XML and Semantic Web Technologies

II. XML / 2. XML Document Type Definitions (DTDs)

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute of Economics and Information Systems
& Institute of Computer Science
University of Hildesheim
http://www.ismll.uni-hildesheim.de

Motivation / Heterogenous Mark-up

```
1 <?xml version="1.1"?>
2 <books>
3   <book year="2004">
4     <authors>
5       <author><fn>Rainer</fn><sn>Eckstein</sn></author>
6       <author><fn>Silke</fn><sn>Eckstein</sn></author></authors>
7     <title>XML und Datenmodellierung</title>
8     <isbn>3-89864-222-4</isbn><book>
9   <book>
10    <authors year="2004">
11      <author><fn>Erik</fn><fn>T.</fn><sn>Ray</sn></author></authors>
12      <title>Learning XML</title>
13  </book>
14 </books>
```

Figure 1: A list of books.

```
1 <?xml version="1.1"?>
2 <books>
3   <book isbn="isbn-1-565-92580-7">
4     <author>Norman Walsh and Leonard Muellner</author>
5     <title>DocBook: The Definitive Guide</title>
6     <year>1999</year></book>
7 </books>
```

Figure 2: Another list of books.

II. XML / 2. XML Document Type Definitions (DTDs)

Document Type Declaration

⟨*DoctypeDecl*⟩ := <!DOCTYPE ⟨*S*⟩ ⟨*Name*⟩
(⟨*S*⟩ ⟨*ExternID*⟩)?
(⟨*S*⟩? [⟨*InternDoctypeDecl*⟩])? ⟨*S*⟩? >

⟨*InternDoctypeDecl*⟩ := (⟨*EntityDecl*⟩
| ⟨*ElementDecl*⟩ | ⟨*AttlistDecl*⟩
| ⟨*NotationDecl*⟩
| ⟨*PEReference*⟩
| ⟨*PI*⟩ | ⟨*Comment*⟩ | ⟨*S*⟩)*

⟨*Name*⟩ specifies the name of the root element.

Document type declarations can be given

- in a file of its own (**external DTD**; see below)
  AND (alternatively as well as additionally)
- in the XML document itself (**internal DTD**).

Entity Declarations

⟨*EntityDecl*⟩ := <!ENTITY ⟨*S*⟩ ⟨*Name*⟩ ⟨*S*⟩
(" ⟨*EntityValue*⟩ " | (⟨*ExternID*⟩ ⟨*NDataDecl*⟩?)) ⟨*S*⟩? >
| <!ENTITY ⟨*S*⟩ % ⟨*S*⟩ ⟨*Name*⟩ ⟨*S*⟩
(" ⟨*EntityValue*⟩ " | ⟨*ExternID*⟩) ⟨*S*⟩? >

Entities generated by 1st line are called
general entities — for usage in character data and attribute values,

Entities generated by 2nd line are called
parameter entities — for usage in DTDs.
(see section 4).

General entities w./o. ⟨*NDataDecl*⟩ are called **parsed entities**,
otherwise **unparsed entities** (see section 3).

Usage of general entities

Parsed entities provide a mechanism for mixed content constants.

```
1 <?xml version="1.1"?>
2 <!DOCTYPE page [
3   <!ENTITY tel "<phone country='+49'>05121 / 883 851</phone>">
4 ]>
5 <page>
6 You can call me at &tel;.
7 </page>
```

Figure 3: Definition and usage of a XML entity.

```
1 <?xml version="1.1" encoding="UTF-8"?>
2 <page>
3    You can call me at <phone country="+49">05121 / 883 851</phone>.
4 </page>
```

Figure 4: Document with resolved entities as seen after parsing.

Non-validating XML parsing

XML documents can be parsed, e.g.,

– with Apache Xerces (http://xml.apache.org/xerces2-j/):

```
xerces ex-entity.xml
```

Non-validating Parsing

- checks if the document is well-formed,
- resolves all general entities.

```
#!/bin/bash
XERCES_HOME=/opt/xml/xerces
java -cp $XERCES_HOME/xercesSamples.jar:$XERCES_HOME/xercesImpl.jar \
    sax.Writer $@
echo
```

Figure 5: Script to run xerces.

Illegal usage of general entities

Parsed entities

- must have a well-formed value,
- can only be used in character data or attribute values (but not, e.g., in start-tags to specify attribute names).

```
<?xml version="1.1"?>
<!DOCTYPE page [
  <!ENTITY plz "Please</s>">
]>
<page>
  <s>&plz;, call me soon.
</page>
```

Figure 6: Illegal entity declaration: replacement text must be well-formed.

```
<?xml version="1.1"?>
<!DOCTYPE page [
  <!ENTITY own "owner='me'" >
]>
<page &own; >
  Hello !
</page>
```

Figure 7: Illegal entity usage: (general) entities can only be used in character data.

External General Entities

⟨*ExternID*⟩ := `SYSTEM` ⟨*S*⟩ `"` ⟨*URI*⟩ `"`
| `PUBLIC` ⟨*S*⟩ `"` ⟨*PublicID*⟩ `"` ⟨*S*⟩ `"` ⟨*URI*⟩ `"`

All external references must have a **system identifier** ⟨*URI*⟩.

The system identifier ⟨*URI*⟩ points to a resource that contains the value of the entity.

- ⟨*URI*⟩ may be relative to the location of its context,
- ⟨*URI*⟩ may not contain a fragment identifier.

The **public identifier** ⟨*PublicID*⟩ is a key that can be resolved by a (system-specific) catalog to an URI (see section 5).

External General Entities / Content Modularization

External general entities can be used the same way as internal general entities.

```
1 <?xml version="1.1"?>
2 <!DOCTYPE article [
3   <!ENTITY intro SYSTEM "intro.xml">
4   <!ENTITY results SYSTEM "results.xml">
5   <!ENTITY outlook SYSTEM "outlook.xml">
6 ]>
7 <article>
8   &intro;
9   &results;
10  &outlook;
11 </article>
```

Figure 8: Master document master.xml

```
1 <h>Introduction</h>
2 Among the most urgent questions ...
```

Figure 9: Included document intro.xml

```
1 <h>Results</h>
2 Based on the idea of the last section ...
```

Figure 10: Included document results.xml

```
1 <h>Outlook</h>
2 In this article we have seen ...
```

Figure 11: Included document outlook.xml

External DTDs

DTDs can be in a file on their own and included via a system or public identifier.

In external DTDs some additional constructs are allowed (see section 4).

```
1 <?xml version="1.1"?>
2 <!DOCTYPE page SYSTEM "me.dtd">
3 <page>
4 You can call me at &tel;.
5 </page>
```

Figure 12: XML document with external DTD.

```
1 <!ENTITY tel "05121 / 883851">
2 <!ENTITY fax "05121 / 883859">
3 <!ENTITY email "schmidt-thieme@ismll.uni-hildesheim.de">
```

Figure 13: External DTD `me.dtd`.

II. XML / 2. XML Document Type Definitions (DTDs)

Document structure can be constrained by specifying

a) the elements allowed
(basic **element declaration**),

b) the attributes allowed for each element
(names, types, and default values; **attribute list declaration**),

c) the contents allowed for each element
(**element content model**).

Well-formed: document matches document production rules and constraints.

Valid: contents and paramerers of all elements match document type.

Element Declaration

⟨*ElementDecl*⟩ := `<!ELEMENT` ⟨*S*⟩ ⟨*Name*⟩ ⟨*S*⟩ ⟨*contentSpec*⟩ ⟨*S*⟩? `>`

⟨*contentSpec*⟩ := `EMPTY` | `ANY` | ⟨*childrenSpec*⟩ | ⟨*mixedSpec*⟩

`EMPTY`: only empty element is allowed:

```
<!ELEMENT available EMPTY>
```

allows only

`<available/>` or `<available></available>`

`ANY`: there are no restrictions on element contents.

Valid Document

```
1 <?xml version="1.1"?>
2 <!DOCTYPE page [
3   <!ELEMENT page ANY>
4 ]>
5 <page/>
```

Figure 14: A minimal valid document.

Element Declaration / children

⟨*childrenSpec*⟩ := (⟨*choice*⟩ | ⟨*seq*⟩) (? | ⋆ | +)?

⟨*choice*⟩ := (⟨*S*⟩? ⟨*cp*⟩ ⟨*S*⟩? (| ⟨*S*⟩? ⟨*cp*⟩ ⟨*S*⟩?)+)

⟨*seq*⟩ := (⟨*S*⟩? ⟨*cp*⟩ ⟨*S*⟩? (, ⟨*S*⟩? ⟨*cp*⟩ ⟨*S*⟩?)*)

⟨*cp*⟩ := (⟨*Name*⟩ | ⟨*choice*⟩ | ⟨*seq*⟩) (? | ⋆ | +)?

| models a choice (or), , models a sequence.

?, ⋆, and + can be used to formulate (simple) **cardinality constraints** (default is exactly 1):

symbol		?	⋆	+
constraint	1	0 or 1	≥ 0	≥ 1

Sequences and Sets (1/4)

```
1 <?xml version="1.1"?>
2 <!DOCTYPE persons [
3   <!ELEMENT persons (person*) >
4   <!ELEMENT person (fn, sn) >
5   <!ELEMENT fn ANY >
6   <!ELEMENT sn ANY >
7 ]>
8 <persons>
9   <person><fn>John</fn><sn>Doe</sn></person>
10  <person><fn>Alice</fn><sn>Meier</sn></person>
11  <person><fn>Bob</fn><sn>Miller</sn></person>
12 </persons>
```

Figure 15: Element with child sequence.

Sequences and Sets (2/4)

```
1 <?xml version="1.1"?>
2 <!DOCTYPE persons [
3   <!ELEMENT persons (person*) >
4   <!ELEMENT person (fn, sn) >
5   <!ELEMENT fn ANY >
6   <!ELEMENT sn ANY >
7 ]>
8 <persons>
9   <person><sn>Doe</sn><fn>John</fn></person>
10  <person><sn>Meier</sn><fn>Alice</fn></person>
11  <person><fn>Bob</fn><sn>Miller</sn></person>
12 </persons>
```

Figure 16: Non-valid document.

Sequences and Sets (3/4)

```
1 <?xml version="1.1"?>
2 <!DOCTYPE persons [
3   <!ELEMENT persons (person*) >
4   <!ELEMENT person (fn | sn)* >
5   <!ELEMENT fn ANY >
6   <!ELEMENT sn ANY >
7 ]>
8 <persons>
9   <person><sn>Doe</sn><fn>John</fn></person>
10  <person><sn>Meier</sn><fn>Alice</fn></person>
11  <person><fn>Bob</fn><sn>von</sn><sn>Miller</sn></person>
12 </persons>
```

Figure 17: Element with child multiset.

Sequences and Sets (4/4)

```
1 <?xml version="1.1"?>
2 <!DOCTYPE persons [
3   <!ELEMENT persons (person*) >
4   <!ELEMENT person ((fn, sn) | (sn, fn)) >
5   <!ELEMENT fn ANY >
6   <!ELEMENT sn ANY >
7 ]>
8 <persons>
9   <person><sn>Doe</sn><fn>John</fn></person>
10  <person><sn>Meier</sn><fn>Alice</fn></person>
11  <person><fn>Bob</fn><sn>Miller</sn></person>
12 </persons>
```

Figure 18: Element with child set.

Element Declaration / mixed content

⟨*mixedSpec*⟩ := `(` ⟨*S*⟩? `#PCDATA` ⟨*S*⟩? (`|` ⟨*S*⟩? ⟨*Name*⟩ ⟨*S*⟩?)* `)*`
| `(` ⟨*S*⟩? `#PCDATA` ⟨*S*⟩? `)`

PCDATA is the historical abbreviation for *parsed character data*.

`#PCDATA` is only allowed in the production rule ⟨*mixedSpec*⟩, i.e., nestings as

```
<!ELEMENT person (#PCDATA | (fn, sn))* >
```

or

```
<!ELEMENT person (name | email | #PCDATA)* >
```

are not well-formed.

Element Declaration / ANY content

```
1 <?xml version="1.1"?>
2 <!DOCTYPE persons [
3   <!ELEMENT persons (person*) >
4   <!ELEMENT person (fn, sn) >
5   <!ELEMENT fn ANY >
6   <!ELEMENT sn ANY >
7 ]>
8 <persons>
9   <person>
10    <fn>John</fn>
11    <sn>
12      <person><fn>Johnny</fn><sn>Doe</sn></person>
13    </sn>
14  </person>
15  <person><fn>Alice</fn><sn>Meier</sn></person>
16  <person><fn>Bob</fn><sn>Miller</sn></person>
17 </persons>
```

Figure 19: Element with ANY contents (valid).

Element Declaration / mixed content

```
<?xml version="1.1"?>
<!DOCTYPE persons [
  <!ELEMENT persons (person*) >
  <!ELEMENT person (fn, sn) >
  <!ELEMENT fn (#PCDATA) >
  <!ELEMENT sn (#PCDATA) >
]>
<persons>
  <person>
    <fn>John</fn>
    <sn>
      <person><fn>Johnny</fn><sn>Doe</sn></person>
    </sn>
  </person>
  <person><fn>Alice</fn><sn>Meier</sn></person>
  <person><fn>Bob</fn><sn>Miller</sn></person>
</persons>
```

Figure 20: Element with mixed / #PCDATA contents (not valid).

Element Declaration / mixed content

```
<?xml version="1.1"?>
<!DOCTYPE article [
  <!ELEMENT article (#PCDATA | h | s)* >
  <!ELEMENT h (#PCDATA | s)* >
  <!ELEMENT s (#PCDATA) >
]>
<article>
  <h>Introduction</h>
  This article aims at giving a <s>new</s> perspective on ...

  <h><s>Related</s> Work</h>
  Miller and Doe 2003 have ...
</article>
```

Figure 21: Element with mixed contents.

Attribute List Declarations

⟨*AttlistDecl*⟩ := <!ATTLIST ⟨*S*⟩ ⟨*Name*⟩
(⟨*S*⟩ ⟨*Name*⟩ ⟨*S*⟩ ⟨*AttType*⟩ ⟨*S*⟩ ⟨*DefaultDecl*⟩)* ⟨*S*⟩? >

⟨*AttType*⟩ := CDATA
| ID | IDREF | IDREFS
| NMTOKEN | NMTOKENS
| (⟨*S*⟩? ⟨*Nmtoken*⟩ ⟨*S*⟩? (| ⟨*S*⟩? ⟨*Nmtoken*⟩ ⟨*S*⟩?)*)
| ENTITY | ENTITIES
| NOTATION ⟨*S*⟩ (⟨*S*⟩? ⟨*Name*⟩ ⟨*S*⟩? (| ⟨*S*⟩? ⟨*Name*⟩ ⟨*S*⟩?)*)

⟨*DefaultDecl*⟩ := #REQUIRED | #IMPLIED | ((#FIXED ⟨*S*⟩)? " ⟨*AttValue*⟩ ")

Attribute type CDATA allows arbitrary character data.

default spec.	constraint	default value
#REQUIRED	must be specified	—
#IMPLIED	can be missing	—
"..."	can be missing	as given
#FIXED "..."	typically missing, but if specified must be default value	as given


```
1 <?xml version="1.1"?>
2 <!DOCTYPE meetings [
3   <!ELEMENT meetings (meeting*) >
4   <!ELEMENT meeting (#PCDATA) >
5   <!ATTLIST meeting
6     date CDATA #REQUIRED
7     room CDATA          "B 26"
8     inst CDATA #FIXED    "ISMLL">
9 ]>
10 <meetings>
11   <meeting date="2009/04/21">XML lecture</meeting>
12   <meeting date="2009/04/27" room="L 057">XML tutorial</meeting>
13 </meetings>
```

Figure 22: Element with three attributes.

```
1 <?xml version="1.1" encoding="UTF-8"?>
2 <meetings>
3   <meeting date="2009/04/21" inst="ISMLL" room="B 26">XML lecture</meeting>
4   <meeting date="2009/04/27" inst="ISMLL" room="L 057">XML tutorial</meeting>
5 </meetings>
```

Figure 23: Parsed document.

Attributes / IDs

attribute type	value constraint
`ID`	• must match production ⟨*Name*⟩. • there are no two elements with the same value of that attribute. • specification of default values is illegal.
`IDREF`	there must be an element with attribute of type `ID` having the same value.
`IDREFS`	a space-separated list of values that are of type `IDREF`.


```
1 <?xml version="1.1"?>
2 <!DOCTYPE books [
3   <!ELEMENT books (book*)>
4   <!ELEMENT book (author+, title, year)>
5   <!ATTLIST book
6     isbn  ID     #REQUIRED
7     cites IDREFS #IMPLIED>
8   <!ELEMENT author (#PCDATA)>
9   <!ELEMENT title (#PCDATA)>
10  <!ELEMENT year (#PCDATA)> ]>
11 <books>
12  <book isbn="isbn-3-89864-222-4" cites="isbn-0-596-00420-6 isbn-1-565-92580-7"
13    <author>Rainer Eckstein</author><author>Silke Eckstein</author>
14    <title>XML und Datenmodellierung</title><year>2004</year></book>
15  <book isbn="isbn-0-596-00420-6">
16    <author>Erik T. Ray</author><title>Learning XML</title><year>2003</year></bo
17  <book isbn="isbn-1-565-92580-7">
18    <author>Norman Walsh and Leonard Muellner</author>
19    <title>DocBook: The Definitive Guide</title><year>1999</year></book>
20 </books>
```

Figure 24: Usage of "ID" and "IDREFS".

Attributes / IDs

```
<?xml version="1.1"?>
<!DOCTYPE books [
  <!ELEMENT books (book | author)*>
  <!ELEMENT book (title, year)>
  <!ATTLIST book
     isbn   ID     #REQUIRED
     cites  IDREFS #IMPLIED
     author IDREFS #REQUIRED>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT year (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
  <!ATTLIST author
     key ID #REQUIRED>
]>
<books>
  <book isbn="isbn-3-89864-222-4"
        author="isbn-0-596-00420-6"
        cites="r.eckstein s.eckstein">
    <title>XML und Datenmodellierung</titl
    <year>2004</year>
  </book>
  <book isbn="isbn-0-596-00420-6"
        author="e.ray">
    <title>Learning XML</title>
    <year>2003</year>
  </book>

  <author key="r.eckstein">
    Rainer Eckstein</author>
  <author key="s.eckstein">
    Silke Eckstein</author>
  <author key="e.ray">
    Erik T. Ray</author>
</books>
```

Figure 25: "IDREF"s can point to *any* "ID".

Attributes / Name tokens and enumerations

Values of attributes of type `NMTOKEN`

- may contain unicode letters, uncode digits, `-`, `.`, or ·,
- contrary to ⟨*Name*⟩s do not have to start with an unicode letter or _,
- contrary to `ID`s and `IDREF`s have not to be unique nor point to anything.

The set of allowed values can be explicitly specified (enumeration).

```
<?xml version="1.1"?>
<!DOCTYPE movies [
  <!ELEMENT movies (movie*)>
  <!ELEMENT movie (title, director)>
  <!ATTLIST movie
     keywords NMTOKENS            #IMPLIED
     rating   (poor|fair|excellent) #IMPLIED>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT director (#PCDATA)> ]>
<movies>
  <movie keywords="alaska gold dance little-tramp"
         rating="excellent">
    <title>The Goldrush</title>
    <director>Charles Chaplin</director></movie>
  <movie keywords="part-talkie capitalism police orphan"
         rating="excellent">
    <title>Modern Times</title>
    <director>Charles Chaplin</director></movie>
</movies>
```

Figure 26: Typical usage of "NMTOKENS" attribute as keywords and of enumerations.

II. XML / 2. XML Document Type Definitions (DTDs)

Unparsed Entities

XML allows the "inclusion"/referencing of non-xml data (**unparsed entities**).

Each such data has to be affiliated with a defined data format (**notation**).

Unparsed entities are included/referenced by **attributes of type "ENTITY"**.

Data formats may also be referenced by **attributes of type "NOTATION"**.

Notation Declarations

Remember: notation ≈ data format.

⟨*NotationDecl*⟩ := <!NOTATION ⟨*S*⟩ ⟨*Name*⟩ ⟨*S*⟩
(⟨*ExternID*⟩ | ⟨*PublicOnlyID*⟩) ⟨*S*⟩? >

⟨*PublicOnlyID*⟩ := PUBLIC ⟨*S*⟩ " ⟨*PublicID*⟩ "

Contrary to external IDs for DTDs and entities, a system identifier may be missing.

Which public and/or system identifiers are associated with which data formats, is application-dependent.
Often URIs to IANA media-types are used (http://www.iana.org/assignments/media-types/).

```
5 <!NOTATION jpg
6     SYSTEM "http://www.iana.org/assignments/media-types/image/jpeg">
```

Figure 27: Notation declaration.

Unparsed Entity Declaration

Unparsed entities are declared using the `NDATA` declaration that specifies the notation of the entity:

⟨*EntityDecl*⟩ := `<!ENTITY` ⟨*S*⟩ ⟨*Name*⟩ ⟨*S*⟩
(. . . | (⟨*ExternID*⟩ ⟨*NDataDecl*⟩?)) ⟨*S*⟩? `>`
| . . .

⟨*NDataDecl*⟩ := ⟨*S*⟩ `NDATA` ⟨*S*⟩ ⟨*Name*⟩

```
5 <!NOTATION jpg
6    SYSTEM "http://www.iana.org/assignments/media-types/image/jpeg">
7 <!ENTITY chaplin SYSTEM "chaplin.jpg" NDATA jpg>
8 <!ENTITY welles SYSTEM "welles.jpg" NDATA jpg>
```

Figure 28: Unparsed entity declaration.

Referencing Unparsed Entities

Unparsed entities **cannot** be referenced using syntax

`&` ⟨*Name*⟩ `;`

(as for parsed entites).

But unparsed entities are included/referenced in XML documents via attributes of type `ENTITY`.

The values of these attributes must be **names of general unparsed entities** (i.e., without leading `&` and trailing `;`).

Notations can also be referenced by attributes of type `NOTATION`.

Unparsed Entities / Example

```
<?xml version="1.1"?>
<!DOCTYPE directors [
  <!ELEMENT directors (director*)>
  <!ELEMENT director (#PCDATA)>
  <!NOTATION jpg
     SYSTEM "http://www.iana.org/assignments/media-types/image/jpeg">
  <!ENTITY chaplin SYSTEM "chaplin.jpg" NDATA jpg>
  <!ENTITY welles SYSTEM "welles.jpg" NDATA jpg>
  <!ATTLIST director
     photo ENTITY #IMPLIED>
]>
<directors>
  <director photo="chaplin">Charles Chaplin</director>
  <director photo="welles">Orson Welles</director>
</directors>
```

Figure 29: Image data referenced by unparsed entities.

Unparsed Entities / Example

```
<?xml version="1.1"?>
<!DOCTYPE directors [
  <!ELEMENT directors (director*)>
  <!ELEMENT director (#PCDATA)>
  <!NOTATION jpg
     SYSTEM "http://www.iana.org/assignments/media-types/image/jpeg">
  <!NOTATION gif
     SYSTEM "http://www.iana.org/assignments/media-types/image/gif">
  <!ENTITY chaplin SYSTEM "chaplin.jpg" NDATA jpg>
  <!ENTITY welles SYSTEM "welles.gif" NDATA gif>
  <!ATTLIST director
     photo ENTITY #IMPLIED
     fmt   NOTATION (gif | jpg) #IMPLIED>
]>
<directors>
  <director photo="chaplin" fmt="jpg">Charles Chaplin</director>
  <director photo="welles" fmt="gif">Orson Welles</director>
</directors>
```

Figure 30: Referencing unparsed entities and notations.

II. XML / 2. XML Document Type Definitions (DTDs)

Parameter entities

Parameter entities are entities for usage in DTDs
(not in the "body" of the XML document).

⟨*EntityDecl*⟩ := ...
| <!ENTITY ⟨*S*⟩ % ⟨*S*⟩ ⟨*Name*⟩ ⟨*S*⟩
(" ⟨*EntityValue*⟩ " | ⟨*ExternID*⟩) ⟨*S*⟩? >

Parameter entities are referenced via

⟨*PEReference*⟩ := % ⟨*Name*⟩ ;

Parameter entities in internal DTDs

In internal DTDs parameter entities can only be used to include external parts of the DTD.

```
<!ELEMENT strong ANY>
<!ELEMENT em ANY>
```

Figure 31: DTD (fragment) `textelements.dtd`.

```
<?xml version="1.1"?>
<!DOCTYPE report [
  <!ELEMENT report (#PCDATA | heading | strong | em)* >
  <!ENTITY % textelements SYSTEM "textelements.dtd" >
  %textelements;
  <!ELEMENT heading (#PCDATA) >
]>
<report>
  <heading>Dates</heading>
  <em>Firm</em> deadline is on <strong>Saturday</strong>.
</report>
```

Figure 32: Parameter entity in internal DTD.

internal/external PE vs. PE in internal/external DTD

Do not confuse

internal parameter entities
vs.
external parameter entities

internal PE: value given between `"..."` in DTD.

external PE: value is contents of a resource referenced via `SYSTEM` or `PUBLIC`.

with

parameter entities in internal DTD
vs.
parameter entities in external DTD

PE in internal DTD: declaration of PE is in XML document, in <!DOCTYPE...> declaration between `[ ... ]`.

PE in external DTD: declaration of PE is in DTD referenced in <!DOCTYPE...> declaration by `SYSTEM` or `PUBLIC`.

Parameter entities in external DTDs

In external DTDs parameter entities can be used almost everywhere and contain

- any part of an attribute default value or
- any part of a declaration that is "properly nested"

```
1 <!ENTITY % textatt "strong | em" >
2 <!ELEMENT page (#PCDATA | heading | %textatt;)* >
3 <!ELEMENT heading ANY>
4 <!ELEMENT strong ANY>
5
6 <!ENTITY % eem "em (#PCDATA)" >
7 <!ELEMENT %eem;>
```

Figure 33: External DTD with advanced usage of parameter entities.

Conditional DTD sections

⟨*ExternDoctypeDecl*⟩ := (InternDoctypeDecl | ConditionalSect)*

⟨*ConditionalSect*⟩ := `<![` ⟨*S*⟩? `INCLUDE` ⟨*S*⟩? `[` ⟨*ExternDoctypeDecl*⟩ `]]>`
| `<![` ⟨*S*⟩? `IGNORE` ⟨*S*⟩? `[` ⟨*IgnoredContents*⟩ `]]>`

`<![INCLUDE[` includes declarations up to next `]]>`,
`<![IGNORE[` ignores declarations up to next `]]>`.

⟨*IgnoredContents*⟩ is any character data not containing `]]>` (c.f. CDATA sections).

```
<!ELEMENT page (#PCDATA | heading | strong | em)* >
<!ELEMENT strong ANY>
<!ELEMENT em ANY>
<![ %plainHeadings; [
  <!ELEMENT heading (#PCDATA) >
]]>
<![ %fancyHeadings; [
  <!ELEMENT heading (#PCDATA | strong | em)* >
]]>
```

Figure 34: DTD `page.dtd` with conditional section.

```
<?xml version="1.1"?>
<!DOCTYPE page SYSTEM "page.dtd" [
  <!ENTITY % plainHeadings "IGNORE" >
  <!ENTITY % fancyHeadings "INCLUDE" >
]>
<page>
  <heading>The <strong>very</strong> beginning</heading>
  ...
</page>
```

Figure 35: XML document using DTD `page.dtd`.

Entity Types

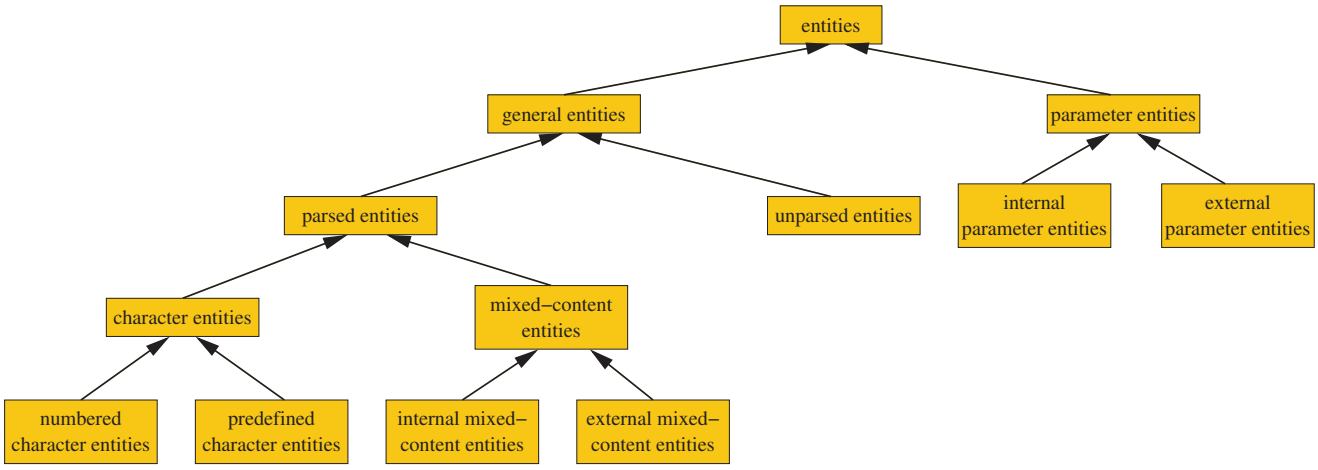

Figure 36: Types of entities.

II. XML / 2. XML Document Type Definitions (DTDs)

Problems with System Identifiers

System identifiers (specified with `SYSTEM`, e.g., for DTDs or entities) may be

- absolute URIs as

  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"

  Advantage: identity of DTD is guaranteed.

  Drawback: DTD is fetched for every parse. Not possible offline.

- relative URIs as

  "DTD/xhtml11.dtd"

  Advantage: DTD is local. Working offline is possible.

  Drawback: DTD has to be reproduced with every project.

Public Identifiers

Public identifiers (specified with `PUBLIC`)

- identify a DTD uniquely, e.g., for XHTML 1.1
  "-//W3C//DTD XHTML 1.1//EN"
  and
- are mapped to URIs by a host-/project-dependent central **catalog**.

XMLCatalog [Wal01] is one implementation of such a catalog.

Public identifiers themselves are not URIs.
But the namespace of public identifiers is mapped to URI space by `urn:publicid`, e.g.,

"urn:publicid:-:W3C:DTD+XHTML+1.1:EN"

XMLCatalog / example (1/2)

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3           "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4 <html>
5   <head>
6     <title>Virtual Library</title>
7   </head>
8   <body>
9     <p>Moved to <a href="http://vlib.org/">vlib.org</a>.</p>
10  </body>
11 </html>
12
```

Figure 37: XHTML document with public DTD identifier.

XMLCatalog / example (2/2)

```
<?xml version="1.1"?>
<!DOCTYPE catalog
  PUBLIC "-//OASIS//DTD Entity Resolution XML Catalog V1.0//EN"
  "http://www.oasis-open.org/committees/entity/release/1.0/catalog.dtd">
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
       prefer="public">
 <public publicId="-//W3C//DTD XHTML 1.1//EN"
        uri="file:///usr/share/sgml/xhtml/xhtml-1.1/DTD/xhtml11-flat.dtd"/>
</catalog>
```

Figure 38: XML catalog for XHTML 1.1 DTD (assumes, that xhtml-1.1 DTD is at given URI locally (true, e.g., for SuSE Linux).

The xerces sample-parser `sax.Writer` has to be modified to take into account catalogs

- compare `EntityResolvingWriter.java` with `sax/Writer.java`.
- run with

```
xercesER -v -l catalog.xml example.xhtml
```


References

[Wal01] Norman Walsh. Xml catalogs. Technical report, OASIS Committee Specification, 6 Aug 2001.