发者、资深顾问以及好朋友表示感谢，他们教会了他如何成为一名出色的程序员，谢谢 Mircea Oancea 和 Alex Katsnelson。还要感谢 Rida Al Barazi，在写本书期间，他确保了项目的进度，避免了 SpinBit 的项目危机。Cloves 要感谢他的朋友，以及本书的另一位作者 Jeffrey Hardy，他出色的工作是由无数辛勤的夜晚所换来的。还要对 Hampton Catlin 表示感谢，是他确保了本书的高质量标准。最后，Cloves 要感谢他的妻子 Jane：在漫长的旅行中是她给予了 Cloves 支持和鼓励；是你所有的爱与友谊才使本书得以成型。

引 言

回想一下，很难记起在 Rails 问世之前 Web 开发是一种什么状况。我仍然对第一次使用 Basecamp（由 37signals 开发的应用程序，Rails 就是从中提取出来的）记忆犹新，当时对它是由什么创建的特别好奇。它明显不是用 PHP 做的，而那时候，PHP 是我开发 Web 应用的首选工具。Rails 要诞生的消息随之也在网络上风生水起，不过那时的 PHP 成熟度已经很高了，而且拥有很强大的面向对象特性。当 Rails 从 Basecamp 提炼出来后，PHP 类似的框架也突然出现。不过，当我听说构造 Basecamp 的速度如此之快时，便不由自主地被它背后所使用的技术所吸引。

当我第一次安装 Rails 的时候，我如饥似渴地跟随 wiki 上一篇又长又晦涩的教程做了一遍，把这个框架运行了起来。不夸张地说，给我留下了极深的印象。在我用完类似的 PHP 框架后，我被 Rails 代码中的分离程度和直观的显示方式所震惊。我以前从来没有见过 Ruby 代码，但却立刻感觉到，这便是我所寻找的编程语言。当我得知 37signals 要举办一个名为“The Building of Basecamp（Basecamp 的构建）”的讲座后，便立刻报了名。在从芝加哥返家的飞机上，我写出了第一个 Rails 应用程序，从此便一发不可收拾。

当 Rails 问世后，由于其典型的破坏性技术，它遭到了很多诋毁者的攻击。前 PHP、Java 开发者 David Heinemeier Hansson 对损害他名誉的问题都做了明确的解释，他要用 Rails 改变这一切。结果是，他思考问题的方式确实震动了业界。以前从来没有任何技术如此强势地占领其地盘，一场近乎神圣的战役，在其他成熟框架的拥趸与这个襁褓中的婴儿间展开了。似乎每一天 Rails 都会赢得一些支持者，最后终于得到了多数思想领导者的支持，天秤开始偏转了。甚至当初的诋毁者最后都认输了，战胜他们的可能是 Ruby、社区、热情、或者是被遗忘的编写优雅代码的快感。

发展到现在这个阶段，除非你生活在岩石下面，否则你一定听说过 Rails 框架。在 Web 开发这个领域，它相当于一个皇室名字。在我写此书的时候，Rails 已经替代，或者至少是影响到当今每一个主流的 Web 应用框架。几乎每一种语言（从 Java、.NET 到 PHP、Python）都拥有一个与 Rails 理念类似的框架。Rails 彻底改变了 Web 开发领域的现状。

我为 Rails 感到骄傲。见证它从早期的特例发展到如今作为现代、成熟框架的主流，我认为《the little engine that could[⊖]》这个故事是最合适的比喻了。从很多方面来说，Rails 便是“the little framework that could”。它突出重围，从质疑中赢得赞誉。我认为在某些方面证明了早期支持者所说的一句话：Rails 将会颠覆 Web 开发领域。

Jeffrey Allan Hardy

⊖ 这是一个英文寓言故事，讲述的是小火车头拉货物过大山的故事，小火车头一边费劲地拉货上山，一边说“I think I can. I think I can. I think I can...”体现了一种永不放弃的精神。——译者注

前 言

至今已经有很多 Rails 入门类书籍了。我们也读了很多，但是却发现大多没有起到很大的帮助。我们认为缺少这么一类书：完全针对初学者，而且还会传授足够的 Rails 文化和理念，并使其成为非常实用的一部分。Rails 是一个很大的主题，很多书籍想传达的内容太多，以至于使它们不能有效地传达出 Rails 的核心功能。本书则会完全不同。

本书特别适合那些有一点或者甚至没有任何 Web 应用开发经验的人，或者有一些经验，但是对 Rails 却是个新手。我们假设你对 Web 标记语言（比如 HTML）已经有所了解，并且你可以自如地安装软件。但是你无需知道如何去编程，如何使用 Web 服务器，如何维护网页状态，或者如何创建和连接数据库。本书将会向你传授 Web 应用运作以及 Rails 快速构建的基础知识。

每个人都是从初学者起步的，我们也不例外。当开始写这本书的时候，我们考虑的是在创建第一个 Rails 应用时认为最实用的内容。我们希望了解什么呢？什么可以令事情变得简单呢？我们开始创作这本书力求把复杂的 Web 开发以深入浅出的方式讲出来，让初学者能够开个好头。

如果你正在寻找一本介绍 Rails 老手高级技巧的书，那么本书可能会令你失望。本书没有探究那些 Ruby 或者 Rails 的复杂细节，没有自关联多态（self-referential polymorphic）连接或者高级缓存技术（advanced caching techniques）等 Rails 主题。相反，我们只关注框架中能让你快速上手的部分。我们让你尽快适应 Rails 环境，而不是把你埋葬在无数的细节中，这也是为什么我们把本书的每一章都设计成围绕框架中的一个具体组件，并且关注那部分最实用的功能。

如果你以前没有编程经验，应当首先阅读附录 A 中的 Ruby 简介。如果你没有接触过数据驱动的应用程序，还应当阅读附录 B 对关系数据库的介绍。当你准备好了，第 1 章将会向你介绍“Rails 之道”（the Rails way），第 2 章将会教你在自己的机器上安装 Ruby 和 Rails。本书的其余部分将会介绍创建实际应用程序的过程，逐个向你介绍 Rails 框架的相关组件。

本书的大部分篇幅都是介绍你最常使用的功能，而在你不常用的功能上，我们所用的篇幅也较少。在 Rails 中你所做的大多数事情都与模型（model）有关，所以你必须要了解 Active Record，它是 Rails 用来与数据库进行通信的函数库。Active Record 很自然地成为 Rails 框架中最庞大的组件，所以你在它身上投入很多时间、我们在它身上铺了很多笔墨（第 4、5 章）也就都在情理之中了。如果你了解如何对某个领域创建模型并知道如何与数据库对象进行有效的操作，那么就为余下的创建工作打下了一个很好的基础。在我们深入研究 Action Pack（第 6 章，它是 Rails 的 Web 组件）的时候，已经创建了完整的模型，并把它用作试运行。在第 6 章，你会学到如何创建控制器与视图，以及如何用视图把你的模型通过 Web 展示出来。第 7 章会阐述如何通过 Ajax 和其他的技术改进用户界面。第 8 章会向你展示应用程序如何通过 Action Mailer 收发邮件。剩下的章节将会向你传授一些非常重要，而且你也必须要了解的知识：对应用程序进行测试；插件的安装、使用与创建；最后

是应用程序的部署以及发布。

Rails 的发展很快。在本书写作期间，它的一些功能又被添加、改进、引起争议或移除掉了，我们力图使本书保持最新并且相关。本书介绍的是 Rails 1.2.3（此版本为当时的稳定版），我们还在未来版本中可能出现的功能上引入了“注意”与“提示”。如果你正在使用更新版本的 Rails，你可能会发现有一些东西已经改变了，但是对于大部分内容而言，你完全不用担心。

固然，Rails 是一个很庞大的框架，它所包含的内容远非一本书所能囊括的。然而，抛开它的大小以及功能强弱不谈，Rails 是一门从理论上非常容易掌握的技术，这都要依仗于它功能上的强大。通过此书，你将会学到所有用 Rails 创建 Web 应用所需的知识。

本书的网站

一定要浏览本书的网站：http://beginningrails.com。除了大多数本书所使用的最新源代码，你还会在上面看到勘误、注意、提示以及其他重要的更新。你还可以与作者进行沟通，询问技术难题并获得所需要的帮助。

目 录

译者序
致谢
引言
前言

第 1 章 Rails 框架简介 …… 1
1.1 Web 应用的兴起 …… 1
1.2 Web 并不是完美的 …… 2
1.3 优秀的 Web 框架 …… 2
1.4 走进 Rails …… 3
1.4.1 Rails 是 Ruby …… 4
1.4.2 Rails 鼓励敏捷 …… 5
1.4.3 Rails 是有主见的软件 …… 6
1.4.4 Rails 是开源的 …… 7
1.5 MVC 模式 …… 8
1.5.1 MVC 循环 …… 8
1.5.2 MVC 的层 …… 9
1.6 构成 Rails 的函数库 …… 11
1.7 Rails 不是尚方宝剑 …… 11
1.8 总结 …… 11
第 2 章 入门 …… 12
2.1 Rails 安装的总体介绍 …… 12
2.2 在 Mac OS X 10.4 Tiger 上安装 …… 13
2.2.1 安装 Apple Developer Tools (Xcode) …… 14
2.2.2 安装 MySQL …… 14
2.2.3 添加 MySQL 到你的 PATH …… 16
2.2.4 安装 Ruby …… 17
2.2.5 安装 RubyGems …… 19
2.2.6 安装 Rails …… 19
2.2.7 自动安装 …… 19
2.3 在 Windows XP 上安装 …… 20
2.3.1 安装 MySQL …… 20
2.3.2 安装 Ruby …… 21
2.3.3 安装 Rails …… 22
2.4 在 Linux 上安装 …… 23
2.4.1 安装 MySQL …… 23
2.4.2 安装 Ruby …… 24
2.4.3 安装 RubyGems …… 24
2.4.4 安装 Rails …… 25
2.5 创建第一个 Rails 应用程序 …… 25
2.5.1 启动内建 Web 服务器 …… 26
2.5.2 生成控制器 …… 27
2.5.3 创建一个 Action …… 28
2.5.4 创建一个模板 …… 28
2.6 总结 …… 30
第 3 章 运行起来再说 …… 31
3.1 项目的总体介绍 …… 31
3.2 创建 events 应用程序 …… 31
3.2.1 创建项目数据库 …… 33
3.2.2 创建 Event 模型 …… 34
3.2.3 创建一个数据库表 …… 35
3.2.4 生成一个控制器 …… 36
3.2.5 用脚手架运行 …… 37
3.2.6 添加更多的域 …… 38
3.2.7 添加校验 …… 39
3.2.8 生成脚手架 …… 41
3.3 总结 …… 43
第 4 章 对数据库进行操作：Active Record …… 44
4.1 Active Record 简介：Rails 的对象 – 关系映射 …… 44

4.1.1 关于 SQL …… 45
4.1.2 Active Record 惯例 …… 46
4.2 Console 简介 …… 46
4.3 Active Record 基础：CRUD …… 49
4.3.1 创建新记录 …… 49
4.3.2 读取（查询）记录 …… 52
4.3.3 更新记录 …… 56
4.3.4 删除记录 …… 57
4.4 当好模型变坏时 …… 59
4.5 总结 …… 60
第 5 章 Active Record 进阶：强化模型 …… 61
5.1 添加方法 …… 61
5.2 使用关联 …… 63
5.2.1 声明关联 …… 64
5.2.2 创建一对一的关联 …… 65
5.2.3 创建一对多关联 …… 68
5.2.4 应用关联选项 …… 71
5.2.5 创建多对多关联 …… 72
5.2.6 创建富多对多关联 …… 76
5.3 为高级查询构建条件 …… 78
5.3.1 使用一个条件 Hash …… 78
5.3.2 使用一个 SQL 片段 …… 79
5.3.3 使用一个数组 …… 80
5.3.4 使用关联代理 …… 81
5.4 应用校验 …… 82
5.4.1 创建自定义校验方法 …… 82
5.4.2 使用内建校验 …… 83
5.5 生成回调 …… 86
5.6 回顾更新的模型 …… 87
5.7 总结 …… 91
第 6 章 Action Pack：对视图和控制器的操作 …… 92
6.1 Action Pack 组件 …… 92
6.1.1 Action Controller …… 92
6.1.2 Action View …… 94
6.1.3 嵌入式 Ruby …… 95
6.1.4 Helpers …… 96
6.1.5 路由 …… 96
6.1.6 Action Pack 请求周期 …… 97
6.2 控制器的走查 …… 98
6.2.1 设置路由 …… 98
6.2.2 生成控制器 …… 101
6.2.3 创建一个模板 …… 103
6.2.4 对布局的操作 …… 104
6.2.5 创建一个注册表单 …… 105
6.2.6 使用表单 helpers …… 106
6.2.7 处理请求参数 …… 109
6.2.8 编写 create action …… 109
6.2.9 渲染响应 …… 110
6.2.10 重定向 …… 110
6.2.11 在模板中显示错误信息 …… 111
6.2.12 添加 edit（编辑）表单 …… 112
6.2.13 应用局部模板保持 DRY …… 113
6.2.14 添加 login（登录）和 logout（退出）action …… 116
6.3 改进 events 控制器和模板 …… 120
6.3.1 清理 events 控制器 …… 121
6.3.2 在 events 模板中使用局部模板 …… 122
6.3.3 为 events 表单添加种类 …… 122
6.3.4 在 events 控制器中处理种类 …… 124
6.4 使用控制器过滤器 …… 125
6.4.1 要求使用过滤器的验证 …… 126
6.4.2 应用过滤器到控制器 …… 127
6.5 收尾工作 …… 129
6.5.1 使用 Action View helpers …… 129
6.5.2 在模板中避开 HTML …… 131
6.5.3 为表述区域格式化 …… 131
6.5.4 添加编辑控制 …… 132
6.5.5 添加自定义 helpers …… 133
6.5.6 为它赋予一些样式 …… 134
6.6 总结 …… 142
第 7 章 用 Ajax 改进交互 …… 144
7.1 Ajax 和 Rails …… 144
7.1.1 Prototype and Helpers …… 144

7.1.2 JavaScript 库 …… 145
7.2 使用 Ajax helpers …… 145
7.2.1 使用 link_to_remote 发起远程调用 …… 146
7.2.2 用 form_remote_tag 远程发送表单 … 151
7.2.3 用 observe_field 和 observe_form 来记录改变 …… 153
7.3 使用 script. aculo. us helpers …… 156
7.3.1 添加自动补全功能 …… 156
7.3.2 添加一个现场编辑器 …… 157
7.3.3 添加可视化效果 …… 159
7.4 使用 RJS 模板 …… 161
7.4.1 在模板中运行 RJS …… 162
7.4.2 在控制器中运行 RJS …… 164
7.5 总结 …… 165
第 8 章 发送和接收邮件 …… 166
8.1 设置 Action Mailer …… 166
8.1.1 配置邮件服务器的设置 …… 166
8.1.2 配置应用程序设置 …… 167
8.2 发送电子邮件 …… 168
8.2.1 对基本的电子邮件进行操作 …… 170
8.2.2 发送 HTML 电子邮件 …… 172
8.2.3 为消息添加多种成分 …… 174
8.2.4 添加附件 …… 175
8.2.5 将收到的电子邮件发送至一个 Rails 进程 …… 175
8.3 总结 …… 177
第 9 章 对你的应用程序进行测试 …… 178
9.1 Rails 处理测试的方式 …… 178
9.2 Rails 应用程序的单元测试 …… 180
9.2.1 测试 Event 模型 …… 181
9.2.2 对校验进行测试 …… 185
9.3 对你的控制器进行功能测试 …… 186
9.3.1 对 Event 控制器进行测试 …… 186
9.3.2 创建一个测试 helper …… 188
9.3.3 运行整套的功能测试 …… 192
9.4 集成测试 …… 194
9.4.1 events 应用程序的集成测试 …… 194
9.4.2 基于故事的测试 …… 197
9.5 运行整套测试 …… 200
9.6 用 rcov 测量测试的覆盖情况 …… 201
9.7 总结 …… 202
第 10 章 用插件来扩展 Rails …… 203
10.1 查找和安装插件 …… 203
10.1.1 查找插件 …… 204
10.1.2 安装插件 …… 204
10.2 在我们的应用程序中使用一个插件 … 205
10.2.1 修改数据库 …… 205
10.2.2 修改应用程序以便使用插件 …… 206
10.3 创建自己的插件 …… 208
10.3.1 使插件可用于应用程序 …… 209
10.3.2 创建插件模块 …… 210
10.3.3 更新控制器和视图 …… 216
10.4 总结 …… 218
第 11 章 部署 Rails 应用程序 …… 219
11.1 用 Capistrano 部署 …… 219
11.1.1 Capistrano 的安装 …… 220
11.1.2 Capistrano 处方 …… 221
11.1.3 部署服务器上的 Capistrano …… 223
11.1.4 自定义 Capistrano 任务 …… 223
11.2 设置服务器架构 …… 224
11.2.1 单一架构 …… 224
11.2.2 代理集群服务器：推广中的新产品 …… 225
11.2.3 立刻变成一位部署高手 …… 226
11.3 总结 …… 227
附录 A Ruby——程序员的知音 …… 228
附录 B 数据库入门 …… 240
附录 C Rails 社区 …… 247

第1章 Rails 框架简介

Rails 是一种基于 Ruby 编程语言的 Web 应用框架。Rails 具有良好的设计和实用性。它会帮助你迅速创建功能强大的网站，而且代码简洁并拥有良好的可维护性。

本书的目标是让你对用 Rails 创建动态 Web 应用拥有一个全面而完整的理解。这意味着，本书不仅仅向你展示了如何使用那些具体的功能以及框架的便捷性，并且向你传授了很多实用的 Ruby 编程知识。不能只把 Rails 看作另一种工具，它还代表了一种思想。为了全面地掌握 Rails，你还必须要了解它的基础架构、文化、美学以及 Web 开发哲学。

如果你还没有听说过它，你一定注意到了近期十分流行的一个词组“the Rails way”（Rails 之道）。它模仿了另一句近些年风靡于 Ruby 社区中的名言“the Ruby way”（Ruby 之道）。“Rails 之道”通常是指最简单的方式（如果你愿意，那将是最近的一条路）。但这并不是说，你不能按照自己的方式去做事情，也不意味着这个框架拥有很多束缚。它只表示如果你脱离了这条康庄大道，就不要指望 Rails 为你带来任何的便捷。如果你曾经闯荡于 UNIX 社区，或许会认为这个理念和一句 UNIX 俗语很相似：“Do the simplest thing that could possibly work（做可能管用的最简单的事）”你会了解这一点的。本章的目的就是把你引入 Rails 之道。

1.1 Web 应用的兴起

Web 应用的重要性逐年递增。正如我们的世界已经联系得越来越紧密，大家所做的越来越多的事情也都是在网上完成的。我们在网上查收电子邮件，处理银行业务。我们通过浏览器自由地在网上听课、分享图片、上传视频、管理项目并与世界各地的人联系在了一起。随着连接速度的提升，带宽的增加，基于 Web 的软件和类似基于网络客户端/服务器的应用将逐渐替代通过传统（阅读、过时的）方式发行的软件。

作为消费者，基于 Web 的软件为我们带来了很大的便捷，令我们可以在更多的地方做更多的事情。基于 Web 的软件可在任何一个支持浏览器（也就是说，所有）的平台上运行，并且什么都不用安装和下载。如果把 Google 的股价当作风向标的话，可以看出 Web 应用确实非常受欢迎。事实是，近几年 Web 上的变化可谓是激动人心，如今的代表便是 Web 2.0 的兴起。全世界的人们都开始重新认识新一代的 Web，开始体会基于 Web 应用的美丽之处。从电子邮件、日历、图片、视频、书签、电子银行、网上支付，我们的生活已完全无法脱离浏览器。

由于分发的便捷，Web 软件市场的变化日新月异。不像传统软件，必须得安装在个体的计算机上，Web 应用的改变可迅速完成，功能也可逐渐完善。没必要再花费几个月甚至数年才发布最终的版本或在投入运行前完成所有的功能。不必在调研和开发上耗费数月，你可以早早把产品上线，然后再慢慢雕琢，甚至功能上也不用那么完善。

你能想像已经发行并运送了成千上万张 CD，却在快递卡车消失于夕阳之下的时候，在软件中

发现了一个 bug？那将是一个非常昂贵的错误！这种软件的分发方式将耗费大量时间才能令你的软件到达用户的手中，因为在公司发布产品之前，必须要确保软件是万无一失的（bug-free）。当然，我们都知道，不可能存在这种万无一失的软件。因此，Web 应用也不可能做到万无一失。不过，对于 Web 应用来说，更正 bug 是很容易去部署的。

当把补丁放到寄存 Web 应用的服务器上，所有用户会同时从更新中受益，通常是没有任何服务中断的。这种级别的安全保证，你是无法从店铺购买的软件中获得的。不用不停地下载服务包，不用安装任何关键的更新。打一次补丁，只要在浏览器的一次刷新中便可完成。还有一个额外的好处，省去了大量用于软件的包装和分发的资源与金钱，开发者可以把更多的时间放在软件质量的提高和创新上。

基于 Web 的软件有以下几点优势：

- 便于分发
- 便于部署
- 便于维护
- 平台无关（platform-independent）
- 可在任何地方读取

1.2 Web 并不是完美的

虽然 Web 是一个很好的平台，但是它也拥有很多局限。其中一个最大的问题便是浏览器本身。当提到浏览器，就会想到其中有几个竞争者，它们对显示网页上的内容都有不同的处理方式。虽然正在逐步统一，兼容标准的状况也在日益改善，但是仍然存在一些遗留问题。甚至到现在，也几乎不可能实现 100% 的跨浏览器兼容。某些可在 IE 下运行的功能，在 Firefox 中却不一定可用，反之亦然。这种缺乏一致性的问题让开发者很难创建真正的跨平台应用，也让用户很难在他们自己的浏览器中运行。

除了浏览器的问题，或许对于 Web 开发来说，最大的问题是其自身的复杂性。典型的 Web 应用拥有许多不确定的部分：协议、端口、HTML、CSS、数据库、服务器、设计师、开发者和众多其他的参与人员，所有这些因素组合在一起，就变得异常复杂。

但是尽管存在这么多的问题，Web 上的新焦点（例如平台）已经体现出 Web 开发领域正在快速发展，并迅速克服这些障碍。它会不断的成熟起来，传统客户端软件的开发工具和过程正向 Web 开发领域转变。

1.3 优秀的 Web 框架

框架集成了众多带你走向 Web 开发之路的工具，它是函数库和工具的集合，目的是简化开发。它设计的核心思想便是高效，一个优秀的框架会为你提供基础而完整的架构，应用程序便是在此基础上创建起来的。

拥有好的框架意味着你的应用程序已经完成了一大块。不必再从头开始，你从已有的基础上继续开发就可以了。如果有一个开发者社区使用同样的框架，那当你需要帮助的时候，便会拥有一个

社区的支持。你同样拥有很强的保证，因为你所使用的“地基”很少有恼人的 bug 和不稳定性的问题，而那些问题会大大减缓你的开发进程。

一个好的 Web 框架可用以下几点来描述：

- 完整堆叠（Full stack）：你所需要用于创建完整应用程序的所有内容都应当包含在这个盒子中。要想安装不同的函数库或者配置多种组件，用一个拖拽便可搞定。不同逻辑层应当无缝接合。
- 开源：框架应当是开源的，最好是位于一个自由并可免费使用的协议下，比如 BSD 或者 MIT。
- 跨平台：一个好的框架应当是平台无关（platform-independent）的。你所工作的平台，属于你个人的选择。你的框架也应当尽可能保持中立。

一个好的 Web 框架会为你提供以下几项内容：

- 容纳万物的地方：结构和规则造就出好框架。换言之，如果一个框架没有良好的结构和一套实用的规则，它便不是个好框架。这个观点是说每件事物在系统里都应有合适的位置。这样便消除了猜测，提高了效率。
- 数据库抽象层：你不需要处理数据库通路的低层次细节，也不必限制于应用某个特定的数据库。一个好框架会为你完成大部分的数据库辅助工作，而且它几乎可以与任何一个数据库协作。
- 为编程出谋划策的文化和美学：与其将框架施加的结构视为限制，不如视为解放。一个好框架会逐渐引导开发者编写代码。框架常常依靠惯例为你作出艰难的决定，而且框架的文化会帮助你少做一些不重要的决定，让你把更多精力放在重要的事情上。

1.4 走进 Rails

Rails 是创建 Web 应用的极品框架。它是完整、开源、跨平台的。它提供了一个强大的数据库抽象层（称为 Active Record），可作用于所有流行的数据库系统。它自带有一套数目可观的默认值，并提供一个被证明很好的、多层次的系统来组织程序文件及其相关内容。

最重要的，Rails 是一套有自己思想的软件。它在 Web 开发艺术上拥有一套自己哲学，并且十分重视它。然而幸运的是，这套哲学是美妙和高效的化身。当你学习 Rails 的时候，会发现它确实使编写 Web 应用程序变得愉快了。

最初 David Heinemeier Hansson 创建 Rails 时，它被塑造成一种 wiki - wiki 应用程序的形式，称为 Instiki。Rails 框架的最初版本实际上取自一个现实世界中运行的应用程序：37signals 公司的 Basecamp。Rails 的创造者去掉了所有的 Basecamp 专有的部分，留下的就叫做 Rails。

因为它取自一个真实的应用程序，而不是建立在脱离实际的象牙塔里，所以 Rails 很实用而且没有多余的功能。它作为框架的目标是解决发生在 Web 开发中 80% 的问题，假定剩下的 20% 是应用程序领域内真正特有的问题的话。这可能会令你惊讶，一个应用程序中多达 80% 的代码是基础架构，但这其实并不像听起来那么牵强。设想一下所有包含在应用程序架构里的工作，从目录结构和命名规则到数据库抽象层和状态的维护。

你将看到 Rails 对于目录结构、文件命名、数据结构、方法参数以及几乎所有问题都有详细明

确的见解。当你编写 Rails 应用程序时，要遵循已经为你列好的规则。这样你便可以专注于那 20% 的真正问题，而不用将精力集中在使应用程序结合在一起的小细节上。

1.4.1 Rails 是 Ruby

现在有很多编程语言。你大概也听到过不少，像 C、C#、Lisp、Java、Smalltalk、PHP 和 Python 都是热门选择。还有一些你可能从没听过：Haskel、IO，或许还有 Ruby。和其他语言一样，Ruby 也是一种编程语言。你可以用它来编写计算机程序，包括（当然并不局限于）Web 应用程序。

在 Rails 到来前，并没有太多人用 Ruby 编写 Web 应用程序。而其他语言，如 PHP 和 ASP 占据了这一领域，大部分 Web 应用都是用它们编写的。Rails 使用 Ruby 这一事实具有重大意义，因为 Ruby 作为编程语言的能力要比 PHP 或 ASP 都强大得多。这很大程度上是 Web 成熟的另一个标志。现在它吸引了一大批的旁观者，更加强大的语言和工具也加入到这个圈子中来。

Ruby 是 Rails 成功的关键。Rails 实际上使用 Ruby 来创建所谓的领域特定语言（domain-specific language，DSL）。这里的领域便是 Web 开发，当你用 Rails 工作时，就好像在用一种专门为架构 Web 应用程序所设计的语言——一种拥有自己的一套规则和语法的语言。Rails 在这一点上做得非常好，以至于有时你很容易就忘记了实际上是在编写 Ruby 代码。这是对 Ruby 功能的证明，而且 Rails 充分利用了 Ruby 的表现力来创建真实美妙的环境。

对于很多开发者，是 Rails 把他们带入 Ruby 之门的，在 Rails 以前，Ruby 这种语言的追随者公认为很少，至少在西方是这样。当 Ruby 逐渐引起日本以外程序员的关注时，Rails 框架将 Ruby 带入了主流。

1994 年 Yukihiro Matsumoto 发明了 Ruby，令人惊奇的是 Ruby 一直处于无人知晓的境地。随着编程语言的发展，Ruby 也逐渐成为最美的语言之一。解释型和面向对象、灵活和表达性强，使用 Ruby 确实是件乐事。Rails 的优雅大部分归功于 Ruby 和渗透在 Ruby 社区中的文化与美学。当你开始使用框架时，你会很快认识到 Ruby，如同 Rails 一样，富有习语和惯例，所有的习语和惯例营造出一个愉快、高效的编程环境。

概括地说，Ruby 可用以下几点来描述：

- 一种解释型、面向对象的脚本语言
- 优雅、简明的语法
- 强大的元程序特征
- 非常适合作为创建 DSL 的宿主语言

本书的附录 A 含有一套完整的 Ruby 入门教程。如果你现在想知道 Ruby 是什么样子，请向后翻到附录 A 看一看。如果开始时 Ruby 看起来有些不合惯例，请不要担心。即使你不是一位程序员，也会发现它很易读。现在你便可以跟随本书进行学习，在你需要解释的时候，参考附录即可。如果你需要更深入地指导，Peter Cooper 撰写了一本很棒的书，名为《*Beginning Ruby: From Novice to Professional*》（Apress，2007）。你也会发现在研究这门语言时，Ruby 社区会为你带来非常大的帮助。请务必访问 http://ruby-lang.org，以获取 Ruby 相关资源。

1.4.2 Rails 鼓励敏捷

Web 应用程序通常并不以敏捷著称，它们拥有应用困难和维护起来像噩梦的名声。也许是为了改变这一结论，Rails 应运而生，协助引领 Web 开发中的敏捷编程运动。开发软件时，Rails 提倡并协助实现以下几项基本原则：

- 个人与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

敏捷宣言[⊖]是由 17 位著名人士（包括 Dave Thomas，Andy Hunt 和 Martin Fowler）讨论得出的一个软件开发领域的“轻量级方法”（lightweight methodologies）。现在，敏捷宣言已被广泛当作敏捷开发的规范定义。

Rails 在设计时就遵循了敏捷之道，它几乎包含每一条敏捷原则。使用 Rails 你可以迅速并毫不费力地响应客户的需求，而且 Rails 在协作开发时工作良好。Rails 之所以能胜任这样的任务，是因为它遵循自己的一套原则，这些原则让敏捷开发成为可能。

Dave Thomas 和 Andy Hunt 在编程技术方面的精华之作《*The Pragmatic Programmer*》（Addison-Wesley，1999）读起来几乎像是在为 Rails 树立路标。Rails 遵循“不要重复你自己”（don’t repeat yourself，简写成 DRY）原则——这是一种快速建立原形的理念，还有“你将不会需要它”（you ain’t gonna need it，简写成 YAGNI）哲学。在纯文本中保存重要数据，惯例重于配置（convention over configuration），在客户和程序员之间搭建桥梁，更重要的是在预期的变化中推迟决策，以上内容在 Rails 中都已经制度化了。这便是 Rails 成为敏捷开发最佳拍档的原因，而且 Dave Thomas 成为 Rails 早期的支持者之一也就不足为奇了。

下面的部分，我们将要体验几条 Rails 的流行语，这将证明 Rails 是多么适用于敏捷开发。虽然我们希望避免变得过于哲学化，但是其中的一些要点对于了解什么使 Rails 变得如此重要是必不可少的。

1. 较少的软件

Rails 哲学的核心原则之一是“较少的软件”（less software）这个概念。较少的软件意味着什么呢？它意味着惯例重于配置，少写代码，去掉那些没有必要却增加系统复杂性的东西。简言之，较少的软件意味着较少的代码、较低的复杂性和较少的错误。

2. 惯例重于配置

惯例重于配置（convention over configuration）意味着程序员需定义只有配置是非惯例的。

编程处处都在作决定。如果你想从零开始编写一个系统，没有 Rails 的帮助，你需要作出很多决定：如何组织你的文件，采用怎样的命名惯例，以及如何处理数据库的访问，这些问题只是其中的一小部分。如果你决定使用数据库抽象层，你需要坐下来编写它，或是至少找一段满足你需求的

⊖ 详细信息可参见网站 http://agilemanifesto.org。

开放源码方案。在你完成所有这些工作后，才能静下心来开始建立你的业务模型。

Rails 可让你立刻开始，它包含一套智能决策，告诉你的程序如何工作，减少你预先需要作出的低层次决策的数量。所以，你可以将精力集中在试图解决的问题上，以便更迅速地完成工作。

Rails 几乎没有配置文件。如果你习惯于其他的框架，这一点可能会令你很吃惊。如果你之前从未使用过框架，你也会惊讶。在一些案例里，配置一个框架要占据一半的工作量。

Rails 依靠常见的结构和命名惯例，而不是配置，所有的这些都遵循了经常被引用的“*最小惊讶原则*”（principle of least surprise，简写为 POLS）。事物以可预测、易解释的方式运行。很多智能的默认遍布于框架的每个角落，使你（开发者）不必再去告诉框架如何工作，这并不是说你不能指示 Rails 如何去工作。实际上，大多数行为是可以按照你的喜好定制的，以满足你特定的需求。但是你会从默认值那儿得到好处的最大化和工作效率的提升，而且 Rails 旨在鼓励你接受默认值并继续前行，解决更有趣的问题。

虽然你可以操控 Rails 设置和环境中的大部分事物，但是你接受的默认值越多，便可以更迅速的开发应用程序并预测它们如何工作。你只需开发而不用作任何明确的配置，这样的开发速度就是 Rails 良好运作的关键因素之一。如果你将文件放入恰当的位置，并按照正确的惯例为它们命名，它们就只管工作了。如果你愿意接受默认值，一般情况下会少写一些代码。

Rails 这样做的原因又回到了“较少的软件”这一概念上。较少的软件意味着少做低层次决策，这会使你作为 Web 开发者的生活变得简单很多。而且简单的确是件好事。

3. 不要重复你自己

Rails 的 DRY 原则是很重要的。DRY 的意思是*不要重复你自己*（don't repeat yourself），它说明系统中的信息应当只在一个地方被描述。

以数据库配置参数为例，当你连接到一个数据库，通常需要一些信息，如用户名、密码和你想要运行的数据库名。将这些连接信息放在每一次数据库查询中似乎是可接受的，而且如果你只建立一至两个连接，它也一定会运行的很好。但是一旦你需要建立更多的连接，你会被大量贯穿在代码中用于保存用户名和密码的实例变量给弄晕。如果你数据库的用户名和密码改变了，便要做很多查找和替换。将连接信息保存在一个单独的文件内，必要时参考它，将是个更好的办法。这样如果信息改变，你需要修改的只是一个文件。这就是 DRY 原则的内容。

系统的副本越多，错误也就有越多的空间可以隐藏。同一信息存在的地方越多，当需要改变时，需要修改的地方也就越多，追踪这些变化也就变得越难。

Rails 是按照尽可能保持 DRY 这一方式来组织的。通常你只需要将信息指定放在一个位置，便可去做其他更重要的事了。

1.4.3 Rails 是有主见的软件

框架将观点编译成代码。Rails 对于你的应用程序如何构建，具有很鲜明的见解——这一点你不要过于惊讶。当你使用 Rails 应用程序时，那些观点便会施加于你，不管你是否意识到了。Rails 发表自己观点的方式之一是通过轻轻地（有时是强烈地）将你推向正确的方向。在讨论惯例重于配置时，我们已经提到过这种鼓励人的形式。你会按照引领做正确的事，原因是如果做错误的事，