XML Document Types and Validation

IIM-I340

April, 2010
Learning Objectives

Understand:

- The need for validation
- Two ways to specify validity:
  - Document Type Definitions (DTDs)
  - XML Schemas
What Is Validation?
And Why Do We Need It?

Examples:
- Bank or purchase order data
- XHTML
- COLLADA
DTDs: Document Type Definitions

- Defines allowable structure of the document type
  - What elements and attributes can it have?
  - How are they to be organized?
Example

bank-flat.dtd
Referencing a Doctype from an XML Document

```xml
<!DOCTYPE foo SYSTEM "foo.dtd">

or

<!DOCTYPE foo SYSTEM "http://foo.org/foo.dtd">

or

<!DOCTYPE foo
  PUBLIC "public-identifier"
  "backup-system-identifier">
```
Defining a Document Type

DTD Structure

```
DTD ::= (element-declaration
    | attribute-list
    | entity-declaration)*
```

- Must contain a declaration for the root element and all subelements
- * means 0 or more
Defining a Document Type

Element Declarations

element-declaration ::=  
<!ELEMENT element-name (element-type*)>

- Sub-elements must occur in the order listed
- element-type is a regular expression built out of element names
- Element names are XML names: start with a letter, may contain letters, digits, hyphens, and maybe periods and underscores
- Special element names:
  - #PCDATA ("parsed character data")
  - EMPTY (no content)
Regular Expressions

- Regular expressions are patterns:
  - | means “or”
  - ()’s group
  - ? means 0 or 1
  - * means 0 or more
  - + means 1 or more.

- Compare database cardinality constraints
- Unix grep utility
Defining a Document Type

Attributes

attribute-list ::= 
   <!ATTLIST element-name attribute-decl*> 

attribute-decl ::= 
   attribute-name attribute-type attribute-default 

attribute-name ::= XML name
Defining a Document Type
Attributes, Continued

attribute-type ::= 
  CDATA | ID | IDREF | IDREFS | enumeration

- CDATA means character data
- ID is an identifier, unique in the document
- IDREF is a reference to another element’s ID
- IDREFS is a list of IDREF separated by spaces
- “enumeration” means an enumerated list of values, e.g.,
  (freshman | sophomore | junior | senior)
attribute-default ::=  
  default-value | #REQUIRED | #IMPLIED

- #REQUIRED means value must be explicitly specified
- #IMPLIED means attribute is not required, but there is no default value
Defining a Document Type

Entities

```
entity-declaration ::= 
   <!ENTITY name "value">
| <!ENTITY name SYSTEM "filename-or-url">
```

- *Entities* are named constants (macros) which can get inserted (called) by the notation `&name;`
- Entities can also be declared in the `!DOCTYPE` declaration in the XML file
Further Examples

- bank-nested.dtd
- bank-attributes.dtd
Limitations of DTD’s

DTDs derive from text processing and are not oriented towards non-textual data.

1. Lack of type constraints on text elements and attributes; e.g., you cannot require that something be a positive integer.

2. Hard to specify unordered sets of subelements.
   - You can manage with something like (a|b|c)* or (a|b|c)+
   - But then it’s not possible to specify that there can be only one a, b, or c

3. ID and IDREF(S) lack typing
   - It isn’t possible to say which kind of element an IDREF(S) refers to
Alternatives to DTDs

- XML Schema
- RELAX NG
XML Schemas

- An alternative to document type definitions

- Advantages:
  - Can constrain types of values used for an element or attribute
  - Can constraint number of occurrences of a subelement
  - Can be less restrictive than DTDs by treating subelements as a set, allowing the order to vary

- Disadvantage:
  - XML, more verbose
Examples

- An XML document referencing an XML Schema: 01c-bank-flat-schema.xml
- The schema itself: bank-flat.xsd