Extensible Stylesheet Language for Transformation (XSLT)

Timothy W. Cole & Myung-Ja Han
University of Illinois at Urbana-Champaign

2015 ALA Annual Conference
ALCTS Preconference
25 June 2015
San Francisco, California
XSLT - Transforming Style Sheets

• Language for transforming XML documents
  • Into HTML, Text, or other XML documents
  • Supported by mainstream browsers (IE, Firefox, Chrome, ...)
  • Applied variously in batch mode, server-side, and client-side

• Invaluable for workflows, interoperability and reusability

```xml
<xsl:template match='//author'>
  <xsl:element name='dc:creator'>
    <xsl:value-of select='lastname'/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select='firstname'/>
  </xsl:element>
</xsl:template>
```
Workflow for XSLT

XML Document -> XSLT Stylesheet -> XSLT Processor -> Output: XML, XHTML, or Text

XSLT processing parsers may be DOM (tree) based, or Streaming (e.g., SAXON-HE).
XSLT is a primarily declarative / functional (pattern & rule-based) programming language
• An XSLT style sheet consists of one or more sets of rules (templates) that specify what not how:
  • e.g., If FOO is found in input, then BAR should appear in output
  • Basic paradigm is pattern matching (inherits from SGML,...)
  • Recursion important in XSLT; variables are fixed once assigned

• Therefore XSLT is a declarative / functional language, rather than an imperative / procedural (instruction-based) language.

• A key attribute of <template> element is match:
  <xsl:template match="/"/> (matches on root of doc)
XSLT Relies on XPath

- XPath expressions are used navigate & identify the values & structures of interest from the source XML;
- XSLT semantics are then applied implement the rules which yield the result tree.
- The atom of the XPath data model is node.
- There are 7 kinds of nodes: document, element, attribute, text, namespace, processing-instruction, and comment
- XPath is expression-based, with core operators & functions
  - Simplest expressions are string & numeric literals: ‘hi’, 7.5
  - Other kinds of expressions: arithmetic, comparison, logical, path, for, conditional, sequence, ...
XPath 2.0 Syntax

XPath assumes a tree structure; node navigation is by axis: e.g., *parent* axis, *child* axis, *ancestor* axis, *descendant* axis, *preceding-sibling* axis, *following-sibling* axis, etc.

XPath is composable – expressions are built step-wise;

‘/’ – is path expression for document root node

‘/oai_dc:dc/dc:creator’ – resolves to node-list of creators

Path expressions can be filtered using predicates ‘[]’

‘/oai_dc:dc/dc:creator[2]/text()’ – text of 2nd creator node


XPath allows use of an abbreviated syntax

XPath 2.0 expressions return sequences of node-lists (0, 1 or more in number) or atomic values (numbers, strings, ...)
XPath Operators & Functions

• Operators
  Arithmetic operators: +, -, *, div, mod, ...
  Comparisons, sequences: =, !=, <, >, <=, >=
  values: eq, ne, lt, gt, le, ge

• Functions
  Arithmetic: count, sum, round, floor, ceiling, ...
  String: substring, substring-before, starts-with, concat, ...
  Node properties: name, local-name, namespace-uri, ...
  Context: position(), last(), current-time(), ...
Using oXygen XPath Builder Window
**Expression** (unabbreviated): /child::oai_dc:dc/child::dc:creator
   (abbreviated):   /oai_dc:dc/dc:creator

*Results in a sequence of 2 nodes:*

<creator> Cole, Timothy W</creator>
<creator> Foulonneau, Muriel</creator>

**Expression** (unabbreviated): /child::oai_dc:dc/child::dc:title/child::text()
   (abbreviated):   /oai_dc:dc/dc:title/text()

*Results in the singleton text node child of <title>:*

Using the Open Archives Initiative protocol ...

**Expression** (unabbreviated): string(/child::oai_dc:dc/child::dc:creator[2])
   (2.0 alternative version):   /child::oai_dc:dc/child::dc:creator[2]/fn:string(.)
   (abbreviated):   string(/oai_dc:dc/dc:creator[2])
   (2.0 abbreviated):   /oai_dc:dc/dc:creator[2]/string(.)

*Results a value of data type string:*

"Foulonneau, Muriel"
**Expression:**
count(//dc:creator)

*Returns integer value:*
2

**Expression:**
//dc:creator[count(//dc:creator)]
Alternative expression: //dc:creator[last()]

*Returns:*
<creator>Foulonneau, Muriel</creator>

**Expression:**
name((/oai_dc:dc/*)[position() = 5])
Alternative expression: name((/oai_dc:dc/*)[5])

*Returns string value:*
"date"
XSLT – Basic concepts

- XSLT elements build the Result Tree(s)
- XSLT uses variables & parameters
- Recursion is used extensively in XSLT
<table>
<thead>
<tr>
<th>XSLT Element</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;xsl:template&gt;</code></td>
<td>Declares rule(s), i.e., sequence constructor(s), for creating result-tree nodes and/or atomic values.</td>
</tr>
<tr>
<td><code>&lt;xsl:variable&gt;</code></td>
<td>Declares a variable (binding between name and a value) that can be referenced during a subsequent step of the transform. Parameter values are set when stylesheet or template is invoked.</td>
</tr>
<tr>
<td><code>&lt;xsl:param&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;xsl:apply-templates&gt;</code></td>
<td>An instruction that allows the invocation of templates matching context node or node pattern specified by select attribute.</td>
</tr>
<tr>
<td><code>&lt;xsl:call-template&gt;</code></td>
<td>Used to invoke templates by name rather than by match. The <code>&lt;xsl:with-param&gt;</code> is optional child of <code>&lt;xsl:call-template&gt;</code>, used to set value of parameter in called template.</td>
</tr>
<tr>
<td><code>&lt;xsl:with-param&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;xsl:element&gt;</code></td>
<td>Instructions used to add a new node or attribute to the result-tree.</td>
</tr>
<tr>
<td><code>&lt;xsl:attribute&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;xsl:text&gt;</code></td>
<td>An instruction used to add atomic string values to result-tree.</td>
</tr>
<tr>
<td><code>&lt;xsl:value-of&gt;</code></td>
<td>Used to construct a new text node in the result tree. Value is value of node specified in select attribute or result of evaluating contained sequence constructor.</td>
</tr>
<tr>
<td><code>&lt;xsl:for-each&gt;</code></td>
<td>Used to transform a sequence node by node by evaluating the contained sequence constructor for each node.</td>
</tr>
<tr>
<td><code>&lt;xsl:if&gt;</code></td>
<td>Contained sequence constructor is evaluated if value of test attribute is True; otherwise skipped.</td>
</tr>
<tr>
<td><code>&lt;xsl:choose&gt;</code></td>
<td><code>&lt;xsl:when&gt;</code> and <code>&lt;xsl:otherwise&gt;</code> are children of <code>&lt;xsl:choose&gt;</code> that each contain a sequence constructor. No more than one sequence constructor is evaluated according value of <code>&lt;xsl:when&gt;</code> test attributes.</td>
</tr>
<tr>
<td><code>&lt;xsl:when&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;xsl:otherwise&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;xsl:document&gt;</code></td>
<td>Used as root of an XML source-tree referenced by the transform (i.e., in addition to the primary source-tree being transformed).</td>
</tr>
</tbody>
</table>
<xsl:value-of>

• Requires select attribute
• Is placeholder for value of the element or computed value selected from document being processed
• Replaced in output stream (result tree) by value
Source XML File:
http://www.w3schools.com/xsl/cdcatalog.xml

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
<cd>
<title>Empire Burlesque</title>
<artist>Bob Dylan</artist>
<country>USA</country>
<company>Columbia</company>
<price>10.90</price>
<year>1985</year>
</cd>
...
</catalog>
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th align="left">Title</th>
<th align="left">Artist</th>
</tr>
<xsl:for-each select="catalog/cd">
<tr>
<td><xsl:value-of select="title"/></td>
<td><xsl:value-of select="artist"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
Link the XSL Style Sheet to the XML Document:
http://www.w3schools.com/xsl/cdcatalog_with_xsl.xml

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  ...
</catalog>
```
Output of XSL-T

<html>
<body>
   <h2>My CD Collection</h2>
   <table border="1">
      <tr bgcolor="#9acd32">
         <th align="left">Title</th>
         <th align="left">Artist</th>
      </tr>
      <tr>
         <td>Empire Burlesque</td>
         <td>Bob Dylan</td>
      </tr>
   </table>
</body>
</html>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" ... 
exclude-result-prefixes="xs oai_dc dc"
version="2.0"

<xsl:output
method="xml" version="1.0"
encoding="UTF-8" indent="yes"
omit-xml-declaration="no"/>

<xsl:template match="/">
Rules & Variables in XSLT are Context Sensitive

• Rules & XPath expressions – e.g., test and select attribute values are resolved relative to the current context node (scope).
  • The context within a template is the matching node, each time
  • Context within an `<xsl:for-each>` (discussed next) becomes the node matching the select attribute value
  • `<xsl:apply-templates>` and `<xsl:call-templates>` shifts rule-processing to a different template, with different context
  • Variables can be global (declared outside all templates), or can be scoped to a specific context …

<xsl:element name="schema:Book">

... 

<xsl:for-each select="/oai_dc:dc/dc:title">
  <xsl:element name="schema:name">
    <xsl:value-of select="."/>
  </xsl:element><!-- closes name-->
</xsl:for-each>

</xsl:element> <!-- closes Book-->
Loops, Branches, and Sorting

**XSL:for-each**

```xml
<xsl:for-each select = "node-set-expression">
... rules, pattern-matching, etc. within this new context
</xsl:for-each>
```

**XSL:if**

```xml
<xsl:if test = "boolean-expression">
... rules, pattern-matching, etc. – context does not change
</xsl:if>
```

**XSL:sort** *(operates on current context node – sorts output)*

```xml
<xsl:sort
select = "string-expression"
data-type = "text" | "number" | "prefixedName" | lang = "langcode"
order = "ascending" | "descending"
case-order = "upper-first" | "lower-first" />
```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <html>
      <body>
        <h2>My CD Collection</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th>Title</th>
            <th>Artist</th>
          </tr>
          <xsl:for-each select="catalog/cd">
            <xsl:sort select="artist"/>
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="artist"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>

My CD Collection

<table>
<thead>
<tr>
<th>Title</th>
<th>Artist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empire Burlesque</td>
<td>Bob Dylan</td>
</tr>
<tr>
<td>The very best of</td>
<td>Cat Stevens</td>
</tr>
<tr>
<td>Greatest Hits</td>
<td>Dolly Parton</td>
</tr>
</tbody>
</table>

Sort On an element
My CD Collection

<table>
<thead>
<tr>
<th>Title</th>
<th>Artist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empire Burlesque</td>
<td>Bob Dylan</td>
</tr>
<tr>
<td>Greatest Hits</td>
<td>Dolly Parton</td>
</tr>
</tbody>
</table>

if test inside <xsl:for-each>
Another branching option

XSL:choose (contains one or more \texttt{xsl:when}, an optional \texttt{xsl:otherwise})

\begin{verbatim}
<xsl:choose>
  <!-- (xsl:when+, xsl:otherwise?) -->
</xsl:choose>
\end{verbatim}

XSL:when (only appears as a child of an \texttt{xsl:choose} element)

\begin{verbatim}
<xsl:when>
  test = "boolean-expression"
</xsl:when>
\end{verbatim}

XSL:otherwise

\begin{verbatim}
<xsl:otherwise>
  <!-- template -->
</xsl:otherwise>
\end{verbatim}
Adding attributes to elements in the result-tree

<xsl:element name="schema:Book">
  <xsl:attribute name="rdf:about">
  </xsl:attribute>
  ...
</xsl:element>  <!-- closes Book-->
References:

W3C XSLT Tutorial:
http://www.w3schools.com/xsl/

Zvon XSL Tutorial:
http://www.zvon.org/comp/r/tut-XSLT_1.html

XPath & XSLT reference:
http://www.w3schools.com/xsl/xsl_w3celementref.asp
http://www.w3schools.com/xsl/xsl_functions.asp

http://www.zvon.org/comp/r/ref-XPath_2.html
http://www.zvon.org/comp/r/ref-XSLT_2.html
Exercise 3 – Create a mapping and an XSLT for transforming a valid MODS records into HTML
http://quest.library.illinois.edu/ALA2015/ALCTS-XML/Exercises/Exercise3/

• Detailed Instructions

• Use your record from Exercise 2 or this sample MODS Record
http://quest.library.illinois.edu/ALA2015/ALCTS-XML/Exercises/Exercise3/Sample3-MODS.xml

• Template for mapping MODS to HTML

• Template4Exercise3 (an XSLT to get you started)
Exercise 3 Solutions

• Our Solution:

  Mods2html-ALA.xsl
  http://quest.library.illinois.edu/ALA2015/ALCTS-ExerciseSolutions/Exercise3/Mods2html-ALA.xsl

  Sample3MODS-HTML.html (result of applying XSLT on server)
  http://quest.library.illinois.edu/ALA2015/ALCTS-ExerciseSolutions/Exercise3/Sample3MODS-HTML.html

  Sample3MODS-XSLTpi.xml (includes stylesheet Processing Instruction; XSLT is applied by the client, e.g., Web browser)
  http://quest.library.illinois.edu/ALA2015/ALCTS-ExerciseSolutions/Exercise3/Sample3-MODS-XSLTpi.xml
Exercise 3
Creating an XSLT that transforms MODSXML to HTML

1. Now you have four MODS metadata (or one MODS collection metadata), think about how you want to display them in a webpage.

2. Create a table that shows mapping between MODS elements to HTML tags.

3. Think about what XSLT logic elements (nodes) you should use. For this process, take a few things into consideration:
   a. Do you want to display an empty HTML tag when the MODS element doesn’t have a value?
   b. Do you want to use the value as is in the MODS element?
   c. Do you want to combine values from multiple MODS elements?
   d. Do you want to add the default value as displayed on the HTML page?
<table>
<thead>
<tr>
<th>HTML Tag</th>
<th>MODS Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td></td>
<td>Book Information</td>
</tr>
<tr>
<td>Title</td>
<td>&lt;titleInfo&gt;&lt;title&gt;</td>
<td>Title of the book</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
XML for catalogers and metadata librarians

Timothy W. Cole
Myung-Ja K. Han

Text

Place: Santa Barbara, California

Date issued: 2013

Physical description:
xiv, 388 pages : illustrations ; 26 cm.

Abstract:
This book provides a foundation of knowledge for catalogers, metadata librarians, and library school students on the Extensible Markup Language (XML)—one of the most commonly listed qualifications in today's cataloger and metadata librarian job postings. How are today's librarians to manage and describe the ever-expanding volumes of resources, in both digital and print formats? The use of XML in cataloging and metadata workflows can improve metadata quality, the consistency of cataloging workflows, and adherence to standards. This book is intended to enable current and future catalogers and metadata librarians to progress beyond a bare surface-level acquaintance with XML, thereby enabling them to integrate XML technologies more fully into their cataloging workflows. Building on the wealth of work on library descriptive practices, cataloging, and metadata, XML for Catalogers and Metadata Librarians explores the use of XML to serialize, process, share, and manage library catalog and metadata records. The authors' expert treatment of the topic is written to be accessible to those with little or no prior practical knowledge of or experience with how XML is used. Readers will gain an educated appreciation of the nuances of XML and grasp the benefit of more advanced and complex XML techniques as applied to applications relevant to catalogers and metadata librarians.”--Publisher's website.
### BOOK INFORMATION

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Role Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jane Smith</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Book Title: *Title of the Book*

Names Associated with the Book:

- John Doe
- Jane Smith
<table>
<thead>
<tr>
<th>Notes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject:</td>
</tr>
<tr>
<td>Identifier:</td>
</tr>
</tbody>
</table>

### HOLDINGS INFORMATION

<table>
<thead>
<tr>
<th>Holdings Library:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Call Number</td>
<td>Shelf Locator</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
<xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:mods="http://www.loc.gov/mods/v3"
    version="1.0">
    <xsl:output method="html" encoding="UTF-8"/>
    <xsl:template match="/">
        <hr width="80%" style="color: #fcb100"/>
        <table cellspacing="3" width="100%">
            <tr>
                <td height="1" width="5%">&#160;</td>
                <td height="1" width="5%">&#160;</td>
                <td height="1">&#160;</td>
                <td height="1">&#160;</td>
            </tr>
            <tr align="left" valign="top">
                <td colspan="4">
                    <h3>BOOK INFORMATION</h3>
                </td>
            </tr>
            <tr align="left" valign="top">
                <td colspan="3">
                    <nobr>
                        <b>Book Title:</b>
                    </nobr>
                </td>
                <td>
                </td>
            </tr>
            <xsl:if test="mods:mods/mods:name">
                <tr align="left" valign="top">
                    <td colspan="3">
                        <nobr>
                            <b>Names Associated with the Book:</b>
                        </nobr>
                    </td>
                    <td>
                        <xsl:for-each select="mods:mods/mods:name">
                            <xsl:value-of select="mods:namePart"/>
                            ,
                            <xsl:value-of select="mods:namePart[2]"/>
                            (<xsl:value-of select="mods:role/mods:roleTerm"/>)
                            <br/>
                        </xsl:for-each>
                    </td>
                </tr>
            </xsl:if>
        </table>
    </xsl:template>
</xsl:stylesheet>
<table>
<thead>
<tr>
<th><strong>Publisher:</strong></th>
<th><strong>Publication Date:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Physical Description:**

<table>
<thead>
<tr>
<th><strong>Language:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Note:**
<td colspan="3">
<nobr>
<Notes/>
</nobr>
</td>
<xsl:for-each select="mods:mods/mods:note">
<td>
<xsl:value-of select="."/>
</td>
</xsl:for-each>
</tr>
</xsl:if>

<xsl:if test="mods:mods/mods:subject">
<tr align="left" valign="top">
<td colspan="3">
<nobr>
<b>Subject:</b>
</nobr>
</td>
<td>
<xsl:for-each select="mods:mods/mods:subject">
<xsl:value-of select="."/><br/>
</xsl:for-each>
</td>
</tr>
</xsl:if>

<xsl:if test="mods:mods/mods:identifier">
<tr align="left" valign="top">
<td colspan="3">
<nobr>
<b>Identifier:</b>
</nobr>
</td>
<td>
<xsl:for-each select="mods:mods/mods:identifier">
<xsl:value-of select='./@type'/><xsl:text>: </xsl:text>
<xsl:value-of select="."/><br/>
</xsl:for-each>
</td>
</tr>
</xsl:if>

<br/>

<tr align="left" valign="top">
<td colspan="3">
<h3>HOLDINGS INFORMATION</h3>
</td>
</tr>
<xsl:for-each select="mods:mods/mods:location/mods:holdingSimple/mods:copyInformation">
<tr align="left" valign="top">
<td colspan="3">
<nobr>
<Holdings Library:></nobr>
</td>
</tr>
</xsl:for-each>
<table>
<thead>
<tr>
<th>Call Number</th>
<th>Shelf Locator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy Number</td>
<td>Item Identifier</td>
</tr>
<tr>
<td>Availability</td>
<td>Note</td>
</tr>
</tbody>
</table>
### BOOK INFORMATION

**Book Title:** XML for catalogers and metadata librarians  
**Names Associated with the Book:** Cole, Timothy W. (author)  
**Publisher:** Libraries Unlimited, an imprint of ABC-CLIO, LLC  
**Physical Description:** xiv, 388 pages : illustrations ; 26 cm  
**Notes:** Includes bibliographical references and index.
| Subject: | XML (Document markup language)  
|         | Cataloging  
|         | Data processing  
|         | Metadata  
| Languages Disciplines: | Library & Information Science / General  
| Languages Disciplines: | Library & Information Science / Cataloging & Classification  
| XML |  

| Identifier: | isbn: 9781598845198 (pbk. : acid-free paper)  
|            | isbn: 1598845195 (pbk. : acid-free paper)  
|            | isbn: 9781610692915 (ebook)  
|            | lccn: 2013006978  

| HOLDINGS INFORMATION: |  
| Holdings Library: | Social Sciences, Health, and Education  
| Call Number: | Z678.93.X54 C65 2013  
| Copy Number: | 1  
| Availability: | Available  

| Available |


XSLT Exercise
Adding an xml-stylesheet Processing Instruction (for styling by the client Web browser)

Available online
http://quest.library.illinois.edu/ALA2015/ALCTS-ExerciseSolutions/Exercise3/Sample3-MODS-XSLTpi.xml

<?xml version="1.0" encoding="utf-8"?>

<?xml-stylesheet type="text/xsl" href="Mods2html-ALA.xsl"?>

  <titleInfo>
    <title>XML for catalogers and metadata librarians</title>
  </titleInfo>
  <name type="personal" usage="primary">
    <namePart type="family">Cole</namePart>
    <namePart type="given">Timothy W.</namePart>
    <role>
      <roleTerm type="text" authority="marcrelator">author</roleTerm>
      <roleTerm type="code" authority="marcrelator">aut</roleTerm>
    </role>
  </name>
  <name type="personal">
    <namePart type="family">Han</namePart>
    <namePart type="given">Myung-Ja K.</namePart>
    <role>
      <roleTerm type="text" authority="marcrelator">author</roleTerm>
      <roleTerm type="code" authority="marcrelator">aut</roleTerm>
    </role>
  </name>
  <typeOfResource>text</typeOfResource>
  <originInfo>
    <place>
      <placeTerm type="code" authority="marccountry">cau</placeTerm>
      <placeTerm type="text" authority="marccountry">Santa Barbara, California</placeTerm>
    </place>
    <dateIssued encoding="marc">2013</dateIssued>
    <issuance>monographic</issuance>
    <publisher>Libraries Unlimited, an imprint of ABC-CLIO, LLC</publisher>
  </originInfo>
  <language>
    <languageTerm authority="iso639-2b" type="code">eng</languageTerm>
  </language>
  <physicalDescription>
    <form authority="marcform">print</form>
    <extent>xiv, 388 pages : illustrations ; 26 cm.</extent>
  </physicalDescription>
  <abstract>"This book provides a foundation of knowledge for catalogers, metadata librarians, and library school students on the Extensible Markup Language (XML)--one of the most commonly listed qualifications in today's cataloger and metadata librarian job postings. How are today's librarians to manage and describe the ever-expanding volumes of resources, in both digital and print formats? The use of XML in cataloging and metadata workflows can improve metadata quality, the consistency of cataloging workflows, and adherence to standards. This book is intended to enable current and future catalogers and metadata librarians to progress beyond a bare surface-level acquaintance with XML, thereby enabling them to integrate XML technologies more fully into their cataloging workflows. Building on the wealth of work on library descriptive practices, cataloging, and metadata, XML for Catalogers and Metadata Librarians explores the use of XML to serialize, process, share, and manage library catalog and metadata records. The authors' expert treatment of the topic is written to be accessible to those with little or no prior practical knowledge of or
experience with how XML is used. Readers will gain an educated appreciation of the nuances of XML and grasp the benefit of more advanced and complex XML techniques as applied to applications relevant to catalogers and metadata librarians." --Publisher's website.


<subject authority="lcsh">
  <topic>XML (Document markup language)</topic>
</subject>

<subject authority="lcsh">
  <topic>Cataloging</topic>
  <topic>Data processing</topic>
</subject>

<subject authority="lcsh">
  <topic>Metadata</topic>
</subject>

<subject authority="bisacsh">
  <topic>LANGUAGE ARTS &amp; DISCIPLINES / Library &amp; Information Science / General</topic>
</subject>

<subject authority="bisacsh">
  <topic>LANGUAGE ARTS &amp; DISCIPLINES / Library &amp; Information Science / Cataloging &amp; Classification</topic>
</subject>

<subject authority="gnd">
  <topic>XML</topic>
</subject>

<classification authority="ddc" edition="23">006.7/4</classification>

<relatedItem type="series">
  <titleInfo>
    <title>Third millennium cataloging</title>
  </titleInfo>
</relatedItem>

<note type="bibliography">Includes bibliographical references and index.</note>

<identifier type="isbn">9781598845198 (pbk. : acid-free paper)</identifier>

<identifier type="isbn">1598845195 (pbk. : acid-free paper)</identifier>

<identifier type="isbn">9781610692915 (ebook)</identifier>

<identifier type="lccn">2013006978</identifier>

<location>
  <physicalLocation>University of Illinois at Urbana-Champaign. Library</physicalLocation>
</location>

<copyInformation>
  <subLocation>Social Sciences, Health, and Education</subLocation>
  <shelfLocator>Z678.93.X54 C65 2013</shelfLocator>
  <note>Available</note>
  <itemIdentifier type="copyNumber">1</itemIdentifier>
</copyInformation>

</mods>