IT350 Web and Internet Programming
Fall 2007
SlideSet #16: XML and Semantic Web

(XML from Chapter 20 of text)
Outline

• Why Structured Data?
• Types of Structured Data
  – XML and Friends
  – RDF and Semantic Web
Structured Data

• Overall Goal – add more “structure” to our otherwise freeform data
• Why?

• HTML/XHTML primarily designed to be rendered for ____________ viewing. We want something that can be exploited by ____________.
<?xml version = "1.0"?>

<!-- Fig. 20.1: article.xml -->
<!-- Article structured with XML -->

<article>
  <title>Simple XML</title>
  <date>July 15, 2003</date>
  <author>
    <firstName>Carpenter</firstName>
    <lastName>Cal</lastName>
  </author>
  <summary>XML is pretty easy.</summary>
  <content>Once you have mastered XHTML, XML is easily learned. You must remember that XML is not for displaying information but for managing information.</content>
</article>
<?xml version="1.0" ?>
<!-- Fig. 20.1: article.xml -->
<!-- Article structured with XML -->
<article>
<title>Simple XML</title>
<date>July 15, 2003</date>
<author>
<summary>XML is pretty easy.</summary>
<content>Once you have mastered XHTML, XML is easily learned. You must remember that XML is not for displaying information but for managing information.</content>
</article>
Another example

```xml
<?xml version = "1.0"?>
<!DOCTYPE letter SYSTEM "letter.dtd">
<letter>
  <contact type = "from">
    <name>John Doe</name>
    <address1>123 Main St.</address1>
    <address2></address2>
    <city>Anytown</city>
    <state>Anystate</state>
    <zip>12345</zip>
    <phone>555-1234</phone>
    <flag gender = "M"/>
  </contact>

  <contact type = "to">
    <name>Joe Schmoe</name>
    <address1>Box 12345</address1>
    <address2>15 Any Ave.</address2>
    <city>Othertown</city>
  </contact>
</letter>
```
<!-- Fig. 20.6: letter.dtd -->
<!-- DTD document for letter.xml -->

<!ELEMENT letter (contact+, salutation, paragraph+, closing, signature)>

<!ELEMENT contact (name, address1, address2, city, state, zip, phone, flag)>
<!ATTLIST contact type CDATA #IMPLIED>

<!ELEMENT name (#PCDATA)>
<!ELEMENT address1 (#PCDATA)>
<!ELEMENT address2 (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT flag EMPTY>
<!ATTLIST flag gender (M | F) "M">

<!ELEMENT salutation (#PCDATA)>
<!ELEMENT closing (#PCDATA)>
<!ELEMENT paragraph (#PCDATA)>
<!ELEMENT signature (#PCDATA)>
XML Validator

Enter a url to load:

[C:\W3HTP3\examples\ch20\letter.xml]

or paste in some XML:

[PASTE]

check the "Validation" box if you want to validate it.

[Validation]

click the "Validate" button to see results.

[Validate]

Your XML is well formed and is validated

- PI: xml
- COMMENT: Fig. 20.3: letter.xml
- COMMENT: Business letter formatted with XML
- SCHEMA: letter
  - ELEMENT: letter
    - ELEMENT: contact
    - ELEMENT: contact
    - ELEMENT: salutation
    - ELEMENT: paragraph
      - TEXT: It is our privilege to inform you about our new database managed with XML. This new system allows you to reduce the load of your inventory list server by having the client machine perform the work of sorting and filtering the data.
    - ELEMENT: closing
    - ELEMENT: signature
Additional XML Technology

- **XML Schema** – like DTD, but in XML and more sophisticated
- **XPath** – way of identifying a particular element inside a document
- **XSL** – render XML for viewing in browser
- **XSLT** – transform one XML doc to another
What’s Missing With XML?

• **XML** is centered around *documents*
  – DTDs/Schemas say what is legal
  – Software can use documents with DTDs it “knows about”

• **RDF** is centered around *statements*
  – Statement is (subject, predicate, object)
  – Example: ( <it350>, location, “MI200” )
  – Example: ( <it350>, type, course )
  – XML-based syntax:
    
    ```
    <course about= http://www.cs.usna.edu/~it350/F05 >
    <location>Sieg 134</location>
    </course>
    ```
  – RDF =
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns="http://www.cs.washington.edu/semweb/rdf-equiv-ns#">
  <rdf:Description>
    <course>
      <rdf:Description>
        <name>Networking Seminar</name>
        <time>1 p.m.</time>
        <location>Sieg 134</location>
        <instructor>
          <rdf:Description>
            <value>Prof. John Fitz</value>
            <office hours>Tue 3-4 p.m.</office hours>
          </rdf:Description>
        </instructor>
        <instructor>
          <rdf:Description>
            <value>Prof. Helen Randolph</value>
            <office hours>Fri 9-10 a.m.</office hours>
          </rdf:Description>
        </instructor>
        ...
      </rdf:Description>
    </course>
  </rdf:Description>
</rdf:RDF>
RDF → Semantic Web?

• XML forces nesting of tags in certain order
  – Vocabulary and structure fixed by the DTD
• With RDF, can place statements anywhere, in any order
  – And can mix vocabulary (types, predicates) from many different documents
• RDF Schema lets us define different schemas for different domains
  – University domain, sailing domain, animal domain, …
  – Software that understands <sailing:knots> can utilize this info in any document that it finds it
  – Crawlers can collect knowledge → RDF databases
• Semantic == meaning
  – “Semantic Web” – collection of data for which software can “know” meaning of many of the terms – and exploit this
  – Aggregate data from many different sites (prices, ratings)
  – Search for info where answer not found on any one page
  – Personal agents to reason about such info to buy/schedule/search for you
Example Semantic Services – Calendar

May 23 - 29, 2004

Current Filter Selection: All Non-Course Events

<table>
<thead>
<tr>
<th>Time</th>
<th>Mon May 24</th>
<th>Tue May 25</th>
<th>Wed May 26</th>
<th>Thu May 27</th>
<th>Fri May 28</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:30</td>
<td></td>
<td>&gt; General Examination: Alex Yates</td>
<td>&gt; Research Seminar (590 NL)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:30</td>
<td></td>
<td>&gt; Research Seminar (590 ET)</td>
<td>&gt; Research Seminar</td>
<td>12:00pm&gt; Lunch with the Chair</td>
<td></td>
</tr>
<tr>
<td>1:30</td>
<td>&gt; Research Seminar (590 ET)</td>
<td>&gt; Research Seminar (590 ET)</td>
<td>&gt; Research Seminar (590 NL)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:30</td>
<td>3:00pm&gt; Research Seminar (590 NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:30</td>
<td>&gt; Research Seminar (590 DM)</td>
<td>&gt; Research Seminar (590 DM)</td>
<td>&gt; Grad Sit</td>
<td>&gt; Final Examination: Yung-Yu Chuang</td>
<td></td>
</tr>
<tr>
<td>4:30</td>
<td>&gt; Matt Mohabbi, Robocup</td>
<td>&gt; Research Seminar (590 SY)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Example Semantic Services – “Who’s Who”

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Portrait</th>
<th>Office/Work Phone</th>
<th>Year Of Study</th>
<th>Advisor</th>
<th>Degree Completed</th>
<th>Research Interests</th>
<th>Hobbies</th>
<th>Birthday</th>
<th>Other Interests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Julie Letchner</td>
<td>Graduate Student</td>
<td></td>
<td></td>
<td>First year</td>
<td>Dieter Fox</td>
<td>B.S. in EECS</td>
<td>context-aware, ubiquitous computing systems</td>
<td>Dancing (ballet, modern, jazz), jogging, reading</td>
<td>June 17</td>
<td></td>
</tr>
<tr>
<td>Alan L. Liu</td>
<td>Graduate Student</td>
<td></td>
<td></td>
<td>1st year grad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>May 7</td>
<td></td>
</tr>
</tbody>
</table>
Example Semantic Services – “Flink”

(see http://flink.semanticweb.org/index.jsp)
Conclusion

• HTML/XHTML has humans in mind
• Both XML and RDF add ________________ to data, making more amenable to ______________ interpretation
• Any one application could always just use XML with a fixed Schema
  – But RDF allows a much more flexible approach
  – Hopefully enabling data to be re-used in unexpected ways
  – “Semantic Web” is all about making smart apps to exploit this data
• Key Challenge – who will make the RDF?
  – Idea 1 – make it rewarding for normal people
  – Idea 2 – smart software to “scrape” it off the web