CODING WITH XML FOR EFFICIENCIES IN CATALOGING AND METADATA

PRACTICAL APPLICATIONS OF XSD, XSLT, AND XQUERY

TIMOTHY W. COLE
MYUNG-JA (MJ) K. HAN
CHRISTINE SCHWARTZ

ALAEditions
CHICAGO 2018

alastore.ala.org
# CONTENTS

**Preface** viii  
**Acknowledgments** ix

<table>
<thead>
<tr>
<th>One</th>
<th>Introduction</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two</td>
<td>A Quick Review of XML Basics</td>
<td>7</td>
</tr>
<tr>
<td>Three</td>
<td>Library Metadata in XML</td>
<td>21</td>
</tr>
<tr>
<td>Four</td>
<td>XML Validation Using Schemas</td>
<td>35</td>
</tr>
<tr>
<td>Five</td>
<td>An Introduction to XPath and XSLT</td>
<td>49</td>
</tr>
<tr>
<td>Six</td>
<td>Cataloging Workflows Using XSLT</td>
<td>75</td>
</tr>
<tr>
<td>Seven</td>
<td>Using XSLT to Connect Libraries to the Semantic Web</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td>Transforming XML Metadata into HTML+RDFa</td>
<td></td>
</tr>
<tr>
<td>Eight</td>
<td>An Introduction to XQuery for Library Metadata Including Use Cases</td>
<td>109</td>
</tr>
<tr>
<td>Nine</td>
<td>XQuery Basics</td>
<td>123</td>
</tr>
<tr>
<td>Ten</td>
<td>XQuery Functions Including Regular Expressions</td>
<td>131</td>
</tr>
</tbody>
</table>
Eleven  Metadata Workflow Using XQuery  139
Creating HathiTrust Submission Files for Monograph and Serials Print Holdings

Twelve  Resources for Further Study  147

APPENDIXES

A  Sample SPINE Metadata Record for an Emblem Book  155
   (discussed in chapter 7)
B  XSLT for Creating Book-Level HTML+RDFa  161
   (discussed in chapter 7)
C  XSLT for Creating Emblem-Level HTML+RDFa  171
   (discussed in chapter 7)
D  Software Configurations for Working with XML, XSD, XSLT, and XQuery  177

About the Authors  185
Index  187
This volume is a direct outgrowth of an all-day pre-conference workshop (www.ala.org/alcts/events/ac/2015/coding) held on June 25, 2015 at the Moscone Convention Center, San Francisco, in conjunction with that year’s Annual Conference of the American Library Association (ALA). The workshop was entitled “Coding for Efficiencies in Cataloging and Metadata: Practical Applications of XML, XSLT, XQuery, and PyMarc for Library Data.” The workshop was sponsored by the ALA’s Association for Library Collections & Technical Services (ALCTS) and cosponsored by the ALA’s Library and Information Technology Association (LITA) Program Planning Committee and the Online Audiovisual Catalogers, Inc. The presenters were Timothy W. Cole, Myung-Ja K. Han, Christine Schwartz, and Heidi Frank.

At the end of the day, the consensus of the presenters was that there hadn’t been time enough to share with the audience, primarily catalogers and metadata librarians, everything they collectively wanted the audience to know about Extensible Markup Language (XML) and how to use it in their work. This was not really unexpected. XML and its ancillary technologies, XML Schema (XSD), the Extensible Stylesheet Language for Transformations (XSLT), and the XML Query Language (XQuery), encompass a great deal of material, and there’s only so much that can be covered in a single-day workshop. The sense that more was needed, combined with the generally positive response to the workshop, led directly to the decision to write this book. The coauthors of this book include three of the four presenters from the 2015 pre-conference workshop (the workshop coverage of PyMarc is not included in this volume).

The goal of this volume is to introduce and illustrate concretely a few of the ways that XML technologies can be used in library cataloging and metadata management settings to enhance workflow effectiveness and efficiency. While still not comprehensive, the material covered in this volume provides a solid
foundation for librarians and library staff who are beginning to or are ready to integrate XML and its related technologies into their day-to-day work. While some prior exposure to XML is helpful, we start with a review of the basics of what XML is and how it is used in library cataloging and metadata workflows, and then move on to talk about workflows for validating XML metadata records, transforming XML metadata into HTML and to meet other kinds of workflow objectives, and querying collections of XML metadata records using XQuery. The final chapter provides information about additional resources for readers who are interested in learning more about advanced concepts, use cases, and techniques. Two of the authors also maintain public GitHub repositories which include XSLT (https://github.com/tcole3/XSLT/) or XQuery (https://github.com/caschwartz) examples.

This is a work written by practitioners, intended for practitioners and especially for librarians new to the field who need to come up to speed quickly on XML and how it is used by libraries today. While by no means the only technology arrow in a modern-day cataloger’s or metadata librarian’s knowledge and skills quiver, a firm understanding of XML remains relevant and helpful for those working in modern bibliographic control or with information discovery services. We hope you find this volume a useful resource, worthy of a place on your reference bookshelf.
INTRODUCTION

The Extensible Markup Language (XML) was introduced by the World Wide Web Consortium (W3C) in 1998. XML, a web-friendly subset of the preexisting Standard General Markup Language (SGML), was developed to facilitate the creation, use, and sharing (especially on the Web) of structured information. It is intentionally similar to the HyperText Markup Language (HTML), albeit more general-purpose, and is less focused on the presentation of information to users than is HTML. Libraries and publishers have widely adopted XML as a way to serialize and exchange bibliographic catalog records and other kinds of structured metadata about the books, special collection items, and other resources they produce, hold, or license. Because of its rigor and underlying approach to modeling structured information like metadata, libraries immediately found XML helpful, but on its own the XML standard is insufficient for developing efficient and general utility workflows that update, analyze, or otherwise process metadata to support library services and interoperability between libraries. Additional extensions and standards were needed (and were created) to support the construction and development of useful XML applications and workflows, that is, to create a full-featured XML ecosystem.
This volume is designed to introduce the XML Schema Definition language (XSD), the Extensible Stylesheet Language for Transformations (XSLT), and the XML Query language (XQuery) to catalogers, metadata librarians, developers, library science students, and others within the library and information science domain who are involved in creating, developing, or otherwise working with catalog record and metadata processing workflows. This introduction is meant to be practical and concrete, and our goal is to inform largely through illustrations drawn from our collective experiences in libraries working with catalog records and metadata in XML.

This volume is an outgrowth of an ALA Annual Conference pre-conference workshop held in San Francisco in June 2015 (www.ala.org/alcts/events/ac/2015/coding). Our objectives in writing this book, as they were in giving the workshop, are to describe and illustrate by example the means and methods of using XSLT and/or XQuery to edit metadata at scale, to streamline and scale up metadata and cataloging workflows, and to extract, manipulate, and construct MARC records and other formats and types of library metadata. We anticipate that this book will be most useful to readers who have encountered XML already in their work or studies, or who are at least familiar with another, related markup language such as HTML. Prior experience with XSD, XSLT, XPath, or XQuery is not required. Some familiarity with the Library of Congress MARC 21 format for bibliographic data (https://www.loc.gov/marc/bibliographic/) will help readers get the most from the examples and illustrations, but it is not expected that every reader will be a MARC expert or full-time MARC cataloger. Similarly, some familiarity with scripting or similar low-level programming is helpful, but not essential. This book is organized and designed to help even XML newbies get started using XSD, XSLT, and XQuery.

**HOW THIS BOOK IS ORGANIZED**

Depending on your level of experience with XML and library metadata, you’ll find that the first third of this volume is either an introduction or a refresher. Chapter 2, “A Quick Review of XML Basics,” spans the basics of XML terminology and the XML data model, starting from first principles and covering the differences between markup and content, elements and attributes, and well-formed and valid XML. The objective of chapter 2 is to lay a solid, shared foundation for reading the rest of the book. Chapter 3, “Library Metadata in XML,” illustrates with examples how common, standard descriptive metadata schemes used in libraries today can be expressed using XML syntax and
XML-compatible semantics. Chapter 4, “XML Validation Using Schemas,” introduces the XML Schema Definition language (XSD) and demonstrates how XSD files can be used to help validate the structure and semantics of XML metadata records that purport to conform to a specific metadata format or standard. While an understanding of XSD is not a prerequisite for using XSLT or XQuery, in practice most cataloging or metadata workflows include validation and quality control components, and XSD is a useful tool when building such workflow components. In addition, an examination of the XSD associated with a particular metadata format makes the creation and development of XSLT and XQuery workflow steps easier by exposing the full range of structures and semantics that can be encountered in a collection of metadata records.

Most of the rest of the book, chapters 5 through 11, focuses on the practical uses of XSLT and XQuery. Chapter 5, “An Introduction to XPath and XSLT,” provides an overview of the XML Path Language (which is foundational for both XSLT and XQuery) and an introduction to creating XSLT stylesheets for manipulating and transforming XML metadata records. Chapters 6 and 7 illustrate the use of XSLT in concrete library workflows. Chapters 8 and 9 introduce XQuery and basic library use cases involving XQuery. Chapter 10 looks at the use of functions in XQuery scenarios, and chapter 11 describes how XQuery is used for a metadata workflow that helps to create submission files for the HathiTrust Digital Library. Finally, chapter 12 provides links to additional resources for the reader wanting to learn more about the advanced features of these technologies.

While containing elements of both, this book is less a ready-reference than a tutorial on its subject. As discussed in chapter 12, the best ready-reference for XSD, XSLT, and XQuery when developing library cataloging or metadata workflows is the Web. The challenge is knowing enough to enter the correct search to find the XPath expression, XSLT instruction, or XQuery function call that will meet your need. This book is intended to give you that needed initial knowledge and to provide you with practical examples and starting points.

**XML TOOLS**

Readers are encouraged to delve deeper into and extend the examples and illustrations provided in this book as they read. There is no substitute for hands-on work with XML, XSD, XSLT, and XQuery when trying to learn and appreciate the nuances and functionalities of these technologies. Trial and error is an excellent teacher in this domain. As discussed in chapter 12, there are a
number of excellent tutorial and reference resources online that can be used to expand on and elaborate the basic discussions found in this book of the syntax and semantics of these standards. Appendix D, “Configurations for Working with XML, XSD, XSL T and XQuery,” which is based on instructions created for the workshop, provides tips for installing useful tools and setting up your environment for working with XML. When it comes to tools for creating, editing, viewing, and manipulating XML metadata, the reader has a number of good options available, both desktop-based and online.

Relatively little infrastructure is required to get started with XML. Files can be created using almost any plain text editor (e.g., Microsoft Notepad, distributed with Windows; TextEdit, distributed with the Apple operating systems; and gedit, a GNOME core application that is ubiquitous on Linux-based computers). Because UTF-8 (the 8-bit version of the Unicode Transformation Format) is the default character encoding for XML, it is even possible to embed non-keyboard characters in your XML using a plain text editor. (But beware of word processors that may use non-Unicode character encodings and special formatting for certain characters, e.g., word processor-specific quote marks, dashes, and the like. These will be displayed differently in an XML-aware application.) Once created and stored on your computer workstation, XML documents can be viewed (and checked for syntax conformance) using most modern web browsers. Figure 1.1 shows a simplified MARC 21 XML record with no associated style information as viewed in a web browser. As mentioned in chapter 5, most web browsers can even process a limited set of XPath expressions and XSL T instructions.

However, for any significant work with XML, XSD, XSL T, and XQuery, XML-specific tools are essential. Chapter 12 includes links to several websites for XML-specific tools, most of which are free, relatively inexpensive, or at least have relatively inexpensive educational/noncommercial purchase or licensing options available. XML editors, parsers, XQuery processors, transformation engines, and similar tool suites—for example, the SyncRO Soft oXygen XML editor, the Altova XMLSpy editor and development environment, and the IVI Technologies Stylus Studio X16 XML enterprise tool suite, to name just three available options—greatly facilitate the creation and development of XML, XSL T, and XQuery documents. These full-featured XML tools provide interfaces that are optimized for creating and editing XML documents, generating and validating XML schemas (XSD files), and developing and testing XSL T stylesheets and XQuery code. Most tools support multiple versions of these standards, including the most recent and up-to-date versions. When editing or creating an XML document that conforms to a particular XSD, these
Figure 1.1 | A simplified MARC XML record viewed in a web browser
tools provide hints and suggestions on screen as you type. Typos and errors in XSLT code are immediately highlighted as you type. Specialized views also are provided for creating and testing XSLT snippets and XQuery expressions.

In addition, for library developers and other programmers, most current scripting and programming languages (e.g., Python, JavaScript, VB.NET, Java) provide libraries with stable application programming interfaces (APIs) for generating and manipulating XML programmatically and for invoking XSLT and XQuery from within your code. This makes it feasible to use, transform, and query XML metadata records at scale and in batch mode.

The essential takeaway from this book is that learning and using XML, XSD, XSLT, and XQuery technologies enables librarians and developers to take advantage of powerful, XML-aware applications, facilitates the interoperability and sharing of XML metadata, and makes it possible to realize the full promise of XML to support more powerful and more efficient library cataloging and metadata workflows.

Note
A QUICK REVIEW OF XML BASICS

Markup languages reveal the structure of information resources. Knowledge of this structure can help a computer application optimize the way information is processed and displayed. For example, web browsers rely in part on the document structure exposed by the HyperText Markup Language (HTML) to know how to format a web page appropriately for the device you are using. Using XML to reveal the structure of a metadata record or other information resource can achieve multiple goals:

• it can enhance search precision;
• it can facilitate the collation of related information resources;
• it can make it easier to identify, differentiate, select, and extract information for ingestion into a database; and
• it can facilitate interoperability.

Markup can reveal and make explicit to a computer application various semantic relationships within and between resources, thereby facilitating automated inferencing and advanced information processing. So, how do markup languages like XML expose structure?
FIRST PRINCIPLES
Ordered Hierarchy of Content Objects

The underlying assumption of markup languages like HTML and XML is that textual information can be modeled as an Ordered Hierarchy of Content Objects (OHCO; an example of a tree data structure in computer science). For instance, the full text of a book might be defined as containing three types of top-level content objects: FrontMatter, Chapter(s), and BackMatter. A FrontMatter object might in turn contain Title, Author(s), and Publication Information content objects. A Chapter might contain Paragraph(s), Table(s), and Figure(s). A Paragraph might contain Sentence(s). And so on. Given this perspective, the full text of a digitized book can be serialized as an ordered hierarchy of content objects. The World Wide Web Consortium (W3C) XML Recommendation defines rules (syntax) for marking up (i.e., serializing) information in accord with the OHCO data model.

In contrast to HTML, which, though it also follows the OHCO data model, limits itself to a fixed set of content object labels (i.e., fixed semantics) and a fixed hierarchical structure, XML is a meta-markup language in that it includes mechanisms for defining your own content object labels (semantics), hierarchy, and content models. (Content models constrain exactly what each type of content object may contain.) Using XML, the same information can be marked up in multiple ways. This flexibility is both a strength and a limitation of XML. Labels, hierarchy, and content models can be selected and defined in accordance with intended use cases. This is a strength of XML. But illustrative of a limitation, interoperability can suffer if different implementers use different labels and content models, especially if content models assume different content object granularities. Generally, though, in the context of library metadata, XML with its OHCO data model is a natural fit for the highly structured information that is prevalent in catalog records and other forms of resource descriptions. XML supports the recognition, indexing, and reuse of the components found in library metadata and catalog records; for example, book title, author name(s), publication information, and so on. Adhering to XML syntax facilitates random access to metadata elements. Assuming that your metadata markup scheme requires each record to have a clearly labeled unique identifier, XML metadata are easy to manage. Assuming you use recognized authorities and select a scheme with sound semantics and content models, your XML metadata can be used to build indexes supportive of robust search, to create rich metadata displays for human readers, to generate text snippets formatted for inclusion in a journal article bibliography, and so on.
There are limitations, of course. The OHCO data model assumes that an XML document instance hierarchy has a single root content object and does not allow overlap within the hierarchy. For example, in marking up a full-text book, you may want to expose both the paragraph and page structure of the book; however, some paragraphs will cross page boundaries, violating the non-overlapping constraints of the OHCO data model. (There are workarounds and this is generally not a significant impediment for metadata markup.) In most library metadata contexts, the expressiveness and functional limitations of the OHCO approach to markup are negligible. On the whole, XML and its inherent OHCO model work well as a way to serialize library metadata.

Nor is OHCO the entire story. There is more to metadata workflows than simply exposing and labeling the structure of a catalog record. XML syntax on its own is not a sufficient solution. Ancillary standards are required to describe how to display and manipulate XML metadata (e.g., transform XML metadata into HTML). This adds complexity. Fortunately, the required ancillary standards, as well as tools based on these standards, exist (as we illustrate in subsequent chapters). But XML syntax is the foundation on which these ancillary standards are based. So in the rest of this chapter we describe the core syntax of XML and illustrate this syntax with a few examples. Libraries continue to use XML for metadata, even as, with an eye to the future, they also investigate complementary technologies (e.g., JavaScript Object Notation [JSON]).

DIVIDING THE WORLD INTO CONTENT AND MARKUP

In XML, there is content and there is markup. Markup is added to expose the structure of the content. Content in XML documents consists of parsed character data (PCData); for example, text, numbers, and references to special characters and non-textual information (entities). By default, PCData is encoded in UTF-8. (Metadata encoded using the MARC-8 standard is converted to UTF-8 for inclusion in XML.) Markup consists of Elements, Attributes, Comments, and Processing Instructions; these are discussed below. In XML, markup is set apart from content by angle brackets (‘<’ and ‘>’), with the exception that certain content may be included within angle brackets as unparsed strings (CDATA) or as the value of an XML Attribute. Figure 2.1 shows a simple example of bibliographic metadata serialized as XML. The content here is all PCData: the title of the book, the names of the authors, the name of the publisher, and the publication date. The string “2013–04–01,”
serialized as an attribute value in this example (more on this syntax later), is provided as an alternate way to express the \texttt{<PublicationDate>} element’s content, “April 2013.” Everything else in figure 2.1 is XML markup.

**HOW DOES XML COMPARE TO HTML?**

As alluded to above, HTML is itself a markup language. So why not just serialize library metadata in HTML? This can be done, of course, but there are drawbacks to such an approach. First, HTML is a display-oriented scheme for making web pages. It is designed to expose structure useful for display. HTML has a fixed set of element and attribute names. It lacks explicit labels and semantics that are useful for describing resources in libraries; for example, there are no HTML labels explicitly intended to contain content like subject headings, call number, place of publication, and so on, and HTML is not easily extensible. The best you can do is rely on conventions (e.g., agree to always put author names, and nothing else, in \texttt{<emph>} elements nested within \texttt{<h2>} elements). Such an approach is difficult to scale or maintain and quickly becomes unworkable for metadata of any complexity or diversity. XML, on the other hand, is highly extensible. In XML, system designers can define all the element and attribute names they need for their application. HTML has the advantage of default rules for displaying content, but with today’s stylesheet languages (e.g., the Cascading Style Sheet [CSS] language), most developers override these default display behaviors anyway, and CSS can be used with XML as easily as with HTML. So the extensibility and flexibility of XML makes more sense when it comes to serializing library metadata, but there are tradeoffs.

XML syntax is stricter than HTML syntax (catalogers sometimes appreciate this since it can reduce ambiguity). This requires more care when authoring XML, but it also facilitates robust validation. In practice, many libraries
use XML behind the scenes for certain workflows and then transform their XML metadata into HTML dynamically at the point of presenting metadata to library users. Among the ways in which XML is stricter than HTML are the following:

- In XML markup labels (e.g., Element and Attribute names) are case-sensitive.
- Attribute values must always be enclosed in quotes.
- Attributes must always have a value.
- Elements must always be explicitly closed (i.e., have a close tag as discussed below).
- XML does not include predefined character references or entities (described below) other than for ‘<’, ‘>’, ‘&’ and quotes (both single and double).

**XML ELEMENTS**

Elements are at the core of XML syntax. Elements delineate content objects. In XML, there are exactly four kinds of content models that can apply to an element:

1. Element content model (the element may only contain other elements)
2. PCData content model (element content is a parsed character string)
3. Mixed content model (may contain other elements with PCData interspersed)
4. Empty content model (may not contain any content; see an example below)

The element and mixed content models expose structural hierarchy through nesting. Figure 2.2 shows an XML fragment illustrating how elements may be nested (i.e., appear within other elements) to convey the hierarchical nature of metadata, in this case publication information. The following XML elements appear in this illustration: `<pubInfo>`, `<publisherName>`, `<placeOfPublication>`, `<yearOfPublication>`, `<city>`, and `<country>`. The elements `<placeOfPublication>` and `<pubInfo>` have element content models, while the other elements have PCData content models. XML element hierarchy is often represented graphically as a simple tree in two dimensions; elements are nodes in this tree view. Nodes with element or mixed content models may create branches in the tree hierarchy, with the terminal node of
each branch then being called a “leaf node.” The element <pubInfo> is at
the root of this XML fragment’s hierarchy. The elements <publisherName>,
<yearOfPublication>, and <placeOfPublication> are all at the second
level of the element hierarchy, with the first two of these also being leaf nodes.
The elements <city> and <country> are at the third level of the hierarchy
and also are terminal leaf nodes. As illustrated here, branches of the tree’s
hierarchy do not have to be (and rarely are) of equal length.

Notice also that in figure 2.2 each element name appears an even number
of times. Here the markup structure containing the first occurrence of an
element name, that is, without the leading ‘/’ character, is referred to as a
start tag. With the exception of empty elements, every start tag must have a
matching close tag. The element name also appears in the close tag where it
is preceded by a ‘/’ character. The W3C XML Recommendation defines rules
for element names; for example, element names may not contain spaces. As
discussed below, an element start tag may contain more than just the element
name; for example, Attributes, as illustrated in figure 2.1.

Empty elements are supported in XML. Empty elements contain no
content. They may appear in an XML metadata record because the element is
required to be present by the markup scheme even when no value is available
for a particular metadata record. For example, if the year of publication were
unknown in the figure 2.2 illustration, this element might appear as:

    <yearOfPublication></yearOfPublication>

Or an empty element may be a placeholder for content stored separately; for
example, as the <img> element in HTML is a placeholder for an image. Or an
element may include content as an attribute value. The presence of an empty
element in an XML metadata record is indicated by no characters or child ele-
ments, not even spaces, between the element’s start and close tag. Alternatively,
an empty element can be indicated by having ‘/’ as the last character of the start tag (in this case no close tag is required), for example:

```
<yearOfPublication />
```
the following, if the attribute authID were defined to be of attribute type ID:

```xml
<Author order="2" authID="a2">M. J. Han</Author>
```

then a consuming application would know that this is the only node in the XML file identified by the string “a2.” (ID type attribute values must be unique within an XML document instance.) Given this certainty, other elements can then use attributes of type IDREF or IDREFS to reference this ID in order to connect one content object (element) to another. This mechanism is particularly useful when serializing information from a relational database in XML. In our illustration, this is useful, for example, when associating an institutional affiliation with the author M. J. Han. Assuming that the attribute refID is of attribute type IDREF, the following node appearing elsewhere in the XML metadata record tells us M. J. Han’s affiliation:

```xml
<Affiliation refID="a2">University of Illinois at Urbana Champaign</Affiliation>
```

Note also that attributes that begin with the characters ‘xml’ or ‘xmlns’ are reserved and may have special meaning; for example, the attribute xml:lang is one way to convey the language of text in a content object. (Additional uses of these special attributes are discussed in chapters 4 and 5.)

**Special Content: Character Encoding, Entities, and Whitespace**

The default character encoding for string content in XML (i.e., PCData) is UTF-8 as defined by the Unicode Consortium. UTF-8 is a way to represent characters in the context of a computer application using 1, 2, 3, or 4 bytes (8, 16, 24, or 32 bits). UTF-8 is a superset of the older American Standard Code for Information Interchange (ASCII) encoding which represented characters
with 7 bits (and so supported less than 128 characters). UTF-8 can be used to encode the more than 65,000 characters in the Unicode Basic Multilingual Plane, thus facilitating the inclusion in XML of multilingual content (e.g., diacritics). In theory, UTF-8 can be used to express any of the more than one million characters in all 17 Unicode planes, but some XML applications do not recognize characters beyond the Basic Plane. While it is possible in XML to override the default character encoding, in order to maximize interoperability and content portability, most implementers do not (or if they do, they limit themselves to UTF-16, which is similar to UTF-8 in design and scope and which like UTF-8 is understood by all conformant XML parsers).

Of course, just because a character can be encoded in UTF-8 doesn’t mean it is easy to enter from the keyboard. When creating or editing the content of a metadata record, catalogers and metadata authors may choose to include such special characters that are not easily entered from the keyboard using XML character references (also known as character entities). In XML a character reference is a kind of markup that serves as a special character placeholder in PCData content and attribute values. Character references begin with an ampersand character, ‘&,’ and end with a semicolon. All Unicode character references begin ‘&#’ and include a number value which is the Unicode code point for the character desired. These code points are found in the Unicode Code Charts. The number can be in base 10 or base 16 (hexadecimal); in the latter case it is prefixed with the character ‘x.’ For the copyright symbol (©), the Unicode Code Point is 0169 (decimal) or 00A9 (hexadecimal). A character reference for this symbol could then be inserted as shown below (the options are equivalent). When either of these character references is resolved and the substitution made, the content becomes: © 2013.

```
<copyright>&#0169; 2013</copyright>
```

or

```
<copyright>&#x00A9; 2013</copyright>
```

Table 2.1 Internal character references built into XML

<table>
<thead>
<tr>
<th>CHARACTER REFERENCE</th>
<th>REPLACED BY</th>
<th>CHARACTER NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>&lt;</td>
<td>less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
<td>greater than</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</td>
<td>ampersand</td>
</tr>
<tr>
<td>'</td>
<td>‘</td>
<td>apostrophe</td>
</tr>
<tr>
<td>&quot;</td>
<td>“</td>
<td>quotation mark</td>
</tr>
</tbody>
</table>
INDEX

A
Adobe Brackets for Mac and Windows, 180
Altova, 73, 150, 181
Anderson, Clifford B., 152
APIs (application programming interfaces)
  defined, 124
  working with, 114–115
arguments, 124
ASCII (American Standard Code for
  Information Interchange), 14
atomic values, 124
attributes
  ID attribute, 13–14
  IDREF attribute, 14
  IDREFS attribute, 14
  overview, 13–14
  schemas, declaring attributes in, 40–43
B
BaseX
  downloading, 182
  installation, 182–183
  Mac installation, 182–183
  metadata records, setting up a database
    of, 183
  overview, 182
  Windows installation, 182
  XQuery main module, working in an,
    183–184
BibFrame, 106
bibliographic records, metadata standards and
  functional requirements for, 30–31
Bing, 92
book-level HTML+RDFa, generating,
  98–101, 161–169
books to use as resource, list of, 152–153
Boolean operators, 129
BracedURILiterals, 57
Brundage, Michael, 152
built-in functions, 131–136
Buxton, Stephen, 152
C
cataloging workflows
  electronic theses and dissertations (ETD)
    metadata transformed to MARC
      (Workflow 2), 81–88
  HathiTrust Digital Library handles
    added into printed books’ metadata
      (Workflow 1), 75–80
    overview, 75
CDATA (character data), 18–19
CDWA-Lite, 150
character encoding, 14–15
character references, 15–16
Code Beauty, 149
Cole, Timothy W., 152
comments, 17
complex documents, modifying, 117
complex type definition, 41–43
computed constructors, 125–126
count ( ) function, 135–136
counting data in XQuery, 115–116
Coursera, 147
Cover Pages, 148
creator of metadata as factor in choosing metadata standards, 31–32
crosswalks between metadata standards, list of resources for, 151–152
CSS (Cascading Style Sheet), 10

D
Darwin Core, 150
data () function, 135
data models, 124–125
debugging. See also testing
  XPath expressions, 56–57
  XSLT stylesheets, 70–73, 149–150
declarative elements in XSLT, 61–62
Definitive XML Schema (Walmsley), 153
digital asset management systems, metadata standards and, 30
digitization of library catalogs, Google and, 22
direct element constructors, 125–126
distinct-values() function, 135–136
DPLA (Digital Public Library of America), 26
DSpace, 30
Dublin Core
  creator name described in, 30–31
  overview, 26–27
  schema locations, 150
  subject elements, 38
Dublin Core Metadata Terms, 44–45

E
EAD (Encoded Archival Description), 150
electronic theses and dissertations (ETD) metadata transformed to MARC (Workflow 2)
mapping from IDEALS metadata to MARC, 81, 83
  overview, 81
  XSLT for transforming ETD metadata to MARC, 81–88
elements
  declaring, 39–43
  empty elements, 12–13
  nesting, 11–12
  overview, 11–13
  start tags, 12
Emblematica Online project (University of Illinois at Urbana-Champaign)
  book and copy information, mapping for, 94–95
book-level HTML+RDFa, generating, 98–101, 161–169
display labels, 98
emblem information, mapping for, 95–97
emblem-level HTML+RDFa, creating, 101–102, 171–175
mapping SPINE metadata to schema.org semantics, 93–97
overview, 90–93
sample SPINE metadata record for an emblem book, 155–160
SPINE schema, 91–93
SPINE XML metadata transformed to HTML+RDFa, 97–106
emblem-level HTML+RDFa, creating, 101–102, 171–175
terms, 15–16
expressions
  defined, 123
XPath
  authoring, 56–57
  debugging, 56–57
  overview, 50–52
  quotes used in, 51–52
  syntax supported by, 52
  types of, 50, 51
XQuery
  conditional expressions (if-then-else), 127–128
  FLWOR expressions, 112, 126–127
  logical expressions (and/or), 129
eXtensible Stylesheet Language
  Transformation. See XSLT (eXtensible Stylesheet Language Transformation)
eXtensible schemas, including and importing into XML schemas, 44

F
Fitzgerald, Michael, 152
flexibility of XML, 8
FLWOR expressions, 112, 126–127
for clause, 126
Freeformatter, 149
Free Webmaster Help, 148
functional requirements for bibliographic records, metadata standards and, 30–31
functions
  defined, 123
XPath, 52, 55, 56
XQuery

- built-in functions, 131–136
  - concat() function, 132
  - contains() function, 132
  - count() function, 135–136
  - data() function, 135
  - distinct-values() function, 135–136
  - matches() function, 134
  - nesting, 133
  - nodes and sequences, functions for working with, 134–136
  - normalize-space() function, 132–133
  - overview, 131
  - regular expressions, functions that use, 134
  - replace() function, 134
  - string() function, 135
  - string-join() function, 133
  - strings, functions for working with, 131–133
  - substring() function, 133
  - translate() function, 133
  - user-defined functions, 136–138

FunctX XQuery Functions, 149

H

HathiTrust project

- MARCXML records, exporting, 139–140
- multi-part print monographs, creating
  - TSV file for, 145
  - overview, 139
- serials, creating TSV file for, 140–143
- single-part print monographs, creating
  - TSV file for, 143–144

Herzog August Bibliothek, 91

Hitchens, Ron, 152

HTML (Hypertext Markup Language)

- overview, 10

XML compared, 10–11

XML metadata transformed into

HTML+RDFa

- book and copy information, mapping for, 94–95
- book-level HTML+RDFa, generating, 98–101, 161–169
- display labels, 98
- Emblematica Online and associated metadata, 90–93
- emblem information, mapping for, 95–97
- emblem-level HTML page, 101–102
- emblem-level HTML+RDFa, 171–175
- mapping SPINE metadata to schema.org semantics, 93–97
- SPINE XML metadata transformed to HTML+RDFa, 97–106

I

IBM developerWorks, 148

- ID attribute, 13–14
- identifiers, 117, 118
- identity transform, 70, 71
- IDEs (integrated development environments), 73

- IDREF attribute, 14
- IDREFS attribute, 14
- instructional elements, 61–62
- ISSN (International Standard Serial Number), 140

- items, 124

J

joins, 127

JSON-LD (JavaScript Object Notation for Linked Data), 107
K
Katz, Howard, 152
Kay, Michael, 152
Kelly, Ann, 152
Kelly, David James, 152
knowledge bases and ready references for XQueries and XSLTs, 148–149

L
Learning XML (Ray), 152
Learning XSLT: A Hands-On Introduction to XSLT and XPath (Fitzgerald), 152
Learn XQuery, 149
Lenz Consulting, 149
let clause, 126
Library Juice Academy, 147
library modules, 117–121
Library Workflow Exchange, 149
Liquid XML Studio, 150
literals, 123
LOD (linked open data), RDF and, 89
logical (and/or) expressions, 129

M
main modules, 124–127
Making Sense of NoSQL: A Guide for Managers and the Rest of Us (McCreary and Kelly), 152
manipulating/repurposing metadata, 113–114
MARC 21, 22, 150
MarcEdit, 150, 178–179
MARCXML
creator name described in, 30–31
exporting MARCXML records in HathiTrust project, 139–140
overview, 22–26
markup in XML documents, 9–10
markup languages
HTML. See HTML (Hypertext Markup Language)
OHCO (Ordered Hierarchy of Content Objects), 8–9
overview, 7
XML. See XML (eXtensible Markup Language)
matches() function, 134
McCreary, Dan, 152
Means, W. Scott, 152
Melton, Jim, 152
Metadata Maker tool, 31–32
metadata schema locations, 150–151
metadata standards
creator of metadata as factor in choosing, 31–32
digital asset management systems and, 30
functional requirements for bibliographic records and, 30–31
library selection of, 29–33
time as factor in choosing, 32–33
METS (Metadata Encoding and Transmission Standard), 150
MIS (Metadata for Images in XML), 150
MODS (Metadata Object Description Schema)
creator name described in, 30–31
invalid MODS example, 45–47
locations, 151
mapping to schema.org for Emblematica Online project, 94–95
overview, 27–29
in SPINE schema, 91
subject elements, 38
monographs, creating TSV file for print, 143–145
multi-part print monographs, creating TSV file for, 145

N
namespaces, 37–39, 57, 61
nesting functions, 133
955 datafield in MARCXML, 24–25, 76
nodes
defined, 124
functions for working with, 134–136
normalize-space() function, 132–133
Notepad++ for Windows, 180
numbers in XQuery, working with, 115

O
OAI-PMH (Open Archives Initiative-Protocol for Metadata Harvesting), 26, 44
OCLC (Online Computer Library Center) control numbers, 142
OHCO (Ordered Hierarchy of Content Objects), 8–9
Omeka, 30
OPAC (online public access catalog), 22
operands, 124
operators
defined, 124
XPath, 52, 56
order by clause, 126
oXygen XML editor (SyncRO Soft)
debugging XSLT, 73
downloading and installing, 177–178
namespace bindings for stand-alone XPath
testing, configuring, 57
website for, 150
XML validation error in, 45, 47

P
parameters
defined, 124
XSLT, 65–67
parsing metadata values with XQuery, 112–113
PBCore 2.1, 151
PCData (parsed character data), 9–10
Perfect XML, 148
PIs (processing instructions), 17–18
PREMIS (PREservation Metadata: Implementation Strategies), 151
Princeton Theological Seminary Library
(Case study)
APIs, working with, 114–115
complex documents, modifying, 117
counting data, 115–116
identifiers, working with, 117, 118
library modules, working with, 117–121
manipulating/repurposing metadata, 113–114
numbers, working with, 115
overview, 111
parsing metadata values, 112–113
programmatically working with library metadata, 111
querying datasets, 112
sequences, working with, 115–116
URIs, working with, 117
programmatically working with library metadata in XQuery, 111
programming concepts and vocabulary, 123–124

Q
qualified Dublin Core, 27
querying datasets with XQuery, 112

Querying XML: XQuery, XPath, and SQL/ XML in Context (Melton and Buxton), 152
quotes used in XPath expressions, 51–52

R
Ray, Erik T., 152
RDFa (Resource Description Framework in Attributes)
overview, 90
testing, 102, 105–106
XML metadata transformed into HTML+RDFa
book and copy information, mapping for, 94–95
book-level HTML+RDFa, generating, 98–101, 161–169
display labels, 98
Emblematica Online and associated metadata, 90–93
eblem information, mapping for, 95–97
eblem-level HTML page, 101–102
eblem-level HTML+RDFa, 171–175
mapping SPINE metadata to schema.org semantics, 93–97
SPINE XML metadata transformed to HTML+RDFa, 97–106
RDFa 1.1 (W3C), 106–107
RDF (Resource Description Framework), 89–90
recursion, 67–70
Reese, Terry, 178
regular expressions, functions that use, 134
replace() function, 134
repurposing/manipulating metadata, 113–114
result-trees, 58–61
return clause, 126
Riley, Jenn, 29
Robie, Jonathan, 152

S
schema.org, 89–90, 92–93
Seeing Standard: A Visualization of the Metadata Universe (Riley), 29
semantic web connection with XSLT
book and copy information, mapping for, 94–95
Index

semantic web connection with XSLT (cont.)
book-level HTML+RDFa, generating, 98–101
display labels, 98
Emblematica Online and associated metadata, 90–93
emblem information, mapping for, 95–97
emblem-level HTML page, 101–102
mapping SPINE metadata to schema.org semantics, 93–97
overview, 89–90
SPINE XML metadata transformed to HTML+RDFa, 97–106
sequences
defined, 125
functions for working with, 134–136
working with, 115–116
serials, creating TSV file for, 140–143
simple Dublin Core, 26–27
simple type definition, 41–43
Simpson, John E., 153
single-part print monographs, creating TSV file for, 143–144
software tips and instructions
BaseX, 182–184
MarcEdit, 178–179
oXygen (SyncRO Soft), 177–178
Stylus Studio XML Editor, 181
text editors for creating and editing XML files, 180–181
web browsers, XML files displaying in, 179–180
XML Spy XML Editor (Altova), 181
Stack Overflow, 149
string() function, 135
string-join() function, 133
strings, XQuery functions for working with, 131–133
Structured Data Testing Tool (Google), 102, 105–106
stylesheets
debugging, 70–73
overview, 61–64
testing, 70–73
Stylus Studio, 73, 150
Stylus Studio XML Editor, 181
substring() function, 133
syntax supported by XPath expressions, 52

T
EI (Text Encoding Initiative), 91, 151
templates for XSLT, 58–61
testing. See also debugging
RDFa (Resource Description Framework in Attributes), 102, 105–106
XSLT stylesheets, 70–73
XPath, 149
Texas Digital Library ETD, 151
text editors for creating and editing XML files, 4, 180–181
Tidwell, Doug, 153
time as factor in choosing metadata standards, 32–33
tools for metadata creation and transformation, 149–150
top-level template rules for XSLT stylesheets, 62
translate() function, 133
TSV (tab-separated values) files, 139–145
tuples, 123
tutorials and courses on XML, 147–148
Tutorials Point, 148
types in XML schemas, declaring, 40–43
Tzag.com, 148

U
Unicode Basic Multilingual Plane, 15
University of Illinois at Urbana-Champaign Library
HathiTrust Digital Library handles added into printed books’ metadata (Workflow 1), 75–80
Metadata Maker tool, 31–32
transforming electronic theses and dissertations (ETD) metadata to MARC (Workflow 2), 81–88
URIs (Uniform Resource Identifiers), 38–39, 76, 117
user-defined functions, 136–138
UTF-8, 9, 14–15

V
valid XML
invalid MODS example, 45–47
overview, 19–20, 36–37
well-formed XML compared, 19–20

alastore.ala.org
variables
defined, 123
XSLT, 64–67
Voyager RESTful APIs (Ex Libris), 139–140
VRA Core 4.0, 151

W
Walmsley, Priscilla, 134, 153
W3C (World Wide Web Consortium) XML Recommendation, 8, 11, 19
W3C (World Wide Web Consortium) XSLT Recommendation, 61
web browsers
debugging XSLT instances with, 72
simplified MARC XML record viewed in, 5
XML files displaying in, 4–5, 179–180
well-formed XML
overview, 19–20, 36
valid XML compared, 19–20
where clause, 126
whitespace characters, 16–17
Wicentowski, Joseph C., 152
workflows for cataloging
HathiTrust Digital Library handles added into printed books’ metadata (Workflow 1), 75–80
overview, 75
transforming electronic theses and dissertations (ETD) metadata to MARC (Workflow 2), 81–88
W3 Schools, 148, 149

X
XML-based metadata standards
Dublin Core, 26–27
MARCBXML, 22–26
MODS (Metadata Object Description Schema), 27–29
overview, 21–22
XML.com, 149
XML (eXtensible Markup Language)
attributes, 13–14
CDATA, 18–19
character encoding, 14–15
character references, 15–16
comments, 17
content, 9–10
declarations, 18
editors, 149–150
elements
empty elements, 12–13
nesting, 11–12
overview, 11–13
start tags, 12
entities, 15–16
flexibility of, 8
HTML compared, 10–11
markup, 9–10
metadata transformed into HTML+RDFa book-level HTML+RDFa, generating, 98–101, 161–169
emblem-level HTML+RDFa, 171–175
namespaces, 37–39, 57, 61
overview, 1
processing instructions, 17–18
schemas
attributes in, declaring, 40–43
elements in, declaring, 39–43
external schemas, including and importing, 44
grouping in, 44–45
namespaces, 37–39
overview, 36–37
types in, declaring, 40–43
tools for, 3–6
valid. See valid XML
well-formed, 19–20, 36
whitespace characters, 16–17
XQuery, XML constructors in, 125–126
XML for Catalogers and Metadata Librarians (Cole and Han), 152
XML in a Nutshell (Harold and Means), 152
XML Prolog, 18
XML Query Language. See XQuery (XML Query Language)
XML Schema Definition Language Part 2, Datatypes, 50
XML Spy XML Editor (Altova), 73, 181
XPath and XPointer (Simpson), 153
XPath and XQuery Functions and Operators 3.1 (W3C Recommendation), 56
XPath Tester, 149

alastore.ala.org
XPath (XML Path Language)
expressions
  authoring, 56–57
  debugging, 56–57
  evaluation of, 54
  overview, 50–52
  quotes used in, 51–52
  syntax supported by, 52
  types of, 50, 51
in XQuery, 126
functions, 52, 55, 56
operators, 52, 56
overview, 49, 111
testers, 149
versions of, 56–57

XQuery: Search across a Variety of XML Data
(Walmsley), 134, 153

XQuery: The XML Query Language
(Brundage), 152

XQuery for Humanists: Coding for Humanists
(Anderson, Robie, and Wicentowski), 152

XQuery from the Experts: A Guide to the W3C
XML Query Language (Katz), 152

XQuery Power, 149

XQuery Wikibook, 149

XQuery (XML Query Language)
  advantages of, 110
  data model, 124–125
  expressions
    conditional expressions (if-then-else), 127–128
    FLWOR expressions, 112, 126–127
    logical expressions (and/or), 129
    XPath expressions, 126
  functions
    built-in functions, 131–136
    concat() function, 132
    contains() function, 132
    count() function, 135–136
    data() function, 135
    distinct-values() function, 135–136
    matches() function, 134
    nesting, 133
    nodes and sequences, functions for working with, 134–136
    normalize-space() function, 132–133
  regular expressions, functions that use, 134
  replace() function, 134
  string() function, 135
  string-join() function, 133
  strings, functions for working with, 131–133
  substring() function, 133
  translate() function, 133
  user-defined functions, 136–138

HathiTrust project
  MARCXML records, exporting, 139–140
  multi-part print monographs, creating TSV file for, 145
  overview, 139
  serials, creating TSV file for, 140–143
  single-part print monographs, creating TSV file for, 143–144
  joins, 127
  main modules, 124–127
  overview, 50, 109–110, 131

Princeton Theological Seminary Library
  (Case study)
  APIs, working with, 114–115
  complex documents, modifying, 117
  counting data, 115–116
  identifiers, working with, 117, 118
  library modules, working with, 117–121
  manipulating/repurposing metadata, 113–114
  numbers, working with, 115
  overview, 111
  parsing metadata values, 112–113
  programmatically working with library metadata, 111
  querying datasets, 112
  sequences, working with, 115–116
  URIs, working with, 117
  structure, 124
  uses for, 110
  XML constructors, 125–126

XSLT: Mastering XML Transformations
(Tidwell), 153
XSLT 2.0 and XPath 2.0: Programmer's Reference (Kay), 152
XSLT (eXtensible Stylesheet Language Transformation)
cataloging workflows
HathiTrust Digital Library handles added into printed books’ metadata (Workflow 1), 75–80
overview, 75
transforming electronic theses and dissertations (ETD) metadata to MARC (Workflow 2), 81–88
debuggers, 149–150
declarative elements, 61–62
identity transform, 70, 71
instructional elements, 61–62
namespaces, 61
overview, 49–50, 58
parameters, 65–67
recursion, 67–70
result-trees, 58–61
stylesheets
debugging, 70–73
overview, 61–64
testing, 70–73
templates, 58–61

XML metadata transformed into HTML+RDFa
book and copy information, mapping for, 94–95
book-level HTML+RDFa, generating, 98–101, 161–169
display labels, 98
Emblematica Online and associated metadata, 90–93
emblem information, mapping for, 95–97
emblem-level HTML page, 101–102
emblem-level HTML+RDFa, 171–175
mapping SPINE metadata to schema.org semantics, 93–97
SPINE XML metadata transformed to HTML+RDFa, 97–106

XSLT Jumpstarter: Level the Learning Curve and Put Your XML to Work (Kelly), 152