

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>

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany,
Course on XML and Semantic Web Technologies, summer term 2012

1/46

XML and Semantic Web Technologies

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 Mueller</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)

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

$$\langle \textit{DoctypeDecl} \rangle := < !\textit{DOCTYPE} \langle \textit{S} \rangle \langle \textit{Name} \rangle \\ (\langle \textit{S} \rangle \langle \textit{ExternID} \rangle)? \\ (\langle \textit{S} \rangle? [\langle \textit{InternDoctypeDecl} \rangle])? \langle \textit{S} \rangle? >$$

$$\langle \textit{InternDoctypeDecl} \rangle := (\langle \textit{EntityDecl} \rangle \\ | \langle \textit{ElementDecl} \rangle | \langle \textit{AttlistDecl} \rangle \\ | \langle \textit{NotationDecl} \rangle \\ | \langle \textit{PEReference} \rangle \\ | \langle \textit{PI} \rangle | \langle \textit{Comment} \rangle | \langle \textit{S} \rangle)^*$$

$\langle \textit{Name} \rangle$ 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

$$\langle \text{EntityDecl} \rangle := \langle !\text{ENTITY} \langle S \rangle \langle \text{Name} \rangle \langle S \rangle \langle S \rangle \langle \text{EntityValue} \rangle \text{"} | (\langle \text{ExternID} \rangle \langle \text{NDataDecl} \rangle ?) \rangle \langle S \rangle ? \rangle$$

$$| \langle !\text{ENTITY} \langle S \rangle \% \langle S \rangle \langle \text{Name} \rangle \langle S \rangle \langle S \rangle \langle \text{EntityValue} \rangle \text{"} | \langle \text{ExternID} \rangle \rangle \langle S \rangle ? \rangle$$

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. $\langle \text{NDataDecl} \rangle$ 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.

```

1#!/bin/bash
2XERCES_HOME=/opt/xml/xerces
3java -cp $XERCES_HOME/xercesSamples.jar:$XERCES_HOME/xercesImpl.jar \
4  sax.Writer $@
5echo

```

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).

```

1<?xml version="1.1"?>
2<!DOCTYPE page [
3  <!ENTITY plz "Please</s>">
4]>
5<page>
6  <s>&plz;, call me soon.
7</page>

```

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

```

1<?xml version="1.1"?>
2<!DOCTYPE page [
3  <!ENTITY own "owner='me'" >
4]>
5<page &own; >
6  Hello !
7</page>

```

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

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 / 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 ...
3
4 Figure 9: Included document intro.xml
5
6 <h>Results</h>
7 Based on the idea of the last section ...
8
9 Figure 10: Included document results.xml
10
11 <h>Outlook</h>
12 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)

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**).

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

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

Element Declaration

$$\langle \textit{ElementDecl} \rangle := \langle \textit{!ELEMENT} \rangle \langle \textit{S} \rangle \langle \textit{Name} \rangle \langle \textit{S} \rangle \langle \textit{contentSpec} \rangle \langle \textit{S} \rangle? \rangle$$

$$\langle \textit{contentSpec} \rangle := \textit{EMPTY} \mid \textit{ANY} \mid \langle \textit{childrenSpec} \rangle \mid \langle \textit{mixedSpec} \rangle$$

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

$$\begin{aligned}
 \langle childrenSpec \rangle &:= (\langle choice \rangle \mid \langle seq \rangle) (? \mid * \mid +) ? \\
 \langle choice \rangle &:= (\langle S \rangle ? \langle cp \rangle \langle S \rangle ? (\mid \langle S \rangle ? \langle cp \rangle \langle S \rangle ?) +) \\
 \langle seq \rangle &:= (\langle S \rangle ? \langle cp \rangle \langle S \rangle ? (, \langle S \rangle ? \langle cp \rangle \langle S \rangle ?) *) \\
 \langle cp \rangle &:= (\langle Name \rangle \mid \langle choice \rangle \mid \langle seq \rangle) (? \mid * \mid +) ?
 \end{aligned}$$

| 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

$$\langle mixedSpec \rangle := (\langle S \rangle? \#PCDATA \langle S \rangle? (| \langle S \rangle? \langle Name \rangle \langle S \rangle?)^*)^* \\ | (\langle S \rangle? \#PCDATA \langle S \rangle?)$$

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

```

1 <?xml version="1.1"?>
2 <!DOCTYPE persons [
3   <!ELEMENT persons (person*) >
4   <!ELEMENT person (fn, sn) >
5   <!ELEMENT fn (#PCDATA) >
6   <!ELEMENT sn (#PCDATA) >
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 20: Element with mixed / #PCDATA contents (not valid).

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany,
Course on XML and Semantic Web Technologies, summer term 2012

20/46

Element Declaration / mixed content

```

1 <?xml version="1.1"?>
2 <!DOCTYPE article [
3   <!ELEMENT article (#PCDATA | h | s)* >
4   <!ELEMENT h (#PCDATA | s)* >
5   <!ELEMENT s (#PCDATA) >
6 ]>
7 <article>
8   <h>Introduction</h>
9   This article aims at giving a <s>new</s> perspective on ...
10
11  <h><s>Related</s> Work</h>
12  Miller and Doe 2003 have ...
13 </article>

```

Figure 21: Element with mixed contents.

Attribute List Declarations

$$\langle \text{AttlistDecl} \rangle := \langle !\text{ATTLIST} \langle S \rangle \langle \text{Name} \rangle (\langle S \rangle \langle \text{Name} \rangle \langle S \rangle \langle \text{AttType} \rangle \langle S \rangle \langle \text{DefaultDecl} \rangle)^* \langle S \rangle? \rangle$$

$$\langle \text{AttType} \rangle := \text{CDATA} \\ | \text{ID} | \text{IDREF} | \text{IDREFS} \\ | \text{NMTOKEN} | \text{NMTOKENS} \\ | (\langle S \rangle? \langle \text{Nmtoken} \rangle \langle S \rangle? (| \langle S \rangle? \langle \text{Nmtoken} \rangle \langle S \rangle?)^*) \\ | \text{ENTITY} | \text{ENTITIES} \\ | \text{NOTATION} \langle S \rangle (\langle S \rangle? \langle \text{Name} \rangle \langle S \rangle? (| \langle S \rangle? \langle \text{Name} \rangle \langle S \rangle?)^*)$$

$$\langle \text{DefaultDecl} \rangle := \# \text{REQUIRED} | \# \text{IMPLIED} | ((\# \text{FIXED} \langle S \rangle)? " \langle \text{AttValue} \rangle ")$$

Attribute type `CDATA` allows arbitrary character data.

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

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany, Course on XML and Semantic Web Technologies, summer term 2012

22/46

XML and Semantic Web Technologies / 2. Constraining Document Structure

```

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.

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany, Course on XML and Semantic Web Technologies, summer term 2012

23/46

Attributes / IDs

attribute type	value constraint
ID	<ul style="list-style-type: none"> • must match production $\langle Name \rangle$. • 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.

XML and Semantic Web Technologies / 2. Constraining Document Structure

```

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

```

1 <?xml version="1.1"?>
2 <!DOCTYPE books [
3   <!ELEMENT books (book | author)*>
4   <!ELEMENT book (title, year)>
5   <!ATTLIST book
6     isbn ID #REQUIRED
7     cites IDREFS #IMPLIED
8     author IDREFS #REQUIRED>
9   <!ELEMENT title (#PCDATA)>
10  <!ELEMENT year (#PCDATA)>
11  <!ELEMENT author (#PCDATA)>
12  <!ATTLIST author
13    key ID #REQUIRED>
14 ]>
15 <books>
16   <book isbn="isbn-3-89864-222-4"
17     author="isbn-0-596-00420-6"
18     cites="r.eckstein s.eckstein">
19     <title>XML und Datenmodellierung</title>
20     <year>2004</year>
21   </book>
22   <book isbn="isbn-0-596-00420-6"
23     author="e.ray">
24     <title>Learning XML</title>
25     <year>2003</year>
26   </book>
27
28   <author key="r.eckstein">
29     Rainer Eckstein</author>
30   <author key="s.eckstein">
31     Silke Eckstein</author>
32   <author key="e.ray">
33     Erik T. Ray</author>
34 </books>

```

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

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany,
Course on XML and Semantic Web Technologies, summer term 2012

26/46

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 `IDs` and `IDREFs` have not to be unique nor point to anything.

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

```
1 <?xml version="1.1"?>
2 <!DOCTYPE movies [
3   <!ELEMENT movies (movie*)>
4   <!ELEMENT movie (title, director)>
5   <!ATTLIST movie
6     keywords NMTOKENS          #IMPLIED
7     rating (poor|fair|excellent) #IMPLIED>
8   <!ELEMENT title (#PCDATA)>
9   <!ELEMENT director (#PCDATA)> ]>
10 <movies>
11   <movie keywords="alaska gold dance little-tramp"
12     rating="excellent">
13     <title>The Goldrush</title>
14     <director>Charles Chaplin</director></movie>
15   <movie keywords="part-talkie capitalism police orphan"
16     rating="excellent">
17     <title>Modern Times</title>
18     <director>Charles Chaplin</director></movie>
19 </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)

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 \approx data format.

$$\langle \text{NotationDecl} \rangle := < !\text{NOTATION} \langle S \rangle \langle \text{Name} \rangle \langle S \rangle \\ (\langle \text{ExternID} \rangle | \langle \text{PublicOnlyID} \rangle) \langle S \rangle ? >$$

$$\langle \text{PublicOnlyID} \rangle := \text{PUBLIC} \langle S \rangle " \langle \text{PublicID} \rangle "$$

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/>).

```

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:

$$\langle \textit{EntityDecl} \rangle := \langle \textit{!ENTITY} \rangle \langle \textit{S} \rangle \langle \textit{Name} \rangle \langle \textit{S} \rangle$$

$$(\dots | (\langle \textit{ExternID} \rangle \langle \textit{NDataDecl} \rangle ?)) \langle \textit{S} \rangle ? >$$

$$| \dots$$

$$\langle \textit{NDataDecl} \rangle := \langle \textit{S} \rangle \textit{NDATA} \langle \textit{S} \rangle \langle \textit{Name} \rangle$$

```

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

$$\& \langle \textit{Name} \rangle ;$$

(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

```

1 <?xml version="1.1"?>
2 <!DOCTYPE directors [
3   <!ELEMENT directors (director*)>
4   <!ELEMENT director (#PCDATA)>
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>
9   <!ATTLIST director
10    photo ENTITY #IMPLIED>
11 ]>
12 <directors>
13   <director photo="chaplin">Charles Chaplin</director>
14   <director photo="welles">Orson Welles</director>
15 </directors>

```

Figure 29: Image data referenced by unparsed entities.

Unparsed Entities / Example

```

1 <?xml version="1.1"?>
2 <!DOCTYPE directors [
3   <!ELEMENT directors (director*)>
4   <!ELEMENT director (#PCDATA)>
5   <!NOTATION jpg
6     SYSTEM "http://www.iana.org/assignments/media-types/image/jpeg">
7   <!NOTATION gif
8     SYSTEM "http://www.iana.org/assignments/media-types/image/gif">
9   <!ENTITY chaplin SYSTEM "chaplin.jpg" NDATA jpg>
10  <!ENTITY welles SYSTEM "welles.gif" NDATA gif>
11  <!ATTLIST director
12    photo ENTITY #IMPLIED
13    fmt NOTATION (gif | jpg) #IMPLIED>
14 ]>
15 <directors>
16   <director photo="chaplin" fmt="jpg">Charles Chaplin</director>
17   <director photo="welles" fmt="gif">Orson Welles</director>
18 </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

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

$$\langle \textit{EntityDecl} \rangle := \dots$$

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

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

Parameter entities are referenced via

$$\langle \textit{PEReference} \rangle := \% \langle \textit{Name} \rangle ;$$

Parameter entities in internal DTDs

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

```

1 <!ELEMENT strong ANY>
2 <!ELEMENT em ANY>

```

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

```

1 <?xml version="1.1"?>
2 <!DOCTYPE report [
3   <!ELEMENT report (#PCDATA | heading | strong | em)* >
4   <!ENTITY % textelements SYSTEM "textelements.dtd" >
5   %textelements;
6   <!ELEMENT heading (#PCDATA) >
7 ]>
8 <report>
9   <heading>Dates</heading>
10  <em>Firm</em> deadline is on <strong>Saturday</strong>.
11 </report>

```

Figure 32: Parameter entity in internal DTD.

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany,
Course on XML and Semantic Web Technologies, summer term 2012

36/46

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

$$\langle \textit{ExternDoctypeDecl} \rangle := (\textit{InternDoctypeDecl} \mid \textit{ConditionalSect})^*$$

$$\langle \textit{ConditionalSect} \rangle := \langle ! [\langle \textit{S} \rangle ? \textit{INCLUDE} \langle \textit{S} \rangle ? [\langle \textit{ExternDoctypeDecl} \rangle]] \rangle$$

$$\mid \langle ! [\langle \textit{S} \rangle ? \textit{IGNORE} \langle \textit{S} \rangle ? [\langle \textit{IgnoredContents} \rangle]] \rangle$$

$\langle ! [\textit{INCLUDE} [\textit{includes declarations up to next }]] \rangle$,

$\langle ! [\textit{IGNORE} [\textit{ignores declarations up to next }]] \rangle$.

$\langle \textit{IgnoredContents} \rangle$ is any character data not containing $]] \rangle$ (c.f. CDATA sections).

```

1 <!ELEMENT page (#PCDATA | heading | strong | em)* >
2 <!ELEMENT strong ANY>
3 <!ELEMENT em ANY>
4 <![ %plainHeadings; [
5   <!ELEMENT heading (#PCDATA) >
6 ]]>
7 <![ %fancyHeadings; [
8   <!ELEMENT heading (#PCDATA | strong | em)* >
9 ]]>

```

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

```

1 <?xml version="1.1"?>
2 <!DOCTYPE page SYSTEM "page.dtd" [
3   <!ENTITY % plainHeadings "IGNORE" >
4   <!ENTITY % fancyHeadings "INCLUDE" >
5 ]>
6 <page>
7   <heading>The <strong>very</strong> beginning</heading>
8   ...
9 </page>

```

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

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany,
Course on XML and Semantic Web Technologies, summer term 2012

40/46

Entity Types

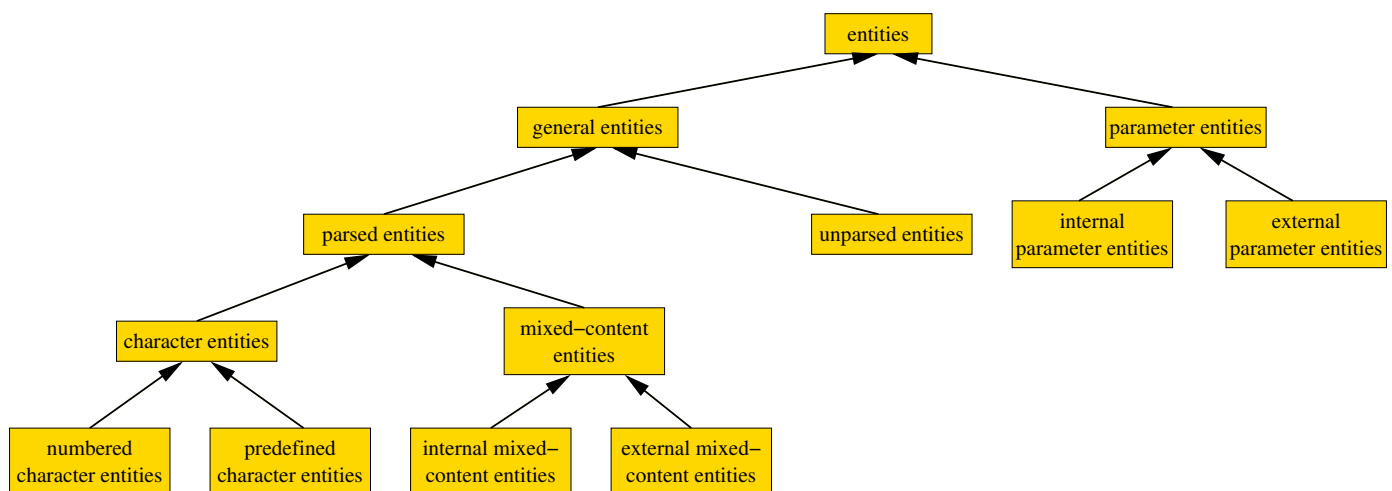


Figure 36: Types of entities.

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 `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>

```

Figure 37: XHTML document with public DTD identifier.

XMLCatalog / example (2/2)

```
1 <?xml version="1.1"?>
2 <!DOCTYPE catalog
3   PUBLIC "-//OASIS//DTD Entity Resolution XML Catalog V1.0//EN"
4   "http://www.oasis-open.org/committees/entity/release/1.0/catalog.dtd">
5 <catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
6   prefer="public">
7   <public publicId="-//W3C//DTD XHTML 1.1//EN"
8     uri="file:///usr/share/sgml/xhtml/xhtml-1.1/DTD/xhtml11-flat.dtd"/>
9 </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
```

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany,
Course on XML and Semantic Web Technologies, summer term 2012

45/46

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

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