Second Edition

Web Application Architecture

Principles, Protocols and Practices

Leon Shklar and Rich Rosen

Web Application Architecture

Second Edition

This book delivers a thorough and rigorous introduction to fundamental architectural elements of the web; this knowledge is a critical foundation for any development engineer working on our advanced web applications. Just as important, this book provides a challenging, non-trivial example application demonstrating current best practices which the reader can work through to connect theory to practice. For our engineers coming from other backgrounds to advanced web development, this book has been a very efficient, effective learning tool and I would recommend it highly to others who desire a deep understanding of web architecture and applications.

Chris Corti, Ph.D.
Cisco Systems, Inc.

Web Application Architecture

Principles, Protocols and Practices

Second Edition

Leon Shklar and Rich Rosen

WILEY
A John Wiley and Sons, Ltd., Publication

Other Wiley Editorial Offices

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 42 McDougall Street, Milton, Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01, Jin Xing Distripark, Singapore 129809

John Wiley & Sons Canada Ltd, 6045 Freemont Blvd, Mississauga, Ontario, L5R 4J3, Canada

Wiley also publishes its books in a variety of electronic formats. Some content that appears
in print may not be available in electronic books.

Library of Congress Cataloging-in-Publication Data

Shklar, Leon
Web application architecture : principles, protocols, and practices / Leon Shklar and Rich Rosen. – 2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-470-51860-1 (pbk.)
1. Web site development. 2. Application software–Development. 3. Web sites–Design. 4. Software architecture.
I. Rosen, Rich. II. Title.
TK5105.888.S492 2009
005.1'2–dc22

2008052051

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 978-0-470-51860-1

Typeset in 10/12.5 Times by Laserwords Private Limited, Chennai, India
Printed and bound in Great Britain by Bell and Bain, Glasgow

To my beautiful girls: my wife Rita and daughter Victoria.

To the memory of my parents, Hasya and Arcady Shklar, and my grandparents, Tasya and Boris Korkin.

Leon Shklar

To my parents, Arthur and Toby, and to Celia. Also, to the memory of my high school maths teacher, Jack Garfunkel, who instilled in his students the value of thinking things through logically, and the value of writing not for those who already know what you're talking about, but for those who don't.

Rich Rosen

Contents

About the Authors

Leon Shklar currently works for Thomson Reuters where he is the head of technology for Reuters Media. Previously, Leon headed up the development team for the online edition of the *Wall Street Journal* at Dow Jones. Prior to joining Dow Jones, he spent six years at Bell Communications Research and almost as long in the world of dot-coms and Internet software. Leon holds a Ph.D. in Computer Science from Rutgers University.

Rich Rosen is a senior developer in the Fixed Income Systems Group at Interactive Data Corporation. Previously, he was an Application Architect at Dow Jones. Rich began his career at Bell Labs, where his work with relational databases and the Internet prepared him for the world of Web application development. He is a co-author of *Mac OS X for Unix Geeks, 4th Edition (O'Reilly)*. Rich holds an M.S. in Computer Science from Stevens Institute of Technology.

Preface to the Second Edition

The expression "web time" connotes a world in which rapid change is the norm, where time is exponentially condensed. Technological advances that once upon a time might have taken years to transpire now occur in a matter of months or even days. What's more, these advances often result in radical paradigm shifts that change the way we interact with our technology, and with the world at large.

The first edition of this book was published in 2003. Since then, there have been many technological advances and paradigm shifts, causing some of what we wrote to become dated. New frameworks such as Ruby on Rails have arisen as a reaction to increasing complexity in the application development process. AJAX has taken client-side interactivity to a new level, blazing new frontiers in web application functionality. Search has become a fundamental part of our everyday web experience. Even the core protocols and markup languages representing the foundation of web technology have evolved since we first wrote about them over five years ago. Back then, who could have imagined the ascendance of today's most popular web applications, such as YouTube, Facebook, eBay, and Wikipedia, or the advances in real-time interactivity that are now commonplace?

For the second edition of this book, we provide new material covering these changes in the web technology landscape, while striving to bring existing content up-to-date. We have included new chapters on search technology and client-side interactivity (JavaScript, DHTML, and AJAX). We have added a second sample application implemented using Ruby on Rails as a complement to the original Struts application, which has been updated for the new edition. The chapters on Internet protocols, markup languages, server and browser architecture, and application development approaches have all been revised and enhanced. It is our hope that this updated edition will provide readers with deeper insights into the principles, protocols, and practices associated with the design and development of web applications.

Leon Shklar
Rich Rosen
October 31, 2008

Acknowledgments

I am grateful to my wife Rita for inspiration and music, and for still being around after all the work that went into the two editions of this book. My special thanks to our daughter Victoria for her insightful ideas throughout the project.

Leon Shklar

Ongoing and everlasting thanks to my wife, Celia, for the joy her singing brings into my life, and for enduring the continued insanity associated with the writing process. Thanks also to my parents and to Celia's parents for their love and support.

Rich Rosen

We would both like to thank the following people for their help:

- our editor, Jonathan Shipley, and his assistants, Georgia King and Claire Jardine, for their professionalism and flexibility throughout the project;
- our technical reviewers, Sue Fitzgerald, Ciarán O'Leary, Ilmi Yoon, Roger Beresford, Yuanzhu Peter Chen, Ray Cheung and Wei Ding, for taking the time to examine the text and provide us with valuable insights and advice;
- our colleagues, Otis Gospodnetich and Keith Kim, for their comments and suggestions for the search chapter, and Heow Eide-Goodman for his comments and suggestions for the chapters on development approaches and Rails application development.

CHAPTER 1

Introduction

IN THIS CHAPTER

- History and pre-history of the web
- From web pages to web sites to web applications
- Web 2.0 and AJAX
- Summary of what is covered in the book

OBJECTIVES

- Offer a historical overview of web technology.
- Introduce the concept of ‘‘web applications.’’
- Explain the focus and purpose of the book.
- Provide a chapter-by-chapter outline.

1.1 History and Pre-History of the Web

Back in 1989, at CERN (the scientific research laboratory near Geneva, Switzerland), Tim Berners-Lee presented a proposal for an information management system that would enable the sharing of knowledge and resources over a computer network. We now know this system as the *worldwide web* (the web). Building on the foundation of existing Internet protocols and services, it lives up to its name as a ubiquitous network providing information and communication services to hundreds of millions of people around the world.

From the very beginnings of Internet technology, many people shared the dream of using the Internet as a universal medium for exchanging information over computer networks. Internet file-sharing services (such as *FTP* and *Gopher*) and message forum services (such as *Netnews*) provided increasingly powerful mechanisms for information exchange and brought us closer to fulfilling those goals.

Ted Nelson's *Xanadu* project aspired to make that dream a reality, but the goals were lofty and never fully realized. Nelson coined the term "hypertext" as a way of describing "non-sequential writing – text that branches and allows choice to the reader." Unlike the static text of print media, hypertext was intended for use with an interactive computer screen. It is open, fluid and mutable, and can be connected to other pieces of hypertext by "links".

It took Tim Berners-Lee to "marry together" (in his words) the notion of hypertext with the power of the Internet, bringing those initial dreams to fruition in a way that the earliest developers of both hypertext and Internet technology might never have imagined. His vision was to connect literally *everything*, in a uniform and universal way. Berners-Lee originally promoted the web as a virtual library, a document control system for sharing information resources among researchers. On-line documents could be accessed via a unique document address, a *universal resource locator* (*URL*). These documents could be cross-referenced via *hypertext links*.

From its humble beginnings, the web has expanded exponentially to serve a wide variety of purposes for a wide variety of people:

- Given its origins, it seems natural that educational institutions and research laboratories were among the very first users of the web, employing it to share documents and other resources across the Internet.
- Popular adoption of the web by individuals followed gradually, originally through on-line services, such as America On-line (now AOL) that slowly but surely integrated with the web and the Internet. Initially, personal usage of the web was limited to e-mail and web surfing. Eventually, people began building their own web sites where they posted everything from on-line photo albums to personal journals that would become known as "blogs."
- Over time, businesses saw the potential of the web and began to employ it for *e-commerce*, providing an on-line medium for buying and selling goods interactively. As more and more people were connected to the web, the draw of a new source of revenue and the fear that competitors would get there first and undercut traditional revenue streams made e-commerce increasingly important to business. E-commerce applications are now being used for everything – from displaying catalogs of available merchandise to providing the means for customers to purchase goods and services securely on-line.
- As the impact of e-commerce grew, the back-end applications supporting e-commerce became increasingly important to the companies using it. The front-end, customer-facing web sites needed to be up to date, synchronized with inventory systems so that customers would know what items were or weren't immediately available. Automated fulfillment systems connected to the on-line ordering mechanisms became commonplace. With that, secure login and registration, including collection of credit card information and payment processing, became an absolute requirement for viable e-commerce sites.
- With the maturing of the web, a tug of war arose between sites offering free and paid content. Google was successful in tilting the balance in favor of free content supported by ads. Even the *Wall Street Journal*, which pioneered the notion of paid subscriptions for on-line content, has been

opening up more and more of its content to non-subscribers. Critical technological challenges for free sites include tracking visitor behavior, providing adaptive personalization, and offering advanced community-building tools.

The web did not come into existence in a vacuum. It was built on top of core Internet protocols that had been in existence for many years prior to the inception of the web. Understanding the relationship between web technology and the underlying Internet protocols is fundamental to the design and implementation of true web applications. In fact, it is the exploitation of that relationship that distinguishes a web page or web site from a web application.

1.2 From Web Pages to Web Sites

The explosive growth of the web at least partially can be attributed to its grass-roots proliferation as a tool for *personal publishing*. The fundamental technology behind the web is relatively simple. A computer connected to the Internet running a *web server* was all that was necessary to serve documents. Both CERN and the *National Center for Supercomputer Applications* (*NCSA*) at the University of Illinois had developed freely available web server software. A small amount of *HTML* knowledge (and the proper computing resources) got you something that could be called a *web site*. Early web sites were just loosely connected sets of pages, branching hierarchically from a home page; now, a web site is much more than just a conglomeration of web pages.

When the web was in its infancy, academic institutions and technology companies owned the only computers that were connected to the Internet and could run server software. In those days, personal computers sitting on people's desks were still a rarity. If you wanted access to any sort of computing power, you used a *terminal* that let you "log in" to a large server or mainframe over a direct connection or dial-up phone line.

Still, creating and posting web pages quickly became popular among people that had access to scarce computing power. The original HTML language was simple enough that, even without the more sophisticated tools we have at our disposal today, it was an easy task for someone to create a web page. (Some would say *too* easy.) In the end, all that was needed to create first generation web pages was a simple text editor.

There is a big difference between a bunch of web *pages* and a web *site*. A web site is more than just a group of web pages that happen to be connected to each other through hypertext links:

- First, there are *content-related* concerns. Maintaining thematic consistency of content is important in giving a site some degree of identity.
- There are also *aesthetic* concerns. In addition to having thematically related content, a web site should also have a common look and feel across all of its pages, so that site visitors know what they are looking at. This means utilizing a common style: page layout, graphic design, and typographical elements.
- Then there are *architectural* concerns. As a site grows in size and becomes more complex, it is critically important to organize its content. This includes not just the layout of content on individual pages, but also the interconnections between the pages (site navigation). If the site becomes so complex that visitors cannot navigate their way through, even with the help of site maps and navigation bars, then it needs to be reorganized and restructured.

1.3 From Web Sites to Web Applications

Initially, what people shared over the Internet consisted mostly of static information found in files. They might edit those files, but there were few truly *dynamic* information services.

Granted, there were a few exceptions: search applications for finding files in on-line archives and services that provided information about the current weather, or the availability of cans from a soda-dispensing machine. (One of the first web applications that Tim Berners-Lee demonstrated at CERN was designed to look up numbers in an on-line phone book using a web browser.) However, for the most part, the information resources on the web were static documents.

The advent of the *dynamic web*, which resulted from the proliferation of dynamic information services, changed all that. The new services ranged from CGI scripts to search engines to packages that connected web applications to relational databases. No longer was it sufficient to build a web site (as opposed to a motley collection of web pages). It became necessary to design a *web application*.

What *is* a "web application?" It is a *client–server* application that uses a web browser as its client program. It delivers interactive services through web servers distributed over the Internet (or an intranet). A web site simply delivers content from static files. A web application can present dynamically tailored content based on request parameters, tracked user behaviors, and security considerations.

Web applications power information portals, retail sites, and corporate intranets. They not only provide information and interact with site visitors, but also collect and update information, maintain access controls, and support on-line transactions.

A prime example of a dynamic web application is an on-line shopping site. Site visitors browse on-line catalogs, search for items they want to purchase, add those items to an on-line shopping cart, and place their orders through a secure interface that encrypts personal information. To enable this functionality, the web application communicates with warehouses that maintain catalog and inventory information, storing this information in a database. Users can search the on-line database to obtain information about products. The application provides a way for users to select items to be added to their shopping carts, maintains shopping cart contents for the duration of the browser session, and records order information in the database. It communicates with outside financial institutions to verify credit card and address information, fulfills the order by directing warehouses to ship the purchased items, and sends confirmation e-mails to purchasers allowing them to track shipment of their orders.

1.4 Web 2.0: On-line Communities and Collaboration

Since 2000, another major trend has arisen in the on-line world, incorporating applications that support user-generated content, on-line communities, and collaborative mechanisms for updating on-line content. This trend is often referred to as *Web 2.0*, because it is closely tied to advances in web technology that herald a new generation of web applications.

In many respects, Web 2.0 is a harking back to the web as a network for presenting personal hyperlinked information. Information flow on the web is no longer one way, from the web site to the surfer. Site visitors contribute information of their own, ranging from reviews and ratings for movies, music, and books to personal journals, how-to information, and news for popular consumption. These journals go by the name *blogs* (short for "web logs") and the whole blogging movement has resurrected the idea of the personal web page and elevated its status.

Personal blogs and community web sites encouraging user input may bear a resemblance to the more personal web of old, but the technology behind them does not. Whereas individual owners could easily maintain the simple static web sites of old, this has become impractical given the enormous volume and increasingly malleable nature of content out there today. Blogs, user forums, and collaborative community sites may look like the simpler web sites of the past, but the underlying functionality supporting them is based on sophisticated web application technology incorporating user authentication, access control, and content management services.

1.5 The Brave New World of AJAX

Another radical change in the web application landscape was the advent of *AJAX* (an acronym that stands for Asynchronous JavaScript and XmlHttpRequest). While the technology advances behind it accumulated gradually, AJAX represents a paradigm shift that has changed the way web applications are conceived and developed.

Tim Berners-Lee's original concept of HTTP was that of a simple stateless request–response protocol. Such a protocol allows for independent processing of requests and responses, without requiring a persistent connection. This simplicity fostered wide acceptance of the HTTP protocol but imposed significant limitations. The only method available for updating page content was to replace the entire page. This limitation was unsatisfying to those who were used to client–server applications that communicated with their associated servers directly and continuously. For instance, in such an application, a server-side change to a value in a spreadsheet cell could immediately be reflected on the screen without refreshing the whole spreadsheet.

HTTP, and the web experience in general, have evolved in a number of ways since their inception, introducing elements that allow web transactions to transcend the limitations of a stateless request–response protocol (e.g., cookies and JavaScript). AJAX is the latest such element, essentially allowing what amounts to an "end-run" to communicate with a web server indirectly: instead of submitting a direct request to see a *new* page from the server, a subordinate background request is submitted (usually through a JavaScript object called *XmlHttpRequest*) and the response to that request is used to dynamically modify the content of the *current* page in lieu of replacing its content entirely.

The main impact of AJAX is that the web experience is brought closer to that of client–server applications, allowing for a more immediate dynamic interaction between the user and the web application.

1.6 Focus of This Book

The purpose of this book is to provide a guide for learning the underlying principles, protocols, and practices associated with designing flexible and efficient web applications. Our target audience is senior undergraduate or graduate students who have already learned the basics of web development and professional developers who have had some exposure to web application development and want to expand their knowledge.

We expect our readers to have *some* familiarity with HTML and JavaScript – not at the level of experienced web designers but enough to create web pages and embed simple JavaScript code to submit forms to a web server. We recommend at least some exposure to Internet protocols, such as

Telnet, FTP, SMTP, HTTP and SSL. We appreciate that many of our readers may not be familiar with Linux or Unix operating systems. However, it is helpful to have some understanding of the command-line interfaces provided by interactive shells, as some of our examples employ such an interface.

As we have said, there is a major difference between web *pages*, web *sites*, and web *applications*. There are excellent resources dedicated to web page and web site design, and the References section at the end of this chapter lists some of the best that we know. When we examined the current literature available on the subject of *web application development*, we found there were three main categories of book currently available.

- technical overviews
- reference books
- focused tutorials.

Technical overviews are usually very high level, describing technology and terminology in broad terms. They do not go into enough detail to enable the reader to design and build serious web applications. They are most often intended for managers who want a surface understanding of concepts and terminology without going too deeply into specific application development issues. Frequently, such overviews attempt to cover technology in broad brushstrokes; their subject may be Java, XML, e-commerce, or Web 2.0. Such books approach the spectrum of technology so broadly that the coverage of any specific area is too shallow for serious application developers.

Reference books are useful, naturally, as references, but not for the purpose of *learning* about the technology. They are great to keep on your desk to look things up once you are already deeply familiar with the technology, but they are not oriented toward elucidation and explanation.

The *focused tutorials* concentrate on the usage of specific platforms and products to develop web applications. Books in this category provide in-depth coverage of very narrow areas, concentrating on how to use a particular language or platform without necessarily explaining the underlying principles. Such books may be useful in teaching programmers to develop applications for a specific platform, but they do not provide enough information about the enabling technologies, focusing instead on the platform-specific implementation. Should a developer be called upon to rewrite an application for another platform, the knowledge acquired from these books is rarely transferable to the new platform.

Given the rate of change of the web technologies, today's platform of choice is tomorrow's outdated legacy system. When new development platforms emerge, developers without a fundamental understanding of the inner workings of web applications have to learn the new platforms from the ground up. The challenge is their lack of understanding of what the systems they implemented using specialized application programming interfaces (*API*s) did behind the API calls. What is missing is the ability to use fundamental technological knowledge across platforms.

What was needed was a book that covered the basic *principles* of good application design, the underlying *protocols* associated with web technology, and the best *practices* for creating scalable, extensible, maintainable applications. With this in mind, we endeavored to write such a book.

The need for such a book is particularly apparent when interviewing job candidates for web application development positions. Too many programmers have detailed knowledge of particular languages and interfaces but they are lost when asked questions about the underlying technologies and how they relate to real problems (e.g., why is it necessary for a server to add a trailing slash to a URL and redirect the request back to itself?). Such knowledge is not purely academic – it is critical when designing and debugging complex systems.

Too often, developers with proficiency *only* within a specific application development platform (such as *Active Server Pages, Cold Fusion, PHP*, or *Perl* CGI scripting) are not capable of transferring that proficiency directly to another platform. The fundamental understanding of core technologies is critical to enable developers to grow with the rapid technological advances in web application development.

What do we have in mind when we refer to the *general principles* that need to be understood in order to properly design and develop web applications? We mean the core protocols and languages associated with web applications. This includes *HyperText Transfer Protocol (HTTP)* and *HyperText Markup Language (HTML)*, which are fundamental to the creation and transmission of web pages, but it also includes older Internet protocols such as *Telnet* and *FTP*, and the protocols used for message transfer such as *SMTP* and *IMAP*. A *web application architect* must also be familiar with JavaScript, XML, relational databases, graphic design and multimedia. Web application architects must be well-versed in application server technology and have a strong background in information architecture.

If you find people with all these qualifications, please let us know: we would love to hire them! Rare is the person who can not only design a web site but can also perform all the other tasks associated with web application development: working with graphic designers, creating database schemas, producing interactive multimedia programs, and configuring e-commerce transactions. More realistically, we can seek someone who is an expert in one particular area (e.g., e-commerce transactions or browser programming) but who also understands the wider issues associated with designing web applications.

We hope that, by reading this book, you can acquire the skills needed to design and build complex applications for the web. There is no "one easy lesson" for learning the ins and outs of application design. Hopefully, this book will enhance your ability to design and build sophisticated web applications that are scaleable, maintainable, and extensible.

We examine various approaches to the process of web application development, covering both client-side presentation technology and server-side application technology. On the client side, we look at both markup languages and programming languages, from *HTML* and *XML* to *CSS* and *JavaScript*. On the server side, we look at the full range of approaches, starting with *Server Side Includes (SSI)* and *CGI*, covering template languages such as *Cold Fusion* and *ASP*, examining the intricacies of *Java Platform, Enterprise Edition (Java EE)*, and finally looking at newer "rapid development" approaches such as *Ruby on Rails*. At each level, we concentrate not only on the particular development platform, but also on the underlying principles that span multiple platforms.

1.7 What Is Covered in This Book

The organization of this book is as follows:

- **Chapter 2: Core Internet Protocols** – This chapter offers an examination of the underlying Internet protocols that form the foundation of the web. It offers some perspectives on the history of TCP/IP, as well as some details about using several of these protocols in web applications.

- **Chapter 3: Birth of the Web: HTTP** – The HTTP protocol is covered in detail in this chapter, with explanations of how requests and responses are transmitted and processed.
- **Chapter 4: HTML and Its Roots** – In the first of two chapters about markup languages, we go back to SGML to learn more about the roots of HTML (and of XML). Rather than providing a tutorial on web design with HTML, the focus is on HTML as a markup language and its place in web application development.
- **Chapter 5: XML Languages and Applications** – This chapter covers XML and related specifications, including XML Schema, XPath, XSLT, and XSL FO, as well as XML applications such as XHTML and WML.
- **Chapter 6: Web Servers** – The operational intricacies of web servers is the topic of this chapter, with in-depth discussion of what web servers must do to support interactions with clients such as web browsers and HTTP proxies.
- **Chapter 7: Web Browsers** – As the previous chapter dug deep into the inner workings of web servers, this chapter provides similar coverage of the inner workings of web browsers.
- **Chapter 8: Active Browser Pages: From JavaScript to AJAX** – Here we cover the mechanisms for providing dynamic interactivity in web pages, including JavaScript, DHTML, and AJAX.
- **Chapter 9: Approaches to Web Application Development** – This chapter contains a survey of available web application approaches, including CGI, Servlets, PHP, Cold Fusion, ASP, JSP, and frameworks such as Struts. It classifies and compares these approaches to help readers make informed decisions when choosing an approach for their project, emphasizing the benefits of using the Model–View–Controller (MVC) design pattern in implementing applications.
- **Chapter 10: Web Application Primer: Virtual Realty Listing Services** – Having examined the landscape of available application development approaches, we decide on Struts along with the Java Standard Tag Library (JSTL). We give the reasons for our decisions and build a sample employing the principles we have been discussing in previous chapters. We then suggest enhancements to the application as exercises to be performed by the reader, including the introduction of an administrative interface component, using Hibernate for object-relational mapping, and using Java Server Faces for presentation.
- **Chapter 11: Web Application Primer: Ruby on Rails** – In this chapter, we revisit the *Virtual Realty Listing Services* application and implement its administrative interface using the Ruby on Rails framework. We describe the general structure of a Rails application, walk through the process of building an application in Rails, and compare the Java EE–Struts approach with Rails in terms of both ease of development and flexibility of deployment.
- **Chapter 12: Search Technologies** – Here we describe not only the process of indexing site content for internal search functionality, but also the mechanisms for structuring a site's content for optimal indexing by external search engines. Jakarta's Lucene and other tools are covered.
- **Chapter 13: Trends and Directions** – Finally, we look to the future, providing coverage of the most promising developments in web technology, as well as speculating about the evolution of web application frameworks.

Chapter 2 starts us off with a study of the core protocols supporting Internet technology, in general, and the web in particular. Although some might see this as review, it is a subject worth going over to gain both a historical and a technological perspective.

1.8 Bibliography

Berners-Lee, Tim, 2000. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*. New York: HarperBusiness.

Gehtland, Justin, Galbraith, Ben and Almaer, Dion, 2006. *Pragmatic AJAX: A Web 2.0 Primer*. Raleigh (NC): Pragmatic Bookshelf.

Hadlock, Kris, 2007. *AJAX for Web Application Developers*. Indianapolis (IN): SAMS Publishing (Developer's Library).

Nelson, Theodor Holm, 1982. *Literary Machines 931*. Sausalito (CA): Mindful Press.

Rosenfeld, Louis and Morville, Peter, 2006. *Information Architecture for the World Wide Web*, 3rd Edition. Sebastopol (CA): O'Reilly & Associates.

Williams, Robin and Tollett, John, 2005. *The Non-Designer's Web Book*, 3rd Edition. Berkeley (CA): Peachpit Press.