Lecture 28: XML

Kenneth M. Anderson
Software Methods and Tools
CSCI 3308 - Fall Semester, 2004

Today's Lecture

- Introduce XML
  - background
  - concepts
  - examples

Quick Introduction

- XML stands for
  - eXstensible Markup Language
- It is a language for creating new languages
- In particular, it is designed to create "tagged" languages similar to HTML
- It is considered "extensible" because it allows the developer to create new tags
  - as compared to HTML where the set of tags has been fixed and new tags are ignored by browsers

Background

- XML was developed to address concerns about HTML
  - In particular, HTML mixes document structure and document presentation in one language
  - This makes it difficult to change a document's presentation while keeping its structure the same
- Originally, HTML was meant to address the same concern; it was just supposed to specify document structure, not presentation
  - but the browser wars quickly changed that!
  - In particular, users cared about the presentation of their information, and quickly demanded presentation features
    - <font>, , <margin>, etc.

An additional problem

- An additional problem can be seen by viewing the HTML source of the the CNN website
  - This page is filled with “headlines” and text/images that support those headlines
  - A “major” headline looks like this
    - <H3><A href="..." class="t1">Earliest certified election results in Florida: 6 p.m. EST</A></H3>
  - A “minor”headline looks like this
    -   • Bush sues 4 counties over absentee ballots
  - Is the difference intuitive? :-)
  - Disclaimer: the above code is taken from a few years back

The problem explained

- The problem is that
  - presentation concerns (i.e. making the web page look good)
  - are overriding structural concerns (i.e. this information is a headline)
- The fact that one paragraph is a headline and another is supporting text is completely lost in the HTML
- If you wanted to write a program to search this web page and list all headlines, you would need to code knowledge of CNN’s presentation rules to figure out where the headlines are hiding
  - To make matters worse, if CNN changes its presentation, you would have to change your program!

The XML approach

- Imagine if the source for CNN’s webpage looked like this
  - <story>
    - <headline class=“important”>Election returns due at 6 PM EST.</headline>
    - <supportingText>Blah Blah Blah…</supportingText>
  - </story>
- Here, structure is preserved
  - It would be very easy to write a program to grab the headlines out of this document
- So, how do we handle presentation?
  - XSLT, which is covered in the next lecture

XML definitions

- An XML document consists of the following parts
  - a Document Type Definition (or DTD)
  - Data
- The DTD defines the structure of the data. A parser can read the DTD and know how to parse the data that follows it
  - As such, XML documents are said to be “self-describing”: all the information for parsing the data is contained in the document itself

Note

- This lecture is presenting a simplified view of the XML standard
  - In particular, the standard supports a number of ways of associating a DTD with an XML document
  - We will cover only one of these mechanisms in this lecture, known as the internal DTD
  - For more information, buy a book on XML, visit <http://www.xml.com/>, or read the XML standard at:
    - <http://www.w3.org/TR/2000/REC-xml-20001006>
    - Note: the spec is not for the "faint of heart". I would recommend starting with an XML book

XML Syntax Rules

- XML imposes a number of syntax rules that make it easier to parse than HTML
  - All tags must be closed, e.g.
    - HTML lets you skip the closing p tag, XML does not.
    - Note: the closing tag must match the opening tag!
    -
 - In HTML, you can have single tags like
 to introduce a horizontal break in the document. The
 tag has no content associated with it; XML requires tags with no content to explicitly end with a trailing slash, hence
.

XML Syntax Rules, continued

- Additional syntax rules
  - All attribute values must be quoted
    - e.g. HTML allows the following
      blah blah blah
    - XML requires the following
      blah blah blah
  - There are many others
    - concerning legal characters, comments, etc. See the spec for details.

Well-Formed XML Documents

- XML documents are considered **well-formed** if they conform to the XML Syntax rules
- Well-formed documents can be parsed by any XML Parser without the need for a DTD
  - It can use the syntax rules to parse the document cleanly, but without the DTD it does not know if the document is **valid**

Valid XML Documents

- An XML document is considered “valid” if
  - (1) it is well-formed and
  - (2) it conforms to the rules specified in its associated DTD
    - That is, if the DTD says that a <p> tag can only contain <b> tags and plain text, then a <p> tag which contains an tag would be considered invalid

Parts of an XML document

- XML declaration
- Document declaration
  - We will be showing a document declaration with an embedded DTD
    - This is only one type of XML document declaration, see the note on slide 9
- Data

XML Declaration

- An XML document begins with this tag
  - <?xml version=“1.0”?>
- The question mark denotes a “processing instruction”
- This instruction is for an XML parser
  - Its provides the parser with additional information about the XML document
- An XML document can contain additional processing instructions
  - The parser will pass these instructions to the client that asked the parser to parse the document

XML Declaration, continued

- Additional attributes
  - encoding
    - <?xml version=“1.0” encoding=“UTF-8”>
    - XML documents can be stored in a variety of character encodings; see the spec for all of the legal values that can be supplied for “encoding”
  - standalone
    - <?xml version=“1.0” standalone=“yes”>
    - yes means document is self-contained
    - no means the DTD is stored externally

Document Declaration

- The document declaration comes after the XML Declaration
- Its tag name is DOCTYPE
  - There are two forms
  - internal
    - <!DOCTYPE greeting [...DTD Goes Here…]>
  - external
    - <!DOCTYPE greeting SYSTEM “greeting.dtd”]>
  - We will cover the first form

DTD Syntax

- The DTD is where you declare the elements (a.k.a. tags) and attributes that will appear in your XML document
- In defining elements, you use **regular expressions** to declare the order in which elements are to appear
- Attributes can be associated with elements and can have default values associated with them
- Lets look at an example

DTD for a Class Gradebook

```
<!DOCTYPE gradebook [
    <!ELEMENT gradebook (class, student*)>
    <!ELEMENT class (name, studentsEnrolled)>
    <!ATTLIST class semester CDATA #REQUIRED>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT studentsEnrolled (#PCDATA)>
    <!ELEMENT student (name, grade*)>
    <!ELEMENT grade (#PCDATA)>
    <!ATTLIST grade name CDATA #REQUIRED>
]>
```


What does this mean?

- This DTD defines a document whose root element is called “gradebook”
- The first element in gradebook has to be a “class” element followed by zero or more student elements
- A Class element contains a name and the number of student’s enrolled
  - It has a required attribute called semester
- A student contains a name and zero or more grades
- A name, a grade, and the studentsEnrolled are declared as having PCDATA or “Parsed Character Data” as their content => this means that they contain strings
  - The grade element also has an attribute called name

An example

```
<?xml version="1.0" ?>
<!DOCTYPE gradebook […insert DTD from slide 19 here]>
<gradebook>
    <class semester="Fall 2004">
        <name>CSCI 3308</name>
        <studentsEnrolled>36</studentsEnrolled>
    </class>
    <student>
        <name>Ken Anderson</name>
        <grade name="lab0">10</grade>
        <grade name="lab1">9</grade>
    </student>
</gradebook>
```


Element Declarations

- Empty Elements
  - <!ELEMENT BR EMPTY>
- Non-Empty Elements
  - <!ELEMENT NAME (CONTENT)>
  - Content contains a regular expression of element names and/or Character Data
  - #PCDATA - strings are parsed for embedded elements (like searching for a <b> tag within a <p> tag in HTML)
  - #CDATA - strings are not parsed for embedded elements

Regular Expressions in Element Declarations

- Element1, Element2
  - Element2 must follow Element1
- Element1?
  - Element1 is optional
- Element1+
  - At least one Element1 tag must appear
- Element1*
  - Zero or more Element1 tags may appear
- Element1 | Element2
  - Either Element1 or Element2 may appear

Examples

- <!ELEMENT p (#PCDATA|B|I|EM|…)>
  - A p tag may contain text, or a B element, or an I element, or …
- <!ELEMENT name (first, middle?, last)
  - A name consists of a first and last name and may contain a middle name
- <!ELEMENT shoppinglist (item+)
  - A shopping list contains one or more items

Attribute Declarations

- Declaring attributes requires that you first declare the associated element
  - You then use the ATTLIST element to declare the attributes
  - <!ELEMENT name (first, middle?, last)>
  - <!ATTLIST name
    - age CDATA #REQUIRED
    - height CDATA #IMPLIED
    - gender (male|female) “female”>
  - This example declares three attributes, one required and two implied (optional), if no gender attribute is specified, it defaults to “female”
  - See the spec. for complete details on ATTLIST tag

Summary

- XML provides the ability to create your own tagged language
- The DTD defines the elements and attributes of the document
- An XML document is “self-describing” because the DTD is embedded directly in the document

Software Engineering Benefits

- XML attacks an accidental difficulty of software engineering
  - Having to define your own file formats
  - Having to write parsers for these formats
- With XML, you can define file formats in a standard way, and any XML parser can be used to parse the file
  - You never have to write a parser again!
  - I threw out hundreds of lines of code from my hypermedia system when I converted my preference files to XML!

More Information

- General XML Information
  - <http://www.xml.com/>
- Free XML Parsers
  - <http://xml.apache.org/>
    - Java and C++ parsers (with bindings for Perl and COM)
  - <http://www.alphaworks.ibm.com/>
    - IBM’s Java and C++ parsers for XML