Lecture 8: XQuery 1.0 and DTD

19 November 2007

David J. Malan
malan@post.harvard.edu
Last Time

HTTP 1.1, JavaServer Pages 2.1, and Java Servlet 2.5

- HTTP 1.1
- n-Tier Enterprise Applications
- JavaServer Pages 2.1
- Java Servlet 2.5
- Project 3
Last Time

Typical J2EE Architecture

Client

Presentation

Business Logic

Data
Last Time

Wahoo!

You Write!

Client Tier

Middle Tier

Back-End Tier

Computer

Login servlet

Prefs servlet

View servlet

UserManager

NewsProvider

webserver

User DB

moreover.com

webserver

Login servlet

Prefs servlet

View servlet

UserManager

NewsProvider

User DB

moreover.com
This Time

- XQuery 1.0
- DTD
- Project 3
XQuery 1.0

History

- Recommendation as of 1/07.
  - “XML is a versatile markup language, capable of labeling the information content of diverse data sources including structured and semi-structured documents, relational databases, and object repositories.”
XQuery 1.0

XPath 2.0

- Sequences
- Data types
- Enhanced function set
- Multiple sources
XQuery 1.0

Path Expressions

- `doc("books.xml")`
- `doc("books.xml")/bib/book/title`
- `doc("books.xml")//title`
- `doc("books.xml")/bib/book[price<50]`

Adapted from http://www.w3schools.com/xquery/xquery_example.asp.
XQuery 1.0

FLWOR Expressions

FLWORExpr ::= 
   (ForClause | LetClause)+ WhereClause? OrderByClause? "return" ExprSingle
XQuery 1.0

FLWOR Expressions

for $x in doc("books.xml")/bib/book
where $x/price>50
order by $x/title
return $x/title

Adapted from http://www.w3schools.com/xquery/xquery_example.asp.
XQuery 1.0

FLWOR Expressions

<bib>
  <book>
    <title>TCP/IP Illustrated</title>
    <author>Stevens</author>
    <publisher>Addison-Wesley</publisher>
  </book>
  <book>
    <title>Advanced Unix Programming</title>
    <author>Stevens</author>
    <publisher>Addison-Wesley</publisher>
  </book>
  <book>
    <title>Data on the Web</title>
    <author>Abiteboul</author>
    <author>Buneman</author>
    <author>Suciu</author>
  </book>
</bib>

Excerpted from http://www.w3.org/TR/xquery/.
XQuery 1.0

FLWOR Expressions

<authlist>
{
  for $a in fn:distinct-values($books//@author)
  order by $a
  return
  <author>
    <name>
      { $a/text() }
    </name>
    <books>
      { for $b in $books//@book[author = $a]
        order by $b/title
        return $b/title
      }
    </books>
  </author>
}
</authlist>

Adapted from http://www.w3.org/TR/xquery/.
XQuery 1.0

FLWOR Expressions

<authlist>
  <author>
    <name>Abiteboul</name>
    <books>
      <title>Data on the Web</title>
    </books>
  </author>
  <author>
    <name>Buneman</name>
    <books>
      <title>Data on the Web</title>
    </books>
  </author>
  <author>
    <name>Stevens</name>
    <books>
      <title>TCP/IP Illustrated</title>
      <title>Advanced Unix Programming</title>
    </books>
  </author>
  <author>
    <name>Suciu</name>
    <books>
      <title>Data on the Web</title>
    </books>
  </author>
</authlist>
XQuery 1.0

Sequence Expressions

```xquery
for $d in doc("depts.xml")//deptno
let $e := doc("emps.xml")//emp[deptno = $d]
where count($e) >= 10
order by avg($e/salary) descending
return
    <big-dept>
        {
            $d,
            <headcount>{count($e)}</headcount>,
            <avgsal>{avg($e/salary)}</avgsal>
        }
    </big-dept>
```

Example excerpted from http://www.w3.org/TR/xquery/.
XQuery 1.0

Conditional Expressions

FOR $h$ IN doc("library.xml")//holding
RETURN
  <holding>
    { $h/title,
      IF ($h/@type = "Journal")
      THEN $h/editor
      ELSE $h/author
    }
  </holding>

Quantified Expressions

FOR $b$ IN doc("bib.xml")//book
WHERE SOME $p$ IN $b$//paragraph SATISFIES
    (contains($p,"sailing") AND
    contains($p,"windsurfing"))
RETURN $b/title

FOR $b$ IN doc("bib.xml")//book
WHERE EVERY $p$ IN $b$//paragraph SATISFIES
    contains($p,"sailing")
RETURN $b/title

XQuery 1.0

Data Types

- **String-related**
  - ENTITIES, ENTITY, ID, IDREF, IDREFS, language, Name, NCName, NMTOKEN, NMTOKENS, normalizedString, QName, string, token

- **Date-related**
  - date, dateTime, duration, gDay, gMonth, gMonthDay, gYear, gYearMonth, time

- **Number-related**
  - base64Binary, byte, decimal, double, float, hexBinary, int, integer, long, negativeInteger, nonPositiveInteger, positiveInteger, short, unsignedLong, unsignedInt, unsignedShort, unsignedByte

- **Err, unrelated**
  - anyURI, boolean, NOTATION, ...

- **User-Defined**
Expressions on Sequence Types

- Instance of
  
  `<a>{5}</a>` instance of `xs:integer`

- Typeswitch
  
  ```
  typeswitch($customer/billing-address)
  case $a as element(*, USAddress) return $a/state
  case $a as element(*, CanadaAddress) return $a/province
  case $a as element(*, JapanAddress) return $a/prefecture
  default return "unknown"
  ```

- Cast and Castable
  
  ```
  if ($x castable as hatsize) then $x cast as hatsize
  else if ($x castable as IQ) then $x cast as IQ
  else $x cast as xs:string
  ```

Examples excerpted from [http://www.w3.org/TR/xquery/](http://www.w3.org/TR/xquery/).
Well-Formedness

<moreovernews>
  [...]  
  <article id="840925179">
    <url>http://c.moreover.com/click/here.pl?x840925179</url>
    <headline_text>Whose Genome Is It, Anyway?</headline_text>
    <source>Discover</source>
    <media_type>text</media_type>
    <cluster>moreover...</cluster>
    <tagline></tagline>
    <harvest_time>Mar 11 2007 8:46AM</harvest_time>
    <access_registration></access_registration>
    <access_status></access_status>
  </article>
  [...]  
</moreovernews>
DTD

Validity

<!ELEMENT moreovernews (article*)>
<!ELEMENT article (url, headline_text, source, media_type, cluster, tagline, document_url, harvest_time, access_registration, access_status)>
<!ATTLIST article id ID #IMPLIED>
<!ELEMENT url (#PCDATA)>
<!ELEMENT headline_text (#PCDATA)>
<!ELEMENT source (#PCDATA)>
<!ELEMENT media_type (#PCDATA)>
<!ELEMENT cluster (#PCDATA)>
<!ELEMENT tagline (#PCDATA)>
<!ELEMENT document_url (#PCDATA)>
<!ELEMENT harvest_time (#PCDATA)>
<!ELEMENT access_registration (#PCDATA)>
<!ELEMENT access_status (#PCDATA)>

Available at http://www.fas.harvard.edu/~cscie259/distribution/projects/project3-7.0/ROOT/dtd/moreovernews.dtd.
DTD

XHTML 1.0 Transitional

```xml
<?xml version="1.0" encoding="iso-8859-1"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>
    <title/>
    <title/>
  </head>
  <body/>
</html>
```

Available at http://www.fas.harvard.edu/~cscie259/distribution/lectures/8/examples8/xhtml.html.
**Overview**

- A DTD is a definition of an XML document's schema
  - Codifies what the structure of a document must be
  - The relationships between the components of the document
  - What data is allowed where
- The DTD language was released as part of the official XML specification
- XML Schema is a more modern, powerful way to accomplish the same goals
- However, DTDs are still widely in use, and are supported as the primary method of validating XML
# DTD

## Motivation

- DTDs, or schemas in general, are contracts for what make a certain type of XML document.
- DTDs allow you to check whether a document "instance" is "valid" with respect to its schema (in contrast with its simply being well-formed).
- DTDs provide a place to specify what belongs in elements, attributes, and what individual elements represent, etc.
- Particularly useful in B2B transactions where agreeing on a data format is important.
- DTDs encapsulate good document design so you can benefit from it:
  - Why reinvent a document standard when there is DocBook?
  - Why reinvent a financial exchange standard when there is OFX?
    [http://www.ofx.net/ofx/specview/SpecView.html](http://www.ofx.net/ofx/specview/SpecView.html)
  - Why reinvent a voice standard when there is VoiceXML?
    [http://www.w3.org/TR/voicexml20/vxml.dtd](http://www.w3.org/TR/voicexml20/vxml.dtd)
To DTD or not to DTD

- It depends on the application
- DTDs (or schemas in general) are crucial when a common understanding of data is important
  - XML makes data interchange easier from a technical standpoint, but it still doesn't eliminate human misunderstandings
  - I say `<price>`, you say `<cost>`
- Writing a DTD can help you design a good data model
  - All the principles of proper data modeling apply to XML as well
- However, DTDs constrain XML flexibility
  - As soon as you have a DTD, your data model is less extensible
  - At least, changes require distribution of a new DTD
A Song Element

<SONG>
  <TITLE>Everyday</TITLE>
  <COMPOSER>Dave</COMPOSER>
  <COMPOSER>Boyd Tinsley</COMPOSER>
  <PRODUCER>Dave Matthews</PRODUCER>
  <PUBLISHER>BMG</PUBLISHER>
  <LENGTH>12:20</LENGTH>
  <YEAR>2001</YEAR>
  <ARTIST>Dave Matthews Band</ARTIST>
</SONG>

Excerpted from http://www.fas.harvard.edu/~cscie259/distribution/lectures/8/examples8/song{1,2}.xml.
A DTD for **SONG** Elements

```xml
<!ELEMENT SONG (TITLE, COMPOSER+, PRODUCER*, PUBLISHER*, LENGTH?, YEAR?, ARTIST+)>  
<!ELEMENT TITLE (#PCDATA)> 
<!ELEMENT COMPOSER (#PCDATA)> 
<!ELEMENT PRODUCER (#PCDATA)> 
<!ELEMENT PUBLISHER (#PCDATA)> 
<!ELEMENT LENGTH (#PCDATA)> 
<!ELEMENT YEAR (#PCDATA)> 
<!ELEMENT ARTIST (#PCDATA)> 
```

Available at http://www.fas.harvard.edu/~cscie259/distribution/lectures/8/examples8/song.dtd.
DTD

The `<!ELEMENT>` Declaration

`<!ELEMENT  element_name  (content_model)>`
DTD

The `<!ELEMENT> Declaration`

- Gives the name and content model of an element
- The name must be unique
- The content model specifies what the valid child content can be
  - `#PCDATA` `<!ELEMENT TITLE (#PCDATA)>`
  - `EMPTY` `<!ELEMENT course EMPTY>`
  - Elements
  - Mixed
  - `ANY` `<!ELEMENT comment ANY>`
DTD

Element Content

- The most sophisticated of content types
- Allows you to specify a regular expression for the allowed child elements
  - `<!ELEMENT SONG (TITLE, COMPOSER+, PRODUCER*, PUBLISHER*, LENGTH?, YEAR?, ARTIST+)>
  - `<!ELEMENT spec (front, body, back?)>
  - `<!ELEMENT div1 (head, (p | list | note)*, div2*)>
DTD

Building Blocks of Regular Expressions

- foo?
  - The foo element must occur 0 times or exactly 1 time.
- foo*
  - The foo element may occur 0 or more times.
- foo+
  - The foo element must occur 1 or more times.
- (foo|bar|baz)
  - Either foo or bar or baz must appear exactly 1 time.
- (foo,bar,baz)
  - 1 instance of foo must occur, followed by 1 instance of bar, followed by 1 instance of baz.
Mixed Content

- When both character and element content can be interspersed, the names of the elements can be constrained, but not their order or number; and #PCDATA must be declared first!

  - `<!ELEMENT p (#PCDATA|a|u|b|i|em)*>`
  - `<p>I am <b>bold</b> and <i>italic</i>.

  - `<!ELEMENT PO (#PCDATA|item|shipdate|qty)*>`
  - `<PO><qty>1</qty> <item>Flowbee</item> was shipped to you on <shipdate>29 March 2003</shipdate>.</PO>`

The Flowbee Precision Home Haircut System is available for purchase at http://www.flowbee.com/.
The `<!ATTLIST>` Declaration

`<!ATTLIST element_name
    attribute_name    attribute_type    default_declaration
    attribute_name    attribute_type    default_declaration
    ...
>

...
<!ATTLIST> **Examples**

- <!ATTLIST termdef
  id ID #REQUIRED
  type CDATA #REQUIRED
  name CDATA #IMPLIED>

- <!ATTLIST list
  type (bullets|ordered|glossary) "ordered">

- <!ATTLIST form
  method CDATA #FIXED "post">

- <!ATTLIST paper
  language CDATA "English"
**DTD**

**Attribute Types**

- **CDATA**
  - Character data, including entities.
- **ID**
  - Must be unique within document (and must start with a letter).}
- **IDREF**
  - Must refer to an ID in document.
- **IDREFS**
  - References one or more IDs, separated by spaces.
- **ENTITY**
  - Must refer to an entity.
- **ENTITIES**
  - References one or more entities, separated by spaces.
- **NMTOKEN**
  - Name token devoid of whitespace.
- **NMTOKENS**
  - Series of one or more NMTOKENS, separated by spaces.
Default Declarations

- **#FIXED**
  - Attribute's value is fixed and must be that specified in DTD.

- **#REQUIRED**
  - The element is required to have the attribute, and the attribute is required to have a value.

- **#IMPLIED**
  - Attribute is optional.
DTD

Where do DTDs go?

- DTDs can be
  - placed in a standalone file known as an "external subset"
  - part of the `<!DOCTYPE>` declaration in the XML document as an "internal subset" (which overrides any declarations in an external subset)

- Examples
  - `<!DOCTYPE SONG [
      <!ELEMENT SONG (TITLE, COMPOSER+, PRODUCER*, 
      PUBLISHER*, LENGTH?, YEAR?, ARTIST+)>
    ]>
  - `<!DOCTYPE SONG SYSTEM "song.dtd">
  - `<!DOCTYPE SONG SYSTEM "song.dtd" [ 
      <!ELEMENT ARTIST (FIRST, LAST)>
      <!ELEMENT FIRST (#PCDATA)>
      <!ELEMENT LAST (#PCDATA)>
    ]>`
DTD

Validation

javax.xml.parsers.SAXParserFactory
org.xml.sax.ErrorHandler
DTD

Whitespace

<foo>
   <bar/>
   <baz/>
</foo>
DTD

Similar XML Constructs

- Entities
  - `<!ENTITY nbsp "&#160;">`
  - `<!ENTITY copyright "Copyright (c) David Malan. All rights reserved."/>

- Notations (http://msxml.com/intro_xml/notation_decl.html)
  - `<!NOTATION GIF SYSTEM "GIF Notation">`
  - `<!ENTITY watAGE_Logo SYSTEM "watage.gif" NDATA GIF>"`
DTD

**Shortcomings**

- Not well-formed XML (though still derived from SGML)
- No built-in data types (*e.g.*, bool, int, float, string, *etc.*).
- No support for custom data types (*e.g.*, phone numbers)
  - No pattern-matching
  - No inheritance
- No support for ranges (*e.g.*, "year must be an integer between 0 and 99", "review can appear as a child of book no more than 10 times", *etc.*).
- Not namespace-aware
- Content models must be deterministic; cannot allow arbitrary ordering of children, as with:
  - `<!ELEMENT foo (((bar,baz,qux) | (bar,qux,baz) | (baz,bar,qux) | (baz,qux,baz) | (qux,baz,baz) | (qux,baz,baz))>`
- ...

Copyright © 2007, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.
Next Time

XML Schema (Second Edition)

- XML Schema (Second Edition)
Lecture 8: XQuery 1.0 and DTD

19 November 2007

David J. Malan
malan@post.harvard.edu