B336 Advanced Internet Computing

The XML Document

Learning Objectives

- Understand the role of the XML document format in XML technologies.

- Know the format and syntax of an XML document, as specified by W3C's XML 1.0 Recommendations.

Learning Objectives

- In the scheme of what we are doing in this unit:
    - We are studying XML as an important set of Internet technologies to use as solutions in different areas (for example, the problem defined in Assignment 2)
    - The XML document format is the first and most basic step in understanding how all the different XML technologies work.

Lecture Outline

- How important is the XML 1.0 document format specifications?
- Definition of a well-formed XML document
- Components of a well-formed XML document

The XML Document Specification

- In last week lecture, we discussed two key factors to the success of XML as a set of technologies:

    1. Standardization
    2. Ease of creating new languages

- These two factors are founded on how good the specifications of the basic XML document format is.

Reference

- The current base XML 1.0 Recommendations: http://www.w3.org/TR/2000/REC-xml-20001006
    - The material in the Unit Reader covers the details of the specifications quite extensively.

The XML Document Specification

- The W3C's XML 1.0 Recommendations specifies two sets of constraints for an XML document:

    1. A **well-formed** document: The XML document conforms to the basic syntax rules.
    2. A **valid** document: Having a DTD to to specify the allowed components (eg. tags, structure) in the XML document.

- In this lecture, we will concentrate on what makes a document well-formed.
    - We will deal with valid documents in the next lecture.

The Importance of XML Document Specification

- Having a well-defined (and perhaps valid) documents will make it:
    - Easy for document publishers and information content creators to create new documents.
    - Easy for software to parse and process the documents.
    - Easy for people to read and understand the documents.

- These are the core design goals of XML (see last week's lecture).

A Well-Formed Document

- The formal definition given in the XML 1.0 specification:

[Definition: A textual object is a **well-formed** XML document if:]

1. Taken as a whole, it matches the production labeled *document*.
2. It meets all the well-formedness constraints given in the XML specification.
3. Each of the parsed entities which is referenced directly or indirectly within the document is well-formed.

The Well-Formedness of the Document and the Parsed Entities

- The 2nd and 3rd condition in the previous overhead basically says that:
    - The document structure must follow what the specifications say, and
    - All sections which are to be parsed must follow what the specification says.
        - » We will deal with non-parsed sections of the document reaching the end of this lecture

The "*Document*" Production

- The first constraint basically says that the document must follow the production rules specified under sub-section titled "*Document*" in the specifications.

- That sub-section says that a document consist of 3 parts:
  - Prolog (optional)
  - A root element
  - Miscellaneous parts (optional)

Miscellaneous Parts

- The miscellaneous parts of an XML document (spread throughout the document) can consist of:
  - Comments. Eg.
    ```
    <!-- Start of the main tag -->
    ```
  - Processing Instructions. Eg.
    ```
    <?xml stylesheet type="text/css" href="mycss.css"?>
    ```
  - White Spaces

The "*Document*" Production

- Matching the "*document*" production implies that:
  - It contains one or more elements.
  - There is exactly one **root** element, no part of which appears in the content of any other element.
  - The elements are all properly nest within each other.

The Prolog

- The prolog exists at the beginning of an XML document. It can consist of:
    - An XML declaration, followed by
    - Miscellaneous parts (as described before), followed by
    - A Document Type Declaration (DTD) - more on this in the next lecture.
- Although formally, the prolog **can** be empty, the W3C (and the specifications itself) recommends that no documents leave out the declaration part.
    - You will be hard pressed to find XML documents on the net which do not have an XML declaration.

The XML Declaration

- Some example declarations:

```
<?xml version="1.0”?>

<?xml version="1.0” standalone=“yes”?>

<?xml version="1.0” standalone=“yes”
 encoding=“UTF-8”?>
```

- The declaration must have at least the “xml” keyword and the “version” attribute.

A summary so far...

- So a well-formed XML document consists of (in order):
    - (Optional) Prolog with
        - » a <?xml …?> declaration
        - » Miscellaenous parts
        - » Document Type Definition
    - A required root element
        - » where all other elements exists within
    - (Optional) Miscellaneous parts

Example

Prolog

Root Element (and nested sub-elements)

```
<?xml version="1.0">
<?xml stylesheet type="text/css" href="mycss.css"?>
<!DOCTYPE poetry [
     <!ELEMENT anthology (poem*)>
     <!ELEMENT poem (title?, author?, stanza+)>
     <!ELEMENT title (#PCDATA) >
     <!ELEMENT author (#PCDATA) >
     <!ELEMENT stanza (line+) >
     <!ELEMENT line (#PCDATA) >
     ]>

<poetry>
  <anthology>
     <poem>
          <title>The SICK ROSE</title>
          <author>William Blake</author>
          <stanza>
               <line>O Rose thou art sick.</line>
               <line>The invisible worm,</line>
               <line>That flies in the night</line>
               <line>In the howling storm,</line>
          </stanza>
          <stanza>
               <line>Has found out thy bed</line>
               <line>Of crimson joy:</line>
               <line>And his dark secret love</line>
               <line>Does thy life destroy.</line>
          </stanza>
     </poem>
  </anthology>
</poetry>
```

Declaration

Processing Instructions (misc parts)

DTD (see next lecture)

Elements in XML

- The content of an XML document (the information the author wants to convey) are broken up into units called *elements*.

- Different types elements are given different names, and tagged with a **start-tag** and **end-tag** with that name.
  - Eg. In the element `<StudentID>12345678</StudentID>`, the information “12345678” of type “StudentID” is marked-up with the tag `<StudentID>`.

Elements in XML

- **All** basic information text must be tagged in an XML file.

- All elements must exist **properly** nested within the root element.

Empty Elements

- Some elements do not have closing tags. These are called “empty” elements.
    - You have seen the concept of empty elements before in HTML, from tags like <hr>, <br>, <img>, etc.
- In XML, tags of empty elements must end with a “/”.
    - Eg, in XHTML, the new XML compliant version of HTML, we have <hr /> and <br /> tags.

Attributes of Elements

- Elements may also contain attributes.

- The attribute **names** and **values** are defined in the start-tag. Eg.

```
<Student ID="12345678" status="enrolled"
        workrate="dead lazy" />
```

- All attribute values must be enclosed in quotes - unlike HTML.

Pre-defined Entity References

- Since some characters like “<“ and “>” are special in XML, you cannot use them in places like attribute values. Eg.
  - `<number attribute=“>5”> 4 </number>` - Wrong!
  - `<number attribute=“>5”> 4 </number>` - Correct!

- There are 5 such pre-defined entity references:

&lt;	The < character
&gt;	The > character
&amp;	The & character
&apos;	The ‘ character
&quot;	The “ character

CDATA Sections

- Since XML is very sensitive to those characters like “<“ and “&”, you can define sections in an XML document which are **not** suppose to be parsed.

- Eg.

```
<![CDATA
        I’m free to use any of my
        own <&*@!]% characters !!!
]>
```

A Summary

- Things to watch out for when constructing a well-formed XML document:
  - Have an XML declaration at the beginning
  - Include at least the root document
  - Include both start and end tags for non-empty elements
  - Use “/” for empty elements tags
  - The root element must contain all other elements
  - Nest all elements properly - no overlaps.
  - Use unique attribute names
  - Use quotes for attribute values
  - Use the pre-defined entity references instead of the original characters.

In the next lecture...

- What makes a **valid** document.
- How to define a new mark-up language using DTD.