June 2003

XHTML Facilitates the Transition from HTML to XML;

Norman Desmarais
Providence College, normd@providence.edu

Against the Grain
c/o Katina Strauch
209 Richardson Avenue
MSC 98, The Citadel
Charleston, SC 29409

SPECIAL REPRINT

VOLUME 15, NUMBER 3 JUNE 2003

Against the Grain

ISSN: 1043-2094

"Linking Publishers, Vendors and Librarians"

Innovations Affecting Us

XHTML Facilitates the Transition from HTML to XML

by **Norman Desmarais** (Acquisitions Librarian, Phillips Memorial Library, Providence College, Providence, RI 02918; Phone: 401-865-2241; Fax: 401-865-2823) <normd@postoffice.providence.edu>

We discussed the eXtensible Markup Language and its benefits in the April, 1999 issue (pp. 86-89, 93). We focused on its applications for e-commerce and its relationship with UN/EDIFACT; but its possible applications are much broader because it is designed to be both human-readable and computer-readable.

XML is a subset of the Standard Generalized Markup Language (SGML) (ISO 8879:1986 as amended and corrected). Initially conceived for use on the World Wide Web, XML can be used for any type of electronic publication. While SGML is a text processing standard that describes how a document should be laid out and structured, XML is a dialect of SGML that describes the information content of a document.

SGML really didn't catch on very well because it is too complicated and requires a steep learning curve that corresponds to high costs. People were also reluctant to incur the expenses of hiring a consultant to implement and manage SGML. Instead, they focused on using the HyperText Markup Language (HTML) which, in its pure form, is an application of SGML with a Document Type Definition (DTD). HTML, as originally conceived, was to be a language for the exchange of scientific and other technical documents, suitable for use by non-document specialists. It addressed the problem of SGML's complexity by specifying a small set of structural and semantic tags that simplified the creation of documents. It added support for hypertext and, later, multimedia.

HTML soon became very popular and rapidly outgrew its original purpose. As it is used in practice, HTML is mostly presentation oriented. It defines how information is displayed, such as the color or font size of a word. It doesn't say anything about the actual meaning of the word. XML, on the other hand, has nothing to do with display. It only describes information.

There has also been rapid invention of new elements for use within HTML (as a standard) and for adapting HTML to vertical, highly specialized, markets. This plethora of new elements has led to compatibility problems for documents which must be accessible across a variety of different software and hardware platforms — platforms which continue to proliferate rapidly. The outlook for HTML's suitability for use on these platforms is somewhat limited.

HTML has a fixed set of tags; but XML lets users define their own tags, making it much more flexible and vendor independent. In other words, users can create XML documents in one application and use them in another without requiring a conversion. Because XML accommodates a virtually unlimited number of tags, it can describe the information content of a document more precisely. An XML tag can describe the meaning of any word or term, such as a person's name, a product name, date, or whatever.

In fact, XML introduces a concept called a namespace which is a method for qualifying the names used in XML documents by associating them with contexts identified by Universal Resource Identifiers (URI). For example, the word "bill" in one context could mean an invoice. In another, it could indicate a piece of proposed legislation. XML namespaces qualifies the tag names used in XML documents by associating them with their source and provides a simple method for qualifying certain names used in XML documents by associating them with namespaces. Thus, an application can distinguish between two (or more) different meanings of the same word.

The current version of HTML (ver. 4.0) will be the last one. It is expected that XML will eventually replace HTML as the language of the Web because it is readable both by humans and by machines; and it allows data processing without human intervention. To facilitate the conversion from HTML to XML, the World Wide Web Consortium (W3C) re-wrote HTML as an XML application and developed XHTML as a sort of bridge language. The W3C approved XHTML on January 26, 2000.

XHTML consists of a family of current and future document types and modules that reproduce and extend HTML 4. These XHTML document types are based on XML and are designed to work with XML-based user agents (browsers and user applications). XHTML is intended to serve as a language to tag content in such a way that user agents that can understand both XML and HTML 4 will be able to use them.

Benefits

XHTML 1.0 offers several benefits:

XHTML documents conform to XML and can be viewed, edited, and validated with standard XML tools.

XHTML documents can be written for new user agents that support XHTML 1.0 in such a way that they operate as well or better than they did before in HTML user agents. Yet, both agents will be able to understand the documents.

XHTML documents can utilize applications (e.g. scripts and applets) that rely upon either the HTML Document Object Model (DOM) or the XML Document Object Model. As the XHTML family evolves, documents conforming to XHTML 1.0 will be more likely to interoperate within and among various XHTML environments, allowing greater confidence in the backward and future compatibility of electronic content.

continued on page 101

Document developers and user agent designers are constantly discovering new ways to express their ideas through new markup. In XML, it is relatively easy to introduce new elements or additional element attributes. The XHTML family is designed to accommodate these extensions through XHTML modules and techniques for developing new modules that conform to XHTML. These modules will permit the combination of existing and new feature sets for developing content and designing new user agents.

Alternate ways of accessing the Internet are constantly being introduced. Some estimates say that 75% of Internet document viewing will occur on these alternate platforms by the year 2002. XHTML aims to operate on a variety of user agents to insure interoperability. Initially, XHTML will use a new user agent and document profiling mechanism that will allow servers, proxies, and user agents to transform the content. Eventually, authors will be able to develop XHTML content for use by any user agent that conforms to XHTML.

Differences with HTML 4

Because XHTML is an XML application, it requires that documents conform to XML syntax. This means that certain practices that were perfectly legal in SGML-based HTML 4 must be changed. First of all, XHTML documents must use lower case for all HTML element and attribute names because XML is case-sensitive. Thus, <li> and <LI> would be different tags.

Authors of XHTML documents should use XML declarations in all their documents. Such a declaration is required when the character encoding of the document is other than the default UTF-8 or UTF-16.

Documents must also be "well-formed." Well-formedness is a new concept introduced by XML. Essentially it means that all elements must have closing tags and that all the elements must nest properly. SGML did not allow overlapping tags; but browsers tolerated it. While HTML and existing browsers would recognize the following example:

here is an emphasized
paragraph.

XML and XHTML would not understand it because the end tags are not properly nested. The example would have to be corrected as follows:

here is an emphasized
paragraph.

Any element that is not empty requires end tags. HTML permits omitting the end tag because subsequent elements imply closure. XHTML does not allow this. The only XML elements that do not require an end tag are those declared in the DTD as EMPTY. However, XML allows a shorthand method for terminating empty tags by ending the start tag with />, such as
<hr/>.

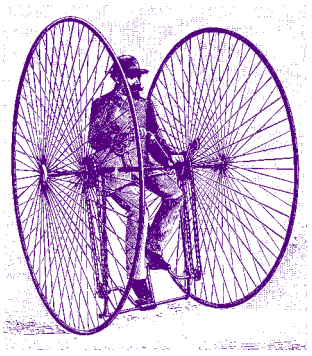
All attribute values must be quoted, even those which appear to be numeric such as: <table rows="3"> and not <table rows=3>. Attribute-value pairs must be written in full. Attribute names such as *compact* and *checked* cannot occur in elements without specifying their value. For example, <dl compact="compact"> would be correct; but <dl compact> would not.

User agents will strip leading and trailing white space (space character, horizontal tab character, and end-of-line codes) from attribute values and map sequences of one or more whitespace characters (including line breaks) to a single inter-word space (an ASCII space character for western scripts).

Authors may use the XHTML namespace with other XML namespaces even though these documents do not conform strictly to XHTML 1.0 documents. However, the W3C still has to address ways to specify how documents involving multiple namespaces will conform to the specification.

Compatibility Issues

Although there is no requirement for XHTML 1.0 documents to be compatible with existing user agents, this is easy to accomplish in practice. While the general recommended MIME labeling for XML-based applications has yet to be resolved, XHTML documents which follow the "HTML Compatibility Guidelines" may be labeled with the Internet Media Type "text/html," as they are compatible with most HTML browsers.



Future Directions

XHTML 1.0 provides the basis for a family of document types that will extend and subset XHTML by defining modules and specifying a mechanism for combining these modules. This mechanism will enable the extension and sub-setting of XHTML 1.0 in a uniform way through the definition of new modules and allow it to support a wide range of new devices and applications.

As the use of XHTML moves from the traditional desktop user agents to other platforms, not all platforms will require all of the XHTML elements. For example a hand held device or a cellphone may only support a subset of XHTML elements. The modularization process breaks XHTML into a series of smaller element sets. These elements can then be recombined to meet the needs of different communities.

Modularization has several advantages:

- It provides a formal mechanism for sub-setting XHTML.
- It provides a formal mechanism for extending XHTML.
- It simplifies the transformation between document types.
- It promotes the reuse of modules in new document types.

A document profile specifies the syntax and semantics of a set of documents. If a document conforms to a document profile, it provides a basis to guarantee interoperability. The document profile specifies the facilities required to process documents of that type, e.g. which image formats can be used, levels of scripting, style sheet support, and so on. This allows various groups of product designers to define their own standard profile. It also allows authors to avoid writing several different versions of documents for different clients.

XHTML expands the use of XML without making existing HTML elements obsolete. It is designed so authors can create Web pages that combine the data structure of XML and the presentation of HTML. It also allows authors to create Web pages without having to go through existing Web pages to strip out the tags and replace them to take advantage of the power of XML. The W3C also provides tools to convert HTML 4 documents into XHTML. XHTML also simplifies the the Web development process by eliminating the need to develop multiple versions of a document based on the type of device upon which it will be used.

For SFX See Librarian

by **Norman Desmarais** (Acquisitions Librarian, Phillips Memorial Library, Providence College, Providence, RI 02918; Phone: 401-865-2241; Fax: 401-865-2823) <normd@providence.edu>

One-stop information searching has been the holy grail for researchers since the appearance of electronic databases. Database aggregators like **Dialog**, **Ovid**, **SilverPlatter**, etc. provided a partial solution with a common user interface that allowed searching several databases with the same familiar interface. Z39.50 expanded that concept to allow researchers to access databases from different vendors, still using a familiar interface. Then came library portals that let librarians vet and brand the resources they opted to provide on their Web pages or library catalogs. **SFX** is the latest innovation to appear on the scene, introduced by **ExLibris** in early 2000.

SFX was developed by **ExLibris** the first company to capitalize on the work of **Herbert Van de Sompel** at the **University of Ghent** in Belgium. **Herbert Van de Sompel** developed the concept of the **OpenURL** which makes **SFX** possible. Soon after **ExLibris** introduced **SFX**, **Endeavor Information Systems** released **LinkFinder**. **EBSCO Publishing** and **Sirsi** unveiled their offerings, **LinkSource** and **Rooms**, at the **American Library Association's Midwinter Meeting** in January, 2003. The objective of these products is to provide seamless access to electronic resources in the fewest steps possible — a high expectation in view of the vast amount of information available on the Internet. Subsequent references to **SFX** should be understood to include these competitor products as well.

SFX is not a search engine; so it does not search databases or the Internet. Rather, it facilitates link-

continued on page 102

ing from a cited item to services or information relevant to that item. It parses the data from the citation to create an **OpenURL** which aims to deliver the appropriate copy to the researcher. **SFX** promises the ability of linking all of a library's resources. The **OpenURL** makes it easy for information resources to be compliant. It also lets the library control its information resources.

OpenURL

An **OpenURL** differs from a static URL in that it is not location dependent. A static link is embedded in the data and requires human or machine matching to be useful. As the number of static links increases, they become more difficult to manage and maintain; so static linking is not practical for large scale applications. A dynamic link, on the other hand, discovers links on the fly. It uses the item's metadata as the basis of the link and controls presentation based on the user and the institution.

An **OpenURL** consists of a base URL followed by a query which identifies the origin of a citation and describes the object sought. The base URL is the URL of the **SFX** server, a service-component that can take an **OpenURL** as input. The base URL will depend on the user (or the institution) and would look something like: *http://sfxserver.uni.edu/sfxmenu*. A question mark separates the base URL from the query which describes the origin of the transported metadata-object (the system that inserts the **OpenURL**, such as **Ovid:Medline** or **EBSCO:MFA**) as well as the metadata-object itself. So an **OpenURL** might look something like: *http://sfxserver.uni.edu/sfxmenu?sid=EBSCO:MFA&issn=12345678&date=1998& volume=12&issue=2&spage=134* where the query identifies the source as **EBSCOhost** and the desired article comes from volume 12, issue 2, dated 1998, of the journal with the ISSN=12345678. The article begins on page 134.

Link Resolver

The server on which the **SFX** or similar software is loaded is called a **Link Resolver**. Basically, the **SFX** software takes data elements from the referent (source that is referenced such as an indexing and abstract service) and creates an **OpenURL** that will then be used to locate the item. It calculates the links using a template or set of rules to construct a link.

The **Link Resolver** also contains a database of links, and information about title lists, coverage and embargo data for aggregators and packages, a list of the library's collections, and rule-based links. This allows the librarian to determine the rules to follow when retrieving and displaying search results. For example, a library may have access to the same journals through more than one service or aggregator. The **Link Resolver** lets the librarian specify the order in which the available services appear and provide the copy or set of services most appropriate for the researcher. If a library has a subscription to a title and access through a pay-per-view service, the librarian can specify that the pay-per-view service not appear as an option. Likewise, a library may routinely display an interlibrary loan or document delivery form for cited articles, but the librarian could suppress the form if the library subscribes to the title.

The **OpenURL** contains elements that identify the user and/or the institution and tells the information provider where the service component is located; so the **Link Resolver** can check a customer's databases and e-journal subscriptions to authenticate users to verify that they have the right to access the resources. When a researcher locates a citation from the *MLA Bibliography*, for example, the **Link Resolver** parses the citation and constructs an **OpenURL** which then serves to locate the item on a target resource such as **JSTOR**.

The **Link Resolver** presents the researcher with context-sensitive links that are dynamically configured on the basis of the institution's e-collections. Such resources are not limited to the full text of e-journals. They could include full-text repositories of any type; abstracting, indexing, and citation databases; OPACs; citations appearing in research articles; interlibrary loan and document delivery services; e-print systems; Web search services; other link resolvers; or A-Z lists from **Serials Solutions**. The **Link Resolver** can also link to **Google** to search for an author or to **Infotrieve** or **Ingenta** to purchase an article.

In summary, the **Link Resolver** takes the citation as input, enhances the incoming metadata, looks up the full text links in the database of links such as **CrossRef** or **EBSCO Article Matcher**, calculates rule-based links, applies filters to eliminate unwanted links, and prepares and presents the link menu to the researcher.

Managing the Link Resolver

The **Link Resolver** serves as the hub of the information wheel. It is more comprehensive and easier to manage than setting up individual static links and easier to keep information up to date. The first thing a librarian should do is to contact the library's information vendors to have them implement their databases. This is a serials management issue. Then, one sets the logic to determine what appears and what doesn't. For example, a librarian may want to specify that if **EBSCOhost** appears as a source don't show **Gale**. If the library subscribes to the full text, don't show the ILL form.

When a journal is bought by a new publisher or there's a change of aggregator, one need only make a change in the **Link Resolver** instead of changing the information for each title affected. The librarian can also customize URLs by determining the text and/or the icon displayed with the link, apply filters to control the display of local and global collections, apply additional rules to control the display such as to specify required fields, or hide resources if the full text is available in the library.

Licensing the software and purchasing and managing the **Link Resolver** can be an expensive proposition, typically running four figures minimum. Smaller institutions might want to consider an option like **Openly Informatics** 1cate software (*http://www.openly.com/1cate/about.html*) where the **Link Resolver** resides at **Openly Informatics** or **EBSCO's LinkSource**, hosted by **EBSCO**. We can also expect to see many information providers offering their own brand of **SFX** technology such as **WilsonLink's SFX**-powered technology.

continued on page 103

Innovations — SFX
from page 102

Having a **Link Resolver** on site allows customizing which resources are available and setting priorities among multiple resources. It gives the librarian maximum control over those available resources and the appearance of the menus and search options. An off-site **Link Resolver** eliminates the costs of buying and maintaining expensive hardware. On-site support staff is not required, reducing the cost of operation and staff time to only a few hours per month. However, the vendor or information provider configures and maintains the server and decides what resources are available. Librarians may have little or no control over what gets offered to their patrons.

OpenURL linking will eventually find its way into every library. Large libraries may opt to license **SFX** or similar software to allow them the greatest flexibility and control over their information resources. Other libraries may rely on **Link Resolvers** hosted by vendors or information providers. Producers of integrated library systems are also building **SFX** capability into their products. **Innovative Interfaces** offers **WebBridge** which can now link to **OCLC's First Search** and **ILLiad Resource Sharing Management** software in addition to all the **OpenURL** resources available to subscribing libraries. **Endeavor Information Systems** plans to integrate **FAST Data Search** into **ENCompass** by mid year and expects **FAST** to eventually replace the search engine for the **Voyager** integrated library system.