服务计算技术丛书

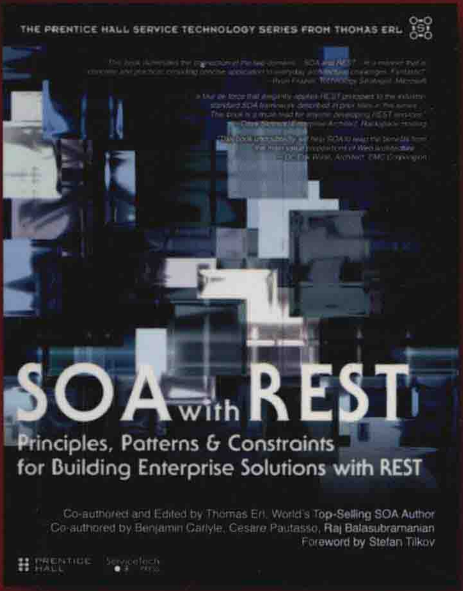
PEARSON

基于REST的SOA技术

——构建企业级方案的原则、模式和约束

（英文版）

〔美〕 Thomas Erl　Benjamin Carlyle
Cesare Pautasso　Raj Balasubramanian 著


THE PRENTICE HALL SERVICE TECHNOLOGY SERIES FROM THOMAS ERL

[illegible]

SOA with REST

Principles, Patterns & Constraints for Building Enterprise Solutions with REST

Co-authored and Edited by Thomas Erl, World's Top-Selling SOA Author
Co-authored by Benjamin Carlyle, Cesare Pautasso, Raj Balasubramanian
Foreword by Stefan Tilkov

PRENTICE HALL　ServiceTech PRESS

SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST

科学出版社

服务计算技术丛书

基于 REST 的 SOA 技术
——构建企业级方案的原则、模式和约束

（英文版）

SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST

〔美〕 Thomas Erl　Benjamin Carlyle　Cesare Pautasso　Raj Balasubramanian 著

科学出版社
北京

图字：01-2013-1326

内 容 简 介

本书是关于设计和构建面向服务解决方案中 RESTful 服务的综合性指南，并且全面介绍了 REST 与 SOA 的关系。本书不仅阐明了 REST 是构建真实面向服务解决方案的合适工具，还说明了面向服务架构模型是 REST 技术架构实现商业价值的必要基础。本书提供了 REST 的约束条件、架构目标与面向服务原则、SOA 规格参数之间的完全映射。本书通过真实的案例，说明在不用折中考虑面向服务解决方案和架构的功率或易管理性的情况下，如何改善 REST 的简洁性、灵活性以及降低费用。

本书可供 IT 架构师、程序开发人员以及对 SOA 和 REST 感兴趣的技术人员参考学习。

图书在版编目(CIP)数据

基于 REST 的 SOA 技术：构建企业级方案的原则、模式和约束 = SOA with REST : Principles, Patterns & Constraints for Building Enterprise Solutions with REST : 英文/ (美)艾尔 (Erl,T.)等著. —北京 : 科学出版社, 2013
（服务计算技术丛书）
ISBN 978-7-03-037225-3
I. ①基… II. ①艾… III. ①互联网络-网络服务器-研究-英文 IV. ①TP368.5
中国版本图书馆 CIP 数据核字(2013)第 055998 号

责任编辑：朱雪玲 / 责任印制：张　倩
封面设计：迷底书装 / 责任校对：朱光兰

科学出版社出版
北京东黄城根北街16号
邮政编码：100717
http://www.sciencep.com
新科印刷有限公司印刷
科学出版社发行 各地新华书店经销
*
2013 年 4 月第 一 版　开本：B5(720×1000)
2013 年 4 月第一次印刷　印张：37 1/2
字数：600 000
定价：120.00 元

Foreword by Stefan Tilkov

When I first heard about REST in early 2002, I was a strong believer in the value of the emerging Web services specifications and standards. I was initially intrigued by the approach, particularly the "uniform interface" idea, but quickly concluded that while REST might be interesting, it was most certainly not applicable in enterprise use cases.

A year or two later, I had started to appreciate the elegance and simplicity of the REST style, and became convinced that in some cases, it was a better choice, while more advanced use cases still required SOAP and WSDL and WS-*. Another year later, I found myself recommending RESTful HTTP over SOAP-style Web services in most situations, and decided that I could call myself a "RESTafarian" by conviction. I had become convinced—and still am—that adhering to the constraints of the REST architectural style not only leads to better systems on the public Web, but also within all kinds of enterprise-internal scenarios.

Today, REST has become mainstream—with all the positive and negative effects this has for any technology. It is now relatively easy to introduce a RESTful approach, even in large enterprises, without raising too many eyebrows. I am still surprised how often it is even viewed as the obvious default choice.

So maybe the REST community should be happy and satisfied, and look forward to all the excellent system development and integration work we will do, both on the public Web and within companies. But there are two problems with this: Not everything that claims to be RESTful actually is so, and often SOA is perceived as the architecture of WS-* style Web services and therefore seen as incompatible with REST.

The Misuse of the "REST" Moniker

Following REST principles when applying Web technologies, such as HTTP and URIs, and using hypermedia formats, such as HTML, leads to systems that are evolvable, dynamic yet stable, and can be connected in new and unforeseen ways. Building or integrating systems in a way that actually deserves being called "RESTful" requires a different kind of design than what most developers have been used to in the past. Choosing REST instead of a Distributed Objects, RPC, or WS-* approach requires expressing an interface's domain using resources, media types, hypermedia, and the uniform interface. The better your understanding of these concepts, the easier you will find it to adhere to the implied constraints, and the more you will be able to exploit REST's benefits.

Due to the fundamental design decision of "transport independence" in the WDSL/SOAP/WS-* architecture, systems built using it can never be RESTful, and very few people would disagree. But it is just as easy to violate the Web's underlying ideas without ever coming near XML or SOAP, and the Internet is filled with examples of this: Tunneling method calls through HTTP GET or POST, ignoring caching and hypermedia, or treating the characters that make up a URI as a development API. Using JSON, HTTP, and "pretty" URIs does not mean a system is RESTful. Don't assume everything called a "REST API" can serve as a role model.

On the Merits of SOA

The label "SOA" has been thoroughly burned, and in my view, this is mostly due to the fact that many made, and continue to make, a strong association between the abstract, high-level concept of SOA and the Web services technology that initially became a popular way to support it. The many discussions you can find on the Web that talk about "REST vs. SOA" enforce this with their apples to oranges comparison. So let us take a step back and look at the original motivation for SOA.

In practically any company, many individual systems make up the complete IT landscape. They run on different technologies, and they have been built using different tools. Some come from commercial vendors, some may have been built in-house. After a short while, you will want to connect them because there very obviously is value in doing so. You can do this in a point-to-point way, exporting some data here, periodically importing it over there, via files, shared databases, or individual integration solutions. This will lead to a brittle, unmanageable landscape.

Because it is so hard to integrate systems, you (and your vendor) will opt for making them fatter and fatter, unless you have to invest an absurd amount of effort to change them. Introducing a centralized integration middleware bottleneck in the middle is not a good solution, although it might seem tempting at first: You become dependent on the single vendor, who will get bought by somebody else, go out of business, or become legacy after you merge with your competitor who has bought another product of this type more recently.

So what do you do instead? You treat the problem in a reasonable way—by reducing the amount of effort for integrating individual systems by limiting the amount of different technologies used at the interface layer, by picking a good interface abstraction, and by modularizing the big applications into smaller chunks. You take care not to depend on a particular vendor, be it a middleware or an applications vendor; you want to make it easy to (re)use stuff. Few will disagree that this is a reasonable approach.

That is, to me, the essence of SOA: A software architecture that is not applied to an individual system, but to a set of systems within a company; focusing on network/wire interfaces, not implementation; standardizing whatever is necessary to ensure smooth communication and serendipitous re-use, but nothing more. When we authored the SOA Manifesto several years ago, it was the emphasis we placed on achieving intrinsic interoperability that embodied, for me, what SOA and service-orientation were truly about, and it was also what aligned it with REST principles.

In my view service-orientation is very much worth pursuing, and RESTful HTTP is the best way we currently have available to do so. No other architecture is as widely supported, enables reuse with similar ease, and has the same kind of support for continued evolution. Yet the use of REST in SOA scenarios is still in its infancy, and best practices and patterns have, so far, been hard to find.

The Contribution of this Book

The authors have a clear and thorough understanding of what REST is and isn't, and this book will provide you with the knowledge needed to distinguish something that actually is RESTful from something that just claims to be. You will learn about SOA and REST fundamentals, and get to know some of the design patterns that combine service-orientation and REST. You will find out how to design and build both services and service consumers.

There is a strong need for guidance in building RESTful systems and integration scenarios—and it is critical that your guide is not satisfied with the low-hanging fruit, but aims for the long-term benefits. I am convinced the authors are among the best guides that you can find, and I am confident you will become convinced that REST and SOA are not only not in conflict, but a perfect match.

—*Stefan Tilkov, innoQ*

Contents at a Glance

Foreword xix
Chapter 1: Introduction 1
Chapter 2: Case Study Background 13

PART I: FUNDAMENTALS 21
Chapter 3: Introduction to Services 23
Chapter 4: SOA Terminology and Concepts 31
Chapter 5: REST Constraints and Goals. 51

PART II: RESTFUL SERVICE-ORIENTATION 65
Chapter 6: Service Contracts with REST 67
Chapter 7: Service-Orientation with REST 93

PART III: SERVICE-ORIENTED ANALYSIS AND DESIGN WITH REST 125
Chapter 8: Mainstream SOA Methodology and REST 127
Chapter 9: Analysis and Service Modeling with REST 139
Chapter 10: Service-Oriented Design with REST 173

PART IV: SERVICE COMPOSITION WITH REST. 229
Chapter 11: Fundamental Service Composition with REST 231
Chapter 12: Advanced Service Composition with REST. 261
Chapter 13: Service Composition with REST Case Study 305

PART V: SUPPLEMENTAL 325
Chapter 14: Design Patterns for SOA with REST 327
Chapter 15: Service Versioning with REST. 343
Chapter 16: Uniform Contract Profiles. 361

PART VI: APPENDICES 381
Appendix A: Case Study Conclusion 383
Appendix B: Industry Standards Supporting the Web 387
Appendix C: REST Constraints Reference 391
Appendix D: Service-Orientation Principles Reference 409
Appendix E: SOA Design Patterns Reference 425
Appendix F: State Concepts and Types 521
Appendix G: The Annotated SOA Manifesto. 533
Appendix H: Additional Resources 547
Index 553

Contents

Foreword xix

CHAPTER 1: Introduction 1

1.1 About this Book 2

Who this Book is For. 2

What this Book Does Not Cover. 3

1.2 Recommended Reading 3

1.3 How this Book is Organized 4

Part I: Fundamentals 4

Chapter 3: Introduction to Services 4

Chapter 4: SOA Terminology and Concepts 5

Chapter 5: REST Constraints and Goals. 5

Part II: RESTful Service-Orientation 5

Chapter 6: Service Contracts with REST. 5

Chapter 7: Service-Orientation with REST. 5

Part III: Service-Oriented Analysis and Design with REST 5

Chapter 8: Mainstream SOA Methodology and REST 5

Chapter 9: Analysis and Service Modeling with REST 5

Chapter 10: Service-Oriented Design with REST 6

Part IV: Service Composition with REST 6

Chapter 11: Fundamental Service Composition with REST. 6

Chapter 12: Advanced Service Composition with REST. 6

Chapter 13: Service Composition with REST Case Study 6

Part V: Supplemental 6

Chapter 14: Design Patterns for SOA with REST. 6

Chapter 15: Service Versioning with REST 6

Chapter 16: Uniform Contract Profiles. 7

Part VI: Appendices 7

Appendix A: Case Study Conclusion 7

Appendix B: Industry Standards Supporting the Web 7

Appendix C: REST Constraints Reference 7

Appendix D: Service-Orientation Principles Reference 7

Appendix E: SOA Design Patterns Reference *7*
Appendix F: State Concepts and Types *7*
Appendix G: The Annotated SOA Manifesto *7*
Appendix H: Additional Resources *8*

1.4 Conventions. 8
Use of the Color Red 8
Design Constraints, Principles, and Patterns: Page References and Capitalization 8
Design Goals: Capitalization 9
Symbol Legend 9

1.5 Additional Information 10
Updates, Errata, and Resources (www.servicetechbooks.com) . . 10
Master Glossary (www.soaglossary.com) 10
Service-Orientation (www.serviceorientation.com) 10
What Is REST? (www.whatisrest.com) 10
Referenced Specifications (www.servicetechspecs.com). 10
The Service Technology Magazine (www.servicetechmag.com). . 10
SOASchool.com® SOA Certified Professional (SOACP) 11
CloudSchool.com™ Cloud Certified (CCP) Professional 11
Notification Service 11

Chapter 2: Case Study Background 13

2.1 How Case Studies Are Used 14

2.2 Case Study Background #1: Midwest University Association (MUA) 14
History 14
IT Environment 14
Business Goals and Obstacles 16
1. Build Reusable Business Services *18*
2. Consolidate Systems and Information *18*
3. Improve Channel Experience *18*
4. Build Services Infrastructure *18*

2.3 Case Study Background #2: KioskEtc Co. 18
History 19
IT Environment 19
Business Goals and Obstacles 19

PART I: FUNDAMENTALS **21**

CHAPTER 3: Introduction to Services **23**

3.1 Service Terminology 24

Service 24

Service Contract. 24

Service Capability 26

Service Consumer 26

Service Agent. 27

Service Composition 27

3.2 Service Terminology Context 29

Services and REST 29

Services and SOA 29

REST Services and SOA 29

CHAPTER 4: SOA Terminology and Concepts **31**

4.1 Basic Terminology and Concepts 32

Service-Oriented Computing 33

Service-Orientation 34

Service-Oriented Architecture (SOA). 37

SOA Manifesto 38

Services 39

Cloud Computing 40

IT Resources 41

Service Models. 41

Agnostic Logic and Non-Agnostic Logic 42

Service Inventory 42

Service Portfolio 43

Service Candidate 44

Service Contract. 44

Service-Related Granularity 45

Service Profiles. 46

SOA Design Patterns 46

4.2 Further Reading 49

Case Study Example 50

CHAPTER 5: REST Constraints and Goals. 51

5.1 REST Constraints. 52
Client-Server. 53
Stateless. 54
Cache. 55
Interface/Uniform Contract. 55
Layered System 56
Code-On-Demand 57

5.2 Goals of the REST Architectural Style................ 58
Performance. 58
Scalability. 59
Simplicity 60
Modifiability 61
Visibility. 61
Portability 62
Reliability 62

Case Study Example 63

PART II: RESTFUL SERVICE-ORIENTATION 65

CHAPTER 6: Service Contracts with REST 67

6.1 Uniform Contract Elements 68
Resource Identifier Syntax (and Resources) 69
URIs (and URLs and URNs). 69
Resource Identifiers and REST Services. 71
Methods 71
Media Types. 73

6.2 REST Service Capabilities and REST Service Contracts . . 75

6.3 REST Service Contracts vs. Non-REST Service Contracts 77
Non-REST Service with Custom Service Contract. 77
REST Service with Uniform Contract 79
HTTP Messaging vs. SOAP Messaging. .81
REST Service Contracts with WSDL?. 82

6.4 The Role of Hypermedia 83
URI Templates and Resource Queries 86
6.5 REST Service Contracts and Late Binding 87
Case Study Example 90

CHAPTER 7: Service-Orientation with REST 93

7.1 "SOA vs. REST" or "SOA + REST"? 95
7.2 Design Goals 97
Increased Intrinsic Interoperability 97
Increased Federation 98
Increased Vendor Diversity Options 99
Increased Business and Technology Alignment 100
Increased ROI 100
Increased Organizational Agility 102
Reduced IT Burden 102
Common Goals 103
7.3 Design Principles and Constraints 104
Standardized Service Contract 104
Service Loose Coupling 105
Service Abstraction 107
Service Reusability 109
Service Autonomy 110
Service Statelessness 111
Service Discoverability 113
Service Composability 114
Common Conflicts 114
Stateful Interactions 115
Service-Specific Contract Details 115
Case Study Example 116

PART III: SERVICE-ORIENTED ANALYSIS AND DESIGN WITH REST 125

CHAPTER 8: Mainstream SOA Methodology and REST 127

8.1 Service Inventory Analysis 131

8.2 Service-Oriented Analysis (Service Modeling) 133

8.3 Service-Oriented Design (Service Contract) 135

8.4 Service Logic Design 137

8.5 Service Discovery 137

8.6 Service Versioning and Retirement 138

CHAPTER 9: Analysis and Service Modeling with REST . . 139

9.1 Uniform Contract Modeling and REST Service Inventory Modeling 141

REST Constraints and Uniform Contract Modeling 144

REST Service Centralization and Normalization 146

9.2 REST Service Modeling 147

REST Service Capability Granularity 148

Resources vs. Entities 149

REST Service Modeling Process 150

Case Study Example 152

Step 1: Decompose Business Process (into Granular Actions) . . 152

Case Study Example 152

Step 2: Filter Out Unsuitable Actions 154

Case Study Example 154

Step 3: Identify Agnostic Service Candidates 155

Case Study Example 157

Event Service Candidate (Entity) 157

Award Service Candidate (Entity) 158

Student Service Candidate (Entity) 158

Notification Service Candidate (Utility) 159

Document Service Candidate (Utility) 159

Step 4: Identify Process-Specific Logic 160

Case Study Example 160
Confer Student Award Service Candidate (Task) 161
Step 5: Identify Resources 161
Case Study Example 162
Step 6: Associate Service Capabilities with Resources and Methods 163
Case Study Example 164
Confer Student Award Service Candidate (Task) 164
Event Service Candidate (Entity) 164
Award Service Candidate (Entity) 165
Student Service Candidate (Entity) 165
Notification Service Candidate (Utility) 166
Document Service Candidate (Utility) 166
Step 7: Apply Service-Orientation 167
Case Study Example 167
Step 8: Identify Candidate Service Compositions 167
Case Study Example 168
Step 9: Analyze Processing Requirements 169
Step 10: Define Utility Service Candidates 170
Step 11: Associate Utility-Centric Service Capabilities with Resources and Methods 171
Step 12: Apply Service-Orientation 171
Step 13: Revise Candidate Service Compositions 171
Step 14: Revise Resource Definitions 171
Step 15: Revise Capability Candidate Grouping 172
Additional Considerations 172

CHAPTER 10: Service-Oriented Design with REST 173

10.1 Uniform Contract Design Considerations 175
Designing and Standardizing Methods 175
Designing and Standardizing HTTP Headers 177
Designing and Standardizing HTTP Response Codes 179
Customizing Response Codes 184
Designing Media Types 186
Designing Schemas for Media Types 188
Service-Specific XML Schemas 189

10.2 REST Service Contract Design 191
Designing Services Based on Service Models 191
Task Services 191
Entity Services 192
Utility Services 193
Designing and Standardizing Resource Identifiers 194
Service Names in Resource Identifiers 195
Other URI Components 196
Resource Identifier Overlap 197
Resource Identifier Design Guidelines 199
Designing with and Standardizing REST Constraints 201
Stateless 201
Cache 202
Uniform Contract 203
Layered System 204
Case Study Example 205
Confer Student Award Service Contract (Task) 205
Event Service Contract (Entity) 207
Award Service Contract (Entity) 207
Student Transcript Service Contract (Entity) 208
Notification and Document Service Contracts (Utility) 209
10.3 Complex Method Design 211
Stateless Complex Methods 214
Fetch Method 214
Store Method 215
Delta Method 217
Async Method 219
Stateful Complex Methods 221
Trans Method 221
PubSub Method 222
Case Study Example 224
OptLock Complex Method 224
PesLock Complex Method 226