

Introduction to XML: DTD

Jaana Holvikivi

Document type definition: structure

Topics:

- Elements
- Attributes
- Entities
- Processing instructions (PI)
- DTD design

DTD

<!-- Document type description (DTD) example (part) -->

```
<!ELEMENT university (department+)>  
<!ELEMENT department (name, address)>  
<!ELEMENT name (#PCDATA)>  
<!ELEMENT address (#PCDATA)>
```

- Document type description, structural description
- one rule /element
 - name
 - content
- a grammar for document instances
- "regular clauses"
- (not necessary)

DTD: advantages

- validating parsers check that the document conforms to the DTD
- enforces logical use of tags
- there are existing DTD standards for many application areas
 - common vocabulary

Well-formed documents

- An XML document is **well-formed** if
 - its elements are properly nested so that it has a hierarchical tree structure, and all elements have an end tag (or are empty elements)
 - it has one and only one root element
 - complies with the basic syntax and structural rules of the XML 1.0 specification:
 - rules for characters, white space, quotes, etc.
 - and its every parsed entity is well-formed

Validity

- An XML-document is valid if
 - it is well-formed
 - it has an attached DTD (or schema)
 - it conforms to the DTD (or schema)
- Validity is checked with a validating parser, either
 - the whole document at once ("batch")
 - interactively

Document type declaration

Shared

```
<!DOCTYPE catalog PUBLIC "-//ORG_NAME//DTD  
CATALOG//EN">
```

- flag(-/+) indicates a less important standard
- ISO -standards start with "ISO
- ORG_NAME the owner of the DTD
- DTD file type
- CATALOG document name
- EN language

the document type definition can be included in the internal database of the processor (no connection needed)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
```

External or internal DTD

Document type declaration format:

```
<!DOCTYPE document_element source location  
  [internal subset of DTD] >
```

internal DTD, DTD and instance in the same file:

```
<!DOCTYPE catalog SYSTEM  
  [<!ELEMENT catalog ... and so on ...]>
```

simple example:

```
<!DOCTYPE Mymessage SYSTEM  
  [<!ELEMENT Mymessage (#PCDATA)]>
```

external DTD, examples

```
<!DOCTYPE dictionary SYSTEM "dictionary.dtd">
```

```
<!DOCTYPE dictionary SYSTEM  
  "http://www.evtek.fi/DTD/dictionary.dtd">
```


External or internal DTD

internal and external:

```
<!DOCTYPE Mymessage SYSTEM "myDTD.dtd"  
[<!ELEMENT Mymessage ... and so on...  
... (#PCDATA)]>
```

Shared document type:

```
<!DOCTYPE Dictionary PUBLIC  
"http://www.evtek.fi/DTD/dictionary.dtd">
```

Element type declaration

- **<!ELEMENT country (capital)>**
- element name
- element content
 - content declaration, content model
- delimiters (**<!**, **>**, **(**, **)**) and keyword (**ELEMENT**)

Sub-elements, children

- Children in specified order:
<!ELEMENT country (cname, capital, population)>
- choice of a child ("pipe" |):
<!ELEMENT country (cname | official_name)>
- optional singular element ? only one or zero:
<!ELEMENT country (cname, capital, population?)>

Cardinality operators

Number of occurrences

* zero or more, optional

<!ELEMENT country (cname, capital, city*)>

+ one or more child elements, required

<!ELEMENT country (cname, neighbour_country+)>

? optional singular element

[none] one required singular element

repeating group:

<!ELEMENT country (cname, (city, city_population)*)>

Element content model

- Data
 - `<!ELEMENT cname (#PCDATA)>`
 - "parsed character data"
- Elements
 - sub-elements (child elements)
- Mixed content
 - data and elements
 - `<!ELEMENT para (#PCDATA | sub | super)*>`
 - #PCDATA must be first in content model, group has options
 - child element sequence, choices and cardinality cannot be specified

Empty element and ANY

- `<!ELEMENT image EMPTY>`
 - in document instance must be `<image/>`
 - not allowed: `<image></image>`
- a regular declaration `<!ELEMENT im (...)>`
 - allows both ways `<im/>`, `<im></im>` or `<im>...</im>`
- `<!ELEMENT some ANY>`
 - element can contain any declared element,
 - flexible, maybe too flexible?

Short example: Dictionary

```
<!ELEMENT dictionary (word_article)*>  
<!ELEMENT word_article (head_word,  
    pronunciation, sense+)>  
<!ELEMENT head_word (#PCDATA)>  
<!ELEMENT pronunciation (#PCDATA)>  
<!ELEMENT sense (definition, example*)>  
<!ELEMENT definition (#PCDATA)>  
<!ELEMENT example (#PCDATA)>
```

Dictionary XML

```
<?xml version="1.0" ?>  
<!DOCTYPE dictionary SYSTEM "dict.dtd">  
<dictionary>  
  <word_article>  
    <head_word>  
      carry  
    </head_word>  
    <pronunciation>  
      kaeri  
    </pronunciation>
```



```
<sense>
  <definition>
    support the weight of and move from place to place
  </definition>
  <example>
    Railways and ships carry goods.
  </example>
  <example>
    He carried the news to everyone.
  </example>
</sense>
<sense>
  <definition>
    wear, possess    </definition>
  <example>
    I never carry much money with me.
  </example>
</sense>
</word_article>
```

```
<word_article>  
  <head_word>gossamer  
  </head_word>  
  <pronunciation>  
  gosomo  
  </pronunciation>  
  <sense>  
    <definition>fine, silky substance of webs made by  
    small spiders  
    </definition>  
  </sense>  
</word_article>  
</dictionary>
```

Content models

- Try to make definitions unambiguous (clear)
 - wrong: (item?, item)
 - right: (item, item?)
 - wrong ((surname, employee) | (surname, customer))
 - right: (surname, (employee | customer))
- <!ELEMENT BookCatalog (Catalog, Publisher+, Book*)>
 - document must have <BookCatalog> top level element
 - contains always one <Catalog> and at least one <Publisher> child
 - <Book> elements may not be present

Attribute declarations ATTLIST

- Attributes can be used to describe the metadata or properties of the associated element
- attributes are also an alternative way to markup data

```
<!ATTLIST country
    population NMTOKEN #IMPLIED
    language CDATA #REQUIRED
    continent (Europe | America | Asia ) "Europe">
```

- CDATA
 - character data, any text
- enumerated values (choice list)
 - `<!ATTLIST country continent (Europe | America | Asia) "Europe">`
 - remember that XML is case sensitive
 - default value given above
 - the parser *may* supply a default value if it is not given

Attribute defaults

- **#REQUIRED**
 - the attribute must appear in every instance of the element
- **#IMPLIED**
 - optional
- enumerated values can have a default
- no default for implied/required

```
<!ATTLIST catalog type CDATA #REQUIRED>
```

```
<!ATTLIST catalog type NMTOKEN #IMPLIED>
```

```
<!ATTLIST catalog type (phone | e-mail)>
```

```
<!ATTLIST catalog type (phone | e-mail) "phone">
```

Attribute types

- NMTOKEN
 - name token <country population = "100">
- NMTOKENS
 - a list of name tokens delimited by white space
 - These types are useful primarily to processing applications. The types are used to specify a valid name(s). You might use them when you are associating some other component with the element, such as a Java class or a security algorithm

```
<!ATTLIST DATA AUTHORIZED_USERS NMTOKENS  
  #IMPLIED>
```

```
<DATA SECURITY="ON"
```

```
  AUTHORIZED_USERS = "IggieeB SelenaS GuntherB">
```

```
element content
```

```
</DATA>
```

Attribute types

- ID
 - attribute value is the *unique* identifier for this element instance, must be a valid XML name
- IDREF
 - reference to the element that has the same value as that of the IDREF
- IDREFS
 - a list of IDREFs delimited by white space

Attribute defaults

- `<!ATTLIST country position #FIXED "independent">`
 - attribute must match the default value
 - why: for example to supply a value for an application
- reserved attributes,
 - `xml:lang`
 - `xml:space`
 - prefix `'xml:'`

Element vs attribute

- When to mark up with an element, when to use attributes?
- Element
 - to describe structures, expandable
 - when shown in the output
 - contents cannot be defined as strictly as with attributes
- Attribute
 - no structure, no multiple values
 - “internal” information
 - default values possible

Entities

- Each XML document is an entity, could comprise of several entities
- document entity
- "subdocuments" = entities
- general entities:
 - internal or external
 - parsed or unparsed (external only)
- a parsed entity can include any well-formed content (replacement text)
- entity declaration
- entity reference
- all unparsed entities must have an associated notation

Internal text entities

- Predefined string
<!ENTITY evitech "Espoo Vantaa Institute of Technology">
I study at the &evitech;
- Single versus double quotes
 - <!ENTITY sent 'His foot is 12" long'>
 - <!ENTITY sent "His foot is 12" long">

Entity reference character string

>	>
<	<
"	"
'	'
&	&

<	<
A	A
<	< (hexadecimal)
࿸	... (Unicode)

CDATA: special characters in XML

```
<action>  
<script language ='Javascript'>  
<![CDATA[  
    function Fhello()  
    {    if (n >1 && m > 8)  
            alert ("Hello");  
    }  
]]>  
</script>  
</action>
```

External entity

- Outside the document entity itself
- within the same resource:
 - `<!ENTITY myfile SYSTEM "extra_files/file.xml">`
- public location
 - `<!ENTITY myfile PUBLIC "... description...">`
 - needs an index
- an unparsed external entity is a reference to an external resource (I.e. an image file)
 - Binary file
 - file type has to be declared
 - `<!ENTITY myphoto SYSTEM "/figures/photo.gif" NDATA GIF>`
 - Use:
Take a look at my photo `<picture name="myphoto"/>`.

- ENTITY

- <!ENTITY mypicture "123.jpg">

- <!ELEMENT pic EMPTY>

- <!ATTLIST pic picfile ENTITY mypicture>

- in the document instance:

- <pic picfile="mypicture"/>

- ENTITIES

- a list of ENTITY names

- Notation

- <!ATTLIST image format NOTATION (TeX | TIFF)>

Entity references, summary

- General parsed entity reference
 - in the document instance
 - not in the DTD
 - entity hierarchy
- unparsed entity
 - no references from the text
 - given as attribute values
- parameter entity
 - in DTD, not in the document instance

Parameter entity

Only usable within a DTD (not in an XML document)

```
<!ENTITY % parapart "(emph | supersc | subsc)">
```

```
<!ELEMENT paragraph (%parapart | bold)>
```

```
<!ELEMENT list (%parapart | item)*>
```

```
<!ELEMENT paragraph (emph | supersc | subsc | bold)>
```


Notation declaration

```
<!NOTATION PIXI SYSTEM "">
```

```
<!NOTATION TIFF SYSTEM "C:\APPS\Show_tiff.exe">
```

Entity declaration refers to notation:

```
<!ENTITY Logo SYSTEM "logo.tif" NDATA TIFF>
```

Notation provides information for an application how to process unparsed entities

Without a DTD:

- Attributes have no default values
- attributes are always text type CDATA
- all attributes are optional
- entities cannot be declared
- only standard entities are possible (')
- element contents are not clearly defined
 - elements, data or mixed

DTD design

- XML often replaces a previous system
- when transforming to XML
 - a standard DTD could be selected (with possible modifications)
 - partners, affiliations
 - a new DTD is designed
- DTD design based on
 - existing document models in the company
 - (representative) model documents
 - other designers consulted

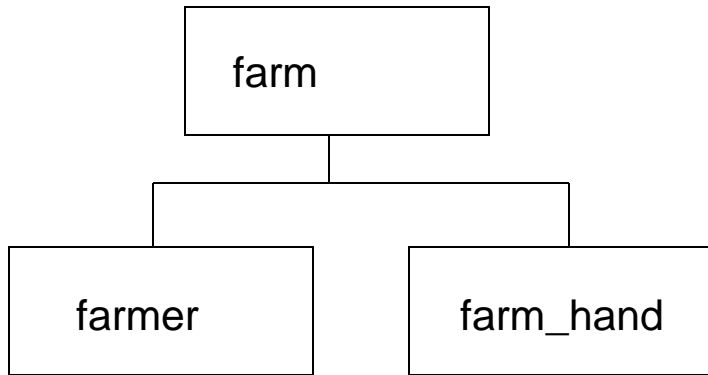
Document analysis

- Document features
 - name, could it be without a name?
 - how many occurrences
 - preceding/ following information, regularity
 - parts of the document
 - standard contents (automatically generated)
- XML document (or parts of it) maybe generated from a data base
 - use data base relation, descriptions and models (UML) when designing DTD

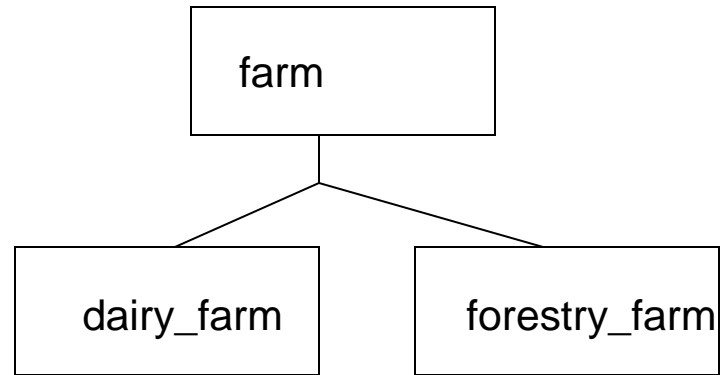
DTD design

- Standard DTD or new?
 - Compatibility and data exchange
 - processing needs, applications
 - future needs, linking
 - consistent names
- Element order, granularity, structure
- element vs. attribute?
- Rules? Order of rules ?
- comments?
- modularity?
- Naming style: short or descriptive, upper or lower case?

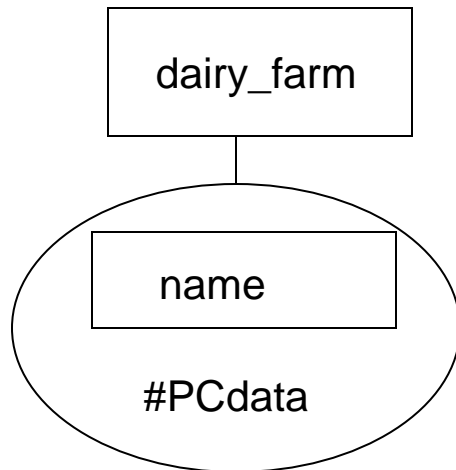
Tree diagrams



<!ELEMENT farm (farmer, farm_hand)>



<!ELEMENT farm (dairy_farm | forestry_farm)>



Standard DTDs

- <http://www.xml.com/pub/rg/DTDs>
 - <http://www.xml.org/>
 - <http://xml.coverpages.org/>
 - <http://www.ebxml.org/specs/ebBPSS.dtd>
 - <http://www.w3.org/QA/2002/04/valid-dtd-list.html>
-
- MathML,
 - CML (chemistry),
 - UXF (UML eXchange Format),
 - SMIL (multimedia),
 - RDF (Resource Description Framework),
 - DocBook, etc.