Introduction

Why this book is needed

"I am left with the feeling that all of the sites I have created are 50% elegance, and 50% nasty kludge."

This quote from a recent Slashdot[1] discussion on PHP development resonated with the audience. Indeed, as many would attest, a web site usually starts simple but quickly grows into a complex, convoluted mess — where you are afraid of making a change for fear of breaking something else.

Why should a web site so quickly become a nightmare of unmaintainable code, visual and semantic inconsistencies, and outright errors embarrassingly visible to the whole world? Many reasons could be quoted, from limitations specific to the particular web development platforms (such as PHP, ASP, or Perl) to fundamental drawbacks of the "web site as an application" paradigm.

1. www.slashdot.org

This book is devoted to one very important way in which the majority of today's web sites are broken — and, of course, to the technology which (if correctly applied) can mend this breakage. The problem I'm speaking about is the lack of a consistently semantic and media-independent representation of web site content; the technology that can help you solve this problem is XML; and the key to applying it in web development is XSLT transformations.

Say what you mean. XML is no panacea. It won't magically make your sites self-maintaining or error-proof. But it will give you a critical advantage: Just as a good programming language allows you to freely express your algorithms, XML makes it possible to actually *say what you mean* in content markup.

The word *content* is the key. XML is actually more important for web development than any programming language — simply because you can have a web site without a dynamic engine of any kind, but there cannot exist a web site without content.

In fact, a lot of the approaches in this book apply not only to web sites but to any XML-based document workflows, such as books or technical documentation. XML stimulates thinking about content *as such*, abstracting it not only from its presentation but from any processing requirements as well.

What you will find in this book

This is not a general XML book; it is a book on one specific application of XML. What you'll find here is as much XML and XSLT as is necessary for a sequence of very practical tasks:

- structuring your web site content into cleanly separated semantic layers;
- developing a custom XML markup vocabulary for each layer;
- automatically validating both markup and structure of content;

- transforming content from source XML to browser-ready HTML using XSLT (optionally with generation of images and other non-HTML objects); and
- integrating the content markup and transformation system with existing web development frameworks and other software.

Building the backbone. The point of using XML in web design is to separate *content* from *presentation*; the above items cover the complete transition from the former to the latter. Simply put, we focus on developing the best source markup for your content and programming the most efficient transformation into your chosen presentation style.

An XML web site may include other components, such as a database, a dynamic engine, or a maintenance back-end. There is a wide range of auxiliary tools and architectures compatible with an XML-based web site. Many of them are mentioned in this book, and a few are explored in detail (notably Cocoon, Chapter 7). However, the content-to-presentation assembly line is the backbone of any XML web site and our main focus of attention in this book.

Usability and portability. In a web development context, the term *usability* normally refers to how easy to use a web site is for a visitor. In this book, however, I would like to redefine this term by focusing on a different aspect of usability that is too often ignored — usability of a web site for its developers, authors, editors, and maintainers. With the Web growing more and more collaborative, this aspect is becoming critical.

Using semantic XML for content markup is already a big step toward liberating web authors from worrying about things they don't need to worry about. But semantic XML is only an idea; how you implement this idea will seriously affect the "authorability" and "maintainability" of your site. This is where this book, with its pervasive ideas of simplification, abbreviation, and readability, might be useful.

Another important theme of the book is *portability*. Again, this term usually describes a web site's viewability and functionality across browsers and platforms. It's not less important, however, that before

a web site gets to your browser, it must be developed and authored — often in different environments and on different platforms. We touch on this server-side aspect of portability with regard to the XML/XSLT workflow.

Who this book is for

Everyone interested in web development or in practical XML/XSLT should find this book interesting. It will be especially useful for web designers, web developers, project managers, as well as webmasters and web site administrators. Whether you are building a modest personal home page or a large dynamic site, learning the XML way of doing things will transform your outlook even if you don't plan to use (all of) this book's techniques.

You need to have a basic knowledge of XML to read and enjoy this book. For most chapters, understanding of XML syntax and common XML-related terms[2] will be enough, but for Chapter 5 you will want to know some XSLT and especially XPath. Expert knowledge of HTML is neither required nor offered. Some familiarity with web development concepts and jargon might be useful but is not necessary.

How this book is organized

Perhaps you've already thumbed through the book, so you might have noticed that it breaks into three main parts. The first part is composed mostly of text and diagrams; the second features lots of example code; the last displays a number of screenshots. This sequence metaphorically reflects the path that we'll follow from manipulating abstract notions, to writing practical markup and code, to launching and maintaining a final working web site.

2. Excluding the syntax and terminology of XML DTDs which (with a few exceptions) are intentionally ignored in favor of more powerful and modern schema languages.

- **Chapter 1** is mostly theoretical; we'll spend some time discussing the basic premises of XML and its applicability to web development. It is recommended that you read this chapter carefully, for its concepts and terminology are used throughout the book. No real harm will be done, however, if you poke into the book's code examples first and return to Chapter 1 later.

  The topics of this chapter include the principles behind using XML with web sites, an overview of relevant XML standards, and a classification of the possible ways to set up an XML web site. Special attention is paid to the dynamic web sites and the approaches to combining XML processing with a dynamic engine.

- **Chapter 2** is dedicated to the foundation of an XML web site — its *source definition*. This includes schemas for all document types used by the web site's XML source plus all the rules and regulations that may be impossible to express in a schema language but that the source must satisfy in order to smoothly transform into a correct web site.

  In this chapter, we'll look at different schema languages and discuss the implementation options for those parts of the source definition that a schema cannot handle. We will also examine the common generic markup constructs, the best approaches to their schematization, and a number of corresponding pitfalls. For instance, in this chapter you'll find insights into the eternal "child elements vs. attributes" dilemma.

- **Chapter 3** is the practical complement to the previous chapter. Here, we'll use the approaches of Chapter 2 to mark up some real web site source documents. Most common elements of web pages, such as text blocks, headings, links, and images, are considered. In most cases, existing standardized vocabularies that you can borrow from are mentioned.

  Some important concepts of the book, such as abbreviating addresses, are introduced in this chapter. This is also the chapter where markup examples start appearing in large numbers, so if you prefer to learn by looking at examples, you might want to

start your reading from this chapter. The last section of the chapter presents summary examples of a page document, a master document, and a Schematron schema that validates both types of documents.

- **Chapter 4** is the first of the two XSLT chapters. It is an introduction aimed at a developer who has had some experience with XSLT 1.0. Here, we'll discuss some of the new stuff that is being introduced in XSLT 2.0 and XPath 2.0 as well as the existing XSLT extensions. A detailed analysis is devoted to the important issue of adapting traditional algorithms to XSLT, which is a functional language without an assignment operator.

- **Chapter 5** is the core of the book — the practical XSLT chapter and the largest of all of them. Lots of XSLT code examples show all aspects of an XML-to-HTML transformation, from setting up the environment and building the page layout to low-level text processing. We'll also revisit our Schematron schema to add some exciting new checks made possible by XPath 2.0.

  This chapter not only uses but extends XSLT. We'll see how a few simple Java classes may drastically advance the capabilities of an XSLT stylesheet. These extensions are used in Chapter 5 for all kinds of tasks, from generating bitmap images via SVG to batch processing all page documents of a site. Again, a section with complete listings of the stylesheet and related bits of code summarizes this chapter.

- **Chapter 6** is where the screenshots are. It is devoted to all kinds of software that will help you run your XML web site after the core validation and transformation components are ready. The focus here is not on specific programs but on classifying the functionality of XML software and the approaches to various practical problems.

  Sections of this chapter discuss the existing XML authoring paradigms and the principles of converting other formats into XML. Also reviewed are various tools and utilities for handling

XML, XSLT, and XPath. The last section explores the use of build tools, such as the `make` utility, in XML web site projects.

- **Chapter 7** is concerned with integrating the XML/XSLT system into a web server setup. We'll briefly discuss using an XSLT processor as a servlet, but the bulk of this chapter is devoted to Apache Cocoon, which crowns the chapter and the book. After learning the principles of building Cocoon web sites, we'll revisit our sample web site from previous chapters to see what it takes to adapt it to run under Cocoon.

Typographic conventions

Designing your own book is a mindbending experience (something that songwriters who author both music and lyrics would probably agree with). In my book, I tried to make the text look rich but consistent, pleasantly dense but varied. Some of the solutions that I came up with may deserve a few words.

Running in from aside. Three levels of numbered headings are used within each chapter. In addition, unnumbered bold run-in headings are often used (as in this paragraph) to break the text into even smaller, manageable pieces.

Semantically, the run-ins are closer to margin notes than headings; usually their goal is not to state the subject but to provide a remark, an aside, a metaphor related to the topic of discussion. Hopefully these run-in headings are memorable enough to serve as landmarks facilitating navigation.

Small but not least. Some paragraphs, with or without run-in headings, are set in a smaller type. They present material that may be skipped in the first reading without any damage to understanding. You can treat the smaller-type fragments as extended footnotes or sidebars.

Cross-references. Bold gray numbers (such as **3.9**) refer to numbered sections of the book. The running headers and footers should make

it easy to find the referenced sections; however, for references that jump especially far, page numbers are also provided.

Syntax coloring without colors. Unlike most computer books with code listings, this one makes use of a concept that has long been commonplace in text editors: syntax coloring. Of course, a black-and-white book page is not really capable of color (except for shades of gray), but instead it can freely use font faces that usually look nicer on paper than on a computer screen. Thus, I have attempted to make code in the book at least as readable as it is in a good text editor by consistently "coloring" syntactic constructions with different font faces.

Essential URLs. All web addresses are given in footnotes in an abbreviated form without `http://`, `index.html`, or trailing slashes.

Slash what? I use forward slashes (`/`) and not backslashes (`\`) as directory separators for *both* Windows and Unix (the latter including Mac OS X). The rationale is simple: Forward slashes are standard on Unix and in URLs, and most Windows tools understand both kinds of slashes anyway.

Notes on terminology

The terminology used in the book is basically standard. Sometimes I simplify the accepted terminology in order to make it more accessible, or I use my own terms instead of those used in authoritative sources; all such cases are noted. Some important terms that may appear confusing or are often misunderstood are commented on below.

Element type, element, or tag? When speaking of XML, many people fail to differentiate between an *element* and an *element type*. Sometimes, a *tag* is also confused with an *element*. For example, this fragment

```
<foo> <foo/> </foo>
```

has *three* tags but *two* elements belonging to *one* element type (and having one *element type name*, `foo`). Note that an element cannot have a name — only an element type can; still, we can refer to an element by its element type name if we identify *which* of the elements of

this type is in question (for example, "in the first `foo` element"). In the XSLT context, an element from the XSLT namespace (e.g., `xsl:template`) is often called an *instruction*.

Stylesheet or transformation? The word *stylesheet* may be misleading when applied to an XSLT program that transforms one XML document into another; the word *transformation* would be more appropriate. (Note that `xsl:stylesheet` and `xsl:transform` are both acceptable as the root element of an XSLT stylesheet.) Still, backed by tradition, I mostly use "stylesheet" or, sometimes, "transformation stylesheet" when referring to the XSLT component of a web site setup.

Stylesheet or style sheet? To avoid confusion with XSLT *stylesheets*, CSS *style sheets* are always spelled thus; this is conformant with both XSLT and CSS specifications.

Document, instance, page, or file? *Document* is a generic term, but I use it only to refer to XML documents, while HTML documents are usually called *pages*. *Instance* is another term often used in XML (it refers to a document being an instance of its document type), but I will stick to "documents" as more familiar. Neither "document" nor "instance" are synonymous with *file*; a document is not necessarily stored in a file at all. Therefore, "file" is used only when real files, handled by the operating system, are involved.

Document element or root element? The XSLT specification uses the term *document element* with the meaning of *root element*. I use the latter term as more descriptive, even though it may be slightly confusing from an XSLT viewpoint because the "root node" of XPath (`/`) is the parent of the node corresponding to the "root element" (e.g., `/page`).

XML Schema or XSDL? *XML Schema* is the W3C recommendation for a schema language. Unfortunately, its name is way too generic for its own good. Even the capital S in "Schema" cannot prevent confusion when you have to speak about XML Schema among other schema languages for XML, and especially when you refer to specific schemas written in that language. So, in conformance with other books in this series, I use the abbreviation *XSDL* (XML Schema Definition

Language) to refer to the language itself and *XSDL schemas* to refer to specific schema definitions.

Yet another abbreviation you may have seen used for the same language is WXS, standing for W3C XML Schema.

URI or URL? This one may confuse even experts at times. *URI* is a more general term than *URL*, but the difference between them — i.e., those URIs that are not URLs — is so insignificant that for practical purposes, these terms are interchangeable. See RFC 2396[3] for more details.

HTML or XHTML? Since this book views HTML mostly as a result of an XSLT transformation, what I mean when speaking of HTML may actually be either HTML or XHTML (any versions). With XSLT, you can output both formats, and modern browsers do not have any problems with either. When there's a meaningful distinction between HTML and XHTML, this is noted.

"Data is" or "data are"? Formally, *data* is the plural of *datum*. In modern English, however, using "data" as singular is more common, as evidenced by statistics reported by Internet search engines. In this book "data" is used as singular.

How this book was created

"Practice what you preach." "Eat your own dogfood." One way or the other, this book itself uses many of the techniques it describes.

The text of the book was written directly in XML using a custom schema inspired by HTML, DocBook, and Charles F. Goldfarb's DTD that is used by many books in this Definitive XML Series. An XSLT transformation stylesheet written by Alina Kirsanova translated the source into XSL-FO and performed all necessary processing, such as importing code examples (stored separately), special character substitutions (**5.4.2.2**), compiling the Index and TOC, and generating cross-references.

3. `www.ietf.org/rfc/rfc2396.txt`

The design for the book was also created by me, with elements borrowed from the other books in the series that we worked on using the same XML/XSLT/XSL-FO technology. The final rendering of XSL-FO into PDF was done by XEP[4] from RenderX.

Code examples (in a total of 11 different formats and XML vocabularies) were parsed by XEmacs + PSGML (**6.1.1.2**) with custom syntax coloring regexps and then saved into XHTML using `htmlize.el`[5] by Hrvoje Nikšić. The resulting files were then translated by a simple stylesheet into a vocabulary understood by the book's main transformation stylesheet.

Acknowledgments

This book could be much worse (all the way down to the point of nonexistence) without the help of the following people to whom I am deeply indebted:

Charles F. Goldfarb, for inventing SGML before I was born, for persuading me that the book project is realistic, and for a detailed review of the manuscript.

Mark Taub, for managing the project and tirelessly pushing me ahead despite my tendency to procrastinate.

G. Ken Holman, for pointing out some finer points of XSLT and catching many ambiguities and outright errors.

Daniel Smith, for useful comments on writing style and presentation of the material.

Vadim Penzin, for useful discussions of database terminology.

Ilya Oussov, for help with Java extension functions.

Alina, for everything.

4. `xep.xattic.com`
5. `fly.srk.fer.hr/~hniksic/emacs/htmlize.el`

Origins of this book

Perhaps the best thing about this book is that it is based on, and in fact was born out of, the author's real-world projects. Since 1998, Dmitry Kirsanov Studio[6] has created web sites for customers around the world. In recent years all the web sites we do are based on XML and XSLT. This book is a snapshot of our current XML experience.

I am a technical writer, freelance XML/XSLT expert, and graphic designer. The book also draws on the magazine and online articles and books I've written. Among these are Dmitry's Design Lab,[7] a monthly column (1997–1999) devoted to exploring creative as well as technical issues pertaining to web design, and *Dmitry Kirsanov's Web Design Book*[8] (1999, in Russian), which has become one of the most influential Russian-language books on web design.

Feedback

I will be grateful for comments, corrections, criticism, or any other form of feedback on this book and its ideas and approaches. Please write me at `dmitry@kirsanov.com`.[9]

An online companion for this book is available.[10] It provides the book's errata, the complete source code of the examples, and other material.

6. `www.kirsanov.com`
7. `www.webreference.com/dlab`
8. `www.kirsanov.com/web.design`
9. I'm relieved that I can, for once, give my email address in plain text — fortunately, spam bots are unable to spider books (yet?) . . . the only downside is that it's not clickable.
10. `authors.phptr.com/Kirsanov` and `www.kirsanov.com/xsltwd`