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

<?xml version="1.1"?>
<books>
  <book year="2004">
    <authors>
      <author><fn>Rainer</fn><sn>Eckstein</sn></author>
      <author><fn>Silke</fn><sn>Eckstein</sn></author>
    </authors>
    <title>XML und Datenmodellierung</title>
    <isbn>3-89864-222-4</isbn>
  </book>
</books>

Figure 1: A list of books.

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

Figure 2: Another list of books.
II. XML / 2. XML Document Type Definitions (DTDs)

1. Mixed Content Constants (Parsed Entities)

2. Constraining Document Structure

3. Referencing Non-XML Data (Unparsed Entities)

4. DTD Modularization (Parameter Entities)

5. Entity Management (XMLCatalog)

Document Type Declaration

\[
\langle \text{DoctypeDecl} \rangle := \langle !\text{DOCTYPE} \rangle \langle S \rangle \langle \text{Name} \rangle \\
(\langle S \rangle \langle \text{ExternID} \rangle )? \\
(\langle S \rangle ? [\langle \text{InternDoctypeDecl} \rangle ]? \langle S \rangle ? > \\
\langle \text{InternDoctypeDecl} \rangle := (\langle \text{EntityDecl} \rangle \\
| \langle \text{ElementDecl} \rangle | \langle \text{AttlistDecl} \rangle \\
| \langle \text{NotationDecl} \rangle \\
| \langle \text{PEReference} \rangle \\
| \langle \text{PI} \rangle | \langle \text{Comment} \rangle | \langle S \rangle )^*
\]

\langle \text{Name} \rangle \text{ 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

\[
\langle \text{EntityDecl} \rangle := \langle !\text{ENTITY} \rangle \langle S \rangle \langle \text{Name} \rangle \langle S \rangle

( " \langle \text{EntityValue} \rangle " | ( \langle \text{ExternID} \rangle \langle N\text{DataDecl} \rangle ? ) \langle S \rangle ? ) >

\| \langle !\text{ENTITY} \rangle \langle S \rangle % \langle S \rangle \langle \text{Name} \rangle \langle S \rangle

( " \langle \text{EntityValue} \rangle " | \langle \text{ExternID} \rangle ) \langle S \rangle ? >
\]

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. \( \langle N\text{DataDecl} \rangle \) are called **parsed entities**, otherwise **unparsed entities** (see section 3).

Usage of general entities

Parsed entities provide a mechanism for mixed content constants.

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

Figure 3: Definition and usage of a XML entity.

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

Figure 4: Document with resolved entities as seen after 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.

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

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

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

Figure 7: Illegal entity usage: (general) entities can only be used in character data.
XML and Semantic Web Technologies / 1. Mixed Content Constants (Parsed Entities)

External General Entities

\[
\langle \text{ExternID} \rangle := \text{SYSTEM} \langle S \rangle \ " \langle \text{URI} \rangle \ " \\
| \text{PUBLIC} \langle S \rangle \ " \langle \text{PublicID} \rangle \ " \langle S \rangle \ " \langle \text{URI} \rangle \ "
\]

All external references must have a **system identifier** \( \langle \text{URI} \rangle \).

The system identifier \( \langle \text{URI} \rangle \) points to a resource that contains the value of the entity.

- \( \langle \text{URI} \rangle \) may be relative to the location of its context,
- \( \langle \text{URI} \rangle \) may not contain a fragment identifier.

The **public identifier** \( \langle \text{PublicID} \rangle \) is a key that can be resolved by a (system-specific) catalog to an URI (see section 5).

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

```xml
<?xml version="1.1"?>
<!DOCTYPE article [
  <!ENTITY intro SYSTEM "intro.xml">
  <!ENTITY results SYSTEM "results.xml">
  <!ENTITY outlook SYSTEM "outlook.xml">
]> 
<article>
  &intro; 
  &results; 
  &outlook; 
</article>
```

Figure 8: Master document master.xml

```xml
<h>Introduction</h>
Among the most urgent questions ...
Figure 9: Included document intro.xml
<h>Results</h>
Based on the idea of the last section ...
Figure 10: Included document results.xml
<h>Outlook</h>
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).

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

Figure 12: XML document with external DTD.

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

Figure 13: External DTD `me.dtd`.

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

1. Mixed Content Constants (Parsed Entities)

2. Constraining Document Structure

3. Referencing Non-XML Data (Unparsed Entities)

4. DTD Modularization (Parameter Entities)

5. Entity Management (XMLCatalog)
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).


Valid: contents and parameters of all elements match document type.
Valid Document

```
<?xml version="1.1"?>
<!DOCTYPE page [ 
  <!ELEMENT page ANY> 
]> 
</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⟩ ) ( ? | * | + )?
```

<table>
<thead>
<tr>
<th>symbol constraint</th>
<th>1</th>
<th>0 or 1</th>
<th>≥ 0</th>
<th>≥ 1</th>
</tr>
</thead>
</table>

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

? , * , and + can be used to formulate (simple) **cardinality constraints** (default is exactly 1):
<?xml version="1.1"?>
<!DOCTYPE persons [
<!ELEMENT persons (person*) >
<!ELEMENT person (fn, sn) >
<!ELEMENT fn ANY >
<!ELEMENT sn ANY >
]>}
<persons>
<person><fn>John</fn><sn>Doe</sn></person>
<person><fn>Alice</fn><sn>Meier</sn></person>
<person><fn>Bob</fn><sn>Miller</sn></person>
</persons>

Figure 15: Element with child sequence.

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

Figure 16: Non-valid document.
<?xml version="1.1"?>
<!DOCTYPE persons [
<!ELEMENT persons (person*) >
<!ELEMENT person (fn | sn)* >
<!ELEMENT fn ANY >
<!ELEMENT sn ANY >
]>
<persons>
<person><sn>Doe</sn><fn>John</fn></person>
<person><sn>Meier</sn><fn>Alice</fn></person>
<person><fn>Bob</fn><sn>von</sn><sn>Miller</sn></person>
</persons>

Figure 17: Element with child multiset.

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

Figure 18: Element with child set.
Element Declaration / mixed content

\[
\langle \text{mixedSpec} \rangle := \begin{cases} 
( \langle S \rangle? \ #\text{PCDATA} \ (S)? \ ( | \langle S \rangle? \ (\langle Name\rangle \ (S)?)^* )^* \\
| ( \langle S \rangle? \ #\text{PCDATA} \ (S)?)^* 
\end{cases}
\]

PCDATA is the historical abbreviation for \textit{parsed character data}.

#PCDATA is only allowed in the production rule \langle mixedSpec \rangle, i.e., nestings as

\[
<!ELEMENT person (#PCDATA | (fn, sn))* >
\]

or

\[
<!ELEMENT person (name | email | #PCDATA)* >
\]

are not well-formed.

Figure 19: Element with ANY contents (valid).
<?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>Johnny</sn><fn>Doe</fn>
</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).

<?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

\[ \langle \text{AttlistDecl} \rangle := \langle \text{<!ATTLIST} \langle S \rangle \langle \text{Name} \rangle \rangle \langle \langle S \rangle \langle \text{Name} \rangle \langle S \rangle \langle \text{AttType} \rangle \langle S \rangle \langle \text{DefaultDecl} \rangle \rangle \ast \langle S \rangle \rangle ? \langle S \rangle \rangle \]

\[ \langle \text{AttType} \rangle := \text{CDATA} \quad | \quad \text{ID} \mid \text{IDREF} \mid \text{IDREFS} \mid \text{NMTOKEN} \mid \text{NMTOKENS} \mid \langle \langle S \rangle \rangle ? \langle \text{Nmtoken} \rangle \langle S \rangle ? \rangle ( | \langle \langle S \rangle \rangle ? \langle \text{Nmtoken} \rangle \langle S \rangle ? \rangle ) \ast ) \mid \text{ENTITY} \mid \text{ENTITIES} \mid \text{NOTATION} \langle S \rangle ( \langle S \rangle ? \langle \text{Name} \rangle \langle S \rangle ? \rangle ( | \langle S \rangle ? \langle \text{Name} \rangle \langle S \rangle ? \rangle ) \ast ) \]

\[ \langle \text{DefaultDecl} \rangle := \#\text{REQUIRED} \mid \#\text{IMPLIED} \mid \langle \langle \#\text{FIXED} \langle S \rangle ? \rangle ? \langle \text{AttValue} \rangle \langle S \rangle ? \rangle \rangle \]

Attribute type \text{CDATA} allows arbitrary character data.

<table>
<thead>
<tr>
<th>default spec.</th>
<th>constraint</th>
<th>default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#REQUIRED</td>
<td>must be specified</td>
<td>—</td>
</tr>
<tr>
<td>#IMPLIED</td>
<td>can be missing</td>
<td>—</td>
</tr>
<tr>
<td>&quot;...&quot;</td>
<td>can be missing</td>
<td>as given</td>
</tr>
<tr>
<td>#FIXED &quot;...&quot;</td>
<td>typically missing, but if specified must be default value</td>
<td>as given</td>
</tr>
</tbody>
</table>

Figure 22: Element with three attributes.

```
<?xml version="1.1"?>
<DOCTYPE meetings [
  <!ELEMENT meetings (meeting*) >
  <!ELEMENT meeting (#PCDATA) >
  <!ATTLIST meeting
date CDATA #REQUIRED
room CDATA "B 26"
inst CDATA #FIXED "ISMLL">
]

<meetings>
  <meeting date="2009/04/21">XML lecture</meeting>
  <meeting date="2009/04/27" room="L 057">XML tutorial</meeting>
</meetings>
```

Figure 23: Parsed document.
### Attributes / IDs

<table>
<thead>
<tr>
<th>attribute type</th>
<th>value constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>- must match production (\langle Name\rangle).</td>
</tr>
<tr>
<td></td>
<td>- there are no two elements with the same value of that attribute.</td>
</tr>
<tr>
<td></td>
<td>- specification of default values is illegal.</td>
</tr>
<tr>
<td>IDREF</td>
<td>there must be an element with attribute of type ID having the same value.</td>
</tr>
<tr>
<td>IDREFS</td>
<td>a space-separated list of values that are of type IDREF.</td>
</tr>
</tbody>
</table>

```xml
<?xml version="1.1"?>
<!DOCTYPE books [
<!ELEMENT books (book*)>
<!ELEMENT book (author+, title, year)>
<!ATTTLIST book
  isbn ID #REQUIRED
  cites IDREFS #IMPLIED>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT year (#PCDATA) > ]>

<books>
  <book isbn="isbn-3-89864-222-4" cites="isbn-0-596-00420-6 isbn-1-565-92580-7"
    <author>Rainer Eckstein</author><author>Silke Eckstein</author>
    <title>XML und Datenmodellierung</title><year>2004</year></book>
  <book isbn="isbn-0-596-00420-6"
  <book isbn="isbn-1-565-92580-7"
    <author>Norman Walsh and Leonard Muellner</author>
</books>
```

**Figure 24:** Usage of "ID" and "IDREFS".
<?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</title>
    <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".

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 IDs and IDREFs 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)

1. Mixed Content Constants (Parsed Entities)

2. Constraining Document Structure

3. Referencing Non-XML Data (Unparsed Entities)

4. DTD Modularization (Parameter Entities)

5. Entity Management (XMLCatalog)
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".

Remember: notation ≈ data format.

\[
\langle NotationDecl \rangle := \langle !NOTATION \langle S \rangle \langle Name \rangle \langle S \rangle \rangle \\
( \langle ExternID \rangle | \langle PublicOnlyID \rangle ) \langle S \rangle ? > \\
\langle PublicOnlyID \rangle := \text{PUBLIC} \langle S \rangle " \langle PublicID \rangle "
\]

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/).

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

Figure 27: Notation declaration.
Unparsed entities are declared using the **NDATA** declaration that specifies the notation of the entity:

```
ENTITY DECL ::= <!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.

Unparsed entities **cannot** be referenced using syntax

```
& ⟨Name⟩ ;
```

(as for parsed entities).

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**.
<?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.gif" NDATA gif>
<!ATTLIST director photo ENTITY #IMPLIED fmt NOTATION (gif | jpg) #IMPLIED>
]>
<directors>
  <director photo="chaplin">Charles Chaplin</director>
  <director photo="welles">Orson Welles</director>
</directors>

Figure 29: Image data referenced by unparsed entities.

<?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>
]>
<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)

1. Mixed Content Constants (Parsed Entities)

2. Constraining Document Structure

3. Referencing Non-XML Data (Unparsed Entities)

4. DTD Modularization (Parameter Entities)

5. Entity Management (XMLCatalog)

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

\[
\langle EntityDecl \rangle := \ldots \\
\mid \langle !ENTITY \langle S \rangle \% \langle S \rangle \langle Name \rangle \langle S \rangle \rangle \\
\quad ("\langle EntityValue \rangle " \mid \langle ExternID \rangle ) \langle S \rangle ? \rangle
\]

Parameter entities are referenced via

\[
\langle PEReference \rangle := \% \langle Name \rangle ;
\]
Parameter entities in internal DTDs

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

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

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

```xml
<?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.

Do not confuse

```
<table>
<thead>
<tr>
<th>Internal parameter entities</th>
<th>External parameter entities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Internal PE:</strong> value given between &quot;...&quot; in DTD.</td>
<td><strong>External PE:</strong> value is contents of a resource referenced via <code>SYSTEM</code> or <code>PUBLIC</code>.</td>
</tr>
</tbody>
</table>
```

**PE in internal DTD:** declaration of PE 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`. 
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"

```
<!ENTITY % textatt "strong | em" >
<!ELEMENT page (#PCDATA | heading | %textatt;)* >
<!ELEMENT heading ANY>
<!ELEMENT strong ANY>

<!ENTITY % eem "em (#PCDATA)" >
<!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)

1. Mixed Content Constants (Parsed Entities)

2. Constraining Document Structure

3. Referencing Non-XML Data (Unparsed Entities)

4. DTD Modularization (Parameter Entities)

5. Entity Management (XMLCatalog)

Problems with System Identifiers

System identifiers (specified with \textit{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 \texttt{PUBLIC})

- identify a DTD uniquely, e.g., for XHTML 1.1
  
  
  
  
- are mapped to URIs by a host-/project-dependent central \texttt{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 \texttt{urn:publicid}, e.g.,

\verb|"urn:publicid:-:W3C:DTD+XHTML+1.1:EN"|

---

Figure 37: XHTML document with public DTD identifier.
<?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

---

Summary

• Document type definitions (DTDs) are a simple method to describe a document type.

• DTDs describe the elements allowed in a document.

• DTDs describe for each element its attributes and their types.

• DTDs describe for each element its contents as (evtl. nested) choices and sequences.

• DTDs provide some elementary support for DTD modularization by using parameter entities.

• Although DTDs are not the schema language of choice to use nowadays, they still occur pervasively and you must know about them.
References