

Cocoon 2 Programming: Web Publishing *with* *XML and Java*[™]

Bill Brogden
Conrad D'Cruz
Mark Gaither




SYBEX

San Francisco • London



Cocoon 2 Programming:
Web Publishing *with*
XML and Java



Cocoon 2 Programming: Web Publishing *with* *XML and Java*[™]

Bill Brogden
Conrad D'Cruz
Mark Gaither



SYBEX

San Francisco • London

Associate Publisher: Richard Mills

Acquisitions and Developmental Editor: Tom Cirtin

Editor: Linda Stephenson

Production Editor: Leslie E.H. Light

Technical Editor: John-Brian Vyncent

Graphic Illustrator: Jeff Wilson, Happenstance Type-O-Rama

Electronic Publishing Specialist: Maureen Forsys, Happenstance Type-O-Rama

Proofreaders: Monique van den Berg, David Nash, Nancy Riddiough, Emily Hsuan, Laurie OConnell, Yariv Rabinovich

Indexer: Nancy Guenther

Cover Designer: Carol Gorska/Gorska Design

Cover Illustrator/Photographer: Akira Kaede, Photodisk

Copyright © 2003 SYBEX Inc., 1151 Marina Village Parkway, Alameda, CA 94501. World rights reserved. No part of this publication may be stored in a retrieval system, transmitted, or reproduced in any way, including but not limited to photocopy, photograph, magnetic, or other record, without the prior agreement and written permission of the publisher.

Library of Congress Card Number: 2002109624

ISBN: 0-7821-4131-5

SYBEX and the SYBEX logo are either registered trademarks or trademarks of SYBEX Inc. in the United States and/or other countries.

Screen reproductions produced with FullShot 99. FullShot 99 © 1991–1999 Inbit Incorporated. All rights reserved. FullShot is a trademark of Inbit Incorporated.

Internet screen shot(s) using Microsoft Internet Explorer 6 reprinted by permission from Microsoft Corporation.

TRADEMARKS: SYBEX has attempted throughout this book to distinguish proprietary trademarks from descriptive terms by following the capitalization style used by the manufacturer.

The author and publisher have made their best efforts to prepare this book, and the content is based upon final release software whenever possible. Portions of the manuscript may be based upon pre-release versions supplied by software manufacturer(s). The author and the publisher make no representation or warranties of any kind with regard to the completeness or accuracy of the contents herein and accept no liability of any kind including but not limited to performance, merchantability, fitness for any particular purpose, or any losses or damages of any kind caused or alleged to be caused directly or indirectly from this book.

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

SOFTWARE LICENSE AGREEMENT: TERMS AND CONDITIONS

The media and/or any online materials accompanying this book that are available now or in the future contain programs and/or text files (the “Software”) to be used in connection with the book. SYBEX hereby grants to you a license to use the Software, subject to the terms that follow. Your purchase, acceptance, or use of the Software will constitute your acceptance of such terms.

The Software compilation is the property of SYBEX unless otherwise indicated and is protected by copyright to SYBEX or other copyright owner(s) as indicated in the media files (the “Owner(s)”). You are hereby granted a single-user license to use the Software for your personal, noncommercial use only. You may not reproduce, sell, distribute, publish, circulate, or commercially exploit the Software, or any portion thereof, without the written consent of SYBEX and the specific copyright owner(s) of any component software included on this media.

In the event that the Software or components include specific license requirements or end-user agreements, statements of condition, disclaimers, limitations, or warranties (“End-User License”), those End-User Licenses supersede the terms and conditions herein as to that particular Software component. Your purchase, acceptance, or use of the Software will constitute your acceptance of such End-User Licenses.

By purchase, use, or acceptance of the Software you further agree to comply with all export laws and regulations of the United States as such laws and regulations may exist from time to time.

Reusable Code in This Book

The author(s) created reusable code in this publication expressly for reuse by readers. Sybex grants readers limited permission to reuse the code found in this publication or its accompanying CD-ROM so long as the author(s) are attributed in any application containing the reusable code and the code itself is never distributed, posted online by electronic transmission, sold, or commercially exploited as a stand-alone product.

Software Support

Components of the supplemental Software and any offers associated with them may be supported by the specific Owner(s) of that material, but they are not supported by SYBEX. Information regarding any available support may be obtained from the Owner(s) using the information provided in the appropriate read.me files or listed elsewhere on the media.

Should the manufacturer(s) or other Owner(s) cease to offer support or decline to honor any offer, SYBEX bears no responsibility. This notice concerning support for the Software is provided for your information only. SYBEX is not the agent or principal of the Owner(s), and SYBEX is in no way responsible for providing any support for the Software, nor is it liable or responsible for any support provided, or not provided, by the Owner(s).

Warranty

SYBEX warrants the enclosed media to be free of physical defects for a period of ninety (90) days after purchase. The Software is not available from SYBEX in any other form or media than that enclosed herein or posted to www.sybex.com. If you discover a defect in the media during this warranty period, you may obtain a replacement of identical format at no charge by sending the defective media, postage prepaid, with proof of purchase to:

SYBEX Inc.
Product Support Department
1151 Marina Village Parkway
Alameda, CA 94501
Web: <http://www.sybex.com>

After the 90-day period, you can obtain replacement media of identical format by sending us the defective disk, proof of purchase, and a check or money order for \$10, payable to SYBEX.

Disclaimer

SYBEX makes no warranty or representation, either expressed or implied, with respect to the Software or its contents, quality, performance, merchantability, or fitness for a particular purpose. In no event will SYBEX, its distributors, or dealers be liable to you or any other party for direct, indirect, special, incidental, consequential, or other damages arising out of the use of or inability to use the Software or its contents even if advised of the possibility of such damage. In the event that the Software includes an online update feature, SYBEX further disclaims any obligation to provide this feature for any specific duration other than the initial posting.

The exclusion of implied warranties is not permitted by some states. Therefore, the above exclusion may not apply to you. This warranty provides you with specific legal rights; there may be other rights that you may have that vary from state to state. The pricing of the book with the Software by SYBEX reflects the allocation of risk and limitations on liability contained in this agreement of Terms and Conditions.

Shareware Distribution

This Software may contain various programs that are distributed as shareware. Copyright laws apply to both shareware and ordinary commercial software, and the copyright Owner(s) retains all rights. If you try a shareware program and continue using it, you are expected to register it. Individual programs differ on details of trial periods, registration, and payment. Please observe the requirements stated in appropriate files.

Copy Protection

The Software in whole or in part may or may not be copy-protected or encrypted. However, in all cases, reselling or redistributing these files without authorization is expressly forbidden except as specifically provided for by the Owner(s) therein.

Contents at a Glance

| | |
|--|------------|
| <i>Introduction</i> | <i>xv</i> |
| Chapter 1: The Cocoon 2 Architecture | 1 |
| Chapter 2: Uses of Cocoon | 21 |
| Chapter 3: A Review of the Essential Technologies | 55 |
| Chapter 4: The Cocoon Serializers | 91 |
| Chapter 5: Logic Control: The Sitemap | 111 |
| Chapter 6: Introducing XSP Usage | 139 |
| Chapter 7: XSP Logicsheets | 167 |
| Chapter 8: Content Generators | 205 |
| Chapter 9: Configuration for Debugging and Optimization | 241 |
| Chapter 10: Patterns of Presentation | 269 |
| Chapter 11: Patterns of Content Generation | 289 |
| Appendix A: Resources | 309 |
| Appendix B: Sitemap Tag References | 315 |
| Glossary | 321 |
| <i>Index</i> | <i>339</i> |

Contents

Introduction

xv

| | | |
|------------------|---|----------|
| Chapter 1 | The Cocoon 2 Architecture | 1 |
| | The Challenges of Web Publishing | 2 |
| | Dynamic Presentation and Browser Wars | 3 |
| | Enterprise Applications and Dynamic Data | 3 |
| | The Challenges of Web Content Management | 4 |
| | Integrating and Formatting Data from Multiple Sources | 4 |
| | Change Management and Content Management | 5 |
| | Taming the Web Beast | 5 |
| | Content Management Systems to the Rescue | 6 |
| | Separation of Concerns | 6 |
| | Web-Publishing Frameworks and CMSs | 7 |
| | Defining CMS | 7 |
| | Web Server Versus CMS | 8 |
| | A Brief Review of Open-Source CMS Offerings | 8 |
| | The Original Cocoon Project | 9 |
| | Cocoon 1—a Simple Solution | 9 |
| | The Cocoon 1 Architecture: Strengths and Drawbacks | 10 |
| | Architecture of the Cocoon 2 Framework | 11 |
| | An XML-Publishing Framework | 11 |
| | Flexible Content Aggregation | 11 |
| | Pipelines and Components | 12 |
| | Introducing the Cocoon Sitemap | 14 |
| | Matching the URI to a Pipeline | 15 |
| | Processing the Requested URI | 16 |
| | Integrating Business Services | 18 |
| | Separation of Presentation, Content, and Logic | 18 |
| | Summary | 19 |

| | | |
|------------------|--|-----------|
| Chapter 2 | Uses of Cocoon | 21 |
| | Setting Up Cocoon for Application Development | 22 |
| | Two Simple Applications | 26 |
| | Hello Cocoon | 26 |
| | Business Card | 29 |
| | Interfacing with a Database | 32 |
| | Setting Up the JDBC Connection to the Database | 32 |
| | Accessing the Databases | 33 |
| | Site Serving Mobile Devices | 38 |
| | A Menu-Driven Site | 42 |
| | Rendering Scalable Vector Graphics | 48 |
| | A Simple Example | 48 |
| | Rendering Complex Graphics | 50 |
| | Summary | 52 |
| Chapter 3 | A Review of the Essential Technologies | 55 |
| | Developing Content | 57 |
| | The Avalon Project | 57 |
| | The Document Object Model (DOM) | 64 |
| | Simple API for XML (SAX) | 67 |
| | Additional Resources for Content Technologies | 71 |
| | Developing Logic | 71 |
| | XSL Transformations (XSLT) | 71 |
| | Extensible Server Pages (XSP) | 75 |
| | Additional Resources for Logic Technologies | 75 |
| | Developing Presentation | 75 |
| | Cascading Style Sheets (CSS) | 76 |
| | Extensible Stylesheet Language (XSL) | 79 |
| | Extensible HTML (XHTML) | 84 |
| | Additional Resources for Presentation Technologies | 87 |
| | Summary | 87 |
| Chapter 4 | The Cocoon Serializers | 91 |
| | Simple Serializers | 92 |
| | The HTML Serializer | 94 |
| | Example Formatting for WML | 95 |

| | | |
|---|-----------------------------------|------------|
| More-Complex Serializers | 98 | |
| Creating a PDF Page Using FOP | 98 | |
| Output of MS Office Formats | 103 | |
| How to Build a Serializer | 105 | |
| The Avalon Project | 106 | |
| The SAX Interface Hierarchy | 107 | |
| The org.apache.cocoon.xml Package | 107 | |
| The org.apache.cocoon.serialization Package | 108 | |
| Looking at Serializer Examples | 108 | |
| Internationalization | 109 | |
| Language Encoding in Java | 109 | |
| Defining Serializer Encoding | 109 | |
| Language Resource Support | 110 | |
| Summary | 110 | |
| Chapter 5 | Logic Control: The Sitemap | 111 |
| Sitemap Design Principles | 112 | |
| The Contents of a Sitemap | 113 | |
| Sitemap Variables | 114 | |
| Subsitemap Operation | 114 | |
| Compiled Versus Interpreted Sitemaps | 115 | |
| The Components in Detail | 115 | |
| Component Management | 116 | |
| Actions | 117 | |
| Generators | 118 | |
| Readers | 121 | |
| Serializers | 122 | |
| Transformers | 123 | |
| Matchers | 127 | |
| Selectors | 129 | |
| The Resources Element | 132 | |
| The Views Element | 133 | |
| The Action-Sets Element | 133 | |
| Pipelines | 134 | |
| Pipeline Error Handlers | 135 | |

| | | |
|------------------|---------------------------------|------------|
| | Pipeline Elements | 135 |
| | Pipeline Control | 138 |
| | Summary | 138 |
| Chapter 6 | Introducing XSP Usage | 139 |
| | A Minimal XSP Page | 141 |
| | How XSP Works | 142 |
| | Creating Dynamic Pages | 143 |
| | Mixed Content and Logic | 143 |
| | Managing User Sessions | 147 |
| | User Session Management Example | 148 |
| | A SQL Example | 158 |
| | Setting Up the SQL Example | 158 |
| | How the Example Works | 159 |
| | Summary | 165 |
| Chapter 7 | XSP Logicsheets | 167 |
| | How a Logicsheet Works | 169 |
| | Built-In XSP Logicsheets | 170 |
| | Action Logicsheet | 170 |
| | Capture Logicsheet | 172 |
| | Cookie Logicsheet | 173 |
| | ESQL Logicsheet | 176 |
| | Formval | 180 |
| | Log Logicsheet | 182 |
| | Request Logicsheet | 183 |
| | Response Logicsheet | 187 |
| | Sel Logicsheet | 188 |
| | Session Logicsheet | 189 |
| | SOAP Logicsheet | 191 |
| | Util Logicsheet | 192 |
| | XScript Logicsheet | 193 |
| | Building a Custom Logicsheet | 194 |
| | myUtil Logicsheet | 195 |
| | Deploying the myUtil Logicsheet | 199 |
| | Summary | 202 |

| | | |
|------------------|---|------------|
| Chapter 8 | Content Generators | 205 |
| | Generators As Cocoon Components | 206 |
| | The Standard Generators | 206 |
| | The Generator for Error Handling | 208 |
| | Using Scripting Languages with BSF | 209 |
| | The JSPGenerator | 217 |
| | The Velocity Generator | 217 |
| | The Directory Generator Group | 217 |
| | The FileGenerator | 220 |
| | Utility Generators | 222 |
| | Debugging and Performance Tuning | 224 |
| | Creating a Custom Generator | 227 |
| | Sitemap Entries for the SurveyGenerator | 228 |
| | Requirements for a Generator | 229 |
| | The SurveyGenerator Code | 231 |
| | The XSLT Transformation for Surveys | 238 |
| | Summary | 239 |
| Chapter 9 | Configuration for Debugging and Optimization | 241 |
| | Tips for Debugging | 242 |
| | Logging in Cocoon | 243 |
| | Formatting the Output of a LogTarget | 244 |
| | Using the Logs to Debug a Problem | 250 |
| | Spying on the Pipeline | 251 |
| | Cleaning Out the Cache | 256 |
| | Optimizing the Cocoon 2 System | 258 |
| | Modifying the Tomcat Configuration File | 259 |
| | Modifying the cocoon.xconf File | 260 |
| | Modifying the sitemap.xmap File | 260 |
| | Modifying the logkit.xconf File | 261 |
| | Deciding What Gets Served from Cache | 264 |
| | Optimizing and Compiling XSPs | 265 |
| | General Recommendations for Application Design | 265 |
| | Summary | 267 |

| | | |
|-------------------|--|------------|
| Chapter 10 | Patterns of Presentation | 269 |
| | Patterns in Web Programming | 270 |
| | Moving Simple Sites to Cocoon | 271 |
| | Separating Presentation from Content | 271 |
| | Conforming to Standards | 272 |
| | Usability Standards | 273 |
| | The Portal Pattern | 275 |
| | Portal Standardization | 280 |
| | The Forms Problem | 281 |
| | XForms: XML-Based Forms | 283 |
| | Cocoon's XmlForms Project | 285 |
| | The Wiki and Blog Phenomena | 285 |
| | A Blog in Cocoon | 286 |
| | A Wiki in Cocoon | 286 |
| | Client Capability | 287 |
| | Summary | 288 |
| Chapter 11 | Patterns of Content Generation | 289 |
| | Cocoon and J2EE | 290 |
| | Cocoon and the Model-View-Controller Pattern | 291 |
| | JBoss—the Open-Source J2EE Server | 292 |
| | Cocoon and Loosely Coupled Systems | 293 |
| | Java Message Service | 293 |
| | JavaSpaces | 294 |
| | Web Services and SOAP | 296 |
| | Getting a SOAP Stock Quote | 297 |
| | Support for SOAP Programming | 298 |
| | Data Sources | 299 |
| | Relational Databases and JDBC | 299 |
| | Data from Stored Objects | 299 |
| | Business Logic | 306 |
| | The UML Design Tool | 306 |
| | Summary | 308 |

| | | |
|-------------------|------------------------------------|------------|
| Appendix A | Resources | 309 |
| | Standards | 310 |
| | XML Standards at W3C | 310 |
| | Related XML Standards | 311 |
| | Java Standards from Sun | 311 |
| | Java Standards from JCP | 312 |
| | Apache Projects | 312 |
| | Tutorials, FAQs, and Other Goodies | 313 |
| Appendix B | Sitemap Tag References | 315 |
| Glossary | | 321 |
| | <i>Index</i> | 339 |

Acknowledgments

I would like to thank the following people for their help: As always, my wife, Rebecca, my unfailing support for many years. My fellow LANWRights, Inc. employees Ed Tittel and Dawn Rader for their guidance and editorial expertise, respectively. Cheryl Applewood for her help early in the project. Tom Cirtin and Leslie Light of Sybex for constructive criticism and discussion. The many active developers of the Apache Cocoon project and the participants on the users and developers mailing lists.

—Bill Brogden

First and foremost, I would like to acknowledge my parents, Wilma and (late) Albert D’Cruz for their role in shaping my life and career. The following for helping me with issues related to the writing of my chapters: Andreas Hartmann (www.cocooncenter.org) for his advice on an example I created, Andrew Oliver of The Triangle Java Users Group for his unwavering support of the Apache Project and a sounding board for my ideas on Cocoon usage, and Meredith Norris (IBM) for helping me surmount some obstacles in getting an example completed on time. My friends and colleagues who encouraged me through the project. A special thank you to my co-authors for their fresh insights and guidance. Cheryl Applewood and Dawn Rader for showing me the ropes in the publishing world. Tom Cirtin and Ed Tittel for taking care of roadblocks and challenges along the way.

—Conrad D’Cruz

I extend my deepest gratitude and thanks to Ed Tittel for his continued friendship, his overwhelming generosity, his steadfast support, and his Sunday night dinners and 9-ball games. Thanks to my co-authors for their patience and understanding. Thanks to the constant voices on the developers mailing lists. Many thanks to my family for believing in me and supporting me through good and bad times. Thanks to my brother, Paul, for his timely guidance and unwavering resolve. Thanks to Scott Bean for being a great friend and great business partner. Lastly, a special thanks to Mikiel Featherston. With his supreme belief in me as a human and a golfing buddy, through holes-in-one and chilly dips, his continued challenging of my faith, through rainbows and floods, and his pure undeniable joy, through crushing football tackles and fumbles, he continues to inspire me. Now, if he could only putt!

—Mark Gaither

Introduction

The technology for production of Internet applications has undergone rapid evolution in the last few years. One of the most dramatic trends has been the rise and standardization of XML as the most widely accepted method of creating and describing content. This has come about by extraordinary vision and collaboration among the creators and users.

XML is a simplified form of the Standard Generalized Markup Language (SGML), which has been a standard for document markup for some time. SGML is very complete, but much too complex to become a useful tool for everyday programmers. The other descendant of SGML is HTML, but HTML went in the direction of describing the appearance of documents rather than the content. Furthermore, when using HTML, you must stick to a given markup scheme. XML is designed as a language for creating your own specialized markup languages, so it is easy to adapt to just about any type of data.

In 1996, the World Wide Web Consortium (W3C) started the design effort that led to the first XML standard. Jon Bosak from Sun Microsystems was one of the main participants, so it is not surprising that most of the early software tools for working with XML were in Java. As a Java programmer, you have access to an awesome array of tools that just keeps on growing.

Since that first standard, XML-related projects have sprung up like mushrooms after a rain. It seems like everybody can find an application for XML in their particular problem domain. Odds are that no matter which industry you work in, some formal or informal group is attempting to standardize an XML vocabulary for you.

Many of these projects have led to standards endorsed by the W3C. In fact, the last time we looked, the W3C had more than 14 XML-related standardization efforts going on. However, a number of significant approaches have become *de facto* standards outside the normal standards bodies.

The Amazing Cocoon Project

The first version of Cocoon (now referred to as Cocoon 1) got started in 1998 by a group of programmers who had worked on several Java projects as part of the Apache Software Foundation. Stefano Mazzocchi took the lead and developed a simple servlet that used one of the first XSL processors to reformat XML formatted documents for the Web. Cocoon 1 grew

and acquired enough utility that it was used at a number of websites, but it had some architectural weaknesses.

Cocoon 2, which is the version this book covers, was a major redesign with a much more general and flexible architecture. It also draws on a number of other projects at the Apache Software Foundation. In fact, a large number of open-source Java projects working with XML-formatted data have found that Cocoon is exactly what they needed as a framework.

Version 2.0.1 was the release when we started the book and version 2.0.2 came along as we were midway through. We have not attempted to cover any features that are not in the release version 2.0.2. These versions are stable enough for commercial website development and there are already a number of public Cocoon-driven sites.

If there is one problem with Cocoon, it is that it has an inspiring effect on programmers, so there is a continuous ferment of suggestions for the next versions. There will probably be another major redesign with 2.1. Your best bet is to select a version and stick with it.

There are so many XML-related initiatives in the world of programming these days that it would be impossible to catalog them. However, due to the stability of the basic XML standard as maintained by the W3C, Cocoon programmers will be well positioned to take advantage of the latest developments.

Support for the Book

LANWrights will be maintaining a website for the book at:

www.lanw.com/books/cocoonbook/

That site will probably be running on Cocoon by the time you read this. We will be posting any errata and revisions to the code samples.

Sybex has published all the code used in this book on their website at www.sybex.com. Search for this book using the title, author or the book number (4131) and click the Downloads button. Once you have accepted the license agreement, you'll be able to download any of the code listed in this book, organized by chapter.

Before we go any further, let's clarify a few things. Nothing in this book is going to teach you XML, XSLT, or any other Xwhatever basics. You will need to be comfortable with XML 1.0 2nd edition, including namespaces. We provide an overview of XSLT, CSS, and other technologies used by Cocoon, but you will need other reference books for complete coverage of these subjects.

You should also be familiar with Java 1.3 or 1.4 and the servlet API. If you don't already have a servlet-capable web server, you should plan on getting one. Currently, the most commonly used servlet engine among Cocoon users is Tomcat.

Contacting the Authors

We would be delighted to hear from readers with suggestions, reports of errors in the text, or your Cocoon-related announcements.

You can contact Bill Brogden at wbrogden@bga.com; his personal web page is at:
www.bga.com/~wbrogden/


You can contact Conrad D'Cruz at conrad.dacruz@netswirl.com; his website is at:
www.netswirl.com

You can contact Mark Gaither at mark@markgaither.com; his website is at:
www.markgaither.com



CHAPTER I

The Cocoon 2 Architecture

- The Challenges of Web Publishing
 - The Challenges of Web Content Management
 - Content Management Systems to the Rescue
 - The Original Cocoon Project
 - Architecture of the Cocoon 2 Framework
- 

Solutions for web publishing abound and have been rapidly evolving to fill the needs of both developers and their customers. All of these solutions come down to systems for managing content and serving it up on the Web upon request in a format easily accessible to a variety of users.

This chapter introduces the Cocoon framework by defining the original design goals and architecture of Cocoon 1 and studying the areas where the framework fell short. The chapter also provides an overview of the Cocoon 2 architecture and lays the foundation for developing robust web applications using the framework. In the course of this chapter and the rest of the book, we highlight the benefits that the Cocoon framework brings to most web-publishing efforts and why most projects benefit from it.

So why Cocoon 2? Because it's a flexible, powerful solution based on XML—the language of the Web—and related technologies. It is part of the Apache Project, making it fully capable of managing content and presentation for large-scale web publishing.

To fully understand Cocoon's place in the scheme of things, however, we must take a brief look at the origins of the World Wide Web and the technologies that supported the adoption of the Web as the *de facto* standard for information repositories around the world. Ever since the mid-1990s, the popularity of the Web has helped to shape the evolution of strategies for business. This chapter emphasizes the associated problems that came about and the issues that have created challenges for technical and business organizations as they tried to adapt to the constantly changing needs that were created by the Web.

Finally, we take a look at the challenges of developing and maintaining web applications and the impact those challenges have had on development teams. We list the features of a good publishing framework and define the additional requirements of a true content management system (CMS). We also list the most popular open-source publishing frameworks currently available and provide references for each.

The Challenges of Web Publishing

The TCP/IP network called the Internet had been in use since the late 1960s by an exclusive group of individuals affiliated with the government, academia, research institutions, and technology companies. The costs and complexity associated with using this network prevented the average person from understanding and using it as a tool for increasing personal productivity. In 1992, the adoption of the Hypertext Markup Language (HTML), derived from the more complex Standard Generalized Markup Language (SGML), caused a lot of excitement in the technology and business communities. The complexity of using the Internet was abstracted by the creation of a network of servers that could be used for publishing documents based on HTML and its parent, SGML. This network of information servers came to be known as the World Wide Web.

Any document could be marked up using a set of standardized tags and published to a web server. A remote client browser application could access this document, interpret the tags, and then display the information. The web developer predefined the format of the document and changing the presentation involved modifying the markup tags in the document. Vendors began providing their web server applications and browsers for free and this resulted in the proliferation of web technologies. The ease with which a developer could mark up a document and the availability of tools for the nontechnical content developer made HTML a very popular language for web publishing. HTML soon came to be the *lingua franca* of the Web.

Dynamic Presentation and Browser Wars

With the adoption and increasing popularity of HTML came the inevitable battle among vendors for control of the standard. In a move to protect their installed bases, vendors began to support nonstandard tags and server configurations. Developing content for consumption within an intranet was slightly less complicated, because most organizations exercised some control over the browser versions deployed within their enterprises. As time went by, the number of browser vendors increased, as did the versions of a particular browser. The nightmare had only just begun for content developers.

HTML was an excellent mechanism for authoring and rendering static information pages with graphical images, but it had some serious limitations. It did not address dynamic presentation or provide any way for the application to access data from external sources. Dynamic presentation and browser customization was made possible by the introduction of support in HTML for scripting languages like JavaScript and VBScript. Server-side technologies, such as the Common Gateway Interface (CGI), Java Servlet technologies, and a few others, enabled dynamic data to be displayed in browsers. Prior to the introduction of these technologies, the website was a tool for information distribution and was only used to support the marketing functions of an organization. Not long after dynamic content made its way into browsers, the industry presented strategies and products to help organizations make money on the Web. Up until that time, only the big companies could afford to conduct business electronically with expensive and proprietary technologies based on the Electronic Data Interchange (EDI) standard. The new web applications allowed even the smallest players to get into the electronic commerce arena.

Enterprise Applications and Dynamic Data

Web application servers became more sophisticated and could now perform multiple tasks, such as serving up information pages customized for a particular browser and embedding data extracted from one or more data sources in those pages. The servers allowed users to initiate transactions and execute business processes—for example, making a retail purchase or placing a bid on an auction site. The user did not need to navigate to each business server, extract individual pieces of information, and manually correlate all relevant data. The service

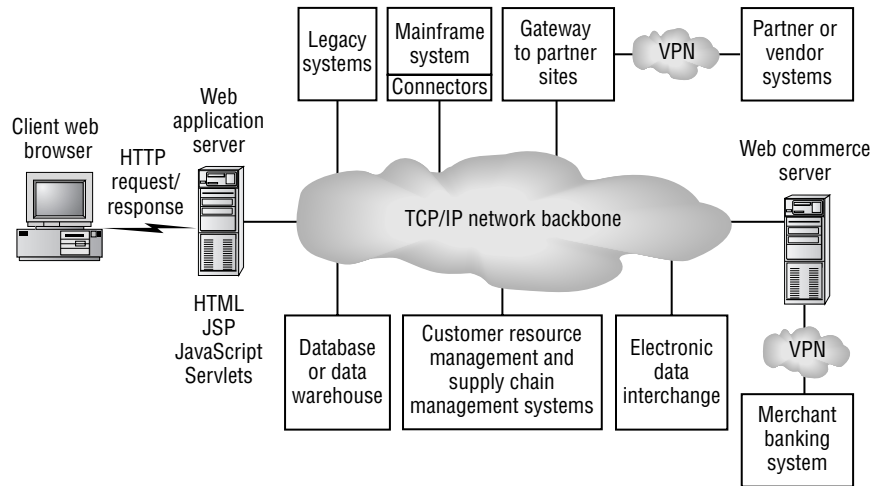
provider had to develop an application that collected all the data behind the scene and presented the user with an integrated and unified web page with the data requested.

The web browser paradigm abstracted the complexity of the organization's technical infrastructure by presenting the results of all operations in the form of a web page with some dynamic presentation data objects. The web server was promoted from the old role of a static electronic brochure to a dynamic catalog of the organization's products and services. It now provided online service delivery and managed user sessions and transactions. It also provided a front end for managing user relationships and communications with the organization.

Figure 1.1 shows a layout of a web application that interfaces with different enterprise services and data stores. It shows the web server as the repository of all the parts of the web application.

FIGURE 1.1:

A typical enterprise web application



The Challenges of Web Content Management

Figure 1.1 shows the complexity and needs of a typical web application. This section looks at the challenges facing web application developers. It discusses the issues associated with integrating enterprise services as well as the problems associated with managing change, and highlights some of the ways development teams are dealing with these issues.

Integrating and Formatting Data from Multiple Sources

One critical issue that presented a challenge when developing web applications that interfaced with enterprise applications and legacy systems was that data in most enterprises was not available in a single, easy-to-use format. The data had to be extracted, modified, and

reformatted before it was suitable for presentation in a web page. Java offered some relief by providing support for developing complex multitier applications. Technologies like Java Database Connectivity (JDBC), Servlets, and Java Server Pages (JSP) provide a means to seamlessly integrate disparate legacy applications and database systems while sticking closely to the web presentation paradigm—that is, output in the form of HTML pages.

This was a double-edged sword and highlighted another critical issue: In an HTML page, content and presentation are so closely tied together that it is difficult for external applications to extract the data embedded in both static and dynamically generated pages. Both these issues were partially addressed by the adoption of Extensible Markup Language (XML) as the standard for modeling data in enterprise and web applications. The use of XML allowed for separation of content and presentation. Because XML documents are normal text files, Java Servlets can be used to dynamically produce XML instead of HTML. Customized presentation was achieved through the use of either Cascading Style Sheets (CSS) or Extensible Stylesheet Language (XSL).

Change Management and Content Management

All these requirements added to the challenges of designing and implementing web applications. As the size and complexity of applications increased, the two issues that moved to the forefront were content management and change management. Content management deals with managing all the objects and parts that go into building the complete application. The use of Java and XML technologies enables the developer to produce robust and easily integrated building blocks for the application. Change management addresses the issues associated with modifying or enhancing the application after it is in production. Some organizations have very stringent requirements, which dictate that changes happen frequently and must be implemented very quickly. This not only includes changes to the dynamic data on the enterprise systems but also how the data gets presented on the pages.

Tighter integration between the web application pages and the enterprise applications or data systems allows for more dynamic data to be available in the final presentation, as in the case of inventory control and purchasing websites. Most content pages have dependencies elsewhere in the same application, or some other application on the same web server, or other applications on remote servers. There is a ripple effect when changes are made to these applications, and often a simple change affects multiple documents, servers, and departments within an organization. The complexity and size of an average dynamic website necessitates either a larger change management team or the automation of the process of change management.

Taming the Web Beast

Most organizations responded to the challenges by increasing the size of the content management team and settling on standardized technologies as the basis for their applications.

An implied challenge was choosing development tools that strictly adhered to the standards and training the entire development team in the use of these tools. Despite all the planning and strategies, content changes rarely could keep pace with the rapidly changing business needs. The limitations of the tools and the change management process often resulted in teams failing in their efforts to manage and maintain web applications. This resulted in a higher cost of ownership, decreased satisfaction on the part of the business sponsors, and an overall reduction in team morale.

Content Management Systems to the Rescue

The Web had proved its potential as a tool for delivery of information and services over the Internet. Organizations of all shapes and sizes quickly jumped on the bandwagon and adopted web strategies. With this came the big push to web-enable products and enterprise applications. Even vendors of proprietary solutions caught the wave and began offering applications designed exclusively for the Web, or at the very least, providing a web front end for their products. The large sites were rich in data and dynamic presentation and created new challenges for the teams designing and maintaining them.

The emphasis was now on teams that were composed of specialists in a particular part of the web application. One such team would specialize in the look and feel of the web pages and the related technologies like HTML and scripting languages. Another team would specialize in integrating enterprise applications into the web application. Newer technologies and standards were hastily put together to support the development and maintenance of large websites. However, this strategy came with its own set of problems. The size of the teams increased and the people-management aspects became a big issue. There were also problems of bridging the training and knowledge gap between teams.

Separation of Concerns

There were sections of the web application that included pieces that mixed the two functions. For example, Java Servlets provided the means to integrate with enterprise applications but would also be used to create the dynamic output to the client. This placed additional pressures on the application integrators because they now had to be concerned with graphical user interface issues. One of the design goals of Java Server Pages (JSP) was to separate content from presentation. However, this came up short and only added to the confusion of the development teams. JSP put additional pressures on the presentation developers because they now had to learn the syntax of Java and had to deal with programming issues. An interim solution was to create cross-functional teams in which members were trained in presentation and application programming, but this was a very expensive solution. There was a need for a solution that would allow the separation of the three areas: content, logic, and presentation.

Web-Publishing Frameworks and CMSs

A web search on the term *content management system* (CMS) reveals a long list of products and frameworks that advertise an easy way to manage the large quantities of information and the services that an enterprise has to offer to its customers. Several of the products and organizations use the term *content management systems* interchangeably with other terms like *content-publishing frameworks*, *web-publishing frameworks*, and *XML-publishing frameworks*.

This section provides a general definition of CMSs that encompasses all the design requirements of a good system. The definition highlights the features of the system that address and solve the challenges created by the surge in the popularity of the Web as a vehicle for delivering information and services. We also name a few of the products available and then shift our focus to the Cocoon architecture. We have chosen to focus on this one framework because it is currently one of the most powerful publishing systems and has the potential of becoming a CMS.

Defining CMS

A CMS is a framework that allows the creator or owner of an information delivery application to effectively manage all the pieces that go into building that application and that define formal processes that support the entire lifecycle of an application. CMSs and web-publishing frameworks offer solutions to most of the common business problems discussed earlier that are associated with developing and maintaining large web applications. They address the issues of rapid application development that offer flexibility in the look and feel as well as added functionality. Using a CMS as the central focal point for web application development strategy allows an organization to produce flexible and scalable applications in a cost effective and timely manner.

A CMS also allows seamless integration of tools and strategies that enable the creation of routine maintenance or upgrades, and the eventual retirement of the application. The CMS integrates data from multiple applications and services and provides a flexible mechanism to format the output and present it to the user. The technologies and strategies might be standards-based or proprietary. The systems that have the term *web* in the name support the standard web technologies and, at the very minimum, use HTML and the Hypertext Transfer Protocol (HTTP). In addition to supporting the Web, the frameworks can support alternative delivery mechanisms, such as dedicated client applications, whether they are stand-alone windows, fat clients, or Java applets embedded in web pages. They also support the integration of the output into another application framework such as Web Services. A few of the products satisfy only a subpart of the definition and might rely on proprietary technologies for the management and delivery of the information and services. Some of the frameworks include specific technologies in the product name itself, such as XML, to emphasize support of the standards for formatting and presenting documents and data. This focuses on technologies that are standards-based and products that are part of the open-source development paradigm.

Web Server Versus CMS

Some of the products attempt to classify regular web information servers as CMSs. This might not be a wrong classification, because creating and managing websites that serve up HTML and images is not trivial given the potential size to which the website can grow. Change management in websites is a significant issue because hyperlinks in HTML pages create interdependencies and any process that enables an easy maintenance strategy needs to get honorable mention. However, the term *content management* is more encompassing and addresses systems and processes for managing information in many different formats and access to enterprise services from multiple sources.

Choosing a framework that offers all the features of a CMS for a website with static HTML content might be overkill, especially if the content is not expected to change over time and the pages do not need to integrate data from external applications. The complexity of configuring and maintaining such HTML content sites using a CMS cannot be cost-justified in the long run. There are several commercial or open-source web servers that are more cost effective and suited for simple websites serving up static HTML pages and images with a little dynamic presentation using scripting languages like JavaScript or VBScript.

A Brief Review of Open-Source CMS Offerings

The following are brief descriptions of some of the open-source CMSs that are currently available and the web addresses of the organizations that support the development of each. It is not a comprehensive list, and their appearance is not meant to be an endorsement over any other open-source products.

XPS Extensible Publishing System from Wyona is an application that uses XML and Java technologies to manage documents and images on the server. Wyona supports XPS at www.wyona.org.

eZ Publish From eZ Systems, eZ Publish is advertised as an information management system with the data residing in a database. eZ Publish is created and maintained by eZ Systems at <http://developer.ez.no>.

Zope Developed by the Zope community, Zope enables teams to collaborate in the creation and management of dynamic web-based business applications. Zope is offered by the Zope community at www.zope.org.

Cofax Created by the Content Object Factory, Cofax is advertised as a web-based text and multimedia publication system. Cofax is available from the Content Object Factory at www.cofax.org.

Midgard From the Midgard Project, Midgard is defined as a toolkit for building dynamic applications for powering e-business and information management processes. Midgard is available at www.midgard-project.org.

MMBase MMBase offers a flexible solution for creating and maintaining big websites easily. MMBase can be found at www.mmbase.org.

OpenCms From the OpenCms Project, OpenCms is advertised as a Java-based web CMS and emphasizes the ease of creating and publishing web content. OpenCms can be found at www.opencms.org.

Why Choose Open-Source Projects?

As stated earlier, we emphasize the open-source frameworks that are helping shape the field of content management. Open-source systems are popular with most enterprises because of the lower costs of ownership. These systems are based on standardized technologies and are vendor independent. Most of the systems have a process for soliciting needs and requirements using a community process. The features invariably are based on needs that solve real problems and not on what a vendor decides is good for you. The developers of the frameworks are professionals in the field who have solved the content management challenges in their work environment. They bring a wealth of knowledge and experience from their careers to the development of these systems.

The Original Cocoon Project

The Cocoon Project started as an attempt to organize and control the documentation of all the projects being run under the Apache umbrella project. The first iteration was simple and was based on proposed technologies that had not yet been standardized. As the technologies evolved, the framework evolved along with it to utilize the new standards and include more developer requirements. The project recently released the second generation of the product, Cocoon 2, which offers more flexibility and features. We start with the first release, Cocoon 1, and examine the design goals, successes, and drawbacks.

Cocoon 1—a Simple Solution

By design, Cocoon 1 (C1) was based on the open technologies adopted by the Apache Software Foundation and utilized existing frameworks. C1 was a publishing framework that was written completely in Java. It was based on technologies standardized by the Worldwide Web Consortium (W3C), such as Document Object Model (DOM) parsing, XML for formatting data, XSLT for transforming data and merging XML documents, and Extensible Stylesheet Language (XSL) for presentation.

Cocoon was originally a very simple Java Servlet with approximately 100 lines of code and the format for the documents was XML. It used the IBM XML4J parser for parsing of XML

documents and the LotusXSL parser for transforming the XML file using an XSL stylesheet. The next chapter goes into the details of the technologies that are the building blocks of the application, but we mention each of them in this section. The framework was defined with the adopted standard at that time. When the need arose for a server engine that would utilize XSL for transforming XML documents, the project was formally adopted by a vote on the jserv-dev mailing list and named the Cocoon Project under the Apache umbrella.

The Cocoon 1 Architecture: Strengths and Drawbacks

The Apache Avalon Project is part of the Jakarta Project and was an effort to allow developers of open-source projects to collaborate and share code easily. It created a common extensible framework and a set of pure Java components that could be extended to create new applications. C1 was based on the Apache Avalon framework and continued to have a simple structure with very little code. It was primarily used to demonstrate the importance of XSL and XML in web-based publishing. The Apache Xerces parser replaced the XML4J parser and the Apache Xalan parser replaced the LotusXSL parser. As the number of developers grew and additional requirements were added to the design goals, the simple servlet evolved into a complete XML-based publishing system. The framework was adopted widely and was used in production websites all over the world. The strengths of the framework were its simplicity and the fact that it was based on existing, popular frameworks.

However, the framework was based on XML technologies that were still in their infancy. The available parsers were based on the DOM and had several critical architectural issues. Performance was an issue because of the use of the DOM parser, which parses the XML document and creates a tree in memory. This also created greater demands on the server resources when multiple documents had to be served up to several concurrent users.

These drawbacks were not unique to the Cocoon Project and were based on the limitations of the technologies available at that time. Developers of other XML-based applications were experiencing similar problems. The W3C is the body that introduces, regulates, and adopts standards for the Web. To address the problems associated with XML, several new standards were proposed and adopted.

One of new standards addressed how XML documents would be parsed using inline Simple API for XML (SAX) events, which would eliminate the need for creating the object model in memory on the application server. Another change involved splitting up the XSL standard to address three different areas: XSL Transformations (XSLT), Formatting Objects (XSL:FO), and XPath for defining subparts of an XML document. Armed with these new standards, the Cocoon developers embarked on a two-year project to redesign the framework that solves the architectural problems of Cocoon 1 while adding new and interesting features that expanded the system's capabilities.