Lecture 16
Introduction to XML

Boriana Koleva
Room: C54
Email: bnk@cs.nott.ac.uk

Overview

- Introduction
- The Syntax of XML
- XML Document Structure
- Document Type Definitions

Introduction

- SGML is a meta-markup language
  - Developed in the early 1980s; ISO standard in 1986
- HTML was developed using SGML in the early 1990s - specifically for Web documents
- Two problems with HTML:
  1. Fixed set of tags and attributes
     - User cannot define new tags or attributes
     - So, the given tags must fit every kind of document, and the tags cannot connote any particular meaning
  2. There are no restrictions on arrangement or order of tag appearance in a document

Introduction

- One solution to the first of these problems:
  - Let each group of users define their own tags (with implied meanings)
  - (i.e., design their own “HTML”s using SGML)
- Problem with using SGML:
  - It’s too large and complex to use, and it is very difficult to build a parser for it
- A better solution: Define a lite version of SGML

XML

- XML is not a replacement for HTML
  - HTML is a markup language used to describe the layout of any kind of information
  - XML is a meta-markup language that can be used to define markup languages that can define the meaning of specific kinds of information
- XML is a very simple and universal way of storing and transferring data of any kind
- XML does not predefine any tags
- XML has no hidden specifications
- All docs described with an XML-derived markup language can be parsed with a single parser

XML

- We will refer to an XML-based markup language as a tag set
  - Strictly speaking, a tag set is an XML application, but that terminology can be confusing
- An XML processor is a program that parses XML documents and provides the parts to an application
- Documents that use an XML-based markup language are XML documents

The Syntax of XML

- The syntax of XML is in two distinct levels:

1. The general low-level rules that apply to all XML documents
2. For a particular XML tag set, either a document type definition (DTD) or an XML schema

General XML Syntax

- XML documents consist of:

1. Data elements
2. Markup declarations
   - instructions for the XML parser
3. Processing instructions
   - for the application program that is processing the data in the document

- All XML documents begin with an XML declaration:

```
<?xml version = "1.0" encoding = "utf-8"?>
```

- XML comments are just like HTML comments

General XML Syntax

- XML names:
  - Must begin with a letter or an underscore
  - They can include digits, hyphens, and periods
  - There is no length limitation
  - They are case sensitive (unlike HTML names)
- Syntax rules for XML: same as those of XHTML
  - Every XML document defines a single root element, whose opening tag must appear as the first line of the document
- An XML document that follows all of these rules is well formed

Simple XML example

```
<?xml version = "1.0">
  <ad>
    <year> 1960 </year>
    <make> Cessna </make>
    <model> Centurian </model>
    <color> Yellow with white trim </color>
    <location>
      <city> Gulfport </city>
      <state> Mississippi </state>
    </location>
  </ad>
```

XML Attributes

- XML document design – add a new attribute to an element or a nested element?
  - In XML, you often define a new nested tag to provide more info about the content of a tag
  - Nested tags are better than attributes, because attributes cannot describe structure and the structural complexity may grow
  - However, attributes should always be used to identify numbers or names of elements (like HTML `id` and `name` attributes)

Attribute Example

```
<!-- A tag with one attribute -->
<patient name = "Maggie Dee Magpie">
  ...
</patient>

<!-- A tag with one nested tag -->
<patient>
  <name> Maggie Dee Magpie </name>
  ...
</patient>

<!-- A tag with one nested tag, which contains three nested tags -->
<patient>
  <name>
    <first> Maggie </first>
    <middle> Dee </middle>
    <last> Magpie </last>
  </name>
  ...
</patient>
```

XML Document Structure

- An XML document often uses two auxiliary files:
  - One to specify the structural syntactic rule
  - One to provide a style specification
- An XML document has a single root element, but often consists of one or more entities
- An XML document has one document entity
  - All other entities are referenced in the document entity

XML Document Structure

- Reasons for entity structure:

1. Makes large documents easier to manage
2. Repeated entities need not be literally repeated
3. Binary entities can only be referenced in the document entities (XML is all text!)

XML Entities

- When the XML parser encounters a reference to a non-binary entity, the entity is merged in
- Entity names:
  - No length limitation
  - Must begin with a letter, a dash, or a colon
  - Can include letters, digits, periods, dashes, underscores, or colons
- A reference to an entity has the form:

```
&entity_name;
```

XML Entities

- One common use of entities is for special characters that may be used for markup delimiters
- These are predefined (as in HTML):

<	&lt;
>	&gt;
&	&amp;
"	&quot;
'	&apos;

- The user-defined entities can be defined only in DTDs

Document Type Definitions (DTDs)

- A DTD is a set of structural rules called declarations
  - These rules specify a set of elements, along with how and where they can appear in a document
- Purpose: provide a standard form for a collection of XML documents and define a markup language for them
- Not all XML documents have or need a DTD
- The DTD for a document can be internal or external

Document Type Definitions (DTDs)

- All of the declarations of a DTD are enclosed in the block of a `DOCTYPE` markup declaration
- DTD declarations have the form:

```
<!keyword … >
```

- There are four possible declaration keywords:
  - `ELEMENT` – to define tags
  - `ATTLIST` – to define tag attributes
  - `ENTITY` – to define entities
  - `NOTATION` – to define data type notations

Declaring Elements

- An element declaration specifies the name of an element, and the element's structure
- If the element is a leaf node of the document tree, its structure is in terms of characters
- If it is an internal node, its structure is a list of children elements (either leaf or internal nodes)
- General form:

```
<!ELEMENT element_name (list of child names)>
```

- E.g.:

```
<!ELEMENT memo (from, to, date, re, body)>
```

Declaring Elements

- Child elements can have modifiers

Modifier	Meaning
+	One or more occurrences
*	Zero or more occurrences
?	Zero or one occurrence

```
<!ELEMENT person (parent+, age, spouse?, sibling*)>
```

- Choices
  - `<!ELEMENT animal (cat | dog)>`
  - `animal` element contains **either** a `cat` child **or** a `dog` child

Declaring Elements

- Parentheses
  - either a choice or a sequence can be enclosed in parentheses to describe a content model

```
<!ELEMENT circle (centre,
(radius |diameter))>
```

- Leaf nodes specify data types, most often `PCDATA` (parsable character data)
  - Data type could also be `EMPTY` (no content) and `ANY` (can have any content)
  - Example of a leaf declaration:

```
<!ELEMENT name (#PCDATA)>
```

Declaring Attributes

- General form:

```
<!ATTLIST el_name at_name at_type [default]>
```

- There are ten different attribute types
  - `CDATA` – any string of characters
  - `ENUMERATION` – list of all possible values for the attribute separated by vertical bars

```
<!ATTLIST date month (January | February | March
| April | May | June| July | August |
September | October | November | December)
#REQUIRED>
```

Declaring Attributes

- Default values:

Value	Meaning
A value	The value, which is used if none is specified in an element.
`#FIXED` value	The value, which every element will have and which cannot be changed.
`#REQUIRED`	No default value is given; every instance of the element must specify a value.
`#IMPLIED`	No default value is given; the value may or may not be specified in an element.

Declaring Attributes

- Attribute specifications in a DTD:

```
<!ATTLIST car doors CDATA "4">
<!ATTLIST car engine_type CDATA #REQUIRED>
<!ATTLIST car price CDATA #IMPLIED>
<!ATTLIST car make CDATA #FIXED "Ford">
```

- An XML element that is valid for above DTD

```
<car doors = "2" engine_type = "V8">
   ...
</car>
```

Declaring Entities

- A general entity can be referenced anywhere in the content of an XML document
- General form of declaration:

```
<!ENTITY [%] entity_name "entity_value">
```

e.g. `<!ENTITY jfk "John Fitzgerald Kennedy">`

A reference: `&jfk;`

- If the entity value is longer than a line, define it in a separate file (an external text entity)

```
<!ENTITY entity_name SYSTEM "file_location">
```

A Sample DTD

- Example DTD
  - http://www.crg.cs.nott.ac.uk/~bnk/Teaching/WPS/planes.dtd
- An XML document valid for planes.dtd
  - http://www.crg.cs.nott.ac.uk/~bnk/Teaching/WPS/planes.xml

DTDs

- XML Parsers
  - Always check for well formedness
  - Some check for validity, relative to a given DTD
    - Called validating XML parsers
- You can download a validating XML parser from: http://xml.apache.org/xerces-j/index.html
- Internal DTDs

```
<!DOCTYPE root_name [
  …
]>
```

- External DTDs

```
<!DOCTYPE XML_doc_root_name SYSTEM
"DTD_file_name">
```

Summary

- Introduction
- The Syntax of XML
- XML Document Structure
- Document Type Definitions
  - Declaring elements
  - Declaring attributes
  - Declaring entities