TER Volume 6, Issue 5, July 1999

ter

telecommunications electronic reviews

Volume 6, Issue 5, July 1999

Contents:

ter issues (/lita/publications/archive/ter)

REVIEW OF: Carole A. Lane. Naked in Cyberspace: How to Find Personal Information Online. Wilton, CT: Pemberton Press, 1997.

by Brad Eden

The Internet and the World Wide Web have greatly enhanced the world's ability for speed of delivery and direct access to information and knowledge in the past five years. The ability to review and access bank accounts, federal records, and personal information has increased the public's awareness of security and privacy issues regarding electronic communication. As such, Carole Lane's book is directed at one of the hottest topics on the Internet today: what can someone find out about someone else just by accessing, surfing, and exploring electronic resources.

The title of this book is appropriate, as Ms. Lane is blunt and straightforward concerning the wealth of information and resources available when searching for personal information electronically. The book is divided into four sections, each subdivided into numerous chapters. "Introduction to Personal Records in Cyberspace" contains five chapters, and is a good general background on that subject. Chapter 2 introduces database searching; chapter 3 looks at the Internet and bulletin board systems as sources of information; chapter 4 provides three typical searches that Ms. Lane has done to show the range and complexity of database searching and detective work needed to find personal information; and chapter 5 examines privacy issues.

Section Two, "How Personal Records are Used," gets into the real nitty-gritty. There are chapters on locating people, pre-employment screening, recruitment and job-searching, tenant screening, asset searching, competitive intelligence research, locating and identifying an expert, prospect and fundraising research, and private investigation. Section Three, "Types of Personal Records," provides information on how to locate biographies, general indices, telephone directories, staff and other professional directories, mailing lists, news, photographic images, quotations, bank records, business credit and company financial records, consumer credit records, criminal justice records, Department of Motor Vehicles records, death records, tax records, medical and insurance records, public records, adoption records, celebrity records, genealogical records, political records, and demographic and statistical records.

Finally, in Section Four, "Where Can I Find More Information," Ms. Lane provides a large bibliographic warehouse of books, periodicals, and organizations to assist one in researching and locating personal information. Fifteen appendices on databases related to these various resources by subject area effectively close the book. Ms. Lane has aptly provided a book that would cause concern to any private citizen. Virtually all information regarding a person's private and public life is available and obtainable by virtually anyone else in the world. Naked in Cyberspace should be required reference material for all reference librarians. Ms. Lane's knowledge and personal experience in this area of investigative work shines through in this book. Her assistance in all areas of database searching and examination in order to uncover personal information is truly astounding. One area of concern, however, is the lack of information beyond a sales pitch for the book on the web page cited in the book for keeping readers current on trends and issues relevant to personal online research. Otherwise, both beginning and experienced reference librarians will find needed information in this book to assist them in locating personal information for their patrons/ clients about either themselves or anyone else they care to examine. Pretty scary!

Dr. Brad Eden (beden@ccmail.nevada.edu (mailto:beden@ccmail.nevada.edu)) is Head of Cataloging for the University of Nevada at Las Vegas.

Understanding Standard Generalized Markup Language

by Charles F. Thomas

Introduction

The terms "SGML" (Standard Generalized Markup Language) and "XML" (Extensible Markup Language) increasingly are encountered by anyone who uses the World Wide Web. Many users have heard the terms, but do not understand what they mean at the conceptual level. This essay leads readers through a basic explanation of markup languages, then progresses quickly from the most familiar markup language in use today to SGML and XML. This is not an instructional text in the application of SGML or XML. Instead, its main intent is to equip readers with a solid understanding of the terms; what they mean, why they are becoming so popular, and their likely impact upon the Internet community.

Markup Languages

If you deal with electronic textual content in any way, by this point you probably have been exposed to the concept of textual markup. Essentially, textual markup occurs anytime a body of raw textual narrative or data is modified by the addition of markers, tags, or other characters that serve to delineate specific elements or features within the text. The use or display of the resulting tagged text totally depends upon the purpose for which it is intended. Any particular set of markup tags can be referred to as a "markup language."

Electronic text is nearly as old as computing itself. For the first few decades that computers existed, any markup of electronic text and data was very specialized, and limited in scope and impact. In recent years, however, the concept of markup languages has become much more familiar. The most familiar markup language in use today is Hypertext Markup Language (HTML). HTML is used to create sites on the World Wide Web (WWW) and consists of approximately 40 commonly used tags. Since the World Wide Web first became a popular medium for electronic publication in 1993- 1995, basic HTML has equipped millions of Internet authors with a simple, yet flexible, tool for the aesthetic, integrated presentation of text and multimedia (primarily text).

To publish a standard WWW document, one merely has to apply this simple markup language to raw text, and place the resulting document online. All of the commercially available WWW browsers are programmed to understand this particular markup language and adhere to particular conventions for displaying tagged documents. For example, one of the most commonly used tags in HTML delineates the beginning and end of paragraphs within a document. Paragraphs are a common feature of much textual information; if HTML was to be of any value, its creators had to include a means of displaying text broken into paragraphs. If the tag is placed at the beginning of every paragraph, and a is placed at the end of each paragraph, HTML browsers automatically will generate space before and after each such tagged text block as a document is displayed.

As the World Wide Web has evolved into a more sophisticated forum for publication, however, HTML's weaknesses also have become more apparent. A wide variety of content and formatting needs increased user expectations for exchanging and manipulating information. Even advancements in computing technology have stretched the functional capabilities of HTML to its limits. In numerous instances, individual

authors now superimpose idiosyncratic conventions upon HTML markup to meet their particular information delivery needs. In most such instances, the obvious flaw of these non-standard conventions is that they are not understood automatically by either users or their computers.

Much of HTML's growing inadequacy results from this markup language's main intended use: a means of controlling the display of tagged information. In other words, most of the available tags for HTML markup are concerned only with the display of a tagged document. Many HTML tags, for example, control font sizes, colors, visual organization of elements of documents, and other visual features. If a WWW author is more concerned with marking and displaying an electronic document so that users can learn more about its cognitive structure or its intellectual content, HTML cannot provide an adequate number of tags for these purposes. Any attempt to use the limited HTML tag set for these purposes is totally arbitrary and is not a viable solution for much of the increasingly diverse and complicated electronic text going online.

If HTML is not an adequate markup language for a growing segment of WWW content, are other markup languages available? As each day passes, the answer increasingly is "Yes!" Numerous communities now are publishing very specialized types of electronic textual information on the WWW, and many of these groups have established a commonly-accepted convention for tagging the content and structure for their information. Widely accepted markup languages now exist for publishing communities such as geneaological societies, museums and archives, technical documentation publishers, government agencies that produce specialized information, institutions transcribing historical texts, and numerous other enclaves of the WWW authoring community.

SGML

If you have paid close attention to this essay, you will recall that all of the commercially available WWW browsers automatically understand how to interpret and display HTML markup. Do they also know automatically how to read and display documents tagged in other markup languages? At the present time, they do not for one simple reason: HTML is a simple and ubiquitous markup language for general application, while all of the other markup languages mentioned serve smaller and more specialized communities of publishers and readers. In other words, producing browsers programmed to read other specific markup languages is not commercially sustainable.

This being the case, how do users read these more specialized documents, and will WWW browsers ever support some or all of these markup languages? The answer to the first half of this question is fairly simple. Communities that use other markup languages presently must rely upon software solutions that are appropriate only for smaller audiences. Some such browsers are free, and others are produced by companies that exist to serve such specialized consumers. The main point to keep in mind, however, is that authoring and reading such documents requires substantially more effort than using a common WWW browser.

The answer to the second half of the above question is almost certainly yes, but not in the way one might anticipate. It is unlikely that any WWW browser ever will be programmed to specifically understand and display, for example, a common geneaology markup language, plus a historical text markup language, plus other markup languages. One reason this would not work is that the number of available markup languages in use would always exceed the capabilities of a commercial browser at any given point.

A discussion of SGML (one of the main purposes of this essay) is now appropriate. One common misconception about SGML is that it is one master markup language containing an enormous set of available tags. Implementers simply pick and choose from this list those tags that meet their authoring needs. If this were the case, then someone could produce a massive WWW browser capable of

understanding the "full" SGML tag set which would therefore be able to understand more limited adaptations of SGML sub-sets by genealogists, scholars, technical publishers, etc. Under this scenario, any community requiring a customized markup language merely would adopt a portion of SGML that by definition would be compliant with the larger standard.

This interpretation, however, is not a correct understanding of SGML. Although the term "markup language" is part of its name, SGML is not a markup language itself. Instead, it is an international standard specification for creating markup languages. In other words, SGML provides specific instructions on how to devise markup languages that meet the needs of specific groups. Its main advantage is that all languages created according to its standards share some common features. These features allow software programmers to create browser applications that can read any SGML-compliant markup language.

Three characteristics distinguish SGML-derived markup languages from other markup languages. First, all SGML tagging languages emphasize descriptive rather than procedural markup; that is, the tags simply state that a particular element or feature exists where it is marked within a body of text. By contrast, HTML markup is more procedural because it indicates to WWW browsers that when a given element is tagged, it must be displayed in a prescribed manner. Descriptive markup's advantage over procedural markup is that the same document can be processed by different software applications and provide different displays or uses of the tagged information depending upon the needs of the user. [1]

A second unique characteristic of SGML markup languages is their shared "document type" concept. Computers process electronic objects and information by being able to distinguish between pre- defined categories of objects. For example, all objects that are supposed to be compatible with a particular database application must be structured and encoded in a certain manner. In the case of tagged electronic textual documents, if all documents in a group adhere to a particular Document Type Definition (DTD), they can be processed in an identical way. [1] DTDs are a key concept behind SGML's strength. DTDs specify how a tagged document must be structured, what markup tags may be used within this type of document, and rules for the application of these tags.

If a community wishes to create its own SGML-compliant markup language, it necessarily must create an accompanying DTD that informs compatible browsers how documents tagged in this language should be structured. The DTD exists as a separate entity from documents tagged according to its rules; browser applications must read these rules before they can open any document that complies with a DTD.

Finally, SGML markup languages are characterized by their independence from any one software or hardware configuration. SGML's creators deliberately built a framework that would permit the transport of tagged information from one system to another without losing information. [1] SGML markup languages deliberately separate tagging the content and structure of information as a function from the display of the marked-up information.

Why is this important? If a markup language is defined in a DTD, any software that can handle SGML, whether it is a word processor, WWW browser application, data extraction utility, or other package, will know how to use the tagging language to interpret and use the document. On the other hand, if the same document is tagged in a markup language that does not comply with an international standard, or even is tagged in one for which many arbitrary, non-standard conventions were used, this information is only reliable in one specific context. Any potential portability of the information will be compromised by the necessity of stripping away the unfamiliar tags and reformatting the information to comply with another software's requirements. In very simple terms, this means that information only has to be created and tagged in an

SGML language once. Subsequently it can be understood and used for a wide variety of purposes. With other languages the information may have to be reformatted numerous times and faces the likelihood of data loss in the process.

As an established international standard (International Organization for Standardization, ISO 8879), SGML changes very slowly. Its rules provide for a great deal of flexibility in creating markup languages, so changes to the standard itself are rare. Although SGML has been used by governments and the publishing industry for many years, only now is it being recognized by a wider world of potential publishers, all of them eager for some easy means of publishing and distributing information that frequently has specialized needs for display and use. Since SGML is a proven framework, developers already can rely upon numerous software applications to assist them in transporting information to a wide variety of users. Because the World Wide Web appears to be the information delivery forum for the foreseeable future, however, the question still lingers, "Will WWW browsers ever support SGML?"

XML

Fortunately, this question can now be answered in the affirmative. The major commercial developers of WWW browsers already are producing beta prototypes of browsers that can read SGML markup languages. However (and this is a significant warning), the community of Internet and WWW planners is trying to cover as much lost ground as possible by making these same prototypes as compliant as possible with other, non-SGML markup languages. Does this mean they are programming each known markup language's rules into the software? Not at all! Instead, the next generation of WWW software will not be just SGML compliant, but will be XML compliant as well.

XML (eXtensible Markup Language) follows conventions very similar to SGML rules. In order for the newest browsers to be able to interpret any markup language, each document tagged in a language must state their document type and reference a location where the rules for this markup language may be found. Since SGML markup languages already require this arrangement, this presents no difficulties. XML browsers, however, will necessitate a few minor changes in SGML and SGML markup languages, so some minor modifications to existing SGML documents will be necessary before they can be viewed in forthcoming browsers.

XML was approved for use by SGML users because its goal of inclusion of SGML and non-SGML markup languages was desirable. Technically, this development means that SGML documents converted to XML for WWW display purposes must be converted back to SGML to be used in other SGML applications. For early implementors, however, this inconvenience reportedly has not been great. As time passes, perhaps more of the non-SGML markup languages will be transformed to comply with ISO 8879, and SGML and XML may become more synonymous.

Conclusion

WWW publishing has become a surprisingly sophisticated forum. Simultaneously, a worldwide community of generous and outspoken people are freely providing massive quantities of useful and specialized information. Commercial interests are also pursuing ways to capitalize on their unique information assets. Both of these groups require newer and better methods for formatting their information, as they have outgrown the capabilities of basic HTML. As a markup language, HTML certainly will continue to be a viable and highly used method of placing simple, informative documentation and information online. SGML, however, provides a means for an infinite number of communities to tailor their own markup languages, while still retaining the benefits of interoperability. SGML's main drawback is that it is not nearly as easy to

learn or implement as HTML, but this should not discourage potential users. XML, the newest thing in markup languages, is a framework very similar to SGML and will serve as a means of bringing much more of the Internet, from the simple to the complex, within reach of anyone who can access the WWW.

Notes:

[1] Sperberg-McQueen, C. M. & Burnard, L., eds. A Gentle Introduction To SGML [Online]. Available from http://www-tei.uic.edu/orgs/tei/sgml/teip3sg/ (http://www-tei.uic.edu/orgs/tei/sgml/teip3sg/).

Charles Thomas (thoma134@tc.umn.edu (mailto:thoma134@tc.umn.edu)) coordinates cataloging and database activities for eight archival and manuscripts collections in the Univeristy of Minnesota Libraries.

table of contents ter issues (/lita/publications/archive/ter)

REVIEW OF: Steven Feuerstein with Bill Pribyl. Oracle PL/SQL Programming 2nd Ed. Sebastopol, CA; O'Reilly & Associates, 1997.

by John Wynstra

Oracle PL/SQL Programming is a typical programming language reference book. It is filled with the information needed to understand the particulars of PL/SQL and to utilize the features of this language to their fullest potential. In addition, this book serves as a good tutorial for the beginning programmer. The author is skilled in the art of simplifying ideas and drawing parallels from the known in order to explain new concepts. This book contains a good balance between hard facts, explanatory sections, examples, and advice.

The book's 7 sections contain a total of 26 chapters, 3 appendices, an index, and a 3 « inch floppy disk featuring program samples and utilities discussed in the book. Its 987 pages provide an organized tutorial for this language and an extensive reference for the topics that are covered. While the 1st edition (1995) featured almost complete coverage of the PL/SQL programming language through release 2.3, the material for the 2nd edition became too bulky for one book and resulted in two books, the second one also by the Steven Feuerstein, entitled Oracle Built-in Packages. That book was written to cover and expand on the material about built-in packages--chapter 15 in the 1st edition.

The first section (chapters 1-3), "Programming in PL/SQL," contains basic introductory material including a general history of the language, a comparison of the different versions, a detailed description of the basic elements of the language, and a set of guidelines for good coding style. Chapter 3, "Effective Coding Style," is very helpful and would be a welcome addition to the stacks of programming language books that seem to ignore the subject completely. His guidelines are offered in the spirit of getting new programmers to establish good coding habits early on and not in the spirit of challenging other approaches to coding style. The goal here is readable programs.

Section II (chapters 4-10), "PL/SQL Language Elements," covers in detail the data types, data structures, program control statements, error handling, and syntax features of PL/SQL in order to familiarize the reader with the pieces and parts of this language. Much of this information is straight-forward reference material, but it is presented in such a way that even a first-time programmer could use the text to learn the principles of programming.

"Built-In Functions" is the topic covered in section III (chapters 11-14). These predefined functions come packaged with PL/SQL and typically address common tasks that every programmer faces. They include basic data-type manipulation functions for characters, strings, dates, numbers, and large objects (e.g., image files) as well data conversion functions from one data-type to another (e.g., date to character or visa versa). Each chapter in this section focuses on all of the functions related to a specific data-type; for example, chapter 11 is titled "Character Functions."

Up to this point the author has dealt with the pieces and parts of this language. Section IV (chapters 15-17), "Modular Code," begins to pull these parts together into the modular components that ultimately make up an application. The author considers this to be an important aspect of building applications and starts the section with an explanation of modularity as a general programming concept. This is followed by a description of the modular structures available in PL/SQL--procedures, functions, anonymous blocks, and packages.

Section V (chapters 18-21), "New PL/SQL8 Features," covers the object-oriented features of Oracle 8. As with previous sections, the author starts with a general discussion of this topic and then delves into specific details related to PL/SQL. Since programmers have certain expectations as to what object-oriented means in terms of programming languages, the author sets about clarifying what object-oriented does and does not mean with regards to this database and PL/SQL. Oracle 8 represents a new breed of databases--object-relational. It is not, however pure object technology, but rather the next step in the development of relational databases.

Section VI (chapters 22-26), "Making PL/SQL Programs Work," is the tips, tricks, tools, and debugging section of this book. The author's years of experience come together in these chapters to steer the reader toward best programming practices and away from some known pitfalls.

The Appendices describe the contents of the companion disk, explain how to call stored procedures from PL/SQL Version 1.1 from within Oracle Developer/2000-based applications, and provide a basic reference list to many popular Oracle built-in packages. Appendix C will likely be the most often referenced section of the book, since it offers a quick summary of the built-in packages. The companion disk contains some useful information, but serves mainly as an advertisement for Revealnet located at http://www.revealnet.com (http://www.revealnet.com/). This web site appears to be an excellent location for keeping current and networking with others in the Oracle and PL/SQL world. The author is a regular contributor to this site.

This book is a must-own for anyone who programs in this language. Steven Feuerstein is an authority on this language and with 18 years of experience as a software developer has much to offer on this subject. The information in this book is presented well and thoroughly. Owners of the 1st edition will want to pick up the 2nd edition in order to get the expanded coverage of the Oracle 8 and PL/SQL version 8.

John Wynstra (wynstra@uni.edu (mailto:wynstra@uni.edu)) is the Library Systems Project Manager at the University of Northern Iowa.

table of contents ter issues (/lita/publications/archive/ter)

About TER