

<<Ruby 最佳实践>>

图书基本信息

书名：<<Ruby 最佳实践>>

13位ISBN编号：9787564119355

10位ISBN编号：7564119357

出版时间：2010-1

出版时间：东南大学出版社

作者：布朗

页数：309

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

前言

In 1993, when Ruby was born, Ruby had nothing. No user base except for me and a few close friends. No tradition. No idioms except for a few inherited from Perl, though I regretted most of them afterward. But the language forms the community. The community nourishes the culture. In the last decade, users increased—hundreds of thousands of programmers fell in love with Ruby. They put great effort into the language and its community. Projects were born. Idioms tailored for Ruby were invented and introduced. Ruby was influenced by Lisp and other functional programming languages. Ruby formed relationships between technologies and methodologies such as test-driven development and duck typing. This book introduces a map of best practices of the language as of 2009. I've known Greg Brown for years, and he is an experienced Ruby developer who has contributed a lot of projects to the language, such as Rupture and Prawn. I am glad he compiled his knowledge into this book.

<<Ruby 最佳实践>>

内容概要

你能够编写真正优雅的Ruby代码吗？

《Ruby最佳实践》正是为想要像专家那样使用Ruby的程序员所准备的。

《Ruby 最佳实践(影印版)》由Ruby项目Prawn的开发者所著，简洁地向你阐释如何使用Ruby编写优美的应用程序接口和领域特定语言。

此外，还包括如何应用函数式编程的思想和技术，使代码更简洁，使你更有效率。

通过《Ruby 最佳实践(影印版)》，你将会学到如何编写可读性更高，表达能力更强的代码，以及许多其他方面的内容。

《Ruby最佳实践》将会帮助你：
* 理解Ruby代码块所蕴含的神秘力量* 学习如何在不破坏原有Ruby代码的情况下进行调整，例如运行时在模块内糅合* 探究测试与调试中的细节，以及如何从易测性出发进行设计* 学习通过让事情保持简单来编写快速代码* 用于文本处理和文件管理的开发策略，包括正则表达式* 理解为什么会发生错误以及错误是如何发生的* 利用Ruby的多语言特性减少文化障碍
《Ruby 最佳实践(影印版)》还包含多个章节对测试代码、设计应用程序接口以及项目维护做了全面介绍。

《Ruby最佳实践》将陪伴你学习如何将这门丰富、优美的语言发挥到极致。

<<Ruby 最佳实践>>

作者简介

Gregory T. Brown是康涅狄格州纽黑文市的一位Ruby爱好者，他的大多数时间都花在了与Ruby语言相关的自由软件项目上。

他是Ruport的原作者，并且是Prawn的作者，该Ruby库被用来生成PDF文档。

书籍目录

Foreword Preface 1. Driving Code Through Tests A Quick Note on Testing Frameworks Designing for Testability Testing Fundamentals Well-Focused Examples Testing Exceptions Run the Whole Suite at Once Advanced Testing Techniques Using Mocks and Stubs Testing Complex Output Keeping Things Organized Embedding Tests in Library Files Test Helpers Custom Assertions Conclusions 2. Designing Beautiful APIs Designing for Convenience: Report 's Table() feature Ruby 's Secret Power: Flexible Argument Processing Standard Ordinal Arguments Ordinal Arguments with Optional Parameters Pseudo-Keyword Arguments Treating Arguments As an Array Ruby 's Other Secret Power: Code Blocks Working with Enumerable Using Blocks to Abstract Pre- and Postprocessing Blocks As Dynamic Callbacks Blocks for Interface Simplification Avoiding Surprises Use attr_reader, attr_writer, and attr_accessor Understand What method? and method! Mean Make Use of Custom Operators Conclusions 3. Mastering the Dynamic Toolkit BlankSlate: A BasicObject on Steroids Building Flexible Interfaces Making instance_eval() Optional Handling Messages with method_missing() and send() Dual-Purpose Accessors Implementing Per-Object Behavior Extending and Modifying Preexisting Code Adding New Functionality Modification via Aliasing Per-Object Modification Building Classes and Modules Programmatically Registering Hooks and Callbacks Detecting Newly Added Functionality Tracking Inheritance Tracking Mixins Conclusions 4. Text Processing and File Management Line-Based File Processing with State Tracking Regular Expressions Don 't Work Too Hard Anchors Are Your Friends Use Caution When Working with Quantifiers Working with Files Using Pathname and FileUtils The tempfile Standard Library Automatic Temporary Directory Handling Collision Avoidance Same Old I/O Operations Automatic Unlinking Text-Processing Strategies Advanced Line Processing Atomic Saves Conclusions 5. Functional Programming Techniques Laziness Can Be a Virtue (A Look at lazy.rb) Minimizing Mutable State and Reducing Side Effects Modular Code Organization Memoization Infinite Lists Higher-Order Procedures Conclusions 6. When Things Go Wrong A Process for Debugging Ruby Code Capturing the Essence of a Defect Scrutinizing Your Code Utilizing Reflection Improving inspect Output Finding Needles in a Haystack Working with Logger Conclusions 7. Reducing Cultural Barriers m17n by Example: A Look at Ruby 's CSV Standard Library Portable m17n Through UTF-8 Transcoding Source Encodings Working with Files Transcoding User Input in an Organized Fashion m17n in Standalone Scripts Inferring Encodings from Locale Customizing Encoding Defaults m17n-Safe Low-Level Text Processing Localizing Your Code Conclusions 8. Skillful Project Maintenance Exploring a Well-Organized Ruby Project (Haml) Conventions to Know About What Goes in a README Laying Out Your Library Executables Tests Examples API Documentation via RDoc Basic Documentation Techniques and Guidelines Controlling Output with RDoc Directives The RubyGems Package Manager Writing a Gem::Specification Working with Dependencies Rake: Ruby 's Built-in Build Utility Conclusions A. Writing Backward-Compatible Code B. Leveraging Ruby 's Standard Library C. Ruby Worst Practices Index

章节摘录

插图：Rake is a very powerful tool that deserves its own chapter or even its own cookbook. There are a ton of useful recipes out there in the wild, so be sure to make the Rakefile one of your first stops in any new codebase you need to review. Understanding and using Rake effectively is key to successfully managing any moderately complex Ruby project, so be sure not to overlook its significance and practical utility. If you want to make the most out of this tool, there are just a few things to keep in mind: Rake provides custom tasks for common needs such as generating RDoc, running unit tests and packaging up a project for distribution. Because these tasks are highly configurable, it is better to use them than to reinvent the wheel. Any other repetitive action that is necessary for maintaining your project can be wrapped in a task to simplify things. Typically, any lengthy command that needs to be run in the shell is fair game for this sort of simplification. Any task that has a preceding desc () call will be listed with a meaningful message in the rake —tasks output for your project. Rake's ability to define prerequisite tasks allows you to build dependency-based workflows that allow you to model multiple-step tasks as needed. Namespaces can be used to segment off tasks into their own subspaces, minimizing the risk of naming clashes. I've tried to stick mainly to the easily overlooked aspects of Rake here, but there is a whole lot that we could have covered and didn't. Be sure to consult the Rake documentation if you're interested in finding out more. Depending on what you were looking for, this chapter may have been a bit different from what you were expecting based on the title. However, what you will find is that the things we've discussed here will really take you far when it comes to improving the maintainability of your projects. Though far from glamorous, things like good documentation, well-organized folders and files, and a way to automate as much of the gruntwork as possible does a whole lot for your projects. Poorly maintained projects can be a huge drain on developer productivity and morale, yet nicely curated code can be downright enjoyable to work with, even if you're brand-new to a project. The tools and techniques we've discussed so far aren't going to make maintenance completely painless, but will still provide a solid foundation to work off of that will grow over time.

<<Ruby 最佳实践>>

媒体关注与评论

“这是一本极为务实的著作，各层次的开发人员都能从中借鉴。

”——Brad Ediger，Madriska Media Group的领袖开发者，同时也是《Advanced Rails》（O'Reilly）的作者 “终于有这样一本书问世了，它不仅教我如何使用Ruby，更教会我如何正确地使用它。每位Ruby爱好者的书架上都该摆上一本《Ruby最佳实践》。

”——Jeremy McAnally，ENTP开发者，同时还是《Ruby in Practice》（Manning）一书的作者 “我敢打赌，通过阅读这本书，你一定学到了可以提高Ruby编程能力的新技巧。

”——James Edward Gray II，代码忍者及Ruby 1.9的CSV标准库的作者

<<Ruby 最佳实践>>

编辑推荐

《Ruby 最佳实践(影印版)》是由东南大学出版社出版的。

<<Ruby 最佳实践>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>