XML and Databases

Chapter 2: XML II: Entities and Marked Sections

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Winter 2022/23

http://www.informatik.uni-halle.de/~brass/xml22/

Objectives

After completing this chapter, you should be able to:

- explain what entities are in XML.
- enumerate the five predefined entities of XML, use them in XML documents.
- explain the purpose of `CDATA` section, use them in XML documents.
- read XML Document Type Definitions (DTDs) that make use of parameter entities.

The appendix is not relevant for the exam.

Contents

1. General Entities
2. Parameter Entities
3. Marked Sections
4. Appendix: More about Entities
5. Appendix: Notations, Unparsed Entities

Entities: Overview (1)

- Entities can be used as macros (abbreviations), e.g. one can declare an entity "`ora`" with the value "`Oracle 8.1.6`" (replacement text):

```
<!ENTITY ora "Oracle 8.1.6">
```

- When the entity is declared, the entity reference

```
&ora;
```

  in the document is replaced by "`Oracle 8.1.6`".

  In SGML, the "`;`" is optional if a character follows that cannot be part of the entity name, e.g. a space. In XML, the "`;`" is always required.

Entities: Overview (2)

- There are different kinds of entities. The above example is a general, internal, parsed entity.
- Entities can be classified as:
  - General: Used in the document.
    Parameter: Used in the DTD.
  - Internal: The value is written in the declaration.
    External: The value is contained in another file.
  - Parsed: The value is SGML/XML text.
    Unparsed: The value is e.g. binary data.

    In SGML, parsed entities are also called SGML entities, other entities are called Non-SGML or data entities.

Entities: Overview (3)

- Of the eight theoretically possible combinations, only five are permitted: Unparsed entities must always be external and general.

  Non-SGML/XML data cannot be directly included in an SGML/XML document and can certainly not be used in the DTD.

- In the SGML/XML literature, entities are seen as the physical units (storage units) of a document.

  I.e. entities are a generalization of files (e.g. they could also be extracted from a database or be computed by a program). Entities are containers for SGML/XML and other data. The main file, where the SGML/XML processing starts, is called the "document entity". In contrast, elements are seen as the logical units of a document.

Entities: Motivation

- Entities reduce the typing effort (abbreviations).
- The entity name might be easier to remember than its replacement text (e.g. `&auml;` stands for `&#228;`).
- Using entities permits simpler updates and leads to higher uniformity.

  If in the above example, the Oracle version changes, one must change only the replacement text in the entity definition (at one place).

- One can also get several versions of a document via differently defined entities.

  E.g. if user interfaces are specified in XML, the language-dependent parts can be defined in entities.

Predefined General Entities

- In XML, the following five entities are predefined:
  - “`&amp;`” for “`&`” (ampersand).
  - “`&lt;`” for “`<`” (less-than symbol).
  - “`&gt;`” for “`>`” (greater-than symbol).
  - “`&apos;`” for “ `'` ” (apostrophe).
  - “`&quot;`” for “`"`” (quotation mark).
- In SGML, these are not predefined. Therefore, they should also be declared in XML.

External Entities

- Entities can also be used as an "include" mechanism for splitting a document into several files:

```
<!ENTITY copyr SYSTEM "copyr.xml">
```

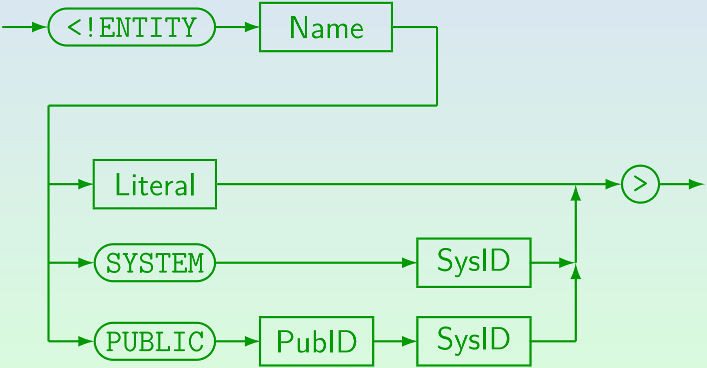
- Then the entity reference "`&copyr;`" in the document is replaced by by the contents of the file "`copyr.xml`".

  The keyword "`SYSTEM`" indicates that the following string gives a system-dependent way to retrieve the entity. In XML this must be a URI, possibly a relative one. There are also public identifiers (see below).

- This is a general, external, parsed entity.

Entity Declaration (1)

General Parsed Entity Declaration:



Entity Declaration (2)

- "Literal" is a string enclosed in single or double quotes. (' or ").
- Parameter entity references and general entity references can be used in the literal.

  Parameter entity references are immediately evaluated, general entity references become part of the replacement text of the entity.
- Entity references are not evaluated in the system and the public identifier.

Contents

1 General Entities

2 Parameter Entities

3 Marked Sections

4 Appendix: More about Entities

5 Appendix: Notations, Unparsed Entities

Parameter Entities (1)

- General entities are used in the document (data).
- However, macros are also useful in the DTD.
- But macros applied in the DTD are not relevant for the user of the DTD, they might even confuse him/her.
- Therefore, two distinct namespaces are used:
  - General entities are substituted in the document.

    And in the default attribute value in the DTD. They can also be used in the declared value of other entities.
  - Parameter entities are substituted in the DTD.

Parameter Entities (2)

- The declaration of parameter entities contains an additional “%”:

```
<!ENTITY % ltypes "(disc|square|circle)">
```

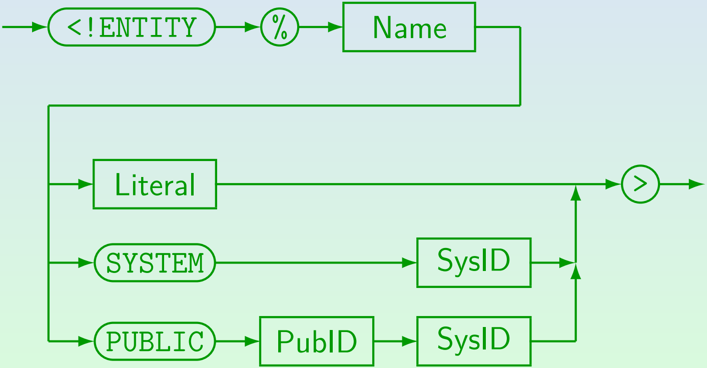
- Correspondingly, a parameter entity reference uses a percent sign “%” instead of the ampersand “&”:

```
%ltypes;
```

- In the document itself, “%” has no special meaning.
- It is even possible to have a general entity and a parameter entity with the same name.

Parameter Entities (3)

Parameter Entity Declaration:



Contents

Marked Sections (1)

- The contents of an IGNORE-section is not processed:

```
<![IGNORE[...]]>
```

- In contrast, the contents of an INCLUDE-section is processed normally:

```
<![INCLUDE[...]]>
```

- One can define an entity which has one of the two values "IGNORE" and "INCLUDE" to get a feature similar to "conditional compilation", e.g.

```
<!ENTITY % solution "INCLUDE">
```

Marked Sections (2)

- Then one can mark sections that are to be included only in certain versions of the document, e.g. solutions are printed only in the edition for the teacher:

```
<![%solution;[...]]>
```

- In XML, conditional marked sections can only be used in the external subset of the DTD and must contain a sequence of complete markup declarations (not arbitrary text).

  In SGML, marked sections can appear in the DTD and in content (the body of the document).

- Conditional marked sections can be nested.

Marked Sections (3)

- Besides these conditional sections, there are also “verbatim” sections, in which markup is not evaluated.
- `CDATA`-sections can contain the characters “`<`”, “`>`” and “`&`” as normal text. They are not interpreted as markup:

  ```
  <![CDATA[...]]>
  ```
- Of course, `CDATA` sections can only be used in the document body, not in the DTD.

Marked Sections (4)

- `CDATA` sections cannot nest.

  The only markup that is interpreted within a `CDATA` section is its end delimiter "`]]>`". The parser would not even notice the begin of another such section.

- `CDATA` sections are normally used for showing example XML/HTML code, which should not be interpreted as markup.

  The alternative would be to escape the special characters "`<`" and "`&`" one by one with entity or character references. Of course, within a `CDATA` section, entity and character references are also not understood.

Contents

1. General Entities
2. Parameter Entities
3. Marked Sections
4. Appendix: More about Entities
5. Appendix: Notations, Unparsed Entities

General Entities: Details (1)

- General parsed entities can be referenced in:
  - the content of an element,
  - attribute value literals (inside quotes),

    This includes default values of attributes defined in the DTD. In XML, only internal (general parsed) entities are allowed in attribute values. It seems that SGML does not have this restriction.
  - the entity value in the definition of an entity.
- E.g. entity references cannot be used instead of an element type or attribute name within a tag.

  As with whitespace, the SGML/XML grammar specifies where entity references can appear. They are not expanded in System/Public IDs.

General Entities: Details (2)

- SGML/XML try to exclude unexpected parsing mode changes after an entity is referenced.
- This is especially important for XML, because XML can be parsed without DTD.

  Then the replacement text for entities might not be known, but still the general structure of the document should be clear.

- The opening delimiter of a tag, comment, etc. must be in the same entity as the closing delimiter.

  I.e. the replacement text of an entity that is referenced in the content cannot contain an unmatched "<" or ">". If the entity is referenced in an attribute value, these characters have no special meaning.

General Entities: Details (3)

- If an entity reference appears in an attribute value, the delimiters (quotes) " and ' are not interpreted in the replacement text.

  I.e. it is not possible that an entity reference in an attribute value suddenly closes the attribute value.

- XML requires also that if the start tag of an element is contained in an entity, the corresponding end tag must be contained in the same entity.

  In SGML, this is only a recommendation, except for elements with content model "CDATA" and "RCDATA", where it is required.

General Entities: Details (4)

- Entities can be used in the definition of other entities:

```
<!ENTITY A "xxx">
<!ENTITY B "yyy &A; zzz">
```

- When the entity declaration is processed, the replacement text is simply stored under the name of the entity (including entity references within it).

- Only when "`&B;`" is called later in the document, the replacement text is inserted, and recursively, any entity references within it are substituted.

General Entities: Details (5)

- Entities must be defined before they are used.
- However, because of the delayed (“lazy”) processing of references, this rule would even be satisfied if the entities “A” and “B” would have been declared in the opposite order.

  Since general entities are declared in the DTD and normally only used in the document, “definition before use” is seldom a problem.
- When an entity is referenced in the default value for an attribute in an ATTLIST declaration, it is immediately evaluated (and must already be defined).

General Entities: Details (6)

- Of course, recursive definitions are forbidden:

  ```
  <!ENTITY X "This is not allowed &X;">
  ```

- If the same entity is defined several times, the first definition counts.

  The “internal subset of the DTD” is processed before the part in external files. Then the external subset can contain a default value for the entity, which can be overridden in the internal subset.

General Entities: Details (7)

- The replacement of entities like "`&amp;`" does not lead to the generation of new entity references.

  E.g. in "`AT&amp;T;`", after the first replacement, the parser does not recursively try to replace "`&T;`". This follows the general rule that the replacement of entities does not lead to the generation of new structures.

- Character references are already replaced when the entity definition is processed (non-recursively).

  E.g. the entity "`amp`" is defined as follows: `<!ENTITY amp "&#38;#38;">`. When this definition is processed, "`&#38;`" is replaced by "`&`". When "`&amp;`" is later used in the text, it is expanded to "`&#38;`", and this is replaced by the character "`&`" which now has no special meaning.

Parameter Entities: Details (1)

- The replacement text of a parameter entity is extended by spaces at the beginning and the end.

  This makes sure that no tokens can merge when parameter entities are replaced.

- In XML, the use of parameter entities in the internal subset of the DTD is quite restricted: A parameter entity reference can only appear in places where an entire declaration would be permitted.

  I.e. there, parameter entities can contain only complete markup declarations. This restriction does not hold for the external subset.

Parameter Entities: Details (2)

- In contrast to general entities (and like character references), parameter entities are immediately replaced, even if they are used in the definition of another entity.
- As for general entities, if a parameter entity replacement text contains the start of a markup declaration ("<"), it must also contain the corresponding end.

Parameter Entities: Details (3)

- Of course, there are also external parameter entities:

```
<!ENTITY % tables SYSTEM "tab.xml">
```

- The contents of the file “tab.xml” is inserted in the DTD where the parameter entity is referenced:

```
%tables;
```

- A large DTD can be constructed in this way out of components stored in different files.

  Also the same component might be reused for different DTDs.

Contents

1. General Entities
2. Parameter Entities
3. Marked Sections
4. Appendix: More about Entities
5. Appendix: Notations, Unparsed Entities

Notations (1)

- An SGML/XML-system can also manage entities (files) that do not contain SGML/XML text.
- E.g. a document often includes pictures in formats like GIF, JPG, PNG, TIFF.
- One can define in SGML/XML that e.g. GIF is a name for a notation (data format).
- Then one can define external entities that use the notation "GIF" (and are therefore not syntactically analyzed).

Notations (2)

- A notation can be declared with a system identifier, which might e.g. refer to a program that could display the data ("helper application"):

  ```
  <!NOTATION GIF SYSTEM "file:///local/bin/xv">
  ```

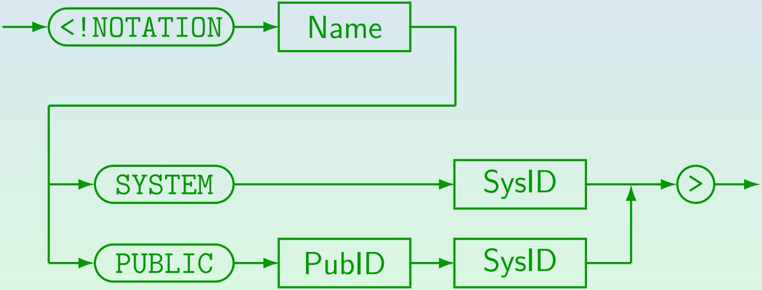
  However, the SGML/XML parser only passes the system identifier to the application program. It depends on this program, how it uses this information.

- In XML, system identifiers must be URIs (without #), enclosed in single or double quotes.

  The standard only says "It is meant to be converted to a URI reference ...". Entity references are not evaluated. The URI can possibly be relative to the location of the entity that contains the declaration.

Notations (3)

Notation Declaration:



Public Identifiers (1)

- Normally, public identifiers refer in some way to further information:

```
<!NOTATION POSTSCRIPT PUBLIC
    "+//ISBN 0-201-18127-4::Adobe//NOTATION
        Postscript Language Ref. Manual//EN">
```

- There is no well-known and generally accepted list of public identifiers for notations.

  But see below for examples. There seems to be no central registry for public identifiers.

- However, e.g. HTML versions have public identifiers mentioned in the respective standards.

Public Identifiers (2)

- In the XML Bible [p. 309] the following public identifier is used for GIF (this method could be generalized to arbitrary MIME types):

  `"-//IETF//NONSGML Media Type image/gif//EN"`

- In my view, the keyword "`NONSGML`" is wrong and must be replaced by "`NOTATION`".

  "`NONSGML`" means a non-SGML data entity.

- The XML Bible uses the following system identifier:

```
"http://www.isi.edu/in-notes/iana/assignments/
                    media-types/image/gif"
```

Public Identifiers (3)

Public identifiers used in the DocBOOK DTD:

- BMP: `"+//ISBN 0-7923-9432-1::Graphic Notation//NOTATION Microsoft Windows bitmap//EN"`
- EPS: `"+//ISBN 0-201-18127-4::Adobe//NOTATION PostScript Language Ref. Manual//EN"`
- GIF87a: `"-//CompuServe//NOTATION Graphics Interchange Format 87a//EN"`
- GIF89a: `"-//CompuServe//NOTATION Graphics Interchange Format 89a//EN"`
- PNG: `"http://www.w3.org/TR/REC-png"`
- TeX: `"+//ISBN 0-201-13448-9::Knuth//NOTATION The TeXbook//EN"`
- WMF: `"+//ISBN 0-7923-9432-1::Graphic Notation//NOTATION Microsoft Windows Metafile//EN"`
- SGML: `"ISO 8879:1986//NOTATION Standard Generalized Markup Language//EN"`
- FAX: `"-//USA-DOD//NOTATION CCITT Group 4 Facsimile Type 1 Untiled Raster//EN"`
- CGM-CHAR: `"ISO 8632/2//NOTATION Character encoding//EN"`
- CGM-BINARY: `"ISO 8632/3//NOTATION Binary encoding//EN"`
- CGM-CLEAR: `"ISO 8632/4//NOTATION Clear text encoding//EN"`

Unparsed Entities (1)

- An unparsed entity is declared in the following way:

  ```
  <!ENTITY clown SYSTEM "clown.gif" NDATA GIF>
  ```

- The keyword “NDATA” (“Non-SGML Data”) must always be followed by a declared notation name.

  SGML has also the keywords “CDATA” and “SDATA” which are, however, not supported in XML.

- In this way, the SGML/XML system “knows” the data format (media type) of each entity and does not have to guess it from file extensions etc.

Unparsed Entities (2)

- Unparsed entities cannot be used in entity references, but one can declare element types that take entities as attributes:

```
<!ELEMENT IMAGE EMPTY>
<!ATTLIST IMAGE SRC ENTITY #REQUIRED>
```

- Then one can write e.g.:

```
<IMAGE SRC="clown"/>
```

The SGML parser then makes system/public identifier (public IDs can normally be mapped to system IDs) of entity and notation available to the application program. The application program can then retrieve the data of the entity by means of the “entity manager” (the layer below the SGML parser, also part of an SGML system).

Unparsed Entities (3)

- Of course, parsed general entities can also be used as attribute values (not only unparsed entities).

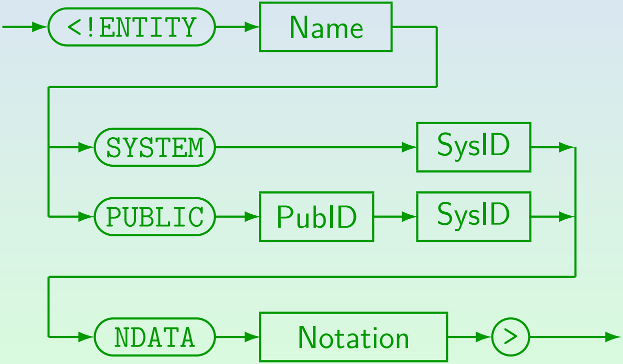
  Only general entities can appear as attribute values (parameter entities have no meaning in the document itself, i.e. the data).

- One cannot restrict the possible notations for entities of an attribute in the attribute declaration.

- In HTML, one cannot define entities (the given DTD cannot be extended). Therefore, the element type “IMG” as an attribute of type “CDATA” which directly contains the URI of the image file.

Unparsed Entities (4)

Unparsed Entity Declaration:



References

- Boc DuCharme: XML — The Annotated Specification.
  Prentice Hall, 1999. ISBN 0-13-082676-6, 339 pages.

- Tim Bray, Jean Paoli, C.M. Sperberg-McQueen: Extensible Markup Language (XML) 1.0, 1998. [http://www.w3.org/TR/REC-xml]
  See also: [http://www.w3.org/XML] [https://www.w3.org/XML/Core/]

- Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan: Extensible Markup Language (XML) 1.1 (Second Edition), W3C Recommendation 16 August 2006, edited in place 29 September 2006. [http://www.w3.org/TR/xml11].

- Elliotte Rusty Harold, W. Scott Means:
  XML in a Nutshell, A Desktop Quick Ref., 3rd Ed.
  O'Reilly, Okt. 2004, ISBN 0-596-00764-7, 689 Seiten, 37 Euro.

- Liora Alschuler: ABCD ... SGML — A User's Guide to Structured Information. International Thomson Computer Press (ITP), 1995, ISBN 1-850-32197-3, 414 pages.

- Charles F. Goldfarb, Yuri Rubinsky: The SGML Handbook. Clarendon Press, 1990.

- Elliotte Rusty Harold: XML 1.1 Bible.
  John Wiley & Sons, 3rd Edition, 2004, ISBN: 0764549863, 1056 pages.