XML Overview

Michelle Lee & Jeff Offutt

http://www.ise.gmu.edu/~offutt/

SWE 432
Design and Implementation of Software for the Web

XML

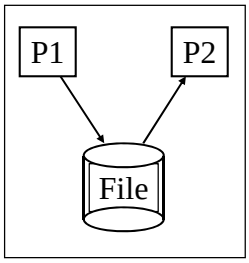
- eXtensible Markup Language
- Markup languages insert "tags" into text files to describe presentation or other information
- SGML: Standard Generalized Markup Language
  - HTML : Visual presentation
  - Latex : Document formatting
  - XML : Data description
- W3C standard: http://www.w3.org/TR/REC-xml/

Why XML?

- Passing data from one software component to another has always been difficult
- The two components must agree on format, types, and organization
- Web software applications have unique requirements for data passing:
  - Very loose coupling
  - Dynamic integration

Passing Data – 1978

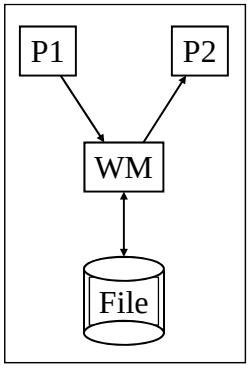
- Program P2 needs to use data produced by program P1
  - Data saved to a file as records (COBOL, Fortran, …)
  - The file format often not documented
  - Source for P1 may not be available
  - Data saved in binary mode – not readable by hand



- Format of file deduced from executing P1 and from trial and error
- MSU Computing Services, 1979: Two weeks of trial and error executions of P1 to understand format of file

Passing Data – 1985

- Program P2 needs to use data produced by program P1
  - Data saved to a file as records (C, Ada, …)
  - The file format often not documented
  - Data saved as plain text
- Both P1 and P2 access the file through a "wrapper module"
- Module needs to repeatedly updated
- Module written by development team
- Data hard to validate
- Mothra, 1985: ~12 data files shared among 15 to 20 separate programs



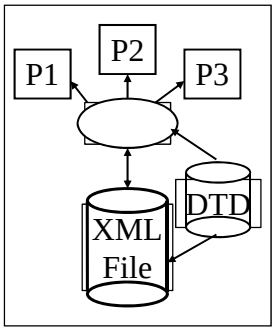
Wrapper Method Problems

- Slow – everything is a file in plain text
- Sharing – Developers of P1 and P2 must agree to share source of WM
- Maintenance – Who has “control” of WM?

- Solution – data sharing that is:
  - Independent of type
  - Self documenting
  - Easy to understand format
  - Especially important for web applications – XML

Passing Data – 21st Century

- Data is passed directly between components
- XML allows for self-documenting data



- P1, P2 and P3 can see the format, contents, and structure of the data
- Free parsers are available to put XML messages into a standard format
- Information about type and format is readily available

Web Services

- A ***Web Service*** is a program that offers software services over the Internet to other programs
  - Internet-based
  - Uses SOAP and XML
  - Peer-to-peer communication
- Web service components can integrate dynamically
  - finding other services during execution
- Web services transmit data that are formatted in XML

How does XML work?

Much simpler than you might think …

XML Is User Defined ("Extensible")

- Programmers can create their own tags
- Tags have been designed for mathematics, formal specifications, resumes, recipes, addresses, …
- Pizza Markup Language (PML):

```
<pizza>
  <topping extracheese="yes">Pepperoni</topping>
  <price>13.00</price>
  <size>large</size>
</pizza>
```

XML Structure

- Containment : Tags can be contained in other tags
- Tag names should be meaningful
- All tags must have an end tag (unlike HTML)

XML Can Easily Be Validated

- Two ways to describe an XML language
  - Document Type Definitions (DTD) : The grammar to define an XML language
  - Schemas : Grammar plus types and facets
- Documents can be checked against the grammar
- Grammar can specify that certain fields are required
- Allows programs to assume the data is formatted correctly, reducing the amount of checking the program must do

XML Structure

- A “well-formed” document is properly tagged, that is, it follows the grammar syntax in the DTD
- Tags must be properly nested
- Indentation is for readability and does not affect the XML
- Empty tags have a shorthand notation:
  <X></X> == <X/>

Parsing XML with Java

- A number of XML parsers exist
- They can be downloaded for free
- They read an XML file and put the contents in a tree whose elements can be accessed

XML vs. HTML

- Unlike HTML, XML tags tell you what the data means, rather than how to display it
- XML elements must be strictly nested, XML can represent data in any level of complexity
- Both XML and HTML allow empty tags; in XML an empty tag must be followed by a forward slash: <emptyTag/>

XML vs. HTML

- XML attribute values must be surrounded by single or double quotes
- HTML does not require quotes for single values
- XML tags are case sensitive
- HTML tags are not

XML Must Be Well-Formed

- A well-formed XML document must be syntactically correct
- All angle brackets are part of tags
  - Use *entity references* < and >
- All tags have an ending tag or are themselves self-ending
  - (<slide> .. </slide> or <slide/>)
- All tags must be fully nested, so this arrangement would produce an error:
  - <slide><image> .. </slide></image>
- A well-formed document might not be valid according to the grammar

XML Example

```
<message>
    <to>you@yourAddress.com</to>
    <from>me@myAddress.com</from>
    <subject>XML Is Really Cool</subject>
    <text>
      How many ways is XML cool? Let me count the
  ways ...
    </text>
 </message>
```

XML Syntax

- XML has strict rules that allows programs to process documents easily because there is less room for ambiguity
- The rules of a language are called the language's syntax
- XML consists of four parts
  1. XML Document
  2. XML grammar (DTD or Schema)
  3. XML Parser
  4. XML Application

Discussed in the next few slides (1–2)

Not discussed in 432 (3–4)

1) XML Document

Consists of prolog, content and markup

- Prolog defines document characteristics
- Content is the character data
- XML uses the following seven types of markups:
  1. Element
  2. Attributes
  3. Comments
  4. Entity references
  5. Processing instructions
  6. Character data sections (CDATA)
  7. Document type declarations (DTD's)

1) XML Document – Prolog

<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>

- <? .. ?> Prolog declaration
- *version*

  Identifies the version of the XML markup language used in the data. This attribute is required.
- *encoding*

  Identifies the character set used to encode the data. "ISO-8859-1" is "Latin-1" the Western European and English language character set. (The default is compressed Unicode: UTF-8.)
- *standalone*

  Tells whether or not this document references an external entity or an external data type specification. If there are no external references, "yes" is appropriate.

1) XML Document – Element

```
<products>  ──────────────────────→ products element
                                       (root element)
    <product> ────────────────────→ product element
        <name>Monitor</name>
        <price>$200</price>
    </product>
    … …
</products>
```

- An XML element can enclose other elements
- All XML documents are built from a single root element
- Empty tag. For example
</BR>

1) XML Document – Attribute

```
<products>
    <product>
        <name>Monitor</name>
        <price currency="US">200</price>
    </product>
    … …
    <!– Here is a comment -->
</products>
```

Attribute value

- In XML, attribute values must be in either single or double quotes
- The "—" string must appear only at the beginning and at the end of the comment

2a) DTD

- A DTD (Document Type Definition) provides a grammar that tells which data structures can occur, in what sequences
- The specification tells you how to write the high-level code that processes the data elements
- DTDs define a set of rules that defines the structure of an XML document

2a) DTD (2)

- Document *declarations* tell parsers which DTD to use when processing a specific XML document

```
<!DOCTYPE authorDoc SYSTEM “author_name.dtd”>
<author_name>Michelle Lee</author_name>
```

- DTD parts include :
  - Element type declarations
  - Attribute list declarations
  - Entity declarations
  - Notations declarations

2a) DTD – Element

- Element type declarations define the structure of classes of elements

- They tell the parser each element’s name, type of information in the element, and associated attribute

```
<!ELEMENT element_name (content_model)>
<!ELEMENT author_name (#PCDATA)>
```

2a) DTD – Element (2)

Variations of content model

- Element with no contents
  <!ELEMENT emptyElement EMPTY>
  <emptyElement/> or <emptyElement></emptyElement>
- Element with one content element
  <!ELEMENT author_name (#PCDATA)>
- Element with many content model descriptors
  <!ELEMENT author_name (first_name, last_name, middle_name?)>
- Element with ANY content model descriptor
  <!ELEMENT author_name ANY>
- Element with one of a list of enumerated values
  <!ELEMENT book_type (mystery | scienceFiction | romance)>

2a) DTD – Element (3)

```
<!DOCTYPE book_entry [
<!ELEMENT book_type (mystery | scienceFiction | romance)>
<!ELEMENT author_name (#PCDATA)>
<!ELEMENT title  (#PCDATA)>
<!ELEMENT lostBook EMPTY>]>
<book_entry>
    <author_name>Arthur C. Clarke</author_name>
    <title>Childhoo's End</title>
    <book_type>scienceFiction</book_type>
   <lostBook/>
</book_entry>
```

2a) DTD – Attribute

- Attribute list declarations

  <!ATTLIST element_name attr1_name attr1_type attr1_defaults attr2_name attr2_type attr2_defaults>
- Acceptable attribute types
  - Strings, Enumerated, ENTITY, ENTITIES, ID, IDREF, IDREFS, NMTOKEN, NMTOKENS, NOTATION
- An attribute default can have one of four possible values:
  1. Required
  2. Implied
  3. Fixed-value
  4. Supplied

2b) Schemas – XML for Book Example

```
<books>
  <book>
    <ISBN>0471043281</ISBN>
    <title>The Art of Software Testing</title>
    <author>Glen Myers</author>
    <publisher>Wiley</publisher>
    <price>50.00</price>
    <year>1979</year>
  </book>
</books>
```

- XML messages are defined by <u>grammars</u>
  - <u>Schemas</u> and DTDs
- Schemas can define many kinds of <u>types</u>
- Schemas include "<u>facets</u>," which refine the grammar

2b) XML Schemas

```
<xs:element name = "books">
   <xs:complexType>
   <xs:sequence>
      <xs:element name = "book" maxOccurs = "unbounded">
         <xs:complexType>
         <xs:sequence>
            <xs:element name = "ISBN" type = "xs:string"/>
            <xs:element name = "price"  type = "priceType"/>
            <xs:element name = "year"   type = "xs:int"/>
         </xs:sequence>
         </xs:complexType>
      </xs:element>
   </xs:sequence>
   </xs:complexType>
</xs:element>
```

facets

```
<xs:simpleType name = "priceType">
   <xs:restriction base = "xs:decimal">
      <xs:fractionDigits value = "2" />
      <xs:maxInclusive value = "1000.00" />
   </xs:restriction>
</xs:simpleType>
```


XML Example

```
<? xml version = "1.0"?>
<products>
    <product>
        <name>Monitor</name>
        <price currency="US">200</price>
    </product>
    <product>
        <name>Hard Drive</name>
        <price currency="US"> 150</price>
    </product>
    <product>
        <name>Keyboard</name>
        <price currency="US"> 50</price>
    </product>
</products>
```