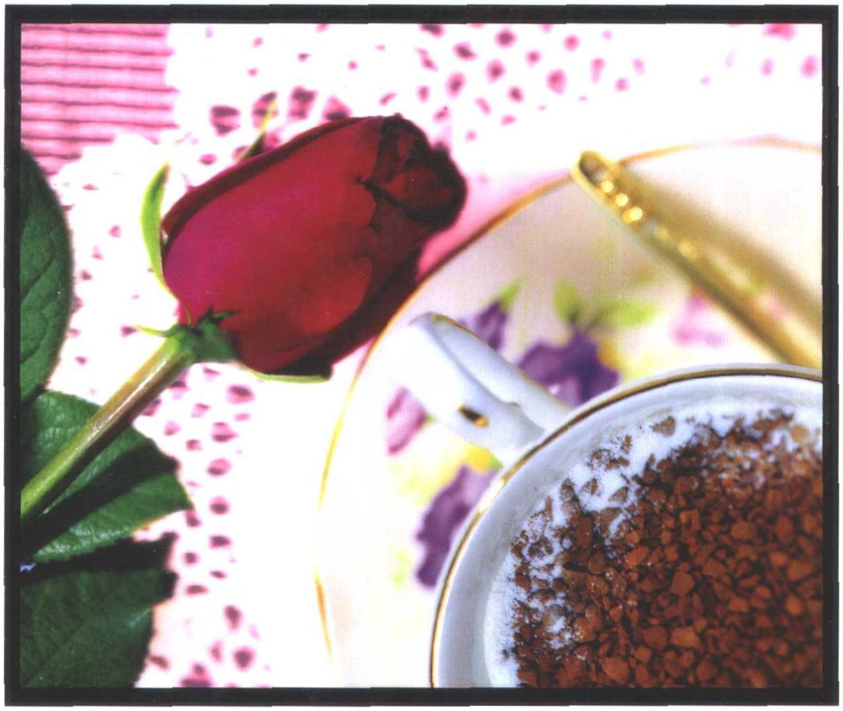
Java程序员书库

用Java处理XML

（影印版）

Processing XML with Java™

A Guide to SAX, DOM, JDOM, JAXP, and TrAX



(美) Elliotte Rusty Harold 编著

Pearson Education

科 学 出 版 社

www.sciencep.com

Java 程序员书库

用 Java 处理 XML

（影印版）

Processing XML with Java™
A Guide to SAX, DOM, JDOM, JAXP, and TrAX

（美）　Elliotte Rusty Harold　编著

科 学 出 版 社
北 京

图字：01-2003-7655 号

内 容 简 介

本书简单回顾了 XML 基础知识，重点介绍了用 Java 处理的 XML 应用程序，包括简单 XML 应用程序、文档对象模型和 JDOM 等。

本书通俗易懂，便于读者掌握用 Java 处理 XML 的技术，适用于 Java 程序员和自学者。

图书在版编目(CIP)数据

用 Java 处理 XML =Processing XML with Java^{TM}: A Guide to SAX,DOM,JDOM,JAXP,and TrAX/（美）Elliotte Rusty Harold 编著.影印本.—北京：科学出版社，2004
（Java 程序员书库）
ISBN 7-03-012495-2

Ⅰ.用…　Ⅱ.H…　Ⅲ.①Java 语言—程序设计—英文②可扩充语言，XML—程序设计—英文
Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2003）第 103030 号

策划编辑：李佩乾/责任编辑：舒　立
责任印制：吕春珉/封面制作：北新华文平面设计室

科学出版社 出版
北京东黄城根北街16号
邮政编码：100717
http://www.sciencep.com
双青印刷厂 印刷
科学出版社发行　各地新华书店经销
*
2004 年 1 月第　一　版　　开本：787×960　1/16
2004 年 1 月第一次印刷　　印张：69 1/4
印数：1—3 000　　字数：1 315 000

定价：108.00 元

(如有印装质量问题,我社负责调换<环伟>)

影印前言

程序设计在于处理复杂性：问题的复杂性和所用的程序设计工具的复杂性。Java 的魅力在于其本身的低复杂性，同时又能很好地处理高度复杂的问题。Java 程序的开发周期只有类似的 C++程序的一半甚至更少，而且 Java 可以方便地处理复杂软件问题：多线程、分布式、跨平台、安全性等。Java 从诞生到现在，已经广泛应用于几乎所有类型软件系统的构建。尤其在基于 Web 的系统开发中，Java 技术具有独特的优势。熟悉 Java 历史的人都知道，Java 的前身——编程语言 Oak 就是致力于电子产品互连的语言，Internet 的发展导致了 Oak 的重生和 Java 的广为流行。现在，J2EE 技术已经成为企业级 Web 应用系统的标准平台。

好的程序设计人员不仅仅要掌握优秀的编程工具，更需要掌握优秀的编程思想。随着面向对象编程技术数十年的发展，人们开始提炼和总结面向对象编程中行之有效的、具有一定普遍意义的方法，即面向对象的设计模式。以 Gamma、Helm、Johnson 和 Vlissides 合著的经典书籍《设计模式》为开端，面向对象设计模式的研究和应用成为面向对象程序设计的重要内容。所有结构良好的面向对象软件系统都包含了大量的设计模式，能否熟练应用设计模式已经成为衡量程序员水平的至关重要的标准。

本丛书收录了与 Java 程序设计和 Java 设计模式相关的经典书籍，反映了应用 Java 开发软件系统的最佳经验总结和最新动态。

《Java 设计模式》采用方便而简洁的编写风格，以可视化的 Java 程序为例，详细介绍了 Gamma、Helm、Johnson 和 Vlissides 合著的经典书籍《设计模式》中列出的所有 23 种模式。通过本书，Java 程序员可以迅速了解和掌握设计模式的内容，并在实践中应用设计模式来创建复杂而健壮的 Java 程序。

《Java 模式应用》介绍了基于模式的开发技巧，演示了使用 Java 开发各种商务系统中的模式应用。书中首先概述设计模式，然后就四种主要模式——创建模式、行为模式、结构模式和系统模式展开了详细的论述。该书还针对系统构建过程中常用的 J2EE、JSP、EJB 和 API 等技术作了介绍，适合具有一定编程基础的 Java 程序员阅读参考。

《J2EE 核心模式》是 Sun Java Center 的资深设计师的 J2EE 亲身实践经验的总结。该书主要描述 J2EE 关键技术（Java Server Pages、Java Servlet、Enterprise Java Beans、Java Message Services 等）的模式、最佳实践、设计策略和经过验证的解决方案。该书介绍了 J2EE 包括的 15 个模式的分类和大量的策略，不仅具有理论深度，而且非常实用。该书内容适合 J2EE 的爱好者、程序员、设计师、开发者和技术管理者。一句话，该书

适合于设计、构建和开发 J2EE 平台应用的所有人。

XML 是新一代文档的标准，Web 页、数据、源码等等，均可以用 XML 文档表示。越来越多的程序员正在使用 Java 来处理 XML 文档。《用 Java 处理 XML》详细论述了如何使用 Java 来读写 XML 文档。该书是目前最新和最全的 Java 处理 XML 技术的介绍，包含内容超过 1000 页的关于 SAX, DOM, JDOM, JAXP, TrAX, XPath, XSLT, SOAP 等的讲解。该书适合于使用 Java 读写 XML 文档的 Java 程序员。其内容从基本概念到高级应用无所不包，特别适合作为手册随时参考。

在《实时 Java》中，作为 RTSJ 专家组的成员之一，Dibble 从 Java 平台特有的实时问题概述开始，依次讲解了 RTSJ 各项主要特性的使用方法。从广泛的实时原理到详细的编程隐患，该书覆盖了构建有效实时程序所需的一切知识。其主要内容包括：与非实时代码的互操作性、实时开发中的取舍以及 JVM 软件的实时问题；垃圾收集、无堆栈访问、物理内存和“不朽”内存以及无堆栈内存的常数时间分配；优先级调度、期限调度以及速率单调分析；闭包、异步传输控制、异步事件以及计时器。这是一本非常实用的指南，适用于有经验的 Java 平台开发人员。

《Java 数据结构与算法分析》介绍了常见的数据结构，如链表、堆栈、队列、树和哈希表等，并对查找和排序进行了算法分析，给出了相应的 Java 实现。该书逻辑结构严谨，主次分明，可用作程序员参考书。

总之，这套书详细介绍了 Java 应用的许多重要方面：从具有普遍性的 Java 数据结构和算法、Java 设计模式、Java 模式应用、J2EE 核心模式，到日益显著的 Java 特殊应用领域（Java 处理 XML 文档和 Java 实时系统开发）。其内容具有一定的理论深度，更有重要的实际参考价值。

有鉴于此，特向 Java 系统开发和应用领域中不同程度的读者推荐这套书，相信每位有心的读者都能得到物超所值的收获。

清华大学经济管理学院管理科学与工程系　朱涛　博士
讲授课程：Java 程序设计，面向对象的分析设计方法

Preface

One night five developers, all of whom wore very thick glasses and had recently been hired by Elephants, Inc., the world's largest purveyor of elephants and elephant supplies, were familiarizing themselves with the company's order processing system when they stumbled into a directory full of XML documents on the main server. "What's this?" the team leader asked excitedly. None of them had ever heard of XML before, so they decided to split up the files among them and try to figure out just what this strange and wondrous new technology was.

The first developer, who specialized in optimizing Oracle databases, printed out a stack of FMPXMLRESULT documents generated by the FileMaker Pro database where all the orders were stored, and began poring over them. "So this is XML! Why, it's nothing novel. As anyone can see who's able, an XML document is nothing but a table!"

"What do you mean, a table?" replied the second developer, well versed in object-oriented theory and occupied with a collection of XMI documents that encoded UML diagrams for the system. "Even a Visual Basic programmer could see that XML documents aren't tables. Duplicates aren't allowed in a table relation, unless this is truly some strange mutation. Classes and objects are what these documents are. Indeed, it should be obvious on the very first pass. An XML document is an object and a DTD is a class."

"Objects? A strange kind of object, indeed!" said the third developer, a web designer of some renown, who had loaded the XHTML user documentation for the order processing system into Mozilla. "I don't see any types at all. If you think this is an object, then it's your software I refuse to install. But with all those stylesheets there, it should be clear to anyone not sedated that XML is just HTML updated!"

"HTML? You must be joking" said the fourth, a computer science professor on sabbatical from MIT, who was engrossed in an XSLT stylesheet that validated all of the other documents against a Schematron schema. "Look at the clean nesting of hierarchical structures, each tag matching its partner as it should. I've never seen HTML that looks this good. What we have here is S-expressions, which is certainly nothing new. Babbage invented this back in 1882!"

"S-expressions?" queried the technical writer, who was occupied with documentation for the project, written in DocBook. "Maybe that means something to those in your learned profession. But to me, this looks just like a FrameMaker MIF file. However, locating the GUI does seem to be taking me a while."

And so they argued into the night, none of them willing to give an inch, all of them presenting still more examples to prove their points, but none bothering to look at the others' examples. Indeed, they're probably still arguing today. You can even hear their shouts from time to time on xml-dev. Their mistake, of course, was in trying to force XML into the patterns of technologies they were already familiar with rather than taking it on its own terms. XML can store data, but it is not a database. XML can serialize objects, but an XML document is not an object. Web pages can be written in XML, but XML is not HTML. Functional (and other) programming languages can be written in XML, but XML is not a programming language. Books are written in XML, but that doesn't make XML desktop publishing software.

XML is something truly new that has not been seen before in the world of computing. There have been precursors to it, and there are always fanatics who insist on seeing XML through database (or object, or functional, or S-expression) colored glasses. But XML is none of these things. It is something genuinely unique and new in the world of computing; and it's possible to understand it only when you're willing to accept it on its own terms, rather than forcing it into yesterday's pigeonholes.

There are a lot of tools, APIs, and applications in the world that pretend XML is something more familiar to developers—that it's just a funny kind of database, or just like an object, or just like remote procedure calls. These APIs are occasionally useful in very restricted and predictable environments; however, they are not suitable for processing XML in its most general format. They work well in their limited domains, but they fail when presented with XML that steps outside the artificial boundaries they've defined. XML was designed to be extensible, but sadly many of the tools designed for XML aren't nearly as extensible as XML itself.

This book is going to show you how to handle XML in its full generality. It pulls no punches. It does not pretend that XML is anything except XML, and it shows you how to design your programs so that they handle real XML in all its messiness: valid and invalid, mixed and unmixed, typed and untyped, and both all and none of these at the same time. To that end, this book focuses on APIs that don't try to hide the XML. In particular, there are three major Java APIs that correctly model XML, as opposed to modeling a particular class of XML documents or some narrow subset of XML. These are

- SAX, the Simple API for XML
- DOM, the Document Object Model
- JDOM, a Java native API

These APIs are the core of this book. In addition, I cover a number of preliminaries and supplements to the basic APIs, including

- XML syntax
- DTDs, schemas, and validity
- XPath
- XSLT and the TrAX API
- JAXP, a combination of SAX, DOM, and TrAX with a few factory classes

And, since we're going to need a few examples of XML applications to demonstrate the APIs, I also cover XML-RPC, SOAP, and RSS in some detail. However, the techniques this book teaches are hardly limited to those three applications.

Who You Are

This book is written for experienced Java developers who want to integrate XML into their systems. Java is the ideal language for processing XML documents. Its strong Unicode support in particular made it the preferred language for many early implementers. Consequently, more XML tools have been written in Java than in any other language. More open source XML tools are written in Java than in any other language. More developers process XML in Java than in any other language.

Processing XML with Java™ will teach you how to

- Save XML documents from applications written in Java
- Read XML documents produced by other programs
- Search, query, and update XML documents
- Convert legacy flat data into hierarchical XML
- Communicate with network servers that send and receive XML data

- Validate documents against DTDs, schemas, and business rules
- Combine functional XSLT transforms with traditional imperative Java code

This book is intended for Java developers who need to do anything with XML. It teaches the fundamentals and advanced topics, leaving nothing out. It is a comprehensive course in processing XML with Java that takes developers from having little knowledge of XML to designing sophisticated XML applications and parsing complicated documents. The examples cover a wide range of possible uses, including file formats, data exchange, document transformation, database integration, and more.

What You Need to Know

This is not an introductory book with respect to either Java or XML. I assume you have substantial prior experience with Java and preferably some experience with XML. On the Java side, I freely use advanced features of the language and its class library without explanation or apology. Among other things, I assume you are thoroughly familiar with the following:

- Object-oriented programming, including inheritance and polymorphism.
- Packages and the CLASSPATH. You should not be surprised by classes that do not have `main()` methods or that are not in the default package.
- I/O including streams, readers, and writers. You should understand that `System.out` is a horrible example of what really goes on in Java programs.
- The Java Collections API including hash tables, maps, sets, iterators, and lists.

In addition, in one or two places in this book I use some SQL and JDBC. These sections are relatively independent of the rest of the book, however, and chances are if you aren't already familiar with SQL, then you don't need the material in these sections anyway.

What You Need to Have

XML is deliberately architecture, platform, operating system, GUI, and language agnostic (in fact, more so than Java). It works equally well on Mac OS, Windows, Linux, OS/2, various flavors of Unix, and more. It can be processed with Python, C++, Haskell, ECMAScript, C#, Perl, Visual Basic, Ruby, and of course Java. No byte-order issues need concern you if you switch between PowerPC, X86, or other architectures. Almost everything in this book should work equally well on any platform that's capable of running Java.

Most of the material in this book is relatively independent of the specific Java version. Java 1.4 bundles SAX, DOM, and a few other useful classes into the core JDK. However, these are easily installed in earlier JVMs as open source libraries from the Apache XML Project and other vendors. For the most part, I used Java 1.3 and 1.4 when testing the examples; therefore, it's possible that a few of the classes and methods used are not available in earlier versions. In most cases, it should be fairly obvious how to backport them. All of the basic XML APIs except TrAX should work in Java 1.1 and later. TrAX requires Java 1.2 or later.

How to Use This Book

This book is organized as an advanced tutorial that can also serve as a solid and comprehensive reference. Chapter 1 covers the bare minimum material needed to start working with XML, although for the most part this is not intended as a comprehensive introduction, but more as a review for readers who already have read other, more basic books. Chapter 2 introduces RSS, XML-RPC, and SOAP, the XML applications used for examples throughout the rest of the book. This is followed by two chapters on generating XML from your own programs (a subject all too often presented as a lot more complicated than it actually is). Chapter 3 covers generating XML directly from code, and Chapter 4 covers converting legacy data in other formats to XML. The remaining bulk of the book is devoted to the major APIs for processing XML:

- The event-based SAX API
- The tree-based DOM API
- The tree-based JDOM API
- XPath APIs for searching XML documents
- The TrAX API for XSLT processing

Finally, the book finishes with an appendix providing quick references to the main APIs.

If you have limited experience with XML, I suggest that you read at least the first five chapters in order. From that point forward, if you have a particular API preference, you may begin with the part that covers the major API you're interested in:

Chapters 6 to 8 cover SAX.

Chapters 9 to 13 cover DOM.

Chapters 14 and 15 cover JDOM.

Once you're comfortable with one or more of these APIs, you can read Chapters 16 and 17 on XPath and XSLT. However, those APIs and chapters do require some knowledge of at least one of the three major APIs.

The Online Edition

The entire book is available online in plain-vanilla HTML at my Cafe con Leche web site. You can find it at *http://www.cafeconleche.org/books/xmljava/*. Every word of this book is there. Nothing has been held back or left out. I do hope you also find the printed book useful and choose to buy it—it's certainly cheaper than the paper and toner you'd use up printing out all 1,120 pages from your laser printer—but you are by no means obligated to do so. My goal is to make this material as broadly available and useful as possible.

The online version has no protection other than copyright law and your own good will. You don't need to register to read it, or to download some special electronic key that becomes invalid when you buy a new laptop (and that probably wouldn't run on Linux or a Mac in the first place). I want people to read and use this book. I do not want to put up silly roadblocks that make it less useful than it could be. I do ask, as a courtesy, that you do not republish the online edition on your own server. Doing so makes it extremely difficult for me to keep the book up to date. If you want to save a few pages on your laptop so you can read this book on an airplane, I don't really mind. But please don't pass out your own copies to anyone else. Instead, refer your friends and colleagues to the web site or the printed book.

Some Grammatical Notes

The rules of English grammar were laid down, written in stone, and encoded in the DNA of elementary school teachers long before computers were invented. Unfortunately, this means that sometimes I have to decide between syntactically correct code and syntactically correct English. When forced to do so, English normally loses. This means that sometimes a punctuation mark appears outside a quotation mark when you'd normally expect it to appear inside, a sentence begins with a lowercase letter, or something similarly unsettling occurs. For the most part, I've tried to use various typefaces to make the offending phrase less jarring. In particular, please note the following:

- *Italicized text* is used for emphasis, the first occurrence of an important term, titles of books and other cited works, words in languages other than English, words as words themselves (for example, *Booboisie* is a very funny word), Java system properties, host names, and resolvable URLs.
- `Monospaced text` is used for XML and Java source code, namespace URLs, system prompts, and program output.
- *`Italicized monospace text`* is used for pieces of XML and Java source code that should be replaced by some other text.
- **`Bold monospaced text`** is used for literal text that the user types at a command line, as well as for emphasis in code.

It's not just English grammar that gets a little squeezed, either. The necessities of fitting code onto a printed page rather than a computer screen have occasionally caused me to deviate from the ideal Java coding conventions. The worst problem is line length. I can fit only 65 characters across the page in a line of code. To try to make maximum use of this space, I indent each block by two spaces and indent line continuations by one space, rather than the customary four spaces and two spaces respectively. Even so, I still have to break lines where I otherwise would prefer not to. For example, I originally wrote this line of code for Chapter 4:

```
result.append("          <Amount>" + amount + "</Amount>\r\n");
```

To fit it on the page, however, I had to split it into two pieces, like this:

```
result.append("          <Amount>");
result.append(amount +"</Amount>\r\n");
```

This wasn't too bad, but sometimes even this wasn't enough and I had to remove indents from the front of the line that would otherwise be present. This occasionally forced the indentation not to line up as prettily as it otherwise might, as in this example from Chapter 3:

```
   wout.write(
 "xmlns='http://namespaces.cafeconleche.org/xmljava/ch3/'\r\n"
   );
```

The silver lining to this cloud is that sometimes the extra attention I give to the code when I'm trying to cut down its size results in better code. For example, in Chapter 4, I found I needed to remove a few characters from this line:

```
OutputStreamWriter wout = new OutputStreamWriter(out, "UTF8");
```

On reflection I realized that nowhere did the program actually need to know that `wout` was an `OutputStreamWriter` as opposed to merely a `Writer`. Thus I could easily rewrite the offending line as follows:

```
Writer wout = new OutputStreamWriter(out, "UTF8");
```

This follows the general object-oriented principle of using the least-specific type that will suit. This polymorphism makes the code more flexible in the future should I find a need to swap in a different kind of `Writer`.

Contacting the Author

I always enjoy hearing from readers, whether with general comments, specific ways I could improve the book, or questions related to the book's subject matter. Because this book is being published in its entirety online, it is possible for me to reprint at least the online edition much faster than can be done with a traditional paper book. Thus corrections and errata are especially helpful because I have a real chance to fix them. Before sending in a correction, please do check the online edition to see if I have already fixed the problem.

Please send all comments, inquiries, bouquets, and brickbats to elharo@metalab.unc.edu. I get a lot of e-mail, so I can't promise to answer them all; but I do try. It's helpful if you use a subject line that clearly identifies yourself as a reader of this book. Otherwise, your message may accidentally get misidentified as spam I don't want or bulk mail I don't have time to read and be dropped in the bit bucket before I see it. Also, please make absolutely sure that your message uses the correct reply-to address and that the address will be valid for at least several months after you send the message. There's nothing quite as annoying as taking an hour or more to compose a detailed response to an interesting question, only to have it bounce because the reader sent the e-mail from a public terminal or changed their ISP. But please do write to me. I want to hear from you.

Elliotte Rusty Harold
Brooklyn, New York
June 7, 2002

Acknowledgments

Thomas Marlin provided me with the original Latin text of the Fibonacci problem you'll find in Chapter 3.

Jason Hunter's encyclopedic knowledge of the Java Servlet API was essential to the design and execution of the servlet code in this book. Donald Sizemore helped me get my servlets installed and running on IBiblio.

Luke Tymowski provided some of the RSS examples and helped me debug various problems with my Cobalt Qube.

Bruce Eckel and Chuck Allison helped me decipher the relative capabilities of Java and C++. Bruce Eckel also helped out with Python, and Matt Sergeant and Brendan McKenna helped out with Perl. Philip Nelson, Robert A. Casola, and Rob Smith helped with Visual Basic. None of these people necessarily agree with what I wrote about those relative capabilities (in fact, more often than not they vehemently disagree; *de linguis non disputandum est*); but I couldn't have done it without them.

Although this is the sixth book I've written about XML, it is the first one I've written in XML. That could not have happened without Norm Walsh's DocBook DTD and XSL stylesheets for DocBook.

Many people helped out with comments, corrections, and suggestions. These include Paymen Aliverdi, Sergey Astakhov, Dagmar Buggle, William Chang, Richard Dedeyan, Paul Duffin, Lacey Anne Edwards, Peter Elliott, Paul Erion, Bernard Farrell, Wei Gao, Scott Harper, Stefan Hässig, Martin Henke, Markus Jais, Oliver Lorpilla, Igor Kostjuhin, Alexander Krumpholz, Wes Kubo, Ramnivas Laddad, Manos Laliotis, Ian Lea, Frank Lee, Ray Leyva, Rob Lugt, Richard Monson-Haefel, Gary Nichols, James Orenchak, Aron Roberts, Carlo Rossi, Raheem Rufai, Arthur E. Salwin, Peter Sellars, Diana Shannon, and Andrew Shebanow. Mike Blackstone deserves special thanks for his copious notes.

Mike Champion, Andy Clark, Robert W. Husted, Anne T. Manes, Ron Weber, and John Wegis did yeomanlike service as technical reviewers. Their comments substantially improved the book.

As always, the folks at the Studio B literary agency were extremely helpful at all steps of the process. David Rogelberg, Sharon Rogelberg, and Stacey Barone should be called out for particular commendation.

This is my first book for Addison-Wesley, but it's not going to be my last. They were all wonderful people to work with, and I look forward to working with them again. Mary T. O'Brien shepherded this book from contract to completion. Alicia Carey ably managed submissions and communications. Jody Thum corrected many of my grammatical failings. Kathy Glidden and John Fuller shepherded the book through the unusual production process writing in XML necessitated. Richard T. Evans produced an excellent index.

Finally, as always, my biggest thanks are due to my wife, Beth, without whose love and understanding this book could never have been completed.

Contents

List of Examples **xiii**
List of Figures **xxiii**
Preface **xxv**
Who You Are xxvii
What You Need to Know xxviii
What You Need to Have xxviii
How to Use This Book xxix
The Online Edition xxx
Some Grammatical Notes xxx
Contacting the Author xxxii
Acknowledgments xxxiii

Part I XML 1

Chapter 1 XML for Data 3
Motivating XML 3
A Thought Experiment 4
Robustness 6
Extensibility 9
Ease-of-Use 11

XML Syntax 13
- XML Documents 13
- XML Applications 15
- Elements and Tags 16
- Text 19
- Attributes 21
- XML Declaration 22
- Comments 23
- Processing Instructions 24
- Entities 25
- Namespaces 28

Validity 32
- DTDs 32
- Schemas 37
- Schematron 41
- The Last Mile 43

Stylesheets 43
- CSS 44
- Associating Stylesheets with XML Documents 45
- XSL 46

Summary 53

Chapter 2 XML Protocols: XML-RPC and SOAP 57

XML as a Message Format 58
- Envelopes 58
- Data Representation 59

HTTP as a Transport Protocol 64
- How HTTP Works 65
- HTTP in Java 68

RSS 73

Customizing the Request 77
- Query Strings 77
- How HTTP POST Works 81

XML-RPC 82
- Data Structures 85
- Faults 89
- Validating XML-RPC 90

SOAP 96
- A SOAP Example 97
- Posting SOAP Documents 98
- Faults 100
- Encoding Styles 102
- SOAP Headers 112
- SOAP Limitations 116
- Validating SOAP 117

Custom Protocols 118
Summary 119

Chapter 3 **Writing XML with Java** **121**

Fibonacci Numbers 122
Writing XML 124
  Better Coding Practices 125
  Attributes 127
  Producing Valid XML 128
  Namespaces 130
Output Streams, Writers, and Encodings 132
A Simple XML-RPC Client 139
A Simple SOAP Client 142
Servlets 145
Summary 149

Chapter 4 **Converting Flat Files to XML** **151**

The Budget 152
The Model 154
Input 156
Determining the Output Format 159
  Validation 165
  Attributes 169
Building Hierarchical Structures from Flat Data 174
Alternatives to Java 191
  Imposing Hierarchy with XSLT 193
  The XML Query Language 196
Relational Databases 201
Summary 208

Chapter 5 **Reading XML** **211**

InputStreams and Readers 211
XML Parsers 216
  Choosing an XML API 218
  Choosing an XML Parser 222
  Available Parsers 226
SAX 229
DOM 234
JAXP 238
JDOM 242
dom4j 246
ElectricXML 248