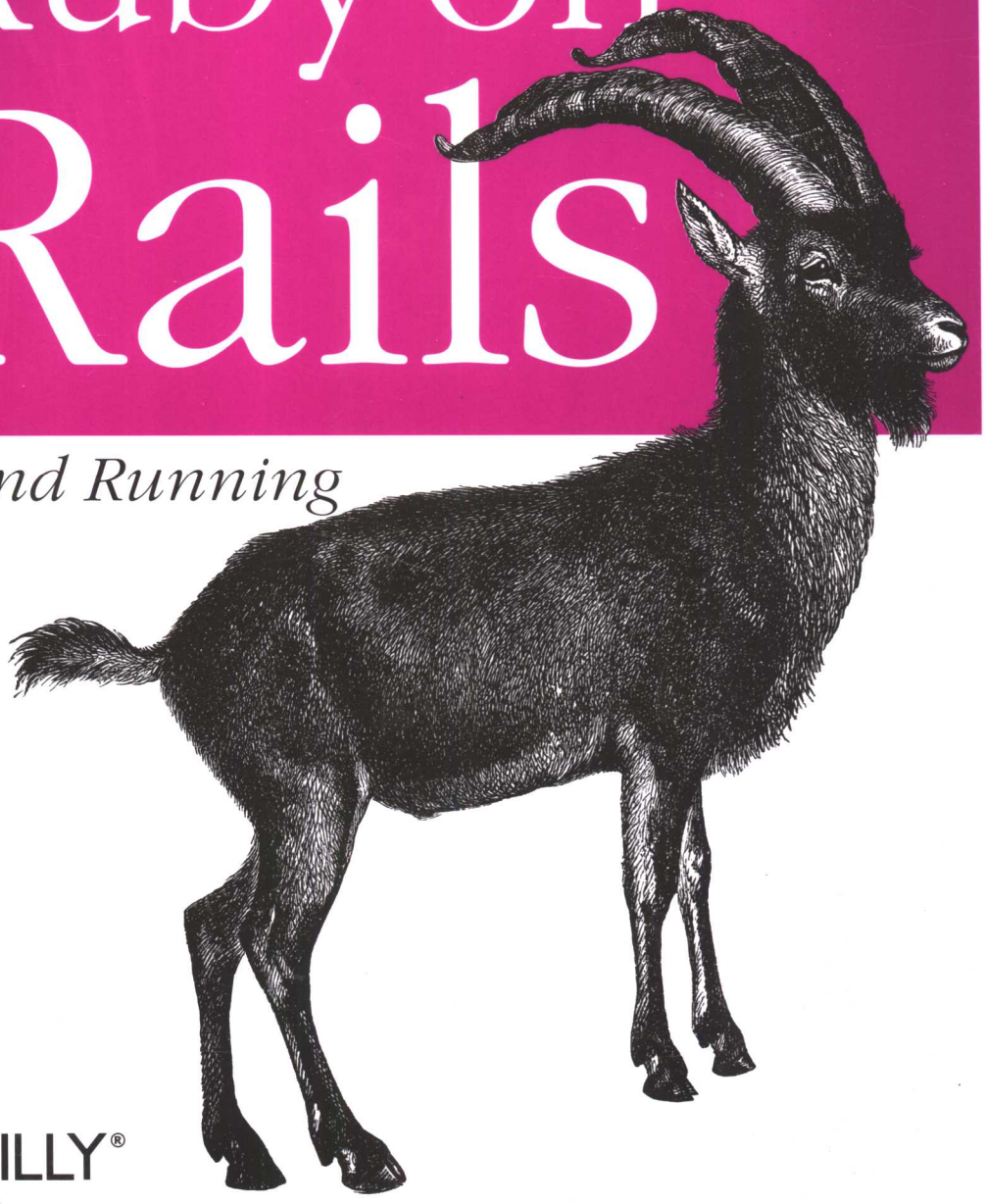
Ruby on Rails: Up and Running（影印版）

Ruby on Rails

Up and Running



O'REILLY®

東南大學出版社

Bruce A. Tate & Curt Hibbs 著

Ruby on Rails: Up and Running（影印版）

Ruby on Rails: Up and Running

Bruce A. Tate & Curt Hibbs

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc.授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

Ruby on Rails: Up and Running /（美）泰特（Tate, B. A.），（美）希布斯（Hibbs, G.）著．—影印本．—南京：东南大学出版社，2006.11

书名原文：Ruby on Rails: Up and Running

ISBN 7-5641-0569-0

Ⅰ.R… Ⅱ.①泰… ②希… Ⅲ.计算机网络－程序设计－英文 Ⅳ.TP393.092

中国版本图书馆CIP数据核字（2006）第115485号

江苏省版权局著作权合同登记

图字：10-2006-258号

书　　名/ Ruby on Rails: Up and Running（影印版）

书　　号/ ISBN 7-5641-0569-0

责任编辑/ 张烨

封面设计/ Karen Montgomery，张健

出版发行/ 东南大学出版社（press.seu.edu.cn）

地　　址/ 南京四牌楼2号（邮政编码210096）

经　　销/ 各地新华书店

印　　刷/ 扬中市印刷有限公司

开　　本/ 787毫米×980毫米　16开本　11.5印张

版　　次/ 2006年11月第1版　2006年11月第1次印刷

印　　数/ 0001-3000册

定　　价/ 26.00元（册）

O'Reilly Media, Inc.介绍

O'Reilly Media, Inc.是世界上在 UNIX、X、Internet 和其他开放系统图书领域具有领导地位的出版公司，同时是联机出版的先锋。

从最畅销的《The Whole Internet User's Guide & Catalog》（被纽约公共图书馆评为二十世纪最重要的50本书之一）到GNN（最早的Internet门户和商业网站），再到WebSite（第一个桌面PC的Web服务器软件），O'Reilly Media, Inc.一直处于Internet发展的最前沿。

许多书店的反馈表明，O'Reilly Media, Inc.是最稳定的计算机图书出版商——每一本书都一版再版。与大多数计算机图书出版商相比，O'Reilly Media, Inc.具有深厚的计算机专业背景，这使得O'Reilly Media, Inc.形成了一个非常不同于其他出版商的出版方针。O'Reilly Media, Inc.所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。O'Reilly Media, Inc.还有许多固定的作者群体——他们本身是相关领域的技术专家、咨询专家，而现在编写著作，O'Reilly Media, Inc.依靠他们及时地推出图书。因为O'Reilly Media, Inc.紧密地与计算机业界联系着，所以O'Reilly Media, Inc.知道市场上真正需要什么图书。

出版说明

随着计算机技术的成熟和广泛应用，人类正在步入一个技术迅猛发展的新时期。计算机技术的发展给人们的工业生产、商业活动和日常生活都带来了巨大的影响。然而，计算机领域的技术更新速度之快也是众所周知的，为了帮助国内技术人员在第一时间了解国外最新的技术，东南大学出版社和美国 O'Reilly Meida, Inc.达成协议，将陆续引进该公司的代表前沿技术或者在某专项领域享有盛名的著作，以影印版或者简体中文版的形式呈献给读者。其中，影印版书籍力求与国外图书“同步”出版，并且“原汁原味”展现给读者。

我们真诚地希望，所引进的书籍能对国内相关行业的技术人员、科研机构的研究人员和高校师生的学习和工作有所帮助，对国内计算机技术的发展有所促进。也衷心期望读者提出宝贵的意见和建议。

最新出版的一批影印版图书，包括：

- 《深入浅出 Ajax》（影印版）
- 《Ajax Hacks》（影印版）
- 《深入理解 Linux 网络内幕》（影印版）
- 《Web 设计技术手册 第三版》（影印版）
- 《软件预构艺术》（影印版）
- 《Ruby on Rails: Up and Running》（影印版）
- 《Ruby Cookbook》（影印版）
- 《Python 编程 第三版》（影印版）
- 《Python 技术手册 第二版》（影印版）
- 《Ajax 设计模式》（影印版）
- 《实用软件项目管理》（影印版）
- 《用户界面设计模式》（影印版）

Preface

The Ruby on Rails phenomenon is sweeping through our industry with reckless disregard for established programming languages, longstanding conventions, or commercial support. You can get a whole lot of information on Ruby on Rails from articles on the Web, excellent books, and even formal coursework. However, there's something missing. How does an established programmer, armed with nothing more than a little Ruby knowledge, go just beyond the basics, and be productive in Rails?

With *Ruby on Rails: Up and Running*, we are not going to reiterate the reference manual or replace Google. Instead, we'll strive to give you the big picture of how Rails applications hold together and tell you where to go for the information that we don't cover in the chapters. You will see how Rails dynamically adds features to all database models, called Active Record objects. By understanding the big picture, you'll be able to make better use of the best reference manuals to fill in the details.

We won't try to make you digest a whole lot of words. Instead, we'll give you the theory in the context of an end-to-end application. We'll walk you through the creation of a simple project—one that is a little more demanding than a blog or shopping cart, but with a simple enough structure that a Rails beginner will be able to quickly understand what's going on.

We're not going to try to cover each new feature. Instead, we'll show you the ones we see as the backbone, forming the most important elements to understand. We will also cover migrations and Ajax in some detail, because you won't find too much information on those two frameworks yet.

In short, we're not trying to build a comprehensive Rails library. We're going to give you the foundation you need to get up and running.

Who Should Read This Book?

Ruby on Rails: Up and Running is for experienced developers who are new to Rails and possibly to Ruby. To use this book, you don't have to be a strong Ruby

programmer. We do expect you to be a programmer, though. You should know enough about your chosen platform to be able to write programs, install software, run scripts using the system console, edit files, use a database, and understand how basic web applications work.

Conventions Used in This Book

The following typographic conventions are used in this book:

Plain text
: Indicates menu titles, menu options, menu buttons, and keyboard accelerators (such as Alt and Ctrl).

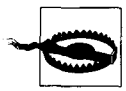
Italic
: Indicates new terms, URLs, email addresses, filenames, file extensions, pathnames, directories, and Unix utilities.

`Constant width`
: Indicates commands, the contents of files, and the output from commands.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values.

This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books *does* require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation *does* require permission.

You can get sample code at the main page for *Ruby on Rails: Up and Running*: *http://www.oreilly.com/catalog/rubyrails/*. You will find a ZIP file that contains the sample

project as it exists after each chapter, with each instance of the sample application numbered by chapter. If you want to skip a chapter, just download the right ZIP file.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Ruby on Rails: Up and Running* by Bruce A. Tate and Curt Hibbs. Copyright 2006 O'Reilly Media, Inc., 978-0-596-10132-9."

If you feel that your use of code examples falls outside fair use or the permission given here, feel free to contact us at *permissions@oreilly.com*.

Platforms

Ruby on Rails is cross-platform, but Unix and Windows shells behave differently. For consistency, we use Windows throughout the book. You can easily run the examples on the Unix or Mac OS X operating systems as well. You'll see a couple of minor differences:

- On Windows, you can specify paths with either the forward slash (/) or backslash (\) character. We'll try to be consistent and use the forward slash to specify all paths.
- On Windows, to run the various Ruby scripts that make up Rails, you need to explicitly type `ruby`. On Unix environments, you don't. If you're running Unix, and you are instructed to type the command `ruby script/server`, feel free to omit the `ruby`.
- On Windows, to run a process in a separate shell, precede the command with `start`. On Unix and Mac OS X, append an ampersand (&) character to run the command in the background.

Safari® Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at *http://safari.oreilly.com*.

How to Contact Us

We have tested and verified the information in this book and in the source code to the best of our ability, but given the amount of text and the rapid evolution of

technology, you may find that features have changed or that we have made mistakes. If so, please notify us by writing to:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

You can also send messages electronically. To be put on the mailing list or request a catalog, send email to:

info@oreilly.com

To ask technical questions or comment on the book, send email to:

bookquestions@oreilly.com

As mentioned in the earlier section, we have a web site for this book where you can find code, errata (previously reported errors and corrections available for public view), and other book information. You can access this web site at:

http://www.oreilly.com/catalog/rubyrails

For more information about this book and others, see the O'Reilly web site:

http://www.oreilly.com

Acknowledgments

Writing a book is a demanding exercise, taking passion, commitment, and persistence. The authors on the cover get all of the glory (and possibly the blame). Many people contribute to a book. We'd like to mention the people who made writing this book such a fulfilling experience.

Collectively, Curt and Bruce would like to thank the outstanding team of reviewers who provided so many great comments, including David Mabelle, Mauro Cicio, Brooke Hedrick, Faisal Jawdat, Shane Claussen, Leo de Blaauw, Anne Bowman, Seth Havermann, Dave Hastings, and Randy Hanford. We'd also like to thank David Geary for fleshing out some of the early ideas in Photo Share.

Ruby on Rails: Up and Running would be nothing without the excellent contributions of the core Ruby on Rails team. We would like to thank David Heinemeier Hansson (the creator of Rails), Florian Weber, Jamis Buck, Jeremy Kemper, Leon Breedt, Marcel Molina, Jr., Michael Koziarski, Nicholas Seckar, Sam Stephenson, Scott Barron, Thomas Fuchs, and Tobias Luetke. Ruby is a fantastic language, and we would like to thank the many who made it so. We throw out specific thanks to Yukihiro Matsumoto (a.k.a. "Matz"), the creator of Ruby, and to Dave Thomas and

Andy Hunt, without whom Ruby might have remained virtually unknown outside of Japan.

Bruce would like to specifically thank Curt, for stepping into this project after it seemed that it was dead. Also, thanks to those at AutoGas who were so instrumental in trying this technology within the context of a real production application—especially the core development team, including Mathew Varghese, Karl Hoenshel, Cheri Byerley, Chris Gindorf, and Colby Blaisdell. Their collective experience shaped this book more than you will ever know. Thanks to my Dutch friend Leo, again, for being such a supportive influence on this book, though you're mostly a Java developer. You have had more influence on me than you might expect. More than anyone else, I would like to thank my family. Kayla and Julia, you are the sparks in my soul that keep the creative fires burning. Maggie, you are my inspiration, and I love you more than I can ever say.

Curt would like to thank his wife, Wasana, for letting him disappear behind his computer screen late into the night (and sometimes into the following day) without complaint. I would also like to thank my friends at O'Reilly, for giving me a forum to spread the word about the incredible productivity advantages of Ruby on Rails. Specifically, I'd like to thank chromatic for publishing my ONLamp.com articles, and Mike Loukides for not giving up when I kept telling him I didn't want to write a book.

Table of Contents

CHAPTER 1

Zero to Sixty: Introducing Rails

Rails may just be the most important open source project to be introduced in the past 10 years. It's promoted as one of the most productive web development frameworks of all time and is based on the increasingly important Ruby programming language. What has happened so far?

- By December 2006, you're likely to see more published books on Rails than any of Java's single flagship frameworks, including JSF, Spring, or Hibernate.
- The Rails framework has been downloaded at least 500,000 times in only its second year, as of May 2006. These statistics compare favorably with the most popular open source frameworks in any language.*
- The Rails community mailing lists get hundreds of notes a day, compared to dozens on the most popular web development frameworks in other languages.
- The Rails framework has caused an explosion in the use of the Ruby programming language, which has been relatively obscure until recently.
- The Rails buzz generates increasingly hot debates on portals that focus on other programming languages. The Java community in particular has fiercely debated the Rails platform.

You don't have to go far to find great overviews of Rails. You can watch several educational videos that show Rails in action, narrated by the founder David Heinemeier Hansson. You can watch him build simple working applications, complete with a backing database and validation, in less than 10 minutes. But unlike the many quick-and-dirty environments you've seen, Rails lets you keep the quick and leave the dirty behind. It lets you build clean applications based on the model-view-controller philosophy. Rails is a special framework.

* The number 500,000 is actually a conservative estimate. Download statistics for a popular delivery vehicle, called *gems*, make it easy to track the number of Rails distributions by gems, but many other distributions exist, such as the Locomotive distribution on Mac OS X. The real download statistics could easily be twice this number.

Sure, Rails has its limitations. Ruby has poor support for object-relational mapping (ORM) for legacy schemas; the Rails approach is less powerful than Java's approach, for example.* Ruby does not yet have flagship integrated development environments. Every framework has limitations, and Rails is no different. But for a wide range of applications, the strengths of Rails far outpace its weaknesses.

Rails Strengths

As you go through this book, you'll learn how Rails can thrive without all of the extensive libraries required by other languages. Ruby's flexibility lets you extend your applications in ways that might have been previously unavailable to you. You'll be able to use a Rails feature called *scaffolding* to put database-backed user interfaces in front of your customers quickly. Then, as you improve your code, the scaffolding melts away. You'll be able to build database-backed model objects with just a couple of lines of code, and Rails will fill in the tedious details.

The most common programming problem in today's typical development project involves building a web-based user interface to manage a relational database. For that class of problems, Rails is much more productive than any other web development framework either of us has ever used. The strengths aren't limited to any single groundbreaking invention; rather, Rails is packed with features that make you more productive, with many of the following features building on one other:

Metaprogramming

Metaprogramming techniques use programs to write programs. Other frameworks use extensive code generation, which gives users a one-time productivity boost but little else, and customization scripts let the user add customization code in only a small number of carefully selected points. Metaprogramming replaces these two primitive techniques and eliminates their disadvantages. Ruby is one of the best languages for metaprogramming, and Rails uses this capability well.†

Active Record

Rails introduces the Active Record framework, which saves objects to the database. Based on a design pattern cataloged by Martin Fowler, the Rails version of Active Record discovers the columns in a database schema and automatically attaches them to your domain objects using metaprogramming. This approach to wrapping database tables is simple, elegant, and powerful.

* For example, Hibernate supports three kinds of inheritance mapping, but Rails supports only single-table inheritance. Hibernate supports composite keys, but Rails is much more limited.

† Rails also uses code generation but relies much more on metaprogramming for the heavy lifting.

Convention over configuration
: Most web development frameworks for .NET or Java force you to write pages of configuration code. If you follow suggested naming conventions, Rails doesn't need much configuration. In fact, you can often cut your total configuration code by a factor of five or more over similar Java frameworks just by following common conventions.

Scaffolding
: You often create temporary code in the early stages of development to help get an application up quickly and see how major components work together. Rails automatically creates much of the scaffolding you'll need.

Built-in testing
: Rails creates simple automated tests you can then extend. Rails also provides supporting code called *harnesses* and *fixtures* that make test cases easier to write and run. Ruby can then execute all your automated tests with the `rake` utility.

Three environments: development, testing, and production
: Rails gives you three default environments: development, testing, and production. Each behaves slightly differently, making your entire software development cycle easier. For example, Rails creates a fresh copy of the Test database for each test run.

There's much more, too, including Ajax for rich user interfaces, partial views and helpers for reusing view code, built-in caching, a mailing framework, and web services. We can't get to all of Rails' features in this book; however, we will let you know where to get more information. But the best way to appreciate Rails is to see it in action, so let's get to it.

Putting Rails into Action

You could manually install all of the components for Rails, but Ruby has something called *gems*. The `gem` installer accesses a web site, Ruby Forge, and downloads an application unit, called a gem, and all its dependencies. You can install Rails through gems, requesting all dependencies, with this command:*

```
gem install rails --include-dependencies
```

That's it—Rails is installed. There's one caveat: you also need to install the database support for your given database. If you've already installed MySQL, you're done. If

* If you want to code along with us, make sure you've installed Ruby and gems. Appendix A contains detailed installation instructions.

MVC and Model2

In the mid-1970s, the MVC (model-view-controller) strategy evolved in the Smalltalk community to reduce coupling between business logic and presentation logic. With MVC, you put your business logic into separate domain objects and isolate your presentation logic in a view, which presents data from domain objects. The controller manages navigation between views, processes user input, and marshals the correct domain objects between the model and view. Good programmers have used MVC ever since, implementing MVC applications using frameworks written in many different languages, including Ruby.

Web developers use a subtly different variant of MVC called *Model2*. Model2 uses the same principles of MVC but tailors them for stateless web applications. In Model2 applications, a browser calls a controller via web standards. The controller interacts with the model to get data and validate user input, and then makes domain objects available to the view for display. Next, the controller invokes the correct view generator, based on validation results or retrieved data. The view layer generates a web page, using data provided by the controller. The framework then returns the web page to the user. In the Rails community, when someone says MVC, they're referring to the Model2 variant.

Model2 has been used in many successful projects spread across many programming languages. In the Java community, Struts is the most common Model2 framework. In Python, the flagship web development framework called Zope uses Model2. You can read more about the model-view-controller strategy at *http://en.wikipedia.org/wiki/Model-view-controller*.

not, go to *http://rubyonrails.org* for more details on Rails installation. Next, here's how to create a Rails project:

```
> rails chapter-1
      create
      create  app/controllers
      create  app/helpers
      create  app/models
      create  app/views/layouts
      create  config/environments
      create  components
      create  db
      create  doc
      create  lib

...
      create  test/mocks/development
      create  test/mocks/test
      create  test/unit
      create  vendor
...
```

```
      create  app/controllers/application.rb
      create  app/helpers/application_helper.rb
      create  test/test_helper.rb
      create  config/database.yml
...
```

We truncated the list, but you get the picture.

Organization

The directories created during installation provide a place for your code, scripts to help you manage and build your application, and many other goodies. Later, we'll examine the most interesting directories in greater detail. For now, let's take a quick pass through the directory tree in the project we created:

app
: This application organizes your application components. It's got subdirectories that hold the view (*views* and *helpers*), controller (*controllers*), and the backend business logic (*models*).

components
: This directory holds *components*—tiny self-contained applications that bundle model, view, and controller.

config
: This directory contains the small amount of configuration code that your application will need, including your database configuration (in *database.yml*), your Rails environment structure (*environment.rb*), and routing of incoming web requests (*routes.rb*). You can also tailor the behavior of the three Rails environments for test, development, and deployment with files found in the *environments* directory.

db
: Usually, your Rails application will have model objects that access relational database tables. You can manage the relational database with scripts you create and place in this directory.

doc
: Ruby has a framework, called *RubyDoc*, that can automatically generate documentation for code you create. You can assist *RubyDoc* with comments in your code. This directory holds all the *RubyDoc*-generated Rails and application documentation.

lib
: You'll put libraries here, unless they explicitly belong elsewhere (such as vendor libraries).