O'REILLY®

# RESTful Rails Development（中文版）

构建开放应用和服务

Silvia Puglisi 著
安道 译

中国电力出版社

# RESTful Rails Development

（中文版）

*Silvia Puglisi* 著

安道 译

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo O'REILLY®

O'Reilly Media, Inc.授权中国电力出版社出版

中国电力出版社

**图书在版编目（CIP）数据**

RESTful Rails开发 /（美）西尔维娅·普里兹（Silvia，Puglisi）著；安道译. -- 北京: 中国电力出版社, 2017.2

书名原文：RESTful Rails Development

ISBN 978-7-5198-0058-1

I. ①R… II. ①西… ②安… III. ①计算机网络－程序设计 IV. ①TP393

中国版本图书馆CIP数据核字（2016）第282864号

北京市版权局著作权合同登记

图字：01-2016-7464号



封面设计/ Ellie Volckhausen，张健

出版发行/ 中国电力出版社（http://www.cepp.sgcc.com.cn）

地　　址/ 北京市东城区北京站西街19号（邮政编码100005）

经　　销/ 全国新华书店

印　　刷/ 北京天宇星印刷厂

开　　本/ 787毫米×980毫米　16开本　16.5印张　309千字

版　　次/ 2017年2月第一版　2017年2月第一次印刷

印　　数/ 0001－3000册

定　　价/ 49.00元（册）

# O'Reilly Media, Inc.介绍

O'Reilly Media通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自1978年开始，O'Reilly一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了Make杂志，从而成为DIY革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版，在线服务或者面授课程，每一项O'Reilly的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

## 业界评论

“O'Reilly Radar博客有口皆碑。”

——Wired

“O'Reilly凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2.0

“O'Reilly Conference是聚集关键思想领袖的绝对典范。”

——CRN

“一本O'Reilly的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照Yogi Berra的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去Tim似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

献给 Aaron，感谢他给我启示。

献给 Sara，她是我生活中的伙伴，陪我一起玩耍。感谢她一直伴我左右。

献给我的家人，感谢他们始终支持我。

献给其他所有挚友，感谢他们相知相守。

# 对本书的赞誉

“吾等目力短亦浅，能见百事待践行。”

——Alan Turing

“保持好奇心。广泛阅读。尝试新鲜事物。人们所指的智慧归根结底说的是好奇心。”

——Aaron Swartz

“语义网并不是独立的网络，而是目前互联网的延伸。在语义网中，信息有明确的意义，可以让人类与电脑协作。”

——Tim Berners-Lee

“这个框架与其他框架的区别在于，它采用了约定优于配置原则，因此应用易于开发，也易于理解。”

——Sam Ruby，ASF 董事会董事

“Rails 是 Ruby 的制胜法宝。”

——Yukihiro Matsumoto，Ruby 创造者

“生活是一个分布式对象系统。然而，人与人之间的交流是分布式超媒体系统，心智、口中说出的话和肢体语言、听到的和看到的，以及想象都是这个系统的一部分。”

——Roy T. Fielding

## 作者简介

**Silvia Puglisi** 是一名软件工程师，生活在西班牙巴塞罗那。她还是加泰罗尼亚理工大学（UPC）遥测工程系信息安全小组的科研工程师，目前在攻读博士。Silvia 之前在 Google 公司工作，任职运营工程师和企业工程师。

她热爱技术和 Web，为了乐趣和生计，喜欢构建开放应用和服务。需要远离电脑屏幕休息眼睛时，她喜欢到海滩放松，还喜欢冲浪。

## 封面介绍

本书封面上的动物是德马雷硬毛鼠（学名 Capromys pilorides，也叫古巴硬毛鼠），这种大型啮齿动物只生活在古巴，以19世纪法国动物学家 Anselme Gaëtan Desmarest 命名。硬毛鼠的栖息地广泛分布于古巴各地，包括红树林、平原和沼泽地。

德马雷硬毛鼠身材矮壮，腿很短，走起路来像鸭子一样一摇一摆；毛粗，呈黑色或棕色；爪子很大，利于爬行。硬毛鼠一般结对生活，最常在白天活动。夜晚，睡在树间或石头间的洼地里（这种动物不挖洞）。硬毛鼠是杂食动物，常吃树皮、树叶、坚果和水果，偶尔也吃蜥蜴和昆虫。在所有啮齿动物中，这种硬毛鼠的胃是最复杂的，有三个隔室。

德马雷硬毛鼠是体型最大的硬毛鼠，平均 12~24 英寸长，体重最高能到 19 磅。以前，古巴人喜欢捕食这种硬毛鼠，因为它们体型硕大，味鲜可口，但是 1968 年立法，禁止未经允许捕杀。如今，德马雷硬毛鼠的数量太多，经常破坏农作物，因此被视作害虫。

克里斯多弗·哥伦布在新大陆最先吃到的肉类可能就是硬毛鼠，因为这种动物在加勒比群岛很常见，而且是土著人的主食。

O'Reilly 出版的图书，封面上很多动物都濒临灭绝。这些动物都是地球的至宝。如果你想知道如何保护这些动物，请访问 *animals.oreilly.com*。

封面图片出自 Lydekker 的《Royal Natural History》。

# 目录

# 前言

本书主要说明如何使用 Rails 设计和开发表述性状态转移（Representational State Transfer，REST）平台。REST 是 Web 的架构风格，由一系列应用于组件、连接器和数据元素的约束组成，用于构建如今更广泛的分布式超媒体系统——万维网。

构建平台比只构建产品或应用更合理，其原因有很多。平台好比生态系统，连通着不同的应用、服务、用户、开发者和合作伙伴。平台得益于直接合作者的输入，有助于创新。平台能提供应用程序编程接口（Application Programming Interface，API）和软件开发工具包（Software Development Kit，SDK），让客户更充分地利用平台。不构建应用而构建平台还有一个原因：Web 正在逐渐变化，它现在的使用方式是人类读者在网页中浏览内容，不过最终肯定会变成服务与客户端（不一定是人类）之间交换数据。2001 年 Tim Berners Lee 在《科学美国人》杂志中发表了一篇文章，题为“The Semantic Web”（语义网），他预言了这种趋势，不过这只是预言的一部分。Web 正在变得更加语义化。过去，软件代理不能“理解”HTML 文档，虽然可以部分解析，但是无法判断文档是博客文章还是其他类型，例如伦敦的公交时刻表。

以前，我们把 Web 看做相互链接的超文本文档，而如今，Web 文档更像是相互链接的数据对象，或称超数据（hyperdata）。Web 应用可以通过人类可读的形式显示超数据，也可以解析超数据，把信息提供给其他服务或应用使用。简单来说，语义网是超文本的正常进化方向。既然要通过 API 处理超数据对象，就得实现不同的通信协议，让各种技术都能访问超数据对象。为了在各种各样的系统之间交换信息，API 要提供一种与语言无关的消息格式，例如 XML 或 JSON，用作所交换消息的容器。细想可知，“超媒体 API”的目的是让任何设备和应用都能访问和使用。因此，这种 API 的架构方式与 Web 类似，伺服和使用 API 的方式也与浏览网页类似。

在众多 Web 开发框架中选择 Rails 也是有原因的。首先，Rails 是使用 Ruby 编程语言开发的。Ruby 易于使用，对 Web 开发者来说，尤其容易。Ruby 完全面向对象，而且是开源的，社区充满生机，开发了大量形形色色的有趣项目和语言库，把开发变得更简单。Ruby on Rails 是个务实的框架，干净利落地实现了模型 - 视图 - 控制器（MVC）模式，易于对实际的场景建模。使用 Rails 能轻易创建应用的骨架，无需重复编写代码，提升了功能的开发速度。Rails 还遵守敏捷开发方法，提倡灵活性、演进式开发和迭代交付。

本书的目的是鼓励开发者们放弃产品，转而有组织地设计平台，而且要立即这样做，并且希望未来 Web 中增加的新服务能更易于发现、更易于集成，从而促进开放信息交换，增进组织间的合作关系。读完每章之后，读者都将学到一些新知识，知道如何构建和有组织地扩展横跨多台设备的多重服务。希望读完本书后你能更好地理解如何构建由不同服务组成的架构，通过协作的 API 和应用共享资源。

# 为什么使用 Rails 而不用 Node.js

过去几年，有很多文章对 Rails 和 Node.js 做了对比。虽然二者都能用于构建 Web 应用，但是 Node.js 和 Rails 之间有一些本质的区别。

首先，Rails 这个 Web 框架有自己的一套见解，而 Node.js 则基于 Chrome 的 JavaScript 运行时构建网络应用（无法独抒己见）。简而言之，Node.js 是服务器端 JavaScript 实现的。

Rails 采用 MVC 架构，交互干净利落，行之有效。在 Node.js 中，必须使用插件和辅助措施才能实现 Rails 开箱即用的那种集成层次。此外，在 Node.js 中虽然前后端都使用 JavaScript 编程，但是开发产品的速度不一定很快。使用 Node.js 开发原型的速度可能很快，可是你必须认真评判 Node.js 能否完美地伸缩产品。记住，使用 Rails 开发一个完整应用的速度与使用 Node.js 开发的速度一样快，有时甚至更快。读到后文你会发现，使用 Rails 构建 REST API 是多么迅速、多么简单。

目前，Rails 社区相当成熟，而且充满活力。Rails 社区依旧不断探索新鲜事物，创建着激动人心的项目，编写着各种各样的模块。使用 Rails 还能学习 Ruby。Ruby 开发者社区生气勃勃，很多新产品都是使用 Ruby 编写的，例如 Logstash（*https://www.elastic.co/products/logstash*）、Chef（*https://www.chef.io/chef/*）、Puppet（*https://puppetlabs.com/*）、Homebrew（*http://brew.sh/*）等。而且，Ruby 程序员的需求量仍在稳步增长。总之，Node.js 是个有趣的技术，可以正式使用，也可以学来玩玩，用于创建原型；而 Rails 更成熟，更适合用于开发长期稳定的项目。

# 为什么（我觉得）你应该阅读本书

读一本书的原因太多了，无法一一列出。开始写作本书时，我设想了一个充满微观通信应用的世界，那些应用给养着数据网络，而数据网络反哺着那些应用。

我想着 REST 式原则，想象着与设备无关的服务，这些服务消费、处理和创建数据流。写作本书之前，我与一些朋友和同事进行了几次讨论，我们谈到了互联网的未来，分析了为什么超媒体被认为是真正的语义网。网上有些博客文章、论坛和社交网络也有讨论这些话题的，看来，程序员、架构师、市场营销人员和普通人对此有不同的观点。有些人抱怨没有合适的方式，无法在业务中采用；有些人抱怨设计简陋，开发人员怠于实践；还有些人异常兴奋，意欲充分释放 REST 式服务的潜能。

本书涵盖以下话题：

- 如何开发 REST 式应用。
- 如何设计 REST 式架构。
- 如何部署 REST 式服务。

因此，如果你有以下诉求，就应该阅读本书：

- 想钻研 REST 式开发。
- 想学习如何设计小型应用生态系统。
- 只想设计 API，连接某些外部服务。

你可能是有几年经验的开发者，或是迫不及待想开始新项目的学生。你可能是一名工程师，想探索创建伟大应用的不同方式，或者想说服领导，让他们看清 REST 式服务和超媒体的无限可能。你可能是项目经理，有些技术背景，想理解 REST 式服务背后的逻辑，或者在市场部门工作，但是想学习如何开放平台，提供给 Web 中众多的服务使用。

如果你对 Web 的未来憧憬满满，如果你强烈希望 Web 保持开放，如果你展望的语义网是通信服务组成的一张网，或者你想编写软件实现这些范式，那么你应该阅读本书。如果你想学习 API 设计和超媒体范式，如果你仅仅对数据网络有热情，抑或你想知道如何快速为新服务创建原型，或者想知道如何在商业项目中应用超媒体模型，那么本书也值得一读。我希望本书能为你当前和未来的项目提供帮助，也希望本书能给你提供指导，让你构建出色的颠覆性应用，让你创建能让 Web 更开放、能让更多设备访问的服务，让你提供设计优雅的超媒体 API，使数据更易于使用和处理。本书是为这样的你而写的，希望你能充分享受。

# 本书内容

这本书不必从头读到尾，当然，如果你愿意，也可以这么做。每一章都是独立的，说明REST 架构和 Rails 开发的一个方面。

*第 1 章 从超文本到超数据*

这一章介绍访问和使用 Web 的方式正在发生的变化，以前网络上都是人类可读的超文本文档，而如今已经变成 Web 应用，既可以显示信息供人类读者阅读，也可以提供端点，供软件代理消费数据。

*第 2 章 Ruby on Rails 入门*

这一章介绍 Ruby on Rails。首先说明如何搭建开发环境，然后介绍一些 RVM 和 rbenv 的基础知识，并且概述 Rails 应用架构的一些基本概念，随后，开发第一个应用。这是一个简单的“Hello Rails”应用，严格来说算不上应用，其实只是个 API。

*第 3 章 初尝 API 设计*

这一章使用维基百科的分类和分类链接数据库转储创建一个简单的 API，带你一览 API 设计的注意事项。最终开发出来的 API 有两个端点。对给定的关键字，这个 API 会返回 JSON 格式的维基百科分类信息或分类图。

*第 4 章 REST 之外的世界*

这一章涵盖 REST 和 CURD（创建、读取、更新、删除）设计，介绍架构约束、资源和表述，以及 HTTP 的语义。我们会扩展前一章开发的分类 API，说明这些概念。

这一章还会说明 Rails 如何处理 REST 逻辑，因为它内置对 CRUD 的支持。

*第 5 章 使用 Rails 设计 API*

这一章扩展之前所介绍的 REST 架构知识，探讨超媒体范式。

我们会使用前面开发的分类 API 做些实际的演练。我们以第 3 章和第 4 章中的 API 为基础，说明使用 Rails 开发 API 的架构和设计。我们将使用分类链接扩展 API，把它变得可探索。

*第 6 章 异步 REST*

这一章探讨 REST 架构中与异步操作有关的方面。异步操作经常用于执行需要一定时间才能完成的操作。我们将介绍一些异步操作的最佳实践。

*第 7 章 测试 REST 式服务*

这一章讨论测试，重点说明如何测试使用 Rails 开发的 REST 式服务。替身是什么？桩件是什么，驭件又是什么？这一章将为你一一解答。这一章还会教你集成测试和测试的最佳实践。

*第 8 章 微服务和微应用*

这一章主要讨论服务导向式架构（Service-Oriented Architecture，SOA）和分布式系统设计的基础知识，并展望细粒度的协作式微服务和微应用的美好未来。我们将举一个例子，说明如何让一系列应用和 API 访问共享的资源。

在那个示例中，有一个 API 用于返回城市中值得一去的地点，然后映射到本书开发的第一个 API 所返回的分类上，最后综合利用这些数据获取城市中值得一走的徒步路线。

*第 9 章 把数据流映射到应用 UI 上*

这一章使用 Ruby on Rails 和 Ember.js 开发一个应用，消费前面几章开发的 API。这一章专门讲解如何把不同的数据源映射到一个应用 UI 上。Walks 应用中的资源将被映射到用户界面上。

*第 10 章 部署 API*

这一章主要讨论部署，以及构建中间件，供外部消费者连接我们的 API。这一章能回答你对如何部署 API 的疑问，还将介绍反向代理等概念。我们将在 Heroku 中配置一个反向代理，使用它把第 5 章开发的 API 和利用那些 API 的应用连接起来。

*第 11 章 管理应用的生态系统*

这一章介绍 API 管理方案，帮你在自己开发和依赖外部服务之间做出抉择。这一章还将介绍密钥配置、角色管理、流量监控，以及 API 和应用的生命周期等概念。

*第 12 章 使用数据流：在自己的应用中集成外部 API*

这一章解说如何把外部服务和 API 集成到自己的应用中。我们将使用开放数据，或者直接使用维基百科、YouTube 或 Twitter 的 API 获取数据，这样就能在分布式平台中导入可用的资源，而不必重复调用。

*第 13 章 开发与设备无关的应用*

这一章扩展之前的示例，让它们支持不同的设备，包括移动平台和物联网。如何让 Web 应用支持移动手机、Arduino 或树莓派，然后有序地集成生成的数据呢？我们将构建一个简单的天气应用，让它访问、消费和生成数据。

*第 14 章 数据分析*

这一章主要讨论对数据流的控制和分析。我们将使用哪些工具开发或集成，以便管控平台中的数据流呢？

*第 15 章 优雅地弹性伸缩*

这一章说明如何优雅地弹性伸缩，主要的关注点是弹性伸缩。这一章特别能引起 Rails 开发者的兴趣，因为 Rails 的伸缩性是个大问题。

*第 16 章 隐私和安全*

这一章专门讨论安全和隐私。我们应该如何防护自己的平台，怎么确认用户的数据是安全的呢？

# 本书不涉及的内容（以及到何处寻找答案）

这不是一本专门讲解 Ruby 的书，也不是讲解 Rails 的书。本书虽然涉及 Rails 开发，但是如果你想学习 Ruby 编程语言或 Rails 框架的核心概念，应该阅读其他专门讲解 Ruby on Rails 或 Ruby 的书。

这不是一本讲解 JavaScript 或 Ember.js 等框架的书，不过我们会使用一些 JavaScript，还会搭建 Ember 环境。

这不是一本专门讲解 Nginx、Redis 或 Lua 的书，不过，讨论 API 管理、弹性伸缩和分析相关的问题时，我们可能会使用或提到这些技术。

这也不是一本讲解隐私或安全的书，不过有一整章会介绍这两个话题（一定要记住，“数据多了，责任也大了”）。在最后一章，我会提供更多的权威资源，以便你学习这些科目。

如果所涉话题超出了本书范畴，我会在这样的注释框中提供额外的资源。

# 资源

本书使用的全部代码可以到 GitHub 中下载，这些代码基于 GPLv3 许可证发布。书中很多地方都提供了所涉话题的权威资源：每一章末尾都有一些链接，供你钻研各章讨论的技术。

# 代码风格

Ruby 和 Ruby on Rails 没有多少代码风格，而其他语言可能有很多，有时可能太多了，每个人都遵守自己的风格。

在本书中，我会采用一些简单的约定，借此希望让代码更易于阅读，并且不解自明。

这些约定是：

- 源码文件使用 UTF-8 编码。