Services for a Changing World

RESTful Web APIs

Leonard Richardson &
Mike Amundsen
Foreword by Sam Ruby

O'REILLY®

RESTful Web APIs

The popularity of REST in recent years has led to tremendous growth in almost RESTful APIs that miss out on many of the architecture's benefits. With this practical guide, you'll learn what it takes to design usable REST APIs that evolve over time. By focusing on solutions that cross a variety of domains, this book shows you how to create powerful and secure applications, using the tools designed for the world's most successful distributed computing system: the World Wide Web.

You'll explore the concepts behind REST, learn different strategies for creating hypermedia-based APIs, and then put everything together with a step-by-step guide to designing a RESTful web API.

- Examine API design strategies, including the collection pattern and pure hypermedia
- Understand how hypermedia ties representations together into a coherent API
- Discover how XMDP and ALPS profile formats can help you meet the web API "semantic challenge"
- Learn close to two-dozen standardized hypermedia data formats
- Apply best practices for using HTTP in API implementations
- Create web APIs with the JSON-LD standard and other Linked Data approaches
- Understand the CoAP protocol for using REST in embedded systems

"A terrific book! RESTful Web APIs covers the most important trends and practices in APIs today."

—**John Musser**
founder of ProgrammableWeb

Leonard Richardson, author of O'Reilly's *Ruby Cookbook*, has created several open source libraries, including Beautiful Soup.

Mike Amundsen has more than a dozen books to his credit, including *Building Hypermedia APIs with HTML5 and Node* (O'Reilly).

Sam Ruby is a co-chair of the W3C HTML Working Group and a Senior Technical Staff Member in the Emerging Technologies Group of IBM.

US $34.99 CAN $36.99
ISBN: 978-1-449-35806-8

twitter: goreillymedia
facebook.com/oreilly

O'REILLY®
oreilly.com

Praise for *RESTful Web APIs*

"This book is the best place to start learning the essential craft of API Design."

—Matt McLarty
Cofounder, API Academy

"The entire time I read this book, I was cursing. I was cursing because as I read each explanation, I was worried that they were so good that it would be hard to find a better one to use in my own writing. You will not find another work that explores the topic so thoroughly yet explains the topic so clearly. Please, take these tools, build something fantastic, and share it with the rest of the world, okay?"

—Steve Klabnik
Author, *Designing Hypermedia APIs*

"Wonderfully thorough treatment of hypermedia formats, REST's least well understood tenet."

—Stefan Tilkov
REST evangelist, author, and consultant

"The best practical guide to hypermedia APIs. A must-have."

— Ruben Verborgh
Semantic hypermedia researcher

RESTful Web APIs

Leonard Richardson and Mike Amundsen
Foreword by Sam Ruby

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

RESTful Web APIs

by Leonard Richardson and Mike Amundsen with a Foreword by Sam Ruby

Copyright © 2013 Leonard Richardson, amundsen.com, Inc., and Sam Ruby. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editors: Simon St. Laurent and Meghan Blanchette
Production Editor: Christopher Hearse
Copyeditor: Jasmine Kwityn
Proofreader: Linley Dolby
Indexer: Judith McConville
Cover Designer: Randy Comer
Interior Designer: David Futato
Illustrator: Rebecca Demarest

September 2013: First Edition

Revision History for the First Edition:

2013-09-10: First release

See *http://oreilly.com/catalog/errata.csp?isbn=9781449358068* for release details.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. *RESTful Web APIs*, the image of Hoffmann's two-toed sloth, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc., was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 978-1-449-35806-8

[LSI]

For Sienna, Dalton, and Maggie. —Leonard

For Milo "The Supervisor," my constant and patient companion throughout this and so many other projects. Thanks, buddy! —Mike

Table of Contents

Foreword

Progressive Disclosure is a concept in User Interface Design which advocates only presenting to the user the information they need when they need it. In many ways, the book you are reading right now is an example of this principle. In fact, it is quite likely that this book wouldn't have "worked" a mere seven years ago.

For you see, the programming world was quite a different place when *RESTful Web Services*, the predecessor of this book, was written. At that time, the term "REST" was was rarely used. And when it was used it was often misapplied, and widely misunderstood.

This was the case despite the fact that the standards upon which REST is based, namely HTTP and HTML, were developed and became IETF and W3C standards in roughly their current form in the second half of the 1990s. Roy Fielding's thesis paper in which he introduced the term REST and on which this book was based was itself published in 2000.

Leonard Richardson and I set out to correct this injustice. To do this, we focused primarily on the concepts underpinning HTTP, and we provided practical guidance on how to apply those concepts to applications.

I'd like to think that we helped kick a few pebbles loose that started the avalanche of support for REST that came forth since that time. REST rapidly took on a life of its own, and in the process has become a buzzword. In fact it now is pretty much the case that presenting a web interface and calling it REST is practically the default. We've definitely come a long way in a few short years.

Admittedly, REST as a term is often over applied, and not always correctly. But all things considered, I am very pleased that the concepts of resources and URIs have successfully managed to infiltrate their way into application interface design. The web, after all, is a resilient place, and these new interfaces, albeit imperfect, are leaps and bounds better than the ones that they replace.

But we can do better.

Now that those building blocks are in place, it is time to take a step back, survey the territory, and build on top of these concepts. The next logical step is to explore media types in general, and hypermedia formats in specific. While the first book focused almost exclusively on the correct application of HTTP, it is time to delve more deeply into the concepts behind hypertext media types like HTML—media types that aren't tightly bound to a single application or even a single vendor.

HTML remains a prime example of a such a hypermedia format, and it continues to hold a special place in web architecture. In fact, my personal journey of discovery has been to take a deep dive into development of the W3C standard for HTML, now branded as HTML5. And while HTML does have a prominent place in this new book, there is so much more to cover on the topic of hypermedia. So while I have remained in touch, Leonard picked up a capable replacement for my role as coauthor in Mike Amundsen.

It has been a pleasure to watch this book be written, and in reading this book I've learned about a number of media types that I had not been exposed to by any other source. More importantly, this book shows what these types have in common, and how to differentiate them, as each has its own specialty.

Hopefully the pebbles that this book kicks loose will have the same effect as its predecessor did. Who knows, perhaps in another seven years it will be time to do this all over again, and highlight some other facet of Representational State Transfer that continues to be under-appreciated.

—Sam Ruby

Introduction

> "Most software systems are created with the implicit assumption that the entire system is under the control of one entity, or at least that all entities participating within a system are acting towards a common goal and not at cross-purposes. Such an assumption cannot be safely made when the system runs openly on the Internet."
>
> — Roy Fielding
> *Architectural Styles and the Design of Network-based Software Architectures*

> "A Discordian Shall Always use the Official Discordian Document Numbering System."
>
> — Malaclypse the Younger and Lord Omar Khayyam Ravenhurst
> *Principia Discordia*

I'm going to show you a better way to do distributed computing, using the ideas underlying the most successful distributed system in history: the World Wide Web. I hope you'll read this book if you've decided (or your manager has decided) that your company needs to publish a web API. It doesn't matter whether you're planning a public API, a purely internal API, or an API accessible by trusted partners—they can all benefit from the philosophy of REST.

This is not necessarily the book for you if you want to learn how to write API *clients*. That's because most existing API designs are based on assumptions that are several years old, assumptions that I'd like to destroy.

Most of today's APIs have a big problem: once deployed, they can't change. There are big-name APIs that stay static for years at a time, as the industry changes around them, because changing them would be too difficult.

But RESTful architectures are designed for managing change. The World Wide Web is made of millions of websites, running atop thousands of different server implementations, and undergoing periodic redesigns. Websites are accessed by billions of users who are using hundreds of different client implementations on dozens of hardware plat-

forms. Your deployment won't look like this howling mess, but the closer you come to web scale, the more familiar this picture will look.

A very simple system is always easy to change. At small scales, a RESTful system has a larger up-front design cost than a push-button solution. But as your API matures and starts to change, you'll really need some way—like REST—of adapting to change.

- An API that's commercially successful will stay available for years on end. Some APIs have hundreds or even thousands of users. Even if the problem domain only changes occasionally, the cumulative effect on clients can be huge.
- Some APIs change all the time, with new data elements and business rules constantly being added.
- In some APIs, each client can change the workflow to suit its needs. Even if the API itself never changes, each client will experience it differently.
- The people who write the API clients usually don't work on the same team as the people who write the servers. All APIs that are open to the public fall under this category. If you don't know what kind of clients are out there, you need to be very careful about making changes—or you need to have a design that can change without breaking all the clients.

If you copy existing designs for your API, you will probably only repeat the mistakes of the past. Unfortunately, most of the improvements are happening below the surface, in experiments and through slow-moving standards processes. I'll cover dozens of specific technologies in this book, including many that are still under development. But my main goal is to teach you the underlying principles of REST. Learn those, and you'll be able to exploit whichever experiments pan out and whichever standards are approved.

There are two specific problems I'm trying to solve with this book: duplication of effort and avoidance of hypermedia. Let's take a look at them.

Duplication of Effort

An API released today will be named after the company that hosts it. We talk about the "Twitter API," the "Facebook API," and the "Google+ API." These three APIs do similar things. They all have some notion of user accounts and (among other things) they all let users post a little bit of text to their accounts. But each API has a completely different design. Learning one API doesn't help you learn the next one.

Of course, Twitter, Facebook, and Google are big companies that compete with each other. They don't *want* to make it easy for you to learn their competitors' APIs. But small companies and nonprofits do the same thing. They design their APIs as though nobody else had ever had a similar idea. This interferes with their goal of getting people to actually use their APIs.

Let me show you just one example. The website ProgrammableWeb has a directory of over 8,000 APIs. As I write this, it knows about 57 microblogging APIs—APIs whose main purpose is posting a little bit of text to a user account.[1] It's great that there are 57 companies publishing APIs in this field, but do we really need 57 different *designs*? We're not talking about something complicated here, like insurance policies or regulatory compliance. We're talking about posting a little bit of text to a user account. Do you want to be the one who designs the 58th microblogging API?

The obvious solution would be to create a standard for microblogging APIs. But there already *is* a standard that would work just fine: the Atom Publishing Protocol. It was published in 2005, and almost nobody uses it. There's something about APIs that makes everyone want to design their own from scratch, even when that makes no sense from a business perspective.

I don't think I can single-handledly stop this wasted effort, but I do think I can break down the problem into parts that make sense, and present some ways for a new API to reuse work that's already been done.

Hypermedia Is Hard

Back in 2007, Leonard Richardson and Sam Ruby wrote the predecessor to this book, *RESTful Web Services* (O'Reilly). That book also tried to address two big problems. One of the problems has been solved; the other is nowhere close to being solved.[2]

The first problem: in 2007, the REST school of API design was engaged in a standoff against a rival school that used heavyweight technologies based on SOAP and questioned the very legitimacy of the REST school. *RESTful Web Services* was a salvo in this standoff, a defense of RESTful design principles against the attacks of the SOAP school.

Well, the standoff is over, and REST won. SOAP APIs are still used, but only within the big companies that were backing the SOAP school in the first place. Pretty much all new public-facing APIs pay lip service to RESTful principles.[3]

Which brings me to the second problem: REST isn't just a technical term—it's also a marketing buzzword. For a long time, REST was a slogan that signified nothing beyond opposition to the SOAP school. Any API that didn't use SOAP was marketed as REST,

1. The full list of ProgrammableWeb APIs tagged with microblogging provides information about each of these APIs.
2. *RESTful Web Services* is now freely available as part of O'Reilly's Open Books Project. You can download a PDF copy of the book from the book's page.
3. If you're wondering, this is why we changed the title. The term "web services" became so tightly coupled with SOAP that when SOAP went down, it took "web services" with it. These days, everyone talks about APIs instead.

even if its design made no sense or betrayed the technical principles of REST. This was inaccurate, confusing, and it gave REST—i.e., REST as a technical term—a bad name.

This situation has improved a lot since 2007. When I look at new APIs, I see the work of developers who understand the concepts I'll be explaining in the first few chapters of this book. Most developers who fly the REST flag today understand resources and representations, how to name resources with URLs, and how to properly use HTTP methods. The first three chapters of this book don't do much but get new developers up to speed.

But there's one aspect of REST that most developers still don't understand: hypermedia. We all understand hypermedia in the context of the Web. It's just a fancy word for links. Web pages link to each other, and the result is the World Wide Web, driven by hypermedia. But it seems we've got a mental block when it comes to hypermedia in web APIs. This is a big problem, because hypermedia is the feature that makes a web API capable of handling changes gracefully.

Starting in Chapter 4, my overriding goal for *RESTful Web APIs* will be to teach you how hypermedia works. If you've never heard of this term, I'll teach it to you along with the other important REST concepts. If you've heard of hypermedia but the concept intimidates you, I'll do what I can to build up your courage. If you just haven't been able to wrap your head around hypermedia, I'll show it to you in every way I can think of, until you get it.

RESTful Web Services covered hypermedia, but it wasn't central to the book. It was possible to skip the hypermedia parts of the book and still design a functioning API. By contrast, *RESTful Web APIs* is effectively a book about hypermedia.

I did it this way because hypermedia is the single most important aspect of REST, and the least understood. Until we all understand hypermedia, REST will continue to be viewed as a marketing buzzword rather than a serious attempt to handle the complexity of distributed computing.

What's in This Book?

The first four chapters introduce the concepts behind REST, as it applies to web APIs.

Chapter 1, Surfing the Web
: This chapter explains basic terminology using a RESTful system you're already familiar with: a website.

Chapter 2, A Simple API
: This chapter translates the lessons of the Web to a programmable API with identical functionality to the website discussed in Chapter 1.

Chapter 3, Resources and Representations
: Resources are the fundamental concept underlying HTTP, and representations are the fundamental concept underlying REST. This chapter explains how they're related.

Chapter 4, Hypermedia
: Hypermedia is the missing ingredient that ties representations together into a coherent API. This chapter shows what hypermedia is capable of, mostly using a hypermedia data format you're already familiar with: HTML.

The next four chapters describe different strategies for designing a hypermedia API:

Chapter 5, Domain-Specific Designs
: The obvious strategy is to design a completely new standard that deals with your exact problem. I use the Maze+XML standard as an example.

Chapter 6, The Collection Pattern
: One pattern in particular—the collection pattern—shows up over and over again in API design. In this chapter, I show off two different standards that capture this pattern: Collection+JSON and AtomPub.

Chapter 7, Pure-Hypermedia Designs
: When the collection pattern doesn't fit your requirements, you can convey any representation you want using a general-purpose hypermedia format. This chapter shows how it works using three general hypermedia formats (HTML, HAL, and Siren) as examples. This chapter also introduces HTML microformats and microdata, which lead in to the next chapter.

Chapter 8, Profiles
: A profile fills in the gaps between a data format (which can be used by many different APIs) and a specific API implementation. The profile format I recommend is ALPS, but I also cover XMDP and JSON-LD.

  In this chapter, my advice begins to outstrip the state of the art at the time this book was written. I had to develop the ALPS format for this book, because nothing else would do the job. If you're already familiar with hypermedia-based designs, you might be able to skip up to Chapter 8, but I don't think you should skip past it.

Chapters 9 through 13 cover practical topics like choosing the right hypermedia format and getting the most out of the HTTP protocol.

Chapter 9, The Design Procedure
: This chapter brings together everything discussed in the book so far, and gives a step-by-step guide to designing a RESTful API.

Chapter 10, The Hypermedia Zoo
: In an attempt to show what hypermedia is capable of, this chapter discusses about 20 standardized hypermedia data formats, most of them not covered elsewhere in the book.

Chapter 11, HTTP for APIs
: This chapter gives some best practices for the use of HTTP in API implementations. I also discuss some extensions to HTTP, including the forthcoming HTTP 2.0 protocol.

Chapter 12, Resource Description and Linked Data
: Linked Data is the Semantic Web community's approach to REST. JSON-LD is arguably the most important Linked Data standard. It's covered briefly in Chapter 8, and I revisit it here. This chapter also covers the RDF data model, and some RDF-based hypermedia formats that I didn't get to in Chapter 10.

Chapter 13, CoAP: REST for Embedded Systems
: This chapter closes out the core body of the book by covering CoAP, a RESTful protocol that doesn't use HTTP at all.

Appendix A, The Status Codex
: An extension of Chapter 11, this appendix provides an in-depth look at the 41 standard status codes defined in the HTTP specification, as well as a few useful codes defined as extensions.

Appendix B, The Header Codex
: Similar to Appendix A, this appendix is also an extension of Chapter 11. It provides a detailed outline of the 46 request and response headers defined in the HTTP specification, as well as a few extensions.

Appendix C, An API Designer's Guide to the Fielding Dissertation
: This appendix includes an in-depth discussion of the foundational document of REST, in terms of what it means for API design.

Glossary
: The glossary contains definitions to terms you'll frequently encounter when working with RESTful web APIs. It's a good place to turn for familiarizing yourself with basic concepts or if you need a quick, at-a-glance reminder of a particular concept's definition.

What's Not in This Book

RESTful Web Services was the first book-length treatment of REST, and it had to cover a lot of ground. Fortunately, there are now over a dozen books on various aspects of REST, and that frees up *RESTful Web APIs* to focus on the core concepts.

sample content of RESTful Web APIs

- Dragonsong here
- *download Princess on the Brink (The Princess Diaries, Book 6)*
- *download When Cultures Collide: Leading Across Cultures (3rd Edition)*
- read online Angel Time (The Songs of the Seraphim, Book 1) online

- http://aircon.servicessingaporecompany.com/?lib/Kort-om-yttrandefrihet.pdf
- http://nautickim.es/books/Peace-Is-Every-Step--The-Path-of-Mindfulness-in-Everyday-Life.pdf
- http://test1.batsinbelfries.com/ebooks/The-Machine-Awakes--Spider-War--Book-2-.pdf
- http://www.freightunlocked.co.uk/lib/Noir-Anxiety.pdf