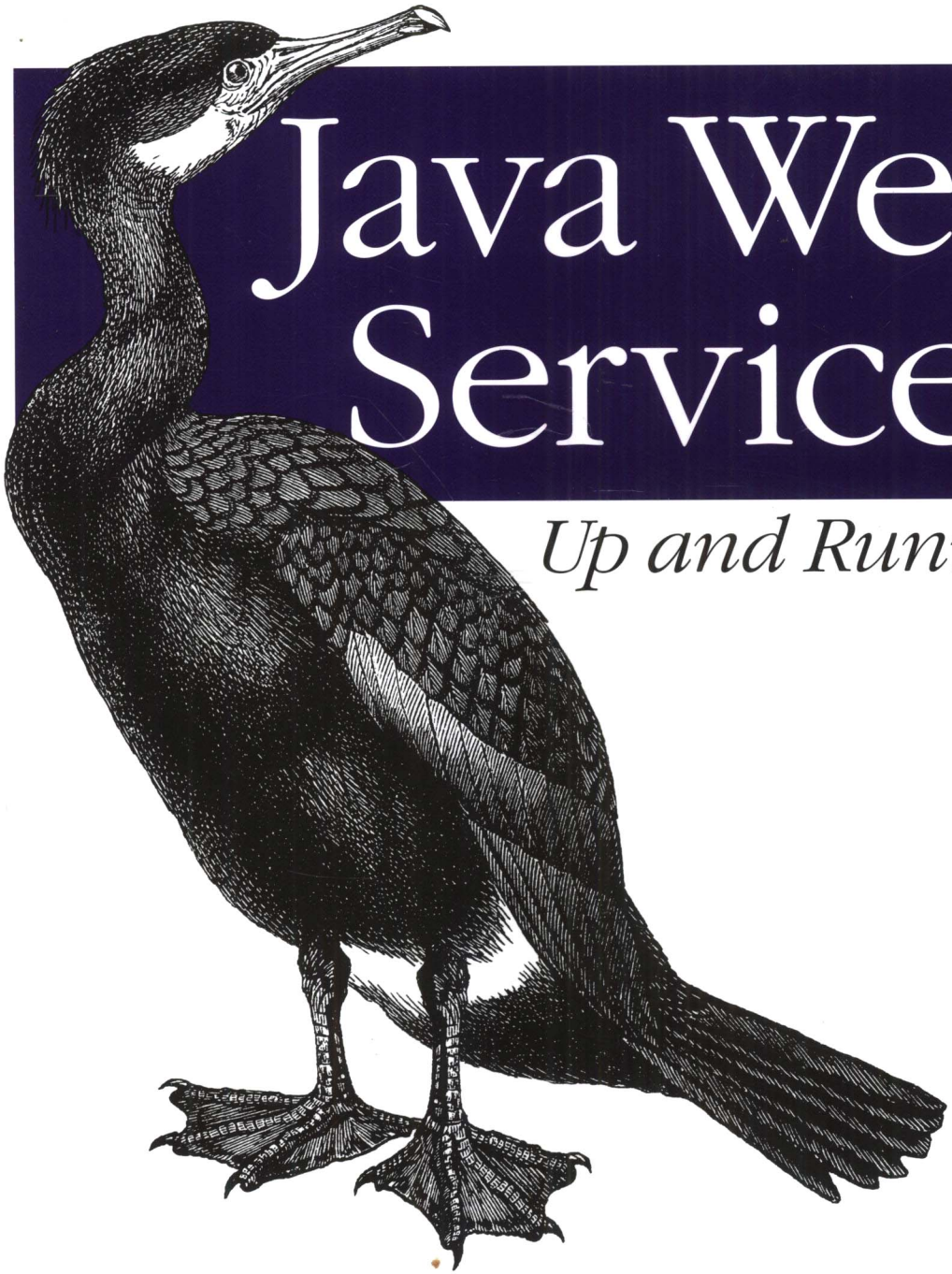
Java Web 服务：构建与运行（影印版）

Covers JAX-WS 2

Java Web Services

Up and Running



Martin Kalin 著

Java Web服务：构建与运行（影印版）

Java Web Services: Up and Running

常州大学山书馆
藏书章

Martin Kalin

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

Java Web服务：构建与运行：英文/（美）卡林(Kalin, M.）著. —影印本. —南京：东南大学出版社，2010.1

书名原文：Java Web Services: Up and Running

ISBN 978-7-5641-1927-0

Ⅰ. J… Ⅱ. 卡… Ⅲ. JAVA语言－程序设计－英文

Ⅳ. TP312

中国版本图书馆CIP数据核字（2009）第205659号

江苏省版权局著作权合同登记

图字：10-2009-243号

©2009 by O'Reilly Media, Inc.

Reprint of the English Edition, jointly published by O'Reilly Media, Inc. and Southeast University Press, 2009. Authorized reprint of the original English edition, 2009 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.

英文原版由O'Reilly Media, Inc. 出版2009。

英文影印版由东南大学出版社出版2009。此影印版的出版和销售得到出版权和销售权的所有者——O'Reilly Media, Inc. 的许可。

版权所有，未得书面许可，本书的任何部分和全部不得以任何形式重制。

Java Web服务：构建与运行（影印版）

出版发行：东南大学出版社
地　　址：南京四牌楼2号　　邮编：210096
出 版 人：江　汉
网　　址：http://press.seu.edu.cn
电子邮件：press@seu.edu.cn
印　　刷：扬中市印刷有限公司
开　　本：787毫米×980毫米　16开本
印　　张：20印张
字　　数：336千字
版　　次：2010年1月第1版
印　　次：2010年1月第1次印刷
书　　号：ISBN 978-7-5641-1927-0
印　　数：1~1600册
定　　价：48.00元（册）

本社图书若有印装质量问题，请直接与读者服务部联系。电话（传真）：025-83792328

Java Web 服务：构建与运行（影印版）

Java Web Services: Up and Running

Preface

This is a book for programmers interested in developing Java web services and Java clients against web services, whatever the implementation language. The book is a code-driven introduction to *JAX-WS* (Java API for XML-Web Services), the framework of choice for Java web services, whether SOAP-based or REST-style. My approach is to interpret JAX-WS broadly and, therefore, to include leading-edge developments such as the Jersey project for REST-style web services, officially known as *JAX-RS* (Java API for XML-RESTful Web Services).

JAX-WS is bundled into the *Metro Web Services Stack*, or *Metro* for short. Metro is part of core Java, starting with Standard Edition 6 (hereafter, core Java 6). However, the Metro releases outpace the core Java releases. The current Metro release can be downloaded separately from *https://wsit.dev.java.net*. Metro is also integrated into the Sun application server, GlassFish. Given these options, this book's examples are deployed in four different ways:

Core Java only
: This is the low-fuss approach that makes it easy to get web services and their clients up and running. The only required software is the Java software development kit (SDK), core Java 6 or later. Web services can be deployed easily using the `Endpoint`, `HttpServer`, and `HttpsServer` classes. The early examples take this approach.

Core Java with the current Metro release
: This approach takes advantage of Metro features not yet available in the core Java bundle. In general, each Metro release makes it easier to write web services and clients. The current Metro release also indicates where JAX-WS is moving. The Metro release also can be used with core Java 5 if core Java 6 is not an option.

Standalone Tomcat
: This approach builds on the familiarity among Java programmers with standalone web containers such as Apache Tomcat, which is the reference implementation. Web services can be deployed using a web container in essentially the same way as are servlets, JavaServer Pages (JSP) scripts, and JavaServer Faces (JSF) scripts. A standalone web container such as Tomcat is also a good way to introduce container-managed security for web services.

此为试读，需要完整PDF请访问：www.ertongbook.com

GlassFish
: This approach allows deployed web services to interact naturally with other enterprise components such as Java Message Service topics and queues, a *JNDI* (Java Naming and Directory Interface) provider, a backend database system and the `@Entity` instances that mediate between an application and the database system, and an *EJB* (Enterprise Java Bean) container. The EJB container is important because a web service can be deployed as a stateless Session EJB, which brings advantages such as container-managed thread safety. GlassFish works seamlessly with Metro, including its advanced features, and with popular *IDE*s (Integrated Development Environment) such as NetBeans and Eclipse.

An appealing feature of JAX-WS is that the API can be separated cleanly from deployment options. One and the same web service can be deployed in different ways to suit different needs. Core Java alone is good for learning, development, and even lightweight deployment. A standalone web container such as Tomcat provides additional support. A Java application server such as GlassFish promotes easy integration of web services with other enterprise technologies.

Code-Driven Approach

My code examples are short enough to highlight key features of JAX-WS but also realistic enough to show off the production-level capabilities that come with the JAX-WS framework. Each code example is given in full, including all of the `import` statements. My approach is to begin with a relatively sparse example and then to add and modify features. The code samples vary in length from a few statements to several pages of source. The code is deliberately modular. Whenever there is a choice between conciseness and clarity in coding, I try to opt for clarity.

The examples come with instructions for compiling and deploying the web services and for testing the service against sample clients. This approach presents the choices that JAX-WS makes available to the programmer but also encourages a clear and thorough analysis of the JAX-WS libraries and utilities. My goal is to furnish code samples that can serve as templates for commercial applications.

JAX-WS is a rich API that is explored best in a mix of overview and examples. My aim is to explain key features about the architecture of web services but, above all, to illustrate each major feature with code examples that perform as advertised. Architecture without code is empty; code without architecture is blind. My approach is to integrate the two throughout the book.

Web services are a modern, lightweight approach to *distributed software systems*, that is, systems such as email or the World Wide Web that require different software components to execute on physically distinct devices. The devices can range from large servers through personal desktop machines to handhelds of various types. Distributed systems are complicated because they are made up of networked components. There

is nothing more frustrating than a distributed systems example that does not work as claimed because the debugging is tedious. My approach is thus to provide full, working examples together with short but precise instructions for getting the sample application up and running. All of the source code for examples is available from the book's companion site, at *http://www.oreilly.com/catalog/9780596521127*. My email address is *kalin@cdm.depaul.edu*. Please let me know if you find any code errors.

Chapter-by-Chapter Overview

The book has seven chapters, the last of which is quite short. Here is a preview of each chapter:

Chapter 1, *Java Web Services Quickstart*

This chapter begins with a working definition of *web services*, including the distinction between SOAP-based and REST-style services. This chapter then focuses on the basics of writing, deploying, and consuming SOAP-based services in core Java. There are web service clients written in Perl, Ruby, and Java to underscore the language neutrality of web services. This chapter also introduces Java's SOAP API and covers various ways to inspect web service traffic at the wire level. The chapter elaborates on the relationship between core Java and Metro.

Chapter 2, *All About WSDLs*

This chapter focuses on the service contract, which is a *WSDL* (Web Service Definition Language) document in SOAP-based services. This chapter covers the standard issues of web service style (`document` versus `rpc`) and encoding (`literal` versus `encoded`). This chapter also focuses on the popular but unofficial distinction between the *wrapped* and *unwrapped* variations of document style. All of these issues are clarified through examples, including Java clients against Amazon's E-Commerce services. This chapter explains how the *wsimport* utility can ease the task of writing Java clients against commercial web services and how the *wsgen* utility figures in the distinction between `document`-style and `rpc`-style web services. The basics of *JAX-B* (Java API for XML-Binding) are also covered. This chapter, like the others, is rich in code examples.

Chapter 3, *SOAP Handling*

This chapter introduces SOAP and logical handlers, which give the service-side and client-side programmer direct access to either the entire SOAP message or just its payload. The structure of a SOAP message and the distinction between SOAP 1.1 and SOAP 1.2 are covered. The messaging architecture of a SOAP-based service is discussed. Various code examples illustrate how SOAP messages can be processed in support of application logic. This chapter also explains how transport-level messages (for instance, the typical HTTP messages that carry SOAP payloads in SOAP-based web services) can be accessed and manipulated in JAX-WS. This chapter concludes with a section on JAX-WS support for

transporting binary data, with emphasis on *MTOM* (Message Transmission Optimization Mechanism).

Chapter 4, *RESTful Web Services*

This chapter opens with a technical analysis of what constitutes a REST-style service and moves quickly to code examples. The chapter surveys various approaches to delivering a Java-based RESTful service: `WebServiceProvider`, `HttpServlet`, Jersey Plain Old Java Object (POJO), and restlet among them. The use of a *WADL* (Web Application Definition Language) document as a service contract is explored through code examples. The *JAX-P* (Java API for XML-Processing) packages, which facilitate XML processing, are also covered. This chapter offers several examples of Java clients against real-world REST-style services, including services hosted by Yahoo!, Amazon, and Tumblr.

Chapter 5, *Web Services Security*

This chapter begins with an overview of security requirements for real-world web services, SOAP-based and REST-style. The overview covers central topics such as mutual challenge and message confidentiality, users-roles security, and WS-Security. Code examples clarify transport-level security, particularly under HTTPS. Container-managed security is introduced with examples deployed in the standalone Tomcat web container. The security material introduced in this chapter is expanded in the next chapter.

Chapter 6, *JAX-WS in Java Application Servers*

This chapter starts with a survey of what comes with a Java Application Server (JAS): an EJB container, a messaging system, a naming service, an integrated database system, and so on. This chapter has a variety of code examples: a SOAP-based service implemented as a stateless Session EJB, `WebService` and `WebServiceProvider` instances deployed through embedded Tomcat, a web service deployed together with a traditional website application, a web service integrated with *JMS* (Java Message Service), a web service that uses an `@Entity` to read and write from the Java DB database system included in GlassFish, and a WS-Security application under GlassFish.

Chapter 7, *Beyond the Flame Wars*

This is a very short chapter that looks at the controversy surrounding SOAP-based and REST-style web services. My aim is to endorse both approaches, either of which is superior to what came before. This chapter traces modern web services from DCE/RPC in the early 1990s through CORBA and DCOM up to the Java EE and .NET frameworks. This chapter explains why either approach to web services is better than the distributed-object architecture that once dominated in distributed software systems.

Freedom of Choice: The Tools/IDE Issue

Java programmers have a wide choice of productivity tools such as Ant and Maven for scripting and IDEs such as Eclipse, NetBeans, and IntelliJ IDEA. Scripting tools and IDEs increase productivity by hiding grimy details. In a production environment, such tools and IDEs are the sensible way to go. In a learning environment, however, the goal is to understand the grimy details so that this understanding can be brought to good use during the inevitable bouts of debugging and application maintenance. Accordingly, my book is neutral with respect to scripting tools and IDEs. Please feel free to use whichever tools and IDE suit your needs. My how-to segments go over code compilation, deployment, and execution at the command line so that details such as classpath inclusions and compilation/execution flags are clear. Nothing in any example depends on a particular scripting tool or IDE.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, filenames, file extensions, and emphasis.

`Constant width`
: Used for program listings as well as within paragraphs to refer to program elements such as variable or method names, data types, environment variables, statements, and keywords.

`Constant width bold`
: Used within program listings to highlight particularly interesting sections and in paragraphs to clarify acronyms.

This icon signifies a tip, suggestion, or general note.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Java Web Services: Up and Running*, by Martin Kalin. Copyright 2009 Martin Kalin, 978-0-596-52112-7."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

When you see a Safari® Books Online icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at *http://safari.oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596521127/

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our website at:

http://www.oreilly.com/

Acknowledgments

Christian A. Kenyeres, Greg Ostravich, Igor Polevoy, and Ken Yu were kind enough to review this book and to offer insightful suggestions for its improvement. They made the book better than it otherwise would have been. I thank them heartily for the time and effort that they invested in this project. The remaining shortcomings are mine alone, of course.

I'd also like to thank Mike Loukides, my first contact at O'Reilly Media, for his role in shepherding my initial proposal through the process that led to its acceptance. Julie Steele, my editor, has provided invaluable support and the book would not be without her help. My thanks go as well to the many behind-the-scenes people at O'Reilly Media who worked on this project.

This book is dedicated to Janet.

此为试读,需要完整PDF请访问: www.ertongbook.com

O'Reilly Media, Inc. 介绍

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，同时是联机出版的先锋。

从最畅销的《The Whole Internet User's Guide & Catalog》（被纽约公共图书馆评为二十世纪最重要的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。

出版说明

随着计算机技术的成熟和广泛应用，人类正在步入一个技术迅猛发展的新时期。计算机技术的发展给人们的工业生产、商业活动和日常生活都带来了巨大的影响。然而，计算机领域的技术更新速度之快也是众所周知的，为了帮助国内技术人员在第一时间了解国外最新的技术，东南大学出版社和美国 O'Reilly Meida, Inc. 达成协议，将陆续引进该公司的代表前沿技术或者在某专项领域享有盛名的著作，以影印版或者简体中文版的形式呈献给读者。其中，影印版书籍力求与国外图书“同步”出版，并且“原汁原味”展现给读者。

我们真诚地希望，所引进的书籍能对国内相关行业的技术人员、科研机构的研究人员和高校师生的学习和工作有所帮助，对国内计算机技术的发展有所促进。也衷心期望读者提出宝贵的意见和建议。

最新出版的影印版图书，包括：

- 《真实世界的 Haskell》（影印版）
- 《卓有成效的程序员》（影印版）
- 《Java Web 服务：构建与运行》（影印版）
- 《并行开发艺术》（影印版）
- 《使用 Perl 实现系统管理自动化 第二版》（影印版）
- 《Java 消息服务 第二版》（影印版）
- 《深入浅出网络管理》（影印版）
- 《Ruby 最佳实践》（影印版）
- 《更快速网站》（影印版）
- 《正则表达式 Cookbook》（影印版）
- 《flex 与 bison》（影印版）

Table of Contents

Preface ix

1. Java Web Services Quickstart 1
What Are Web Services? 1
What Good Are Web Services? 3
A First Example 4
The Service Endpoint Interface and Service Implementation Bean 4
A Java Application to Publish the Web Service 6
Testing the Web Service with a Browser 7
A Perl and a Ruby Requester of the Web Service 10
The Hidden SOAP 11
A Java Requester of the Web Service 13
Wire-Level Tracking of HTTP and SOAP Messages 14
What's Clear So Far? 16
Key Features of the First Code Example 16
Java's SOAP API 18
An Example with Richer Data Types 23
Publishing the Service and Writing a Client 25
Multithreading the Endpoint Publisher 27
What's Next? 30

2. All About WSDLs 31
What Good Is a WSDL? 31
Generating Client-Support Code from a WSDL 32
The @WebResult Annotation 35
WSDL Structure 36
A Closer Look at WSDL Bindings 38
Key Features of Document-Style Services 39
Validating a SOAP Message Against a WSDL's XML Schema 42
The Wrapped and Unwrapped Document Styles 43
Amazon's E-Commerce Web Service 46
An E-Commerce Client in Wrapped Style 47

An E-Commerce Client in Unwrapped Style 52
Tradeoffs Between the RPC and Document Styles 55
An Asynchronous E-Commerce Client 57
The wsgen Utility and JAX-B Artifacts 59
A JAX-B Example 60
Marshaling and wsgen Artifacts 65
An Overview of Java Types and XML Schema Types 67
Generating a WSDL with the wsgen Utility 68
WSDL Wrap-Up 69
Code First Versus Contract First 69
A Contract-First Example with wsimport 70
A Code-First, Contract-Aware Approach 76
Limitations of the WSDL 79
What's Next? 80

3. SOAP Handling 81
SOAP: Hidden or Not? 81
SOAP 1.1 and SOAP 1.2 81
SOAP Messaging Architecture 82
Programming in the JWS Handler Framework 84
The RabbitCounter Example 85
Injecting a Header Block into a SOAP Header 85
Configuring the Client-Side SOAP Handler 91
Adding a Handler Programmatically on the Client Side 92
Generating a Fault from a @WebMethod 94
Adding a Logical Handler for Client Robustness 95
Adding a Service-Side SOAP Handler 97
Summary of the Handler Methods 101
The RabbitCounter As a SOAP 1.2 Service 102
The MessageContext and Transport Headers 104
An Example to Illustrate Transport-Level Access 104
Web Services and Binary Data 109
Three Options for SOAP Attachments 111
Using Base64 Encoding for Binary Data 111
Using MTOM for Binary Data 116
What's Next? 119

4. RESTful Web Services 121
What Is REST? 121
Verbs and Opaque Nouns 124
From @WebService to @WebServiceProvider 125
A RESTful Version of the Teams Service 126
The WebServiceProvider Annotation 126

Language Transparency and RESTful Services 132
Summary of the RESTful Features 136
Implementing the Remaining CRUD Operations 136
Java API for XML Processing 138
The Provider and Dispatch Twins 148
A Provider/Dispatch Example 149
More on the Dispatch Interface 153
A Dispatch Client Against a SOAP-based Service 157
Implementing RESTful Web Services As HttpServlets 159
The RabbitCounterServlet 160
Requests for MIME-Typed Responses 165
Java Clients Against Real-World RESTful Services 167
The Yahoo! News Service 167
The Amazon E-Commerce Service: REST Style 170
The RESTful Tumblr Service 173
WADLing with Java-Based RESTful Services 177
JAX-RS: WADLing Through Jersey 182
The Restlet Framework 186
What's Next? 191

5. Web Services Security 193
Overview of Web Services Security 193
Wire-Level Security 194
HTTPS Basics 195
Symmetric and Asymmetric Encryption/Decryption 196
How HTTPS Provides the Three Security Services 197
The HttpsURLConnection Class 200
Securing the RabbitCounter Service 203
Adding User Authentication 211
HTTP BASIC Authentication 212
Container-Managed Security for Web Services 212
Deploying a @WebService Under Tomcat 213
Securing the @WebService Under Tomcat 215
Application-Managed Authentication 217
Container-Managed Authentication and Authorization 219
Configuring Container-Managed Security Under Tomcat 220
Using a Digested Password Instead of a Password 223
A Secured @WebServiceProvider 224
WS-Security 227
Securing a @WebService with WS-Security Under Endpoint 229
The Prompter and the Verifier 236
The Secured SOAP Envelope 237
Summary of the WS-Security Example 238

What's Next? 238

6. JAX-WS in Java Application Servers 239
Overview of a Java Application Server 239
Deploying @WebServices and @WebServiceProviders 244
Deploying @WebServiceProviders 245
Integrating an Interactive Website and a Web Service 250
A @WebService As an EJB 252
Implementation As a Stateless Session EJB 252
The Endpoint URL for an EBJ-Based Service 256
Database Support Through an @Entity 256
The Persistence Configuration File 258
The EJB Deployment Descriptor 260
Servlet and EJB Implementations of Web Services 261
Java Web Services and Java Message Service 262
WS-Security Under GlassFish 265
Mutual Challenge with Digital Certificates 266
MCS Under HTTPS 266
MCS Under WSIT 269
The Dramatic SOAP Envelopes 276
Benefits of JAS Deployment 280
What's Next? 281

7. Beyond the Flame Wars 283
A Very Short History of Web Services 283
The Service Contract in DCE/RPC 284
XML-RPC 285
Standardized SOAP 286
SOAP-Based Web Services Versus Distributed Objects 287
SOAP and REST in Harmony 288

Index 291

此为试读,需要完整PDF请访问: www.ertongbook.com