XML Basics

lesson #xml01

James L. Parry
B.C. Institute of Technology

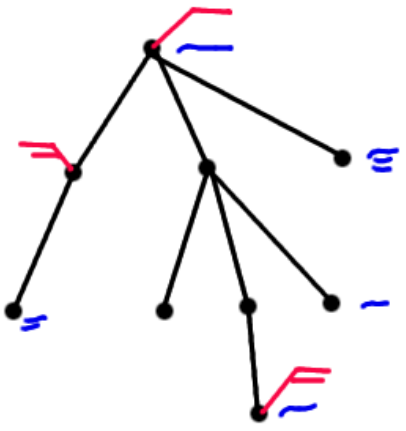
XML BASICS (Refresher)

X M L stands for eXtensible Markup Language

Used for storing object or data structure state.

Remember COMP1536!

Visualizing XML: elements (dots) may have children (below), may have text values (blue), and may have set of attributes (red)



Agenda

1. What is XML?
2. XML Structure
3. Entities
4. Attributes
5. Design Strategies

WHAT IS XML?

- Way of digitally representing data structures
- Adds tags to text data so that it can be processed by any application and is human-readable
- Application can understand data's meaning and how to process it
- Can be used to extend HTML, store object state, or even to define new "languages"

XML Documents

- Documents are based on a logical tree structure
  - Documents can be recursively broken down into elements
  - Elements can have attributes
- Documents of the same "type" have the same structure
- Physically, a document can be broken up using entities - think separation into files

XML Markup

- Represents the logical structure
- Connects/contains the physical entities
- An XML document is made up of markup and character data
- Markup processed by XML parser
- Character data passed on to application
- Markup is found between < and > just like HTML
- Reserved characters: & and ; pair can also be significant

An Example XML Document

```
<shoes>
  <item uid="S121">
    <manufacturer>Nike</manufacturer>
    <model>hightop</model>
    <designer>Steve Nash</designer>
    <price>125.00</price>
    <in-stock>176</in-stock>
  </item>
</shoes>
```

XML Design Goals

- Simple
- Human readable
- Platform neutral

A More Complex Example

```
<?xml version="1.0"?>
<!DOCTYPE MEMO SYSTEM "memo.dtd">
<memo>
<from>
  <name>Paul Prescod</name>
  <email>papresco@prescod.com</email>
</from>
<to>
  <name>Charles GoldFarb</name>
  <email>charles@sgmlsource.com</email>
</to>
<subject>Another Memo Example</subject>
<body>
<paragraph> Charles, I wanted to suggest that
we
<emphasis>not</emphasis> use the typical memo
example in our book. Memos tend to be used
anywhere a small, simple document type is
needed,
but they are just <emphasis>so</emphasis>
boring!
</paragraph>
</body>
</memo>
```

XML STRUCTURE

There are three ways to look at the structure of an XML document:

- Tree structure, conceptually
- Text document
- DOM

These all have to do with different ways of thinking about the pieces of an XML document:

- directives (document type & entities)
- elements (with attributes)

Logical Structure

- Tree structured hierarchy
- Every document has a single outermost element, called the "root" or "document" element
- Every element can contain from 0 to many nested elements, defined using particular XML tags
- Can be navigated in a predictable way

Physical Structure

- Character string
- Stored in 1 or more files
- XML document can be broken down physically into pieces of text or sub strings

Document Contents

- XML signature (often forgiven)
- DOCTYPE directive (optional)
  - Entity directives (optional)
- Nested elements (at least one)
  - Attributes for elements (optional)
  - Embedded entity references (optional)

Document Object Model

- Tree structure, where each node ...
- Has a text value (or can be empty)
- May have attributes
- May have child nodes

There is a Javascript API for this.

The W3C prescribes language-neutral API for interfaces

ENTITIES

- An abbreviation or short form for some text
- Allow a document to be broken up into multiple storage objects or files (external)
- Allow substitution within a file (internal)
- An entity reference substitutes the entity text for the abbreviation

Entity Declaration

- Directives define them
- Abbreviation is entity name
- Long form is entity content
  `<!ENTITY dtd "document type definition">`
- Can build symbolic constants using entity references
  `<!ENTITY inverted-exclamation "&#161;">`

parsed Entities

- If an entity contains XML that should be parsed by the XML processor, it is called a parsed entity
- Simple example of parsed entity with markup:
  `<!ENTITY dtd "<term> document type definition</term>">`
- This is also an example of an internal entity

External Entities

- Content of entity can come from another file...
  `<!ENTITY intro-chapter SYSTEM "http://www.megacorp.com/intro.xml">`
- Keyword "SYSTEM" lets processor know that the next thing in the declaration is a URI
- Content of the entity comes from URI

Entity Reference

- A document will use an entity through an entity reference
- & and ; combination is used, around the name of the entity to substitute
- The reference will be replaced by the entity contents

```
<!DOCTYPE MAGAZINE[
<!ENTITY title "Painters Quarterly">
]>
<MAGAZINE>
<TITLE>&title;</TITLE>
<p>Welcome to the first issue of &title;.
&title; is targeted at the amateur
painter.</p>
</MAGAZINE>
```

Unparsed External Entities

- Used for data such as images
- Application does not expect the XML processor to parse this data
- NDATA indicates that this is an unparsed entity
- GIF Indicates type of data

```
<!ENTITY picture SYSTEM
  "http://www.home.org/mydog.gif"
  NDATA GIF>
```

ATTRIBUTES

- An attribute is a name followed by an equal sign then an attribute value
- Allows an author to attach extra information to the elements in a document
- An attribute value can be any characters except the ones that start markup
- Used to store information about a tag
- Used for values that are unique to element as a whole and unlikely to change (see example)

Attributes vs Elements

- Attributes cannot contain elements or sub-attributes
- Specified once and in any order
- Elements must occur in order specified and may be specified more than once
- Attributes are properties, elements are parts

Structure and Use

- Text strings with no explicit structure
- Or simple lists of strings
- Different elements can have attributes with the same name; you should have same semantics
- Attributes are considered immutable ... second or subsequent attribute settings are ignored

Attribute Types

- Attribute types enforce lexical and semantic constraints
- Simplest type is the `StringType` which is made up of character data, `CDATA`
- Any string of characters
- Tokenized types:
  - Name token or tokens
  - ID and IDREF - used for cross referencing
  - Entity - can refer to external unparsed entities like graphics file

Attribute Example

```
<customer cust_id="12345">
  <name salute="Mr" nickname="Bobby">Smith, Robert</name>
  <title>Manager</title>
  <company>My Company Inc.</company>
</customer>
```

XML STRATEGIES

- Goals to shoot for:
  - Logical or intuitive
  - Constrainable
  - Easy to process
- Bad things?
  Nothing is right for all uses
- Strategies
  - Element centric
  - Attribute centric
  - Combination

Congratulations!

You have completed lesson #xml01: XML Basics

If you would take a minute to provide some feedback, we would appreciate it!

The next activity in sequence is: xml02 Constraining XML

You can use your browser's back button to return to the page you were on before starting this activity, or you can jump directly to the course homepage, organizer, or reference page.