

CmpE 473

Internet Programming

Pınar Yolum
pyolum@cmpe.boun.edu.tr

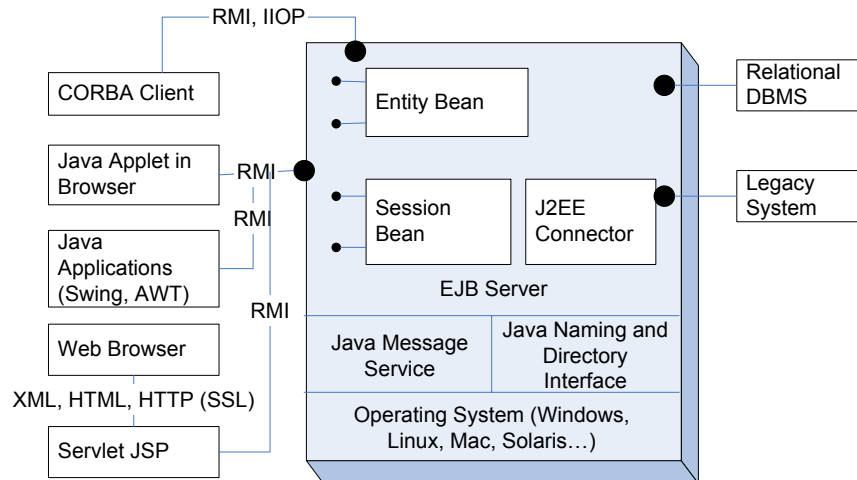
Department of
Computer Engineering
Boğaziçi University

Chapter 8

XML

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

J2EE Technology



Spring 2005— Pinar Yolum

3

Highlights of this Chapter

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

Spring 2005— Pinar Yolum

4

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

Spring 2005— Pinar Yolum

5

XML Basics

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

Spring 2005— Pinar Yolum

6

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

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"?>

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

Spring 2005— Pinar Yolum

9

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

Spring 2005— Pinar Yolum

10

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

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>
```

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>
```

Spring 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>
```

Spring 2005— Pinar Yolum

14

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?

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

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

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">
```

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

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();
...
```

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

SAX Example [Simeoni 2003]

```
class MemberProcess extends DefaultHandler {
public void startElement (String uri, String n, String
    qName, Attributes attrs) {
    if (n.equals("member")) code =
        attrs.getValue("code");
    if (n.equals("project")) inProject = true;
    buffer.reset(); }
public void endElement (String uri, String n, String
    qName) {
    if (n.equals("project")) inProject = false;
    if (n.equals("member") && !inProject)
        name = buffer.toString().trim(); } }
```

Programming with XML

- Current approaches concentrate on structure but ignore meaning
 - Difficult to construct and maintain
 - Treat everything as a string
 - Inadequate type checking can hide errors
- Emerging approaches (e.g., JAXB) provide superior binding from XML to programming languages
 - Primitives such as unmarshal to materialize an object from XML

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

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

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

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, ...