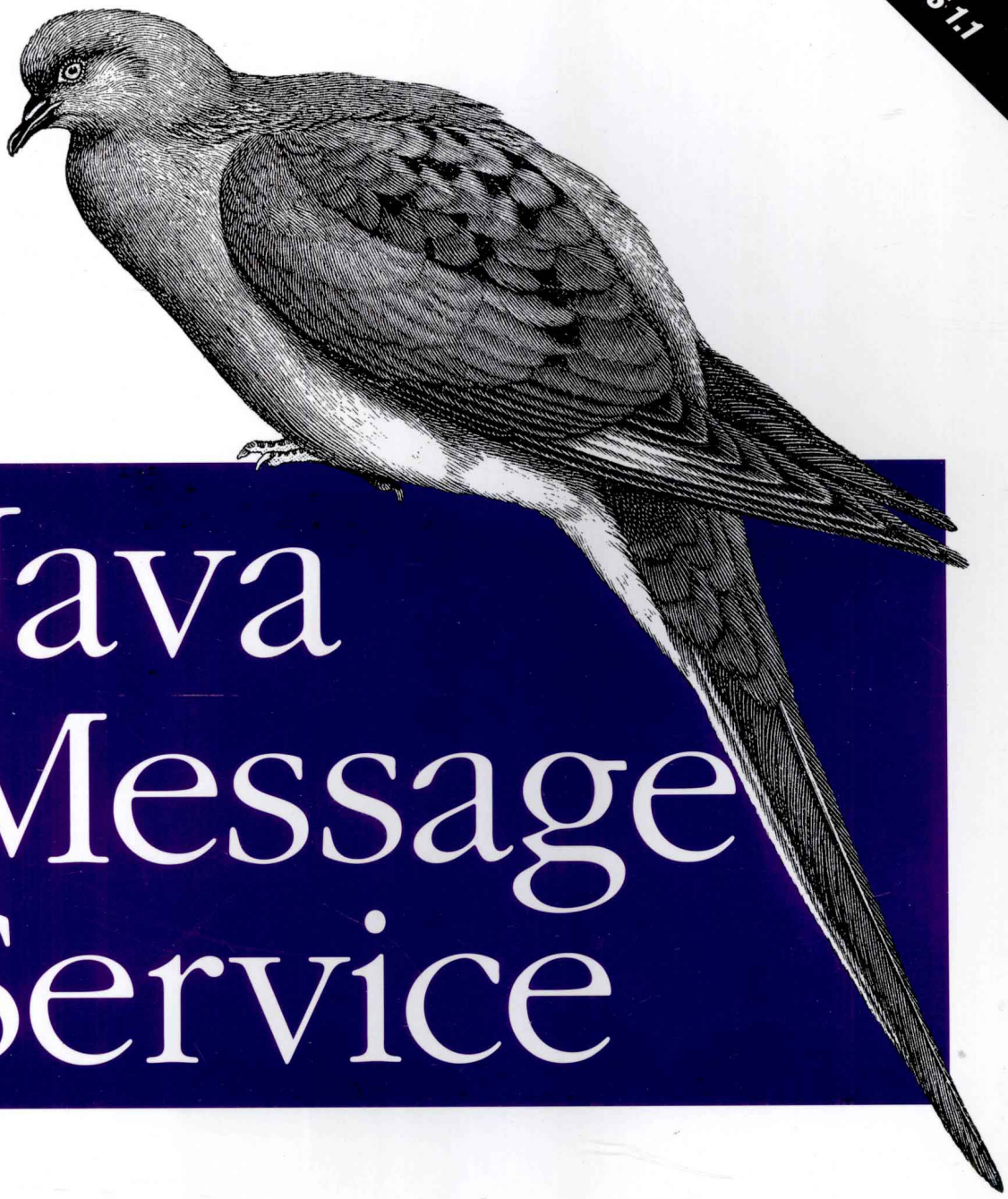
*Java*消息服务（影印版）

第二版
Updated for JMS 1.1



Java Message Service

REILLY®
[illegible]大學 出版社

Mark Richards, Richard Monson-Haefel & David A. Chappell 著

第二版

Java消息服务（影印版）

Java Message Services

Mark Richards, Richard Monson-Haefel
David A. Chappell

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消息服务：第2版：英文/（美）布朗（Brown, T.G.）等著.—影印本.—南京：东南大学出版社，2010.1
书名原文：Java Message Service, 2E
ISBN 978-7-5641-1930-0

Ⅰ.J… Ⅱ.布… 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205632号

江苏省版权局著作权合同登记
图字：10-2009-245号

Java消息服务：第2版（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：20.75印张
字　　数：349千字
版　　次：2010年1月第1版
印　　次：2010年1月第1次印刷
书　　号：ISBN 978-7-5641-1930-0
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消息服务（影印版）

Java Message Services

Foreword

For close to a decade now, I've been a fan of messaging-based systems. They offer a degree of reliability, flexibility, extensibility, and modularity that a traditional RPC or distributed object system simply cannot. Working with them takes a bit of adjustment, because they don't quite behave the same way that an architect or designer expects a traditional n-tier system to behave. This is not to say that they're better or worse; they're just different. Instead of invoking methods on objects directly, where the object can hold conversational state or context, now the message itself has to be self-contained and state-complete.

Which raises an important point.

For any given developer with respect to any given technology, there are four distinct stages.

The first is the Ignorant. We may know the technology exists, or not, but beyond that we remain entirely ignorant about its capabilities. It's a collection of letters, at best, often mentioned in conjunction with other technologies that may or may not matter to what we're doing on a daily basis.

The second is the Explorer. Something piques our curiosity, voluntarily or not. We begin some initial forays into the jungle, perhaps downloading an implementation or reading a few articles. We begin to understand the basic framing of where this thing sits in the broad scheme of things and maybe how it's supposed to work, but our hands-on experience is generally limited to the moral equivalent of "Hello World" and a few other samples.

The third is the Journeyman. After running many of the samples and reading a few articles, we realize that we understand it at a basic level and begin to branch out to writing code with it. We feel reasonably comfortable introducing it into production code and reasonably comfortable debugging the stupid mistakes we'll make with it. We're not experts, by any means, but we can at least get the stuff to compile and run most of the time.

The last, of course, is the Master. After building a few systems and seeing how they react under real-world conditions, we have a deep gestalt with it and can often predict how the tool or technology will react without even running the code. We can see how

this thing will interact with other, complementary technologies, and understand how to achieve some truly miraculous results, such as systems that resist network outages or machine failures. When the Java Message Service (JMS) API was first released, back in 1999, before any noncommercial/open-source implementations were available, I distinctly remember looking at it, thinking, "Well, it seems interesting, but it's not something I can use without a real implementation," and setting my printed copy of the specification off to one side for later perusal. My transition to Explorer and Journeyman came a few years later, as I came to understand the power of messaging systems, partly thanks to the few implementations out, partly thanks to my own exploration of other messaging systems (most notably MSMQ and Tibco), but mostly due to the person who wrote this second edition of *Java Message Service*.

I'm still well shy of Master status. Fortunately, both you and I know somebody who is not.

Mark Richards has spent the last several years living the messaging lifestyle, both as an architect and implementor as well as a leader and luminary: the first in his capacity as a consultant, the second in his capacity as a speaker on the No Fluff Just Stuff (NFJS) tour. He has a great "take" on the reasons for and the implications of building message-based systems, and he brings that forth in this nearly complete rewrite of Richard Monson-Haefel and Dave Chappell's first edition. Even if you're in the Ignorant stage of JMS, Mark's careful walkthrough of the basics, through implementation and then the design pros and cons of messaging will bring you to the Journeyman stage fast and leave you with the necessary structure in place to let you reach that Master stage in no time at all.

And that, my friend, is the best anybody can ask of a book.

Happy messaging.

—Ted Neward
Principal Consultant,ThoughtWorks
December 10, 2008, Antwerp, Belgium

Preface

When I was presented with the opportunity to revise *Java Message Service*, I jumped at the chance. The first edition, published by O'Reilly in 2000, was a bestseller and without a doubt the definitive source for JMS and messaging in general at that time. Writing the second edition was an exciting chance to breath new life into an already great book and add new content that was relevant to how we use messaging today. What I failed to fully realize when I took on the project was just how much messaging (or, more precisely, how we use messaging) has changed in the past 10 years. New messaging techniques and technologies have been developed, including message-driven beans (as part of the EJB specification), the Spring messaging framework, Event-Driven Architecture, Service-Oriented Architecture, RESTful JMS interfaces, and the Enterprise Service Bus (ESB), to name a few. The somewhat minor book project that I originally planned quickly turned into a major book project.

My original intent was to preserve as much of the original content as possible in this new edition. However, based on changes to the JMS specification since the first edition was written, as well as the development of new messaging techniques and technologies, the original content quickly shrank. As a result, you will find that roughly 75% of this second edition is new or revised content.

The JMS specification was updated to version 1.1 a couple of years after the printing of the first edition of this book. While not a major change to the JMS specification, the JMS 1.1 specification was nevertheless a significant step toward fixing some of the deficiencies with the original JMS specification. One of the biggest changes in the specification was the joining of the queue and topic API under a unified general API, allowing queues and topics to share the same transactional unit of work. However, the specification change alone was not the only factor that warranted a second edition of the book. As the Java platform has matured, so has the way we think about messaging. From new messaging technologies and frameworks to complex integration and throughput requirements, messaging has changed the way we think about and design systems, particularly over the past 10 years. These factors, combined with the specification changes, are the reasons for the second edition.

With the exception of the `Chat` application found in Chapter 2, all of the sample code has been changed to reflect more up-to-date messaging use cases and to illustrate some additional features of JMS that were not included in the first edition.

I added several new chapters that were not included in the first edition, for obvious reasons. You will find new sections in the first chapter on the JMS API, updated messaging use cases, and a discussion of how messaging has changed how we design systems. You will also find new chapters on message filtering, Java EE and message-driven beans, Spring JMS and message-driven POJOs, and messaging design.

In addition to adding new chapters, I significantly revised the existing chapters. Because I updated the sample code used to illustrate various points throughout the book, I was in turn forced to rewrite much of the corresponding text. This provided me with the opportunity to add additional sections and topics, particularly in Chapter 4, *Point-to-Point Messaging*, and Chapter 5, *Publish-and-Subscribe Messaging*. I also reversed these chapters from the first edition with the belief that it is easier to jump into messaging with the point-to-point messaging model using queues rather than the publish-and-subscribe messaging model using topics and subscribers.

I hope you find the new edition of this book helpful in terms of understanding the Java Message Service and messaging in general.

—Mark Richards

Who Should Read This Book?

This book explains and demonstrates the fundamentals of Java Message Service. It provides a straightforward, no-nonsense explanation of the underlying technology, Java classes and interfaces, programming models, and various implementations of the JMS specification.

Although this book focuses on the fundamentals, it's no "dummy's" book. While the JMS API is easy to learn, the API abstracts fairly complex enterprise technology. Before reading this book, you should be fluent with the Java language and have some practical experience developing business solutions. Experience with messaging systems is not required, but you must have a working knowledge of the Java language.

Organization

The book is organized into 11 chapters and 4 appendixes. Chapter 1 explains messaging systems, messaging use cases, centralized and distributed architectures, and why JMS is important. Chapters 2 through 6 go into detail about developing JMS clients using the two messaging models, point-to-point and publish-and-subscribe, including how to filter messages using message selectors. Chapters 7 and 10 should be considered "advanced topics," covering deployment and administration of messaging systems. Chapter 8 provides an overview of the Java 2, Enterprise Edition (Java EE) with regard

to JMS, including coverage of message-driven beans as part of the Enterprise JavaBeans 3.0 specification. Chapter 9 covers the Spring Framework as it relates to messaging. Finally, Chapter 11 provides some insight into many of the design considerations and anti-patterns associated with messaging.

Chapter 1, *Messaging Basics*
: Defines enterprise messaging and common architectures used by messaging vendors. JMS is defined and explained, as are its two programming models, publish-and-subscribe and point-to-point. Many of the use cases and real-world scenarios for messaging are described in this chapter, as are the basics of the JMS API.

Chapter 2, *Developing a Simple Example*
: Walks the reader through the development of a simple publish-and-subscribe JMS client.

Chapter 3, *Anatomy of a JMS Message*
: Provides a detailed examination of the JMS message, the most important part of the JMS API.

Chapter 4, *Point-to-Point Messaging*
: Examines the point-to-point messaging model through the development of a simple borrower and lender JMS application. Also covers some of the finer points of the point-to-point messaging model, including message correlation, dynamic queues, load balancing, and queue browsing.

Chapter 5, *Publish-and-Subscribe Messaging*
: Examines the publish-and-subscribe messaging model through the enhancement of the borrower and lender application developed in Chapter 4. This chapter also covers durable subscribers, nondurable subscribers, dynamic durable subscribers, and temporary topics.

Chapter 6, *Message Filtering*
: Provides a detailed explanation of message filtering using message selectors.

Chapter 7, *Guaranteed Messaging and Transactions*
: Provides an in-depth explanation of advanced topics, including guaranteed messaging, transactions, acknowledgments, message grouping, and failures.

Chapter 8, *Java EE and Message-Driven Beans*
: Provides an overview of the Java 2, Enterprise Edition (Java EE) version 3.0 with regard to JMS and includes coverage of message-driven beans (MDBs).

Chapter 9, *Spring and JMS*
: Provides a detailed explanation of the Spring Framework with regards to JMS, including the Spring JMS Template and message-driven POJOs (MDPs).

Chapter 10, *Deployment Considerations*
: Provides an in-depth examination of features and issues that should be considered when choosing vendors and deploying JMS applications.

Chapter 11, *Messaging Design Considerations*
: Provides insight into and explanation of several design considerations, including the use of internal versus external destinations, request/reply processing, and a discussion of some of the more common messaging anti-patterns.

Appendix A, *The Java Message Service API*
: Provides a quick reference to the classes and interfaces defined in the JMS package.

Appendix B, *Message Headers*
: Provides detailed information about message headers.

Appendix C, *Message Properties*
: Provides detailed information about message properties.

Appendix D, *Installing and Configuring ActiveMQ*
: Provides detailed information about installing and configuring ActiveMQ to run the examples in this book.

Software and Versions

This book covers Java Message Service version 1.1. It uses Java language features from the Java 6 platform. Because the focus of this book is to develop vendor-independent JMS clients and applications, I have stayed away from proprietary extensions and vendor-dependent idioms. Any JMS-compliant provider can be used with this book; you should be familiar with that provider's specific installation, deployment, and runtime management procedures to work with the examples. To find out the details of installing and running JMS clients for a specific JMS provider, consult your JMS vendor's documentation; these details aren't covered by the JMS specification. We have provided the details for running the examples with ActiveMQ, a popular open source JMS provider, in Appendix D.

The source code examples and explanation in Chapter 8 refer to the Enterprise JavaBeans 3.0 (EJB 3) specification. The source code examples and explanation in Chapter 9 refer to version 2.5 of the Spring Framework

The examples developed in this book are available through the book's catalog page at *http://oreilly.com/catalog/9780596522049/examples*. These examples are organized by chapter. Special source code modified for specific vendors is also provided. These vendor-specific examples include a *readme.txt* file that points to documentation for downloading and installing the JMS provider, as well as specific instructions for setting up the provider for each example.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Used for filenames, pathnames, hostnames, domain names, URLs, email addresses, and new terms when they are defined.

`Constant width`
: Used for code examples and fragments, class, variable, and method names, Java keywords used within the text, SQL commands, table names, column names, and XML elements and tags.

`Constant width bold`
: Used for emphasis in some code examples.

`Constant width italic`
: Used to indicate text that is replaceable.

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

The term *JMS provider* is used to refer to a vendor that implements the JMS API to provide connectivity to its enterprise messaging service. The term *JMS client* refers to Java components or applications that use the JMS API and a JMS provider to send and receive messages. *JMS application* refers to any combination of JMS clients that work together to provide a software solution.

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 Message Service*, Second Edition, by Mark Richards, Richard Monson-Haefel, and David A. Chappell. Copyright 2009 Mark Richards, 978-0-596-52204-9."

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://my.safaribooksonline.com/*.

How to Contact Us

We have tested and verified the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). Please let us know about any errors you find, as well as your suggestions for future editions, by writing to:

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, or any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596522049/

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

bookquestions@oreilly.com

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

http://www.oreilly.com

Acknowledgments

These acknowledgments are from Mark Richards and refer to the second edition of this book.

No one ever writes a book alone; rather, it is the hard work of many people working together that produces the final result. There are many people I would like to acknowledge and thank for the hard work and support they provided during the project.

First, I would like to recognize and thank my editor, Julie Steele, for putting up with me during the project and doing such a fantastic job editing, coordinating, and everything else involved with getting this book to print. I would also like to thank Richard Monson-Haefel for doing such a great job writing the first edition of this book (along with David Chappell), and for providing me with the opportunity to write the second edition.

To my good friend and colleague, Ted Neward, I want to thank you for writing the Foreword to this book during your very busy travel schedule and for providing me with insight and guidance throughout the project. Your suggestions and guidance helped bring this new edition together. I also want to thank my friends, Neal Ford, Scott Davis, Venkat Subramaniam, Brian Sletten, David Bock, Nate Shutta, Stuart Halloway, Jeff Brown, Ken Sipe, and all the other No Fluff Just Stuff (NFJS) gang, for your continued support, lively discussions, and camaraderie both during and outside the NFJS conferences. You guys are the greatest.

I also want to thank the many expert technical reviewers who helped ensure that the material was technically accurate, including Ben Messer, a super software engineer and technical expert; Tim Berglund, principle software developer and owner of the August Technology Group, LLC; Christian Kenyeres, principle technical architect at Collaborative Consulting, LLC; and last (but certainly not least), Ken Yu and Igor Polevoy. I know it wasn't easy editing and reviewing the manuscript during the holiday season (bad timing on my part, I'm afraid), but your real-world experience, advice, comments, suggestions, and technical editing helped make this a great book.

To the folks at the Macallan Distillery in Scotland, thank you for making the best single malt Scotch in the world. It helped ease the pain during those long nights of writing, especially during the winter months.

Finally, I would like to acknowledge and thank my lovely wife, Rebecca, for her continued support throughout this book project. You mean the world to me, Rebecca, and always will.

Acknowledgments from the First Edition

These acknowledgments are carried over from the first edition of this book and are from the original authors, Richard Monson-Haefel and David A. Chappell.

While there are only two names on the cover of this book, the credit for its development and delivery is shared by many individuals. Michael Loukides, our editor, was pivotal to the success of this book. Without his experience, craft, and guidance, this book would not have been possible.

Many expert technical reviewers helped ensure that the material was technically accurate and true to the spirit of the Java Message Service. Of special note are Joseph Fialli, Anne Thomas Manes, and Chris Kasso of Sun Microsystems; Andrew Neumann and

Giovanni Boschi of Progress; Thomas Haas of Softwired; Mikhail Rizkin of International Systems Group; and Jim Alateras of ExoLab. The contributions of these technical experts are critical to the technical and conceptual accuracy of this book. They brought a combination of industry and real-world experience to bear and helped to make this the best book on JMS published today.

Thanks also to Mark Hapner of Sun Microsystems, the primary architect of Java 2, Enterprise Edition, who answered several of our most complex questions. Thanks to all the participants in the JMS-INTEREST mailing list hosted by Sun Microsystems for their interesting and informative postings.

Special appreciation goes to George St. Maurice of the SonicMQ tech writing team for his participation in organizing the examples for the O'Reilly website.

Finally, the most sincere gratitude must be extended to our families. Richard Monson-Haefel thanks his wife, Hollie, for supporting and assisting him through yet another book. Her love makes everything possible. David Chappell thanks his wife, Wendy, and their children, Dave, Amy, and Chris, for putting up with him during this endeavor.

David Chappell would also like to thank some of the members of the Progress SonicMQ team—Bill Wood, Andy Neumann, Giovanni Boschi, Christine Semeniuk, David Grigglestone, Bill Cullen, Perry Yin, Kathy Guo, Mitchell Horowitz, Greg O'Connor, Mike Theroux, Ron Rudis, Charlie Nuzzolo, Jeanne Abmayr, Oriana Merlo, and George St. Maurice—for helping to ensure that the appropriate topics were addressed, and addressed accurately. And special thanks to George Chappell for helping him with "split infinitives."

About the Authors

Mark Richards is an accomplished author and conference speaker working as a hands-on SOA and enterprise architect in the financial services industry. In addition to numerous published articles, he is the author of *Java Transaction Design Strategies* (C4Media), contributing author of *97 Things Every Software Architect Should Know* (O'Reilly), and contributing author of *No Fluff, Just Stuff Anthology* Volumes 1 and 2 (Pragmatic Bookshelf). He is a recognized authority on messaging, Service-Oriented Architecture, and transaction management. Mark is a regular speaker on the NFJS Software Symposium series and speaks at conferences around the world.

Richard Monson-Haefel is the author of the first five editions of *Enterprise Java Beans* (O'Reilly), the first edition of *Java Message Service* (O'Reilly), and is one of the world's leading experts and book authors on enterprise computing.

David A. Chappell is vice president and chief technologist for SOA at Oracle Corporation. He is well noted for authoring *Java Web Services* (O'Reilly), *Professional ebXML Foundations* (Wrox), and the first edition of *Java Message Service* (O'Reilly).

Colophon

The animal on the cover of *Java Message Service*, Second Edition, is a passenger pigeon (*Ectopistes migratorius*), an extinct species. In the mid-1800s, passenger pigeons were the most numerous birds in North America. Several flocks, each numbering more than two billion birds, lived in various habitats east of the Rocky Mountains. Flocks migrated en masse in search of food, without regard to season, and a good food source could keep a flock in one place for years at a time. John James Audubon observed that nearly the entire passenger pigeon population once stayed in Kentucky for several years and was seen nowhere else during this time.

Whole flocks roosted together in small areas, and the weight of so many birds—often up to 90 nests in a single tree—resulted in the destruction of forests, as tree limbs and even entire trees toppled. (The accumulated inches of bird dung on the ground didn't help.) Such roosting habits, combined with high infant mortality and the fact that female passenger pigeons laid a single egg in a flimsy nest, did not bode well for the long-term survival of the species.

It was humans harvesting passenger pigeons for food, however, that drove them to extinction. In 1855, a single operation was processing 18,000 birds per day! Not even Audubon himself was concerned that the pace might have an adverse effect on the birds' population, but the last passenger pigeon died in the Cincinnati Zoo in 1914.

The cover image is a 19th-century engraving from the Dover Pictorial Archive. The cover font is Adobe ITC Garamond. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSansMonoCondensed.

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：建构与运行》（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）