RESTful Web Services

Web Services for the Real World

David Heinemeier Hansson 作序推荐

RESTful Web Services

中文版

Leonard Richardson & Sam Ruby 著

*W3China*徐涵 李红军 胡伟 译

O'REILLY®

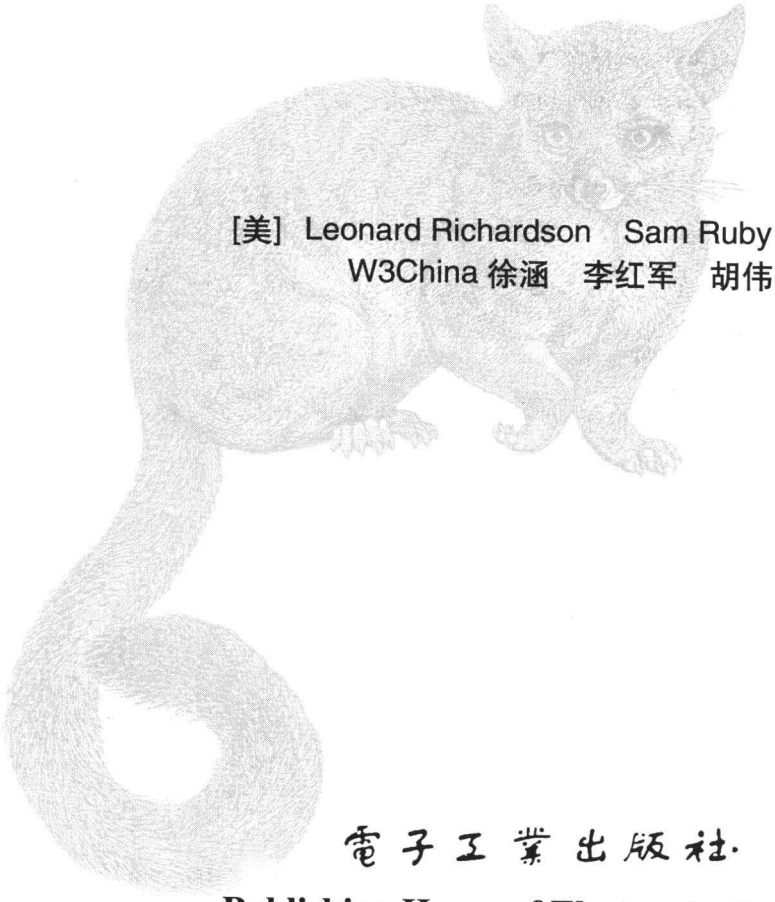
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

TP393.09/182
2008

O'REILLY®

RESTful Web Services 中文版

RESTful Web Services



[美] Leonard Richardson Sam Ruby 著
W3China 徐涵 李红军 胡伟 译

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内容简介

本书向读者介绍了什么是 REST、什么是面向资源的架构（Resource-Oriented Architecture，ROA）、REST 式设计的优点、REST 式 Web 服务的真实案例分析、如何用各种流行的编程语言编写 Web 服务客户端、如何用三种流行的框架（Ruby on Rails、Restlet 和 Django）实现 REST 式服务等。不仅讲解 REST 与面向资源的架构（ROA）的概念与原理，还向读者介绍如何编写符合 REST 风格的 Web 2.0 应用。本书详实、易懂，实战性强，提供了大量 RESTful Web 服务开发的最佳实践和指导，适合广大的 Web 开发人员、Web 架构师及对 Web 开发或 Web 架构感兴趣的广大技术人员与学生阅读。

978-0-596-52926-0 RESTful Web Services, First Eition. Copyright © 2007 by O'Reilly Media, Inc. Simplified Chinese edition, jointly published by O'Reilly Media, Inc. and Publishing House of Electronics Industry, 2008. Authorized translation of the English edition, 2007 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

本书中文简体版专有出版权由 O'Reilly Media, Inc.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：01-2007-3949

图书在版编目（CIP）数据

RESTful Web Services 中文版／（美）理查森（Richardson,L.），（美）露比（Ruby,S.）著；徐涵，李红军，胡伟译.—北京：电子工业出版社，2008.5
书名原文：RESTful Web Services
ISBN 978-7-121-06227-8
I. R… II.①理…②露…③徐…④李…⑤胡… III.互联网络－网络服务器－程序设计 IV.TP368.5

中国版本图书馆 CIP 数据核字（2008）第 036263 号

责任编辑：何　艳
项目管理：梁　晶
封面设计：Karen Montgomery　　张　健
印　　刷：北京市天竺颖华印刷厂
装　　订：三河市金马印装有限公司
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
开　　本：787×980　1/16　印张：28　字数：590 千字
印　　次：2008 年 5 月第 1 次印刷
定　　价：69.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
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。
服务热线：（010）88258888。

序
Foreword

自从架构师们发现了另一个可以搞复杂了再卖给大公司的点子以来，Web 服务领域就处在一条快速成为超级新秀的道路上。不过谢天谢地，还没有完全迷失方向。对 HTTP 的再度重视正在兴起，而且在 REST 的旗号下，HTTP 显示出了取代那些大公司试图强加在人们头上的技术的相当实力；REST 是一套简单的原则，开发者们可以根据这些原则按贴近 Web 的方式来把应用连接起来。

《RESTful Web Services》将教你如何使用这些原则。它的讲解是实实在在的，无夸大之辞，也没有绕弯子——那种做法，已经害得一批 Web 开发者认为 Web 服务难得只有依靠大公司来做了。每位从事 Web 开发的人员都应该读读这本书。

—David Heinemeier Hansson

前言
Preface

复杂的系统总是由简单的系统演变而来的。

—John Gall

Systemantics

我们写这本书，是要告诉你一项令人瞩目的新技术。嗬，它很热门，它会彻底改变我们编写分布式系统的方式。我们要讲的是万维网（World Wide Web，简称 Web）。

没错，Web 不是什么新技术，也不如昔日那么火了，而且从技术角度来看，它并不是那么令人瞩目。但它的确改变了我们许多。这 10 年来，Web 已经改变了我们生活的方式；不过，更多潜在的改变将等待我们。Web 是简单的、无所不在的；然而，它作为分布式编程平台的潜力却被忽视了。我们编写本书的目的，就是要让大家体验 Web 的这种潜力。

说 Web 作为分布式编程平台的潜力被忽视了，这听上去也许令人感到诧异。毕竟，本书还要与其他 Web 服务相关书籍竞争。问题是，大部分如今的“Web 服务”都与 Web 毫无干系。它们采用像 COM、CORBA 那样的重量级分布式对象访问架构——这与 Web 的简单性背道而驰。如今的“Web 服务”架构重复或忽略了 Web 赖以成功的每一种特性。

其实并不需要那样。Web 背后的（underlying）技术足以支撑强大的远程服务——其实那些服务就存在着，而且我们每天都在使用它们。这种服务可以延伸到巨大的规模——其实这已经实现了。以 Google 搜索引擎为例，它不就是一个对海量数据库进行查询并返回结构化搜索结果的远程服务吗？通常，我们不把网站（web site）当作“服务（service）”来看，因为网站的最终用户是人，而服务是程序之间的对话。但网站就是服务。

每个 Web 应用（包括每个网站）都是一个服务（service）。只要遵从 Web 的理念而不是违反它，只要不把 Web 特有的能力隔离在很多层抽象之下，你就能够让可编程应用（programmable applications）利用这种能力。现在是让“Web 服务”回归“Web”理念的时候了。

此为试读，需要完整PDF请访问：www.ertongbook.com

令网站易于被上网者使用的那些特性，同样也令 Web 服务 API 易于被程序员所使用。为寻找服务的设计原则，我们可以从网站的设计原则着手，并思考如果网站的使用者不是人而是程序，那最终将得到什么样的设计原则。

本书就是这么做的。我们的中心目标，就是展示 Web 基础技术（HTTP 应用协议、URL 命名标准、XML 标记语言）的强大能力、适用场合及局限等。本书主要讲的是 Web 背后的（underlying）一套设计原则——表示性状态转移（Representational State Transfer），或简称为 REST。我们率先为“REST 式（RESTful）”Web 服务提出了最佳实践（best practices）。我们不会采用含糊或臆断性的语言，相反，将用具体的建议来取代那些坊间传言（folklore）和隐性知识。

我们引入了面向资源的架构（Resource-Oriented Architecture，ROA）作为用于设计 REST 式 Web 服务（RESTful web services）的一组切合实际的原则。我们还会教你如何编写客户端程序来调用 REST 式服务。我们将采用一些真实的 REST 式服务作为案例，比如：Amazon S3（Simple Storage Service）、各种 Atom 发布协议的变形，以及 Google Maps 等。我们也会举一些流行的但不符合 REST 原则的例子（比如 del.icio.us 的社会性书签 API），然后对它们进行重构。

简单的 Web
The Web Is Simple

为何我们对 Web 如此着迷，以至于认为它能解决所有问题？也许我们上当了，成为炒作的受害者。尽管 HTTP 并不是最受欢迎的 Internet 协议，但 Web 无疑是被炒作得最凶的一种 Internet 技术。据统计，全球 Internet 流量中的大部分源自电子邮件（归因于垃圾邮件）或 BitTorrent（归因于侵犯版权）。假如明天 Internet 就不复存在，那么人们最为怀念的将是电子邮件。那为何要如此重视 Web 呢？是什么使得 HTTP——一种为物理实验室之间传递研究记录而设计的协议——同时能够适合分布式 Internet 应用呢？

实际上，说 HTTP 是为**某某**目的而设计的，对它是一种极大的恭维。已经有人说了，HTTP 与 HTML 是“Internet 协议里的放屁坐垫（Whoopee Cushion）与欢乐蜂鸣器（Joy Buzzer），只能搞些小把戏”（译注 1）——这还是一个**喜爱**它们的人说的。（注 1）第一版 HTTP（即 HTTP 0.9）的确看上去像是搞小把戏，比如下面这个客户端与服务器交互的例子：

客户端请求	服务器响应
`GET /hello.txt`	`Hello, world!`

译注 1：屁坐垫和欢乐蜂鸣器都是廉价的整人玩具。放屁坐垫（Whoopee Cushion）是一种充气坐垫，当人坐上去后会发出放屁似的声音。欢乐蜂鸣器（Joy Buzzer）是一种装置，当佩戴此装置的人与人握手时，它会发出剧烈振动和蜂鸣声。这个比喻的言下之意是说，HTTP 与 HTML 是不起眼的，只能搞些小把戏，做不了正经事。

注 1：Clay Shirky，《In Praise of Evolvable Systems》（*http://www.shirky.com/writings/evolve.html*）。

就这么简单。你连接到服务器，把文档路径给它，然后服务器就把文档内容返回给你。HTTP 0.9 差不多就只支持这么些了，看似只是毫无特色地照搬比它复杂一点的文件传输协议（FTP）。

令人意外的是，答案基本就是这样。我们可以半开玩笑地讲，HTTP 是特别适合分布式 Internet 应用的，因为它没有值得一提的特性。你讲要什么，它就给什么。跟功夫片里的手法同出一辙，（注 2）HTTP 的缺点转化成了优势，它的简单性转化成了强大能力。

HTTP 0.9 是特地设计成那么简单的。从 HTTP 0.9 中我们可以看到**可寻址性**（*addressability*）和**无状态性**（*statelessness*）——正是这两条基本设计原则，令 HTTP 较其同类更加优秀，并得以延伸到今天如此巨大的规模。在 HTTP 0.9 所缺少的特性中，大部分已被证实是多余的，甚至是有副作用的（其实，添加那些特性将有损于 Web），而其余的许多特性已在 HTTP 1.0 和 1.1 版中实现了。Web 赖以成功的另两项重要技术是 URL 和 HTML（以及后来的 XML）——它们在许多重要方面也是简单的。

显然，这些“简单的”技术在功能上已足够强大，它们支持 Web 及 Web 上的应用。在本书中，我们进一步提出：万维网（World Wide Web）是一个简单而灵活的分布式编程环境。而且我们认为其原因是：为人类使用而设计的 human web，跟为软件程序调用而设计的“programmable web”没有本质区别。我们认为，如果 Web 能很好地为人类所用，那么它也同样能很好地为程序所用。我们只要多作一些考虑即可。计算机程序擅长于构建和解析复杂的数据结构，但是在理解文档方面，它们无法做到像人类一样灵活。

复杂的大 Web 服务
Big Web Services Are Not Simple

用作构建 Web 服务的协议与标准有很多，它们大多是构筑在 HTTP 之上的。这些标准被统称为 WS-*标准栈。它们包括 WS-Notification、WS-Security、WSDL 及 SOAP 等。在本书中，我们用“大 Web 服务（Big Web Services）”来称呼这些技术，以较为礼貌地表达我们对它的鄙视。

本书不对这些标准作详细介绍。我们觉得，实现 Web 服务并不一定非得用大 Web 服务（Big Web Services），相反，用 Web 就足够了。我们相信，Web 基础技术足以成为默认的分布式服务平台。

对于某些 WS-*标准（例如 SOAP），你可以做到使用它们，而不违反 REST 及面向资源的架构（Resource-Oriented Architecture）。但实际上，它们被用于实现基于 HTTP 的远程过

注 2：《Legend of The Drunken Protocol》（1991）

程调用（Remote Procedure Call，RPC）。有时，采用 RPC 式架构是合适的；有时，有比 Web 的优点更为重要的需求得考虑。这没关系。

不必要的复杂性是我们所不期望的。有些场合，采用普通老式的 HTTP（plain old HTTP）就足以应付了，但程序员或公司却经常采用大 Web 服务（Big Web Services）。这样做的效果是，HTTP 成为一种用于传输庞大 XML 负载（payload）的协议，而“真正的”描述信息在 XML 里。最终的服务变得太过复杂、难于调试，而且要求客户端必须与服务端具备同样的配置环境。

大 Web 服务（Big Web Services）确实有个优点：现今的开发工具使你只需点一下鼠标，就可以根据代码生成 Web 服务（尤其是 Java 或 C#开发）。在用这些工具来生成符合 WS-* 标准栈的 RPC 式 Web 服务时，REST 式 Web 服务的简单性这一优点对你来说也许已经不重要了，因为这些工具已经隐藏了所有的复杂性，不用让你操心。毕竟带宽和 CPU 都便宜。

假如是在同质（homogeneous）环境中，且服务都在同一个防火墙之后的话，那么这种做法是可行的。如果你的机构有足够的政治影响力，可以要求防火墙外的其他机构也采用跟你们同样的方式。但是，假如期望服务逐渐发展到 Internet 规模，就必须能够应付没有事先考虑到的客户端，比如采用你从没想到的自定义软件栈（software stacks）来访问服务。用户会试图把你的服务与其他你没听说过的服务集成起来。貌似很难？这在 Web 上已经司空见惯了。

抽象（abstraction）也不是完美的。每增加一个层次，就多一些故障点（failure point）、互操作难题和可伸缩性问题。现代化开发工具能够隐藏复杂性，但它们无法为引入复杂性解释原因——它们就知道不断地增加层次。要令服务融入 Web，就要在适应性（adaptability）、可伸缩性（scalability）及可维护性（maintainability）方面多加注意。而简单性（simplicity）——HTTP 0.9 被忽视的优点——则是这三者的先决条件。系统越复杂，出错时纠正起来就越困难。

如果你提供 REST 式 Web 服务，可以把复杂性投入“实现更多特性、实现不同服务之间的交互”中。成功地提供服务，意味着不光把服务构筑在 Web“之上”，而且应该令服务融入 Web：设计服务时，应遵循跟“良好的网站设计所采用的”一样的原则。跟基本的 Web 协议靠得越拢，这就越容易实现。

关于 REST
The Story of the REST

REST 虽然简单，但它是良好定义的（well-defined）；而且，不能因为“Web 服务和网站是一样的东西”，就以简单性作为“把 Web 服务实现成功能贫乏的网站”的借口。可惜到目前为止，主要的 REST 参考文献只有 Roy Fielding 2000 年的博士论文第 5 章——就博士论文来说，那是很好的资料；不过对于大部分现实问题，它并没有给出回答。（注 3）这是因为，那篇博士论文不是把 REST 作为一种架构来诠释，而是把 REST 当作一种评判架

构的方法来介绍的。“REST 式（RESTful）”这个术语就像“面向对象（object-oriented）”这个术语一样。一门语言、一种框架或一个应用也许是采用面向对象的方法设计的，但这并不一定确保其架构是面向对象的架构。

即使采用像 C++或 Ruby 这样面向对象语言，也有可能写出不是真正面向对象的程序出来。按照 REST 的标准，HTTP 在理论上是很好的。（这是理所当然的，因为 Fielding 参与了 HTTP 标准的编写，而且他在博士论文里描述了 Web 的架构。）不过，在现实中，网站、Web 应用及 Web 服务等经常与 REST 原则相违背。那么，在对具体的 Web 服务进行设计时，你怎样才能确信自己正确运用了 REST 原则呢？

大部分关于 REST 的信息来源都是非正式的：邮件列表（mailing list）、Wiki、博客等（我在附录 A 中给出了一些很好的链接）。直至如今，REST 的最佳实践（best practices）还只是一些坊间传言（folklore）。所需要的，是一个以 REST 元架构（meta-architecture）为基础的具体架构（architecture）：一套为设计“体现了 Web 的潜力”的服务而制定的基本准则——我们将在第 4 章介绍这样一种架构，即面向资源的架构（Resource-Oriented Architecture，简称 ROA）。它并不是唯一的高层（high-level）REST 式架构，但是我们认为，用它来设计“易于为客户端所用”的 Web 服务是很好的。

我们通过制定这个面向资源的架构（ROA），把来自坊间传言（folklore）的经验提炼为 Web 服务设计的最佳实践（best practices）。它们仅作为一个建议的基线（suggested baseline）。如果你曾经力图领会 REST，希望我们的架构能够令你自信地认为所做的是“真正的”REST。我们也希望，面向资源的架构（ROA）能够帮助整个社区加快提出并系统化最佳实践我们希望让程序员可以轻易地构建优雅的、符合设计用途的、融入 Web 的（而不是仅仅构筑在 Web 之上的）分布式 Web 应用。

但我们知道，你光掌握这些技术还是不够的。我们两人都曾有过这样的经历：所工作的机构，在主要的架构决策上没有按照我们的意思去做。如果没机会采用 REST 式架构，你是无法在 REST 式架构方面取得成功的。因为，除了技术知识以外，我们还必须教你怎样用言辞去推荐 REST 式解决方案。我们已经把面向资源的架构（ROA）定位为一种替代 RPC 式架构（即如今的 SOAP+WSDL 服务所采用的）的简单方案。RPC 式架构通过一个复杂的、编程语言式的接口，来暴露其内部**算法**（*algorithms*）——这种接口，每个服务都各不相同。而 ROA 则通过一个简单的、文档处理接口，来暴露其内部**数据**（*data*）——这种接口是统一的。我们将在第 10 章对这两种架构加以比较，并教你如何来推荐采用 ROA。

注 3：Fielding, Roy Thomas. *Architectural Styles and the Design of Network-Based Software Architectures*, Doctoral dissertation, University of California, Irvine, 2000 (*http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm*)。**译注**：这篇博士论文已经被国内同行翻译为中文版，可以从这里下载：*http://www.redsaga.com/opendoc/REST_cn.pdf*。

统一两种 Web
Reuniting the Webs

程序员把网站作为 Web 服务来用，这已经有好多年了——当然，是非正式的。（注 4）要让计算机理解供人类阅读的网页，是比较困难的，但这并没有阻止程序高手们用自动化客户端来抓取网页，然后用屏幕抓取程序来提取有关信息。随着时间的推进，出现了 RSS、XML-RPC 和 SOAP 这些对程序员友好的技术——它们用于按正式认可的方式来暴露网站功能。这些技术形成了一种 programmable web，它是对 human web 的扩展，它为软件程序提供了方便。

我们写这本书的最终目的，是统一 programmable web 与 human web。我们设想的是单个互联的网络：一个“在一组服务器上运行，采用一套协议并遵从一组设计原则”的万维网（World Wide Web）。那是一个既可被人也可被计算机程序使用的网络。

若不是得益于分配不当的国防资金、创新的工程项目、“差点则更好”（worse-is-better）的工程实践、大科学、天真的自由理想主义、古怪的自由主义政治、技术迷信，以及那些认为自己找到了发财之路的程序员和投资者的汗水与资本，因特网（Internet）和万维网（World Wide Web）也许不一定会诞生。

Web 作为一个简单的、开放的（至少现阶段）和基本统一的网络应用平台，它容纳了大量的人类知识，并为人类所努力钻研的许多领域提供支持。我们认为，现在应该开始认真考虑将网站设计原则应用于服务设计，以便自动化客户端可以访问那些信息和算法。假如同意的话，本书将教你怎么做。

本书的内容
What's in This Book?

在本书中，我们主要关注一些实际问题：如何设计与实现 REST 式 Web 服务（RESTful Web Services），以及调用它们的客户端。其次我们还关注于理论方面：什么是 REST 式架构风格？为什么 Web 服务应该尽量符合 REST 风格？我们的讨论不会涵盖方方面面，但是力图切入如今的重点话题。因为这是同类书中的第一本，我们会不断反复这一关键问题——即如何设计一个 REST 式服务。

前 3 章，将从客户端的角度来介绍 Web 服务，并告诉你 REST 式服务有什么特别不同。

注 4：一个早期的例子，请参见 Jon Udell 于 1996 年发表在《Byte》杂志上的文章《On-Line Componentware》（*http://www.byte.com/art/9611/sec9/art1.htm*）。注：“A powerful capability for ad hoc distributed computing arises naturally from the architecture of the Web.”——大家注意了，这句话 1996 年就讲了。

第 1 章　Programmable Web 及其分类

我们在这一章对 Web 服务作总体性介绍。Web 服务运行于 Web 上，请求外部服务器提供数据或运行算法的程序。我们将展示三种常见的 Web 服务架构：REST 式架构、RPC 式架构及 REST-RPC 混合架构。我们会为每一种架构举一个 HTTP 请求/响应的例子，并给出典型的客户端代码。

第 2 章　编写 Web 服务客户端

在这一章，我们教你如何用 HTTP 库和 XML 解析器来为现有 Web 服务编写客户端。我们将介绍一个流行的 REST-RPC 服务——del.icio.us Web 服务，并用 Ruby、Python、Java、C#及 PHP 等语言来示范其客户端代码。对于其他一些语言，我们虽然没有展示代码，但会推荐一些适用的 HTTP 库和 XML 解析器。JavaScript 和 Ajax 将在第 11 章中作单独介绍。

第 3 章　REST 式服务有什么特别不同？

我们将吸取第 2 章的经验，并把这些经验运用于一个纯 REST 式服务——Amazon S3（Simple Storage Service）。我们将在构建 S3 客户端时举例说明一些重要的 REST 概念：资源（resource）、表示（representation）、统一接口（uniform interface）。

下面 6 章是本书的核心，它们关注于如何设计和实现自己的 REST 式服务。

第 4 章　面向资源的架构

本章正式介绍 REST：不是抽象地介绍，而是以一个具体的 Web 服务架构为背景进行介绍。我们的架构基于四个重要的 REST 概念：资源、资源名称、资源表示（representation），以及资源间的链接。它的服务应根据四个 REST 特征来评判：可寻址性、无状态性、连通性和统一接口。

第 5 章　设计只读的面向资源的服务

我们提出了“把想法或需求转变为 REST 式资源”的系列步骤。这些资源是只读的，也就是说客户端可以从服务获取数据，但是不能改变服务的数据。我们以一个设计地图服务（类似 Google Maps）的例子来说明这些步骤。

第 6 章　设计可读写的面向资源的服务

我们扩展了上一章的步骤，以允许客户端创建、修改并删除资源。我们以为地图服务新增两种资源（用户账户和自定义地点）为例，示范这些步骤。

第 7 章　一个服务实现

我们把一个 RPC 式服务（即第 2 章那个 del.icio.us REST-RPC 混合服务）重构为一个纯 REST 式服务。然后，我们把这个服务实现为一个 Ruby on Rails 应用。通过这一章，你可以把所学到的知识操练一遍！

此为试读，需要完整PDF请访问：www.ertongbook.com

第 8 章　REST 和 ROA 最佳实践

在这一章，我们把之前给出的关于服务设计的建议汇集起来，并补充一些新的建议。我们将告诉你如何利用 HTTP 的标准特性来解决常见问题和进行优化。我们还将为棘手的特性（比如事务，也许你认为这没法用 REST 式 Web 服务来实现）给出面向资源的设计。

第 9 章　服务的技术构件

我们在这一章描述在 REST 的三大技术（HTTP、URI 和 XML）之上的其他技术：有些是用于传递状态的文档格式，比如 XHTML 及其微格式（microformat）；有些是用于为客户端状态推进的超媒体格式，比如 WADL；有些是用于构建 REST 式 Web 服务的控制流，比如 Atom 发布协议。

最后 3 章涉及一些专门话题，这些专题各自都可单独作为一本书来讨论。

第 10 章　面向资源的架构 VS 大 Web 服务

将拿我们的架构及一般的 REST 架构跟其他著名架构来进行比较。我们认为 REST 式 Web 服务较“基于 SOAP、WSDL 和 WS-*”的服务而言，更为简单、可伸缩性更好、更易于使用、更加符合 Web 的理念，且更能应付各种各样的客户端。

第 11 章　将 Ajax 应用作为 REST 客户端

在这一章，我们将以 Web 服务来阐述用于 Web 应用的 Ajax 架构：一个 Ajax 应用，就是一个在浏览器里运行的 Web 服务客户端。本章是对第 2 章内容的延伸。我们教你如何用 `XMLHttpRequest` 和标准的 JavaScript 库来编写用于 REST 式 Web 服务的客户端。

第 12 章　REST 式服务框架

在最后这一章里，我们将讨论三种流行的框架：Ruby on Rails、Restlet（用于 Java）和 Django（用于 Python）。它们可以简化 REST 式 Web 服务的开发。

我们还有 3 个附录，希望对你有用。

附录 A　REST 相关资源与 REST 式资源

第一部分列出了一些与 REST 式 Web 服务有关的标准、教程和社区。第二部分给出了一些现有的、公开的、可供你调用和学习的 REST 式 Web 服务。

附录 B　42 种常见的 HTTP 响应代码

这部分解释标准 HTTP 的每一个响应代码（以及一个扩展），并解释何时你会在 REST 式 Web 服务里用到这些代码。

附录 C　常见的 HTTP 报头

这部分向你解释各种 HTTP 报头，包括每个标准的 HTTP 报头，以及一些用于 Web 服务的扩展报头。

你应该阅读哪些部分？

本书的组织，适于那些对 Web 服务的概况感兴趣的读者：有些读者通过实践学习 Web 服务，但他们对于 Web 服务没有太多经验。如果你属于这种情况，可以采取一条最简化的阅读路线，即从本书开头一直读到第 9 章，然后在剩下的章节中选择你所感兴趣的部分。

如果你经验比较丰富，那么可以采取另一种阅读路线。如果你关心如何为现有服务编写客户端，那么可以只关注第 1、2、3 和 11 章——那些关于服务设计的章节对你可能帮助不大。如果你想要创建自己的 Web 服务，或者想知道 REST 到底是什么意思，你不妨从第 3 章开始阅读。如果你想比较 REST 与 WS-*，那么你可以选第 1、3、4 和 10 章作为起点。

说明
Administrative Notes

本书有两位作者（Leonard 和 Sam），在本书的 1-11 章，我们将把这两个身份合并为第一人称“我”。而在最后一章（第 12 章）里，这个第一人称“我”代表的是一大群人，因为有众多 Django 和 Restlet 开发者们参与到“展示如何用他们的框架来构建 REST 式服务”中来。

我们假定你是一名胜任的程序员，但不假定你在 Web 编程方面有任何经验。我们在书中介绍的内容并不限定于某一编程语言。我们提供了以各种语言编写的 REST 式客户端及服务的示例代码。除非为了示范特定的框架或语言，否则我们将采用 Ruby（*http://www.ruby-lang.org*）语言来描述。

我们选用 Ruby 是因为：即使对于不懂 Ruby 的程序员来说，它也是简洁易读的。（因为这是一门很好的语言，而且它的名字跟 Sam 的姓有着令人困惑的联系。）Ruby 的标准 Web 框架——Ruby on Rails——也是最主要的一种 REST 式 Web 服务实现平台之一。如果你不懂 Ruby 也没关系，我们在代码中嵌入了许多有助于理解 Ruby 语法的注释。

本书的示例程序可以从本书官方网站（*http://www.oreilly.com/catalog/9780596529260*）上下载，其中包括第 7 章的 Rails 应用的完整代码，以及第 12 章中 Restlet 和 Django 应用的相应代码。在书中，许多客户端我们只给出了 Ruby 实现；但在示例程序中，你还可以找到它们的 Java 实现。这些客户端程序采用了 Restlet 库，并且是由 Restlet 开发者 Jerome Louvel 和 Dave Pawson 编写的。若相对 Ruby 而言，你更熟悉 Java，那么这些代码对你掌握代码背后的概念会有帮助。特别值得注意的是，示例代码里还包括一个完整的第 3 章 Amazon S3 客户端的 Java 实现。

一些约定
Conventions Used in This Book

书中字体将采用以下约定：

Italic（斜体）

表示新术语、URI、Email 地址、文件名及文件扩展名。

`Constant width`（等宽字体）

用于程序代码，以及变量、函数名、数据库、数据类型、环境变量、声明和关键字等。

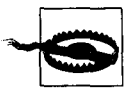
`Constant width bold`（等宽粗体）

用于命令行及其他应由用户逐字输入的文本。

`Constant width bold`（等宽斜体）

用于应替换为用户提供或由上下文决定的值的文本。

这个图标表示一则技巧、提示或普通的注解。



这个图标表示一则警告或注意。

联系我们

How to Contacts Us

我们已尽力核验本书所提供的信息，尽管如此，仍不能保证本书完全没有瑕疵，而网络世界的变化之快，也使得本书永不过时的保证成为不可能。如果读者发现本书内容上的错误，不管是赘字、错字、语意不清，甚至是技术错误，我们都竭诚虚心接受读者指教。如果您有任何问题，请按照以下方式与我们联系。

奥莱理软件（北京）有限公司

北京市　西城区　西直门南大街 2 号　成铭大厦 C 座 807
邮政编码：100035
网页：*http://www.oreilly.com.cn*
E-mail：*info@mail.oreilly.com.cn*

与本书有关的在线信息如下所示。

http://www.oreilly.com/catalog/9780596529260（原书）
http://www.oreilly.com.cn/book.php?bn=978-7-121-06227-8（中文版）

致谢
Acknowledgments

最后，我们要感谢那些令我们得以直接在 HTTP 上编程的人们。对于 Sam 来说，是 Rael Dornfest 和他的 Blosxom 博客应用。Leonard 的经验来自于 20 世纪 90 年代中期编写屏幕抓取应用。他感谢那些把网站作为 Web 服务来设计的人们，特别是那位在线连环画“Pokey the Penguin”的匿名作者。

我们曾有过这样的想法，不过是 Roy Fielding 充实了它。对于我们，那只是一种感觉，而 Roy Fielding 则在他的博士论文里命名并定义了它。Roy 的理论基础正是我们所赖以为基础的。

在本书编写过程中，我们从 REST 社区获得了巨大的帮助。我们非常感激给我们反馈的人：Benjamin Carlyle、David Gourley、Joe Gregorio、Marc Hadley、Chuck Hinson、Pete Lacey、Larry Liberto、Benjamin Pollack、Aron Roberts、Richard Walker 和 Yohei Yamamoto。还有一些人通过他们的作品在不知不觉中帮助我们：Mark Baker、Tim Berners-Lee、Alex Bunardzic、Duncan Cragg、David Heinemeier Hansson、Ian Hickson、Mark Nottingham、Koranteng Ofosu-Amaah、Uche Ogbuji、Mark Pilgrim、Paul Prescod、Clay Shirky、Brian Totty 和 Jon Udell。当然，书中的所有看法及可能存在的错误与疏漏都是我们自己的责任。

在本书编写过程中，编辑 Michael Loukides 为我们提供了很多帮助与智慧。我们还要感谢 Laurel Ruma 和其他所有为本书出版付出劳动的 O'Reilly 同事们。

最后，需要特别感谢的是 Jerome Louvel、Dave Pawson 和 Jacob Kaplan-Moss。他们在 Restlet 和 Django 方面的经验令我们有了本书的第 12 章。

O'Reilly Media, Inc.介绍

为了满足读者对网络和软件技术知识的迫切需求，世界著名计算机图书出版机构O'Reilly Media, Inc. 授权电子工业出版社，翻译出版一批该公司久负盛名的英文经典技术专著。

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，同时也是在线出版的先锋。

从最畅销的《The Whole Internet User's Guide & Catalog》(被纽约公共图书馆评为20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。

此为试读，需要完整PDF请访问：www.ertongbook.com