Document Modeling

Prof. B.D. Chaudhary

August 2012

Table of contents

Document Modeling

- Define a Markup Language for documents
- Define a grammar for documents
- It is also called as XML Application Modeling
- Markups may appear in the documents
- It describes the restrictions which each document instance has to honor

Document Type Definitions

- Document Type Denition is written in formal syntax
- Describes elements, attributes, entities, and contents which may appear in the documents
- Validating Parser compares documents to their DTDs
- Validation is an optional step

Document Type Definitions (Contd...)

- The DTD does not say:
  - What is the document's root element?
  - How many instances of each element?
  - What character data are inside elements look like?
  - What is the meaning of an element?

A Simple DTD Example

```
<!ELEMENT person (name, profession*) >
<!ELEMENT name (first_name, last_name) >
<!ELEMENT first_name (#PCDATA) >
<!ELEMENT last_name (#PCDATA) >
<!ELEMENT profession (#PCDATA) >
```

A Simple DTD Example (Contd...)

- The DTD describes person element
- The person element has two children elements/sub-elements
- Sub-elements are: name and profession
- A person may have “zero or more” profession
- The name has two sub-elements

Document Type Declaration

- A valid document includes a reference to its DTD
- The DTD declaration is included in prolog
- <!DOCTYPE person SYSTEM "http://www.mnnit.ac.in/xml/dtds/person.dtd" >
- The DTD is generally stored in separate file
- Optionally, it may have extension .dtd

Document Type Declaration (Contd...)

- Root element is person
- The DTD can be found at the URL
- Relative URL may be used if DTD is on same site
- File name may be used if the document and the DTD are in the same directory
- DTD may be stored at several URLs

Some Remarks about the Example DTD

- Every element declaration on separate line for readability purpose
- They may be on the same line
- Each person element must contain exactly one name child element, followed by "zero or more" profession elements
- name must come before profession

Example Invalid Document

```
<person>
    <profession>Computer programmer</profession>
    <profession>Mathematician</profession>
</person>
<profession>Computer programmer</profession>
<name>
    <first_name>Alan</first_name>
    <last_name>Turing</last_name>
</name>
```

An Alternative DTD

```
<!ELEMENT first_name (#PCDATA) >
<!ELEMENT last_name (#PCDATA) >
<!ELEMENT profession (#PCDATA) >
<!ELEMENT name(first_name, last_name) >
<!ELEMENT person(name, profession*) >
```

Internal DTD Subset

- A document and its DTD may be in the same file
- It is convenient to modify and check
- The internal DTD subset is contained between [and]
- An example internal DTD subset

Internal DTD Subset (Contd...)

An example Internal DTD Subset:

```
<?xml version = “1.0”? >
<!DOCTYPE person [
<ELEMENT person(name, profession*) >
<ELEMENT....... >
......
]>
```

External DTD Subset

- All declarations that are not contained in the internal subset and comes from outside
- Internal and external subsets must be *compatible*
- Neither can override the element or attribute declarations the other makes
- However, entity declarations may be overridden
- Together they form complete DTD
- standalone attribute should have value “no”

External DTD Subset (Contd...)

An example External DTD Subset:
<?xml version = “1.0” encoding = “UTF8” standalone =“no”? >
<!DOCTYPE person SYSTEM “http://.../xml/ex1.dtd” >

External DTD Subset (Contd...)

- Factors to be considered to make decision about internal and external subsets
- External DTD can be used with multiple documents
- Document becomes concise
- Easier to maintain DTD
- Internal DTD: Completely independent document

Public IDs

- Standard DTDs may be stored at several URLs
- Such DTDs may be associated with public ID
- The public ID uniquely identies XML application
- URL is also given as backup
- An example declaration is given below:

Public IDs (Contd...)

Public IDs: An Example:
<!DOCTYPE rss PUBLIC
"//N etscapeCommunications//DT DRSS0.91//EN"
"http://my.netscape.com/publish/.../...dtd" >

Validating a Document

- A validating processor is required to read the external DTD subset
- A non-validating processor may read the subset
- Microsoft Internet Explorer 5(IE5) have built-in XML parser MSXML
- When an xml document is loaded into IE5, it is parsed by MSXML

Validating a Document (Contd...)

- On-line validators
- The Brown University Scholarly Technology Group's XML Validation form at http://www.stg.brown.edu/service/xmlvalid
- Richard Tobin's XML well-formedness checker and validation at http://www.cogsci.ed.ac.uk/%7Erichard/xml-check.html
- The document and associated DTD must be placed on publicly accessible web server

DTD Syntax

- Rules presented here are not exhaustive
- A DTD is not required to have a prolog
- Syntax: EBNF
- A DTD is not an XML document
- A DTD may have optional declaration
- Declarations may be for character set, ...

DTD Syntax (Contd...)

- The DOCTYPE declaration may trigger syntax error
- You may use white spaces liberally
- The order of declarations are important
- For duplicate declarations, the first takes precedence

Element Declarations

- XML may be used to define structure and to store the contents of documents
- XML document or document only
- XML document is viewed as tree
- Elements divide the document into its constituent parts

Element Declarations (Contd...)

- Elements with no content restrictions
- <!ELEMENT contain-anything ALL >
- Elements containing only character data
- It does not contain elements
- <!ELEMENT name (#PCDATA) >
- PCDATA: Parsed Character Data

Element Declarations (Contd...)

- Elements containing only elements
- Content consists only of elements
- <!ELEMENT article(title, (para | sect)+) >
- Symbols used in element content model
- **,**(stand for AND), |(for OR), **()** for grouping, **?**(renders the preceding element or group optional)

Element Declarations (Contd...)

- + requires at least one of the preceding element
- * stipulates any number of times
- (#PCDATA | name)*

Elements with Mixed Content

- A mixture of both elements and character data
- <!ELEMENT para (#PCDATA|name|xref)* >

An Example:
<!ELEMENT article (title, subtitle?, author*, (para|table|list)+, bibliography?)>
?: Zero or one times
*: Zero or more times
+: One or more times

Attribute List Declarations

- Elements may have attributes
- All attributes of an element should be declared at one place using attribute declarations
- For an element,attribute names must be unique
- <!ATTLIST element name
  attname1 atttype attdesc1
  attname2 atttype attdesc2 >

Attribute List Declarations (Contd...)

- attname: Attribute name
- attytype: Attribute type
- Ten attribute types:
  - CDATA, ID, IDREF, IDREFS, ENUMERATION, NMTOKEN, NMTOKENS, ENTITY, ENTITIES, NOTATION
- attdesc: Attribute description

Attribute List Declarations (Contd...)

An Example:
<ATTLIST memo id ID #REQUIRED
security (high|low) “high”
keywords NMTOKENS #IMPLIED
>

Attribute List Declarations (Contd...)

- #REQUIRED: Attribute must be specied
- #IMPLIED: Attribute is optional and has no default value
- (high | low): May take either value; default "high"
- ENUMERATION not a keyword and does not appear in declaration

Attribute Data Types

- ID: Unique identier
- Value must be XML name
- Guaranteed to be unique in the document
- No other attribute can have this value
- Each element gets unique label
- Example: id="ISBN-12456-98-123"

Attribute Data Types (Contd...)

- IDREF: Similar to ID
- It refers to ID of another element
- Error if no element with given ID
- IDREFS: More than one value of ID type attribute

Attribute Data Types (Contd...)

- NMTOKEN: A name token
- May contain alphanumeric and/or ideographic characters and the punctuation marks , -, and .
- All allowed characters can be first character
- XML Name: Only letters, ideographs, and can be rst character
- Example part no="XI-123"
- NMTOKENS: Several name tokens separated by white spaces

Attribute Data Types (Contd...)

- CDATA: Character Data
- Any character can be used
- Example equation="1+2+3=3+2+1"
- Attributes may have xed value
- color #FIXED "black"

Notations

- A notation type attribute contains the name of a notation declared in the documents DTD
- Syntax: <!NOTATION name identifier
- Used for labeling non-textual data
- Also to label textual data in specic format
- identier:An external identifier that has some meaning to the XML processor
- Meaning is processor dependent

Entity Declarations

- general entity:
  <!ENTITY abc “The abc group” >
- To reference above entity: &abc;
- External general entity: <!ENTITY man PUBLIC “//Acme Gadets//Textmanual23//EN”
  “http://www.acmegadgets.com/manuals/prod23.html” >
  <!ENTITY man SYSTEM “/pub/docs/manuals/prod...”
- The entity is referenced as &man;

Entity Declarations (Contd...)

- Nonparsed external entity:
  <!ENTITY logo PUBLIC //NONXMLlogo//EN
  http://www.acme/.../logo.gif NDATA gif >
  <!ENTITY logo SYSTEM images/logo.gif NDATA... >
- Entity reference: &logo;
- Parameter entity: Holds text from a DTD
- Can be used in either internal or external subset

Parameter Entity

- A simple substitution for DTD text:
  $<$ ENTITY % paratext "(#P CDATA | emph | acronym) * "
  $>$
- Entity referenced as: %paratext;
- External parameter entity:
  $<$!ENTITY % tables PUBLIC "-//Acme...//EN"
  "/xmldtdds/tables2.1.dtd" $>$
  $<$!ENTITY % tables SYSTEM "http : //www....." $>$
- % is used in denition and reference

An Example of Parameter Entity

```
<!ENTITY % content "para|note|warning" >
<!ENTITY % id.att "id ID #REQUIRED" >
<!ELEMENT chapter (title, epigraph, (%content; )+) >
<!ATTLIST chapter %id.att; >
<!ELEMENT appendix (title, (%content; )+) >
<!ATTLIST appendix %id.att; >
```

An Example: Checkbook Document

```
<?xml version="1.0"?>
<!DOCTYPE checkbook SYSTEM "checkbook.dtd">
<checkbook>
    <deposit type="direct-deposit">
        <payor>Bob's Bolts</payor>
        <amount>987.32</amount>
        <date>21-6-00</date>
        <description category="income">Paycheck</description>
    </deposit>
    <payment type="check" number="980">
        <payee>Kimora's Sports Equipment</payee>
        <amount>132.77</amount>
        <date>23-6-00</date>
```

An Example: Checkbook Document (Contd...)

```
    <description category="entertainment">Kendo equipment
    </description>
</payment>
<payment type="atm">
    <amount>40.00</amount>
    <date>24-6-00</date>
</payment>
<payment type="debit">
    <payee>Lone Star Cafe</payee>
    <amount>36.86</amount>
    <date>26-6-00</date>
    <description category="food">Lunch with Greg
    </description>
</payment>
```

An Example: Checkbook Document (Contd...)

```
    <payment type="check" number="981">
        <payee>Wild Oats Market</payee>
        <amount>47.28</amount>
        <date>29-6-00</date>
        <description category="food">Groceries</description>
    </payment>
    <payment type="debit">
        <payee>Barnes and Noble</payee>
        <amount>58.79</amount>
        <date>30-6-00</date>
        <description category="work">O'Reilly Books
        </description>
    </payment>
</checkbook>
```

The DTD for Checkbook Example

```
<!––
A simple checkbook DTD
––>
<! –– parameter entities ––>
<!ENTITY %basic.content '#PCDATA' >
<!ENTITY %entry.content 'amount, date, description?' >
<! –– main elements –– >
<!ELEMENT checkbook (deposit | payment)* >
<!ELEMENT deposit (payor, %entry.content; ) >
<!ATTLIST deposit type(cash | check | direct-deposit | transfer)
#REQUIRED >
<!ELEMENT payment (payee?, %entry.content; ) >
<!ATTLIST payment type(atm | check | 1debit) #REQUIRED >
```

The DTD for Checkbook Example (Contd...)

```
<!-- basic elements -- >
<!ELEMENT amount (%basic.content; )* >
<!ELEMENT date (%basic.content; )* >
<!ELEMENT payee (%basic.content; )* >
<!ELEMENT payor (%basic.content; )* >
<!ELEMENT description (%basic.content; )* >
<!ATTLIST description
category(cash | entertainment | food | income | work) 'food' >
```