

Data-centric XML

Document Types

Purpose

- Document Type Definitions define a vocabulary
 - set of allowed element names
 - set of attributes per element name
 - data type given for each attribute
 - content model: elements and data allowed inside the content of the element
- Validation: checking the conformance of a document
- Association of semantics: explanation of the meaning of each element, for a certain kind of processing

Things not specified

- root element of the document
 - Some DTDs (e.g. DocBook) are used with different root elements (e.g. book, article)
- number of instances of each element
- structure of the character data
- semantics of each element
 - specified in natural language; e.g. DocBook gives “processing expectations”

An Example

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

DTD Usage Example

```
<?xml version="1.0" standalone="no" ?>  
<!DOCTYPE person SYSTEM "http://cafeconleche.org/dtds/  
person.dtd">  
<person>  
  <name>  
    <first_name>Alan</first_name>  
    <last_name>Turing</last_name>  
  </name>  
  <profession>computer scientist</profession>  
  <profession>mathematician</profession>  
</person>
```

Document Identifier

- **SYSTEM**: meaningful only on the local system
 - XML: must be URI Reference (RFC2732)
 - no fragment identifier
 - relative identifiers are relative to the location of the original resource
- **PUBLIC**: intended to be meaningful across systems
 - inherited from SGML
 - located on the local system by means of catalogs
 - FPI: Formal Public Identifier

Formal Public Identifier

Syntax: prefix//owner-identifier//text-class text-description//
language//display version

- prefix: + (registered), – (unregistered), ISO (reserved to ISO)
- owner-identifier: organization issuing FPI
 - IDN allows to use domain names
- text-class: DOCUMENT, DTD, ELEMENTS, ENTITIES, NONSGML, NOTATION, ...
- text-description: free form text
- language: ISO code
- display version (optional): distinguishes different forms

FPI Examples

```
-//OASIS//DTD DocBook V3.1//EN  
-//W3C//DTD XHTML 1.0 Strict//EN  
-//W3C//ENTITIES Latin 1 for XHTML//EN  
ISO 646//NOTATION IS 646-IRV//EN  
+//IDN python.org//DTD XML Bookmark Exchange  
Language 1.0//EN//XML
```


Internal DTD Subset

```
<?xml version="1.0"?>
<!DOCTYPE person [
  <!ELEMENT person (name, profession*)>
  <!ELEMENT name (first_name, last_name)>
  <!ELEMENT first_name (#PCDATA)>
  <!ELEMENT last_name (#PCDATA)>
  <!ELEMENT profession (#PCDATA)>
]>
<person>
  <name><first_name>Alan</first_name><last_name>Turing</
  last_name></name>
</person>
```

DTD Subsets

- external subset specified through system or public identifier
- internal subset included in document
- must not have overlapping element definitions
- internal subset occurs before external subset, so internal definitions of entities and attribute lists take precedence

Validation

- Process of checking all validity constraints
- validating processor must read external DTD subset
 - non-validating processor may still read external subset, to find entity definitions
- access to external entities resolves either through public identifier or system identifier, at the processor's (or application's) choice

Element Specifications

[45] elementdecl ::= '<!ELEMENT' S
Name S contentspec S? '>'

- VC: element names must be unique

[46] contentspec ::= 'EMPTY' | 'ANY' | Mixed | children

- Elements with EMPTY content model are valid if they have no content
 - for interoperability, empty-element tag should be used iff content model is EMPTY
- Elements with ANY content model are valid if all child elements have been declared

Element Content

[47] children ::= (choice | seq) ('?' | '*' | '+')?

[48] cp ::= (Name | choice | seq) ('?' | '*' | '+')?

[49] choice ::= '(' S? cp (S? '|' S? cp)+ S? ')'

[50] seq ::= '(' S? cp (S? ',' S? cp)* S? ')'

- content is valid if it is possible to trace through the content model, following choices and sequences appropriately
 - for compatibility, the content model must be deterministic
- space (S) is allowed around child elements

Mixed Content

[51] Mixed ::= '(' S? '#PCDATA'
(S? '|' S? Name)* S? ')*'
| '(' S? '#PCDATA' S? ')'

- Names of child nodes, unordered
- VC: element names must not appear twice

Attribute Declarations

```
<!ATTLIST image1 source CDATA #REQUIRED>  
<!ATTLIST image2  source CDATA  #REQUIRED  
                    width CDATA  #REQUIRED  
                    height CDATA #REQUIRED  
                    alt   CDATA  #IMPLIED>
```

Attribute List Syntax

[52] AttlistDecl ::= '<!ATTLIST' S Name AttDef* S? '>'

[53] AttDef ::= S Name S AttType S DefaultDecl

- multiple AttlistDecl for the same Name are merged
- for multiple declarations of the same attribute, only the first declaration is binding

Attribute Types

- Three kinds of types: strings, tokenized lists, and enumerations

[54] `AttType ::= StringType | TokenizedType | EnumeratedType`

Character Data Attributes

[55] StringType ::= 'CDATA'

- contains arbitrary text
- references are expanded; otherwise, data is uninterpreted
- default type for a non-validating parser

Tokenized Attributes

[56] TokenizedType ::= 'ID'
| 'IDREF'
| 'IDREFS'
| 'ENTITY'
| 'ENTITIES'
| 'NMTOKEN'
| 'NMTOKENS'

ID

- Unique identification of elements within a document
- VC: Must match Name production;
in a document, all values of this type must be unique
- VC: At most one ID attribute per element type
- VC: Default value must be #REQUIRED or #IMPLIED

```
<!ATTLIST employee social_security_number ID  
#REQUIRED>
```

```
<employee social_security_number="_078-05-1120">...
```

IDREF

- refers to elements with an ID
- VC: there must be an attribute of type ID with the same value

```
<!ATTLIST team_member person IDREF #REQUIRED>
```

```
<team_member person="_078-05-1120">
```

IDREFS

- List of multiple IDs, space separated
- VC: must match production Names; individual names must be ID values

ENTITY/ENTITIES

- Refers to unparsed entities (not yet discussed)
- VC: Value must match Name production; must refer to unparsed entity declaration
- ENTITIES: likewise list of unparsed entity names

NMTOKEN(S)

- VC: value must match production Nmtoken(s)
- used to constrain attributes to “identifier-like” things:
 - allows “.cshrc”, “March”, “2003”
 - disallows “March 2003”, “Sally had a lamb”

Enumerated Attributes

[57] EnumeratedType ::= NotationType | Enumeration

[58] NotationType ::= 'NOTATION' S '(' S? Name (S? '|' S? Name)* S? ')'

- VC: Names must be notation names; attribute values must match one of the names (examples given later)
- VC: Each element must have at most one attribute of notation type
- VC: For compatibility, empty elements must not have notation attributes

[59] Enumeration ::= '(' S? Nmtoken (S? '|' S? Nmtoken)* S? ')'

- VC: attribute values must match one of the Nmtokens

```
<!ATTLIST date month (Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec) #IMPLIED>
```

```
<!ELEMENT date EMPTY>
```

```
<date day="20" month="Oct" year="2003"/>
```

Attribute Defaults

[60] DefaultDecl ::= '#REQUIRED' | '#IMPLIED'
| (('#FIXED' S)? AttValue)

- VC: #REQUIRED attributes must be specified on all elements
- WFC: AttValue must not contain '<'
- VC: AttValue must be follow lexical constraints of the attribute type
- VC: values of #FIXED attributes must match the AttValue

<!ATTLIST termdef

id ID #REQUIRED

name CDATA #IMPLIED>

<!ATTLIST list

type (bullets|ordered|glossary) "ordered">

<!ATTLIST form

method CDATA #FIXED "POST">

Attribute Value Normalization

1. Line breaks are normalized to #xA
2. For each character/reference,
 1. replace character references with referenced characters
 2. replace entity references recursively with replacement text
 3. replace white space (#x20, #xD, #xA, #X9) with a space character
3. For non-CDATA attributes, remove leading and trailing space, and replace sequences of space with a single #x20

General Entities

- Text replacement mechanism
- Predefined: gt, lt, amp, quot, apos
- User-defined: Using entity declarations
`<!ENTITY super "supercalifragilisticexpialidocious">`
...
`&super;`
- Replacement text can contain further markup (elements and references)
- Can be internal to the DTD, or external
`<!ENTITY footer SYSTEM "http://www.oreilly.com/
boilerplate/footer.xml">`

Entity Declarations

[70] EntityDecl ::= GEDecl | PEDecl
[71] GEDecl ::= '<!ENTITY' S Name S EntityDef
S? '>'
[72] PEDecl ::= '<!ENTITY' S '%' S Name S
PEDef S? '>'
[73] EntityDef ::= EntityValue | (ExternalID
NDataDecl?)
[74] PDef ::= EntityValue | ExternalID

- General entities: usable anywhere inside character data for replacement text
- Parameter entities: usable only in DTD, to allow parameterization of DTD
- General entities are either parsed or unparsed (NDATA)

Internal Entities

- Defined through EntityValue

[9] EntityValue ::= ''' ([^%&"] | PEReference | Reference)* '''
| "" ([^%&'] | PEReference | Reference)* ""

- Internal entities are always parsed

External Entities

[75] ExternalID ::= 'SYSTEM' S SystemLiteral
| 'PUBLIC' S PubidLiteral S SystemLiteral

[76] NDataDecl ::= S 'NDATA' S Name

- Parser may use PubidLiteral to obtain alternative URI
- Otherwise, SystemLiteral must be used to retrieve resource
 - SystemLiteral is encoded as UTF-8, non-ASCII characters are escaped using %HH
 - non-validating parser may refuse resource download, and report the reference instead (providing declaration details if available)
- Presence of NDataDecl indicates unparsed entity
- VC: Name in NDataDecl must be a declared notation

Parsed Entities

- Must be well-formed, i.e. match production extParsedEnt
[78] extParsedEnt ::= TextDecl? content
- TextDecl (<?xml ...?>) must be used to denote non-UTF-8 entities
- Production content guarantees that markup cannot split across replacement texts, and that start-tag and end-tag must be balanced

Unparsed Entities and Notations

```
<!NOTATION gif SYSTEM "image/gif">
```

```
<!NOTATION jpeg SYSTEM "image/jpeg">
```

```
<!NOTATION png SYSTEM "image/png">
```

```
<!ENTITY turing_getting_off_bus
```

```
    SYSTEM "http://www.turing.org.uk/turing/pi1/bus.jpg"
```

```
    NDATA jpg>
```

- usage of unparsed entity references only in attributes of type entity

```
<!ELEMENT image EMPTY>
```

```
<!ATTLIST image source ENTITY #REQUIRED>
```

```
...
```

```
<image source="turing_getting_off_bus"/>
```

- no further processing of entity by parser; application must interpret notation and download the resource

Notation Syntax

[82] NotationDecl ::= '<!NOTATION' S Name S (ExternalID | PublicID) S? '>'

[83] PublicID ::= 'PUBLIC' S PubidLiteral

- XML processor must pass notation name and identifiers to the application
 - optionally, processor may resolve public id into system identifier indicating processor for the application
- VC: Notation names must be unique within the document

Further Notation Usage

- Processing Instruction Targets

```
<!NOTATION tex “/usr/local/bin/tex”>
```

- Notation attributes

```
<!ATTLIST image type NOTATION (gif | jpeg | png)>
```

Parameter Entities

- Macro replacement mechanism in DTDs
- allows multiple usage of the same content model
- also allows parametrization, by means of conditional inclusion

PE Example (XHTML)

```
<!ENTITY % coreattrs
  "id          ID          #IMPLIED
  class       CDATA       #IMPLIED
  style       %StyleSheet; #IMPLIED
  title       %Text;      #IMPLIED"
>
<!ENTITY % attrs "%coreattrs; %i18n; %events;">
<!ENTITY % Block "(%block; | form | %misc;)*">
<!ELEMENT body %Block;>
<!ATTLIST body
  %attrs;
  onload      %Script; #IMPLIED
  onunload    %Script; #IMPLIED
>
```

PE Syntax

[72] PEDecl ::= '<!ENTITY' S '%' S Name S
PEDef S? '>'

[74] PDef ::= EntityValue | ExternalID

[69] PReference ::= '%' Name ';' ;

- External PEs: recursively downloaded in validating processor; allow modular definition of DTD

```
<!ENTITY % HTMLlat1 PUBLIC
```

```
"-//W3C//ENTITIES Latin 1 for XHTML//EN"
```

```
"xhtml-lat1.ent">
```

```
%HTMLlat1;
```

- VC: entity in PReference must be declared
- WFC: PDefs must not be recursive, and must occur only in DTDs

Parameterization

- Redeclaration of PEs in internal subset
 - first declaration is binding
 - can be used to add or remove attributes from attribute lists, or change the content model, if the DTD allows it
- In addition, conditional inclusion allows omitting parts of the DTD

Conditional Inclusion

- INCLUDE vs. IGNORE

```
<![IGNORE[  
  <!ELEMENT production_note (#PCDATA)>  
]]>
```

```
<![INCLUDE[  
  <!ELEMENT production_note (#PCDATA)>  
]]>
```

- Conditional inclusion: define PE that expands to either INCLUDE or IGNORE

```
<!ENTITY % notes_allowed "INCLUDE">  
<![%notes_allowed;  
  <!ELEMENT production_note (#PCDATA)>  
]]>
```


Comparison with SGML

- More Keywords (beyond DOCTYPE, ELEMENT, ATTLIST, NOTATION):
 - SHORTREF, USEMAP as a macro mechanism
- Optional markup minimization
 - can omit either start tag or end tag (need to declare minimizable tags in DTD)
 - Can minimize end tags to </>
 - Can omit semicolons
 - Can omit quotes/apostrophes in attribute values
 - Can omit attribute names
- More attribute types (NUMBER(S), NUTOKEN(s))