

CmpE 494: Service-Oriented Architectures and Web Services

Pinar Yolum
pinar.yolum@boun.edu.tr

Department of
Computer Engineering
Boğaziçi University

Chapter 3 XML

Based largely on
Service-Oriented Computing: Semantics, Processes, Agents
– Munindar P. Singh and Michael N. Huhns, Wiley, 2004
Examples from www.w3schools.com

Highlights of this Chapter

- XML and Vocabularies
- Well-Formedness
- Namespaces and Qualified Names
- XML Extensions
- XML Schema
- XML Query Languages
- XPath
- XSLT
- Limitations

Fall 2005— Pinar Yolum

3

Markup History

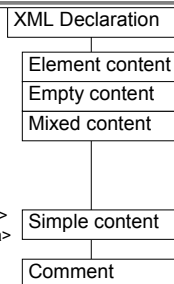
- None
- Ad hoc tags
- SGML (Standard Generalized Markup L): complex, few reliable tools
- HTML (HyperText ML): simple, unprincipled, mixes structure and display
- XML (eXtensible ML): simple, yet extensible subset of SGML to capture new vocabularies
 - Machine processible
 - Comprehensible to people: easier debugging

Fall 2005— Pinar Yolum

4

XML Basics

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book>
  <title>My First XML</title>
  <prod id="33-657" media="paper"></prod>
  <chapter>Introduction to XML
    <para>What is HTML</para>
    <para>What is XML</para>
  </chapter>
  <chapter>XML Syntax
    <para>Elements must have a closing tag</para>
    <para>Elements must be properly nested</para>
  </chapter>
  <!-- Example comment -->
</book>
```



Fall 2005— Pinar Yolum

5

Parsing

- An XML document maps to a parse tree.
 - Each tag has to end and end once
 - Contrast with HTML <p>
 - Nesting structure (one root)
 - Every other element under this root
 - Each attribute occurs at most once; quoted string
 - <note date="12/11/2002">
 - Tags are case sensitive
 - White space preserved
- Well-formed XML documents can be parsed

Fall 2005— Pinar Yolum

6

Viewing

- XML only has content
- Provide information on how to display the content
- For Cascading Style Sheets
 - `<?xml-stylesheet type="text/css" href="cd_catalog.css"?>`
- XSL (the eXtensible Stylesheet Language)
 - Written in XML
 - `<?xml-stylesheet type="text/xsl" href="simple.xsl"?>`

Fall 2005—Pinar Yolum

7

Namespaces

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

- URI: Uniform Resource Identifier
 - Identify resources of any kind; including resources that are not network-accessible
- URL: Uniform Resource Locator

Fall 2005—Pinar Yolum

8

Validating

- Applications have an explicit or implicit syntax for their particular XML-based tags
 - If explicit, may be expressed in DTDs and XML Schemas
 - Best referred to definitions elsewhere
 - XML Schemas, expressed in XML, are superior to DTDs
 - When docs are produced by external components, they should be validated

Fall 2005—Pinar Yolum

9

XML Schema

- A data definition language for XML: defines a notion of *schema validity*
 - Same syntax as regular XML documents
 - Local scoping of subelement names
 - Incorporates namespaces
 - Types
 - Primitive (built-in): string, integer, float, date, ...
 - Primitive (built-in): ID (key), IDREF (foreign key)
 - simpleType constructors: list, union
 - Restrictions: intervals, lengths, enumerations, regex patterns
 - Flexible ordering of elements
 - Key and referential integrity constraints

Fall 2005—Pinar Yolum

10

XML Example

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Fall 2005—Pinar Yolum

11

XML Schema Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.w3schools.com"
  xmlns="http://www.w3schools.com" elementFormDefault="qualified">
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string" default="pinar"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string" fixed="test"/>
        <xs:element name="body" type="xs:string"/>
        <xs:element name="signDate" type="xs:date"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Fall 2005—Pinar Yolum

Simple Type Restrictions

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="100"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="gender"> <xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:pattern value="male|female"/>
  </xs:restriction>
</xs:simpleType>
</xs:element>
```

Fall 2005—Pinar Yolum

13

XML Schema: complexType

- Four kinds
 - Empty elements
 - Contain only other elements
 - Contain only text
 - Contain both other elements and text
- Specifies types of elements with structure:
 - Must use a *compositor* if >1 subelements
 - Subelements with types
 - Min and max occurrences (default 1) of subelements
- Elements with text content not easy: ignore
- EMPTY elements: easy. Example?

Fall 2005—Pinar Yolum

14

XML Schema: Compositors

- *Sequence*: ordered
 - Can occur within other compositors
 - Allows varying min and max occurrence
- *All*: unordered
 - Must occur directly below root element
 - Max occurrence of each element is 1
- *Choice*: exclusive or
 - Can occur within other compositors

Fall 2005—Pinar Yolum

15

XML Schema: Key Namespaces

- <http://www.w3.org/2001/XMLSchema>
 - Conventional prefix: xsd
 - Terms for defining schemas: schema, element, attribute, ...
 - The tag schema has an attribute targetNamespace
- <http://www.w3.org/2001/XMLSchema-instance>
 - Conventional prefix: xsi
 - Terms for use in instances: schemaLocation, null
- targetNamespace: user-defined

Fall 2005—Pinar Yolum

16

XML Schema Instance Doc

```
<?xml version="1.0"?>
<note xmlns="http://www.w3schools.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3schools.com
  note.xsd">
```

Fall 2005—Pinar Yolum

17

Document Object Model (DOM)

- Basis for parsing XML, which provides a node-labeled tree in its API
 - Conceptually simple: traverse by requesting tag, its attribute values, and its children
 - Processing program reflects document structure
 - Can edit documents
 - Inefficient for large documents: parses them first entirely to build the tree even if a tiny part is needed

Fall 2005—Pinar Yolum

18

DOM Example [Simeoni 2003]

```
Element s = d.getDocumentElement();
NodeList l =
    s.getElementsByTagName("member");
Element m = (Element) l.item(0);
int code = m.getAttribute("code");
NodeList kids = m.getChildNodes();
Node kid = kids.item(0);
String tagName =
    ((Element)kid).getTagName();
...
```

Fall 2005—Pinar Yolum

19

Simple API for XML (SAX)

- Parser generates a sequence of events:
 - startElement, endElement, ...
- Programmer implements these as callbacks
 - More control for the programmer
- Processing program does not reflect document structure
- Read
 - <http://www.saxproject.org/quickstart.html>
 - <http://java.sun.com/j2ee/1.4/docs/tutorial/doc/>
 - JAXP: <http://java.sun.com/webservices/jaxp/index.jsp>

Fall 2005—Pinar Yolum

20

SAX Example [Saxproject.org]

```
class MySAXApp extends DefaultHandler {
public void startElement (String uri, String name, String
qName, Attributes atts) {
    if ("".equals (uri))
        System.out.println("Start element: " + qName);
    else System.out.println("Start element: {" + uri + "}" +
name);
}
public void endElement (String uri, String name, String
qName) {
    if ("".equals (uri))
        System.out.println("End element: " + qName);
    else System.out.println("End element: {" + uri + "}" +
name);
}
}
```

Fall 2005—Pinar Yolum

21

Uses of XML

- Exchanging information across software components
- Storing information in nonproprietary format
- XML documents represent structured descriptions:
 - Products, services, catalogs
 - Contracts
 - Queries, requests, invocations (as in SOAP)
- Data-centric versus document-centric (irregular, heterogeneous data, depend on entire doc for app-specific meaning) views

Fall 2005—Pinar Yolum

22

Data-Centric View

```
<relation>
<tuple><attr1>V11</attr1>...
<attrn>V1n</attrn></tuple>
...
<tuple><attr1>Vm1</attr1>...
<attrn>Vmn</attrn></tuple>
</relation>
```

- Extract and store into DB via mapping to DB model
- Regular, homogeneous tags
- May be expensive if repeatedly parsed and instantiated

Fall 2005—Pinar Yolum

23

Document-Centric View

- Storing docs in DBs
 - Use character large objects (clobs) within DB
 - Store paths to external files containing docs
 - Combine with some structured elements with search conditions for both structured elements and unstructured clobs or files
 - Heterogeneity also complicates mappings to traditional typed OO programming languages

Fall 2005—Pinar Yolum

24

Directions

- Limitations of XML
 - Doesn't represent meaning
 - Enables multiple representations for the same information; transform if models known
- Trends: sophisticated approaches for
 - Querying and manipulating XML, e.g., XSLT
 - Binding to PLs and DBs
 - Semantics, e.g., RDF, OWL, ...

Fall 2005—Pinar Yolum

25

XML Query Languages

- XSL: **eXtensible Stylesheet Language**
 - XPath: Defining parts of XML documents
 - XSLT: For transforming XML documents
 - XSL-FO: For formatting XML documents

Fall 2005—Pinar Yolum

26

XPath

- Address parts of XML documents
- Uses the logical structure of XML documents
- Model XML documents as trees with nodes
 - Element nodes
 - Attributes nodes
 - Text nodes (PCDATA)
 - Comments
 - Root node: above root of document
 - Namespace nodes
 - Processing instruction nodes
 - Comment nodes

Fall 2005—Pinar Yolum

27

Data Model

- Parent in XPath is like parent as traditionally in computer science
- XPath basics:
 - Element nodes, comment nodes, processing instruction nodes and text nodes are children
 - An attribute is not the child of its parent
 - Makes a difference for certain kinds of recursion (e.g., apply-templates discussed in XSLT)

Fall 2005—Pinar Yolum

28

XPath Paths

- Path expressions to select nodes in XML document

Expression	Description
element	All child nodes of the node
/	From the root node
//	Nodes in the document from the current node that matches the selection
.	The current node
..	The parent of the current node
@	Attributes

Fall 2005—Pinar Yolum

29

XPath Paths

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

Example: What happens if you select

- bookstore
- /bookstore
- bookstore/book
- //book
- bookstore//book
- //@lang

Fall 2005—Pinar Yolum

30

XPath Navigation

- Predicates embedded in square brackets
- Select children according to position, e.g., [j], where j could be 1 ... last()
- //title[@lang]: All titles with attribute lang
- /bookstore/book[price>35.00]: All the book elements of bookstore with price>35
- Wildcard, *:
 - * matches any element node
 - @*: finds all attribute values
 - Node() matches any node of any kind
- AND
 - //book/title | //book/price: Title AND price elements of all book elements

Fall 2005—Pinar Yolum

31

XPath Queries

- Incorporate selection conditions in XPath
 - Attributes: //Song[@genre="jazz"]
 - Elements: //Song[starts-with(//group, "Led")]
 - Existence of attribute: //Song[@genre]
 - Existence of subelement: //Song[group]
 - Boolean operators: and, not, or
 - Set operator: union (|); none others
 - Arithmetic operators: >, <, ...
 - String functions: contains(), concat(), length(),
 - Aggregates: sum(), count()
- Example:
 - http://www.w3schools.com/xpath/xpath_examples.asp
 - http://www.w3schools.com/xpath/tryit.asp?filename=try_xpath_select_cdnodes

Fall 2005—Pinar Yolum

32

XSLT

- A functional programming language
- Transform one XML into another format XML, HTML, XHTML
- Add (or remove) new elements
- Rearrange or sort elements
- XML Source Tree → XML Result tree
- Use XPath to define matching pattern

Fall 2005—Pinar Yolum

33

XSLT Stylesheets

- A stylesheet specifies transformations on a document

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"
  href="URL-to-dot-xsl"?> <!-- the sheet to use -->
<main-tag>
...
</main-tag>
```
- Use the XSLT namespace, conventionally abbreviated as xsl
- Includes primitives:
 - Copy-of
 - <for-each select="...">
 - <if test="...">
 - <choose >

Fall 2005—Pinar Yolum

XSLT Templates: 1

- A pattern to specify where a given transform should apply
 - This match only works on the root:
<xsl:template match="/">
 - ...
 - </xsl:template>
- Value-of Element
 - Select the value and add it to the transformation

Fall 2005—Pinar Yolum

35

XSLT Templates: 2

- <xsl:for-each>
 - Apply to every node of a specified element
 - <xsl:for-each select="catalog/cd">
- Filtering output
 - Add extra constraint
 - <xsl:for-each select="catalog/cd[artist='Bob Dylan']">
 - = (equal)
 - != (not equal)
 - < (less than)
 - > (greater than)

Fall 2005—Pinar Yolum

36

XSLT Templates: 3

- Sort the selected nodes
 - `<xsl:sort select="artist"/>`
- If test: Apply based on condition

```
<xsl:if test="price > 10">
<tr>
  <td> <xsl:value-of select="title"/> </td>
  <td><xsl:value-of select="artist"/></td>
</tr>
</xsl:if>
```

Fall 2005—Pinar Yolum

37

XSLT Templates: 4

- Choose (if-then-else)
 - Used with `<xsl:when>` and `<xsl:otherwise>`
- ```
<xsl:choose>
 <xsl:when test="price > 10">
 <td bgcolor="#ff00ff">
 <xsl:value-of select="artist"/></td>
 </xsl:when>
 <xsl:otherwise>
 <td><xsl:value-of select="artist"/></td>
 </xsl:otherwise>
</xsl:choose>
```

Fall 2005—Pinar Yolum

38

## XSLT Templates: 5

- Can be applied recursively on the et-children via
  - `<xsl:apply-templates/>`
- By default, if no other template matches, recursively apply to et-children of current node (ignores attributed) and to root:
  - `<xsl:template match="*/"/>`
  - `<xsl:apply-templates/>`
  - `</xsl:template>`
- Can over-apply; to override the default, may need an empty template:
  - `<xsl:template match="..."/>` <!-- e.g., match all text() -->

Fall 2005—Pinar Yolum

39

## Example 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>
```

Fall 2005—Pinar Yolum

40

## Example XSL

```
<?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>
```

Fall 2005—Pinar Yolum

## Tools

- Many open and commercial tools
  - XMLSPY: XML and XML Schema edit & validate; XSL edit & transform
  - Xml.apache.org: Tools for integration with Java
  - Stylusstudio.com: XSLT editor
  - Eclipse: XML plug-ins

Fall 2005—Pinar Yolum

42