Documentation of DTD dtddoc

A Language for Defining and Documenting DTDs

Burkhardt Renz
University of Applied Sciences Gießen-Friedberg
Wiesenstr. 14
D-35390 Gießen
Burkhardt.Renz@mni.fh-giessen.de

Revision 1.14 October 2006

This document was generated using the technique described herein — bootstrapping `dtddoc`.

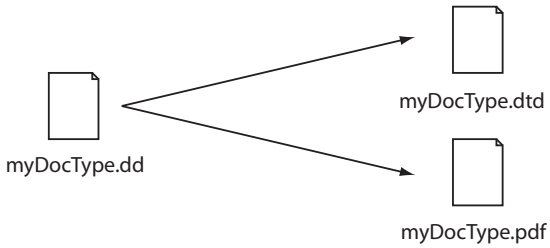
Generated by dd2doc.xsl from dtddoc.dd.

1 An XML Language for Defining and Documenting DTDs

Keeping source code and documentation together in the same file is always a good idea. In the case of a document type definition this can simply be done by adding comments to the dtd. But the result is unsatisfactory: as a textfile the annotated dtd file has a poor layout and is not really usable as a reference.

`dtddoc` is a XML language to define the document type definition *and* its documentation in a common source file, from which the dtd and the documentation is generated. The document you are currently reading is generated from a `dtddoc` file.

Let's assume the source file, written in the `dtddoc` XML language, is named `myDocType.dd`. This file is then transformed to the actual document type definition `myDocType.dtd` and to the documentation file `myDocType.pdf`.

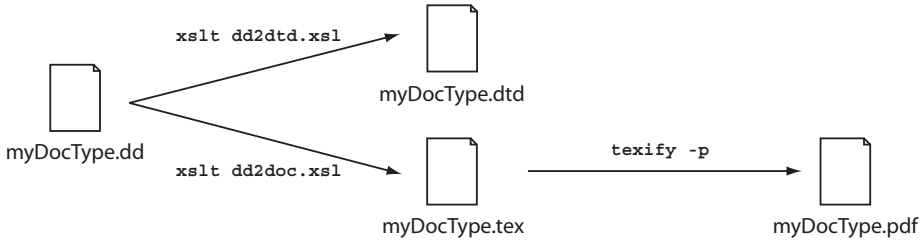


The generation of the document type definition is achieved by the XSL transformation `dd2dtd.xsl`.

To produce the documentation file, we need two steps:

1. The XSL transformation `dd2doc.xsl` generates a LaTeX file.
2. LaTeX renders the documentation to a pdf file.

The complete production process is shown in the following figure



Tools used:

- gnu make,
- XSLT processor,
- LaTeX with packages: inputenc, babel, array, graphicx, color, ae, fontenc, url, and hyperref, and
- TeX .

As one can see in the makefile, I used the MKS Toolkit, Microsoft msxsl, and the MikTex TeX installation on Windows NT.

`dtddoc` does not support all possibilities of markup declarations defined by the XML Recommendation. In `dtddoc` one can declare

- parameter entities for classes of elements that can be used as options (repeatable or-groups) typically in mixed content elements,
- parameter entities for enumerations of types of attributes,
- parameter entities for classes of attributes, and
- elements with their attribute lists.

The restriction to these elements of a document type definition is not only due to the simplicity of `dtddoc`, but supports a certain style to define document

type definitions. The four types of entities give the dtd a structure, that makes it clearer and better maintainable.
On the other hand, there is an escape mechanism, i.e. one can define content in dtddoc which is copied verbatim to the dtd.

2 Reference

2.1 Parameter entities for elements

2.1.1 inline.class

```
<!ENTITY % inline.class
    "#PCDATA | em | q | tt | v | ul | ol | img | nl | gt | lt
    | amp | ding">
```

Description

This parameter entity contains the inline elements.

Verbatim reproduced in dtd:

```
 <!- This is an example for the escape mechanism
    of dtddoc:-> <!NOTATION TEX PUBLIC "+//ISBN
    0-201-13448-9::Knuth//NOTATION The TeXbook//EN"> >
```

2.2 Parameter entities for enumerations

2.2.1 bool

```
<!ENTITY % bool
    '( true | false )'>
```

Description

The boolean values.

true	true
false	false

2.2.2 type.enum

```
<!ENTITY % type.enum
    '( CDATA | ID | IDREF | IDREFS | ENTITY | ENTITIES |
    NMTOKEN | NMTOKENS | enum )'>
```

Description

This parameter entity defines the types of attributes, how they are given from the XML specification.

CDATA	Character data, but not the CDEnd sequence
ID	Unique name within a XML document
IDREF	Must match an ID
IDREFS	Must match IDs
ENTITY	Must match an unparsed entity declared in the DTD
ENTITIES	
NMTOKEN	Any mixture of name characters
NMTOKENS	
enum	The values are given by an enumeration. The values are defined by the parameter entity specified in the attribute enum-pe-ref

2.2.3 default.enum

```
<!ENTITY % default.enum
    '( req | impl | fixval | val )'>
```

Description

This parameter entity defines the attribute defaults.

req	The attribute is required
impl	The attribute is implied
fixval	The attribute has a fixed value
val	The attribute has a default value

2.2.4 imgext.enum

```
<!ENTITY % imgext.enum
    '( pdf | png | jpg | jpeg | tif | tiff )'>
```

Description

The extensions of image files. The file formats are given by pdftex, which is used to render the document.

pdf	Portable Document Format, default
png	Portable Network Graphics
jpg	Joint Photographic Experts Group
jpeg	Joint Photographic Experts Group
tif	Tagged Image File Format
tiff	Tagged Image File Format

2.3 Parameter entities for attributes

2.3.1 name.att

```
<!ENTITY % name.att
    ' name NMTOKEN #REQUIRED '>
```

Description

Attribute name

name	name of element

2.3.2 lang.att

```
<!ENTITY % lang.att
    ' xml:lang NMTOKEN #IMPLIED '>
```

Description

Attribute for language

xml:lang	language of contents in nodeset elements, at the moment the dd2doc.xsl uses de and en

2.3.3 repeat.att

```
<!ENTITY % repeat.att
    ' repeat CDATA #IMPLIED '>
```

Description

Attribute for repetition of elements

repeat	*, + or ? according to DTD syntax

2.4 Hierarchy of elements in dtddoc

This section contains the elements that build the hierarchical structure of dtdoc.

2.4.1 dtddoc

```
<!ELEMENT dtddoc
    ( docinfo, description, reference ) >
```

```
<!ATTLIST dtddoc
    %name.att;
    %lang.att;>
```

Description

The root element. Use the lang.att to control the language for the documentation.

2.4.2 docinfo

```
<!ELEMENT docinfo
    ( author, subject?, revision, date?, keywords?, note?  )
    >
```

Description

Header with document information: author, subject, revision and keywords.

2.4.3 author

```
<!ELEMENT author
    ( name, address?, email ) >
```

Description

Author

2.4.4 name

```
<!ELEMENT name
    ( %inline.class; )* >
```

Description

Author's name

2.4.5 address

```
<!ELEMENT address
    ( %inline.class; )* >
```

Description

Author's address

2.4.6 email

```
<!ELEMENT email
    ( #PCDATA ) >
```

Description

Author's e-mail

2.4.7 subject

```
<!ELEMENT subject
    ( %inline.class; )* >
```

Description

Subject

2.4.8 revision

```
<!ELEMENT revision
    ( %inline.class; )* >
```

Description

Revision, normally a pattern for rcs keyword substitution.

Example

```
<revision>$Revision: 1.14 $<revision/>
```

2.4.9 date

```
<!ELEMENT date
    ( %inline.class; )* >
```

Description

Date of revision

2.4.10 keywords

```
<!ELEMENT keywords
    ( %inline.class; )* >
```

Description

Keywords, separated by semicolon

2.4.11 note

```
<!ELEMENT note
    ( %inline.class; )* >
```

Description

Note on the `dtddoc` file, the aim, the author(s)...

2.4.12 description

```
<!ELEMENT description
    ( title?, desc* ) >
```

Description

Description of DTD, each subelement desc builds a paragraph in the documentation.

2.4.13 desc

```
<!ELEMENT desc
    ( %inline.class; )* >
```

Description

Paragraph in documentation, may contain rich text.

2.4.14 reference

```
<!ELEMENT reference
    ( sect-choice-pe*, sect-enum-pe*, sect-attribute-pe*,
    sect-element+ ) >
```

Description

This section contains the definitions of parameter entities and elements with their attributes.

2.4.15 sect-choice-pe

```
<!ELEMENT sect-choice-pe
    ( title?, desc?, (choice-pe | dtd-esc )+ ) >
```

Description

This section contains the parameter entities for choices of elements. It is a widely used convention to name these parameter entities with a suffix `.class`.

2.4.16 sect-enum-pe

```
<!ELEMENT sect-enum-pe
    ( title?, desc?, (enum-pe | dtd-esc )+ ) >
```

Description

This section contains the definition of enumerations used for values of attributes. With `dtddoc` the only possibility to give a series of values is to define a enumeration parameter entity and to reference it in the attribute definition. Often the extension `.enum` is used.

2.4.17 sect-attribute-pe

```
<!ELEMENT sect-attribute-pe
    ( title?, desc?, (attribute-pe | dtd-esc )+ ) >
```

Description

This section contains the parameter entities for attributes. The role of these entities are expressed by the extension `.att`.

2.4.18 sect-element

```
<!ELEMENT sect-element
    ( title?, desc?, (element | dtd-esc )+ ) >
```

Description

Eventually follows the definition of elements and their attributes. There may be several of these sections, to group the elements into categories.

2.4.19 title

```
<!ELEMENT title
    ( %inline.class; )* >
```

Description

Title

2.5 Elements in reference

2.5.1 choice-pe

```
<!ELEMENT choice-pe
    ( desc?, pcdata?, (element-ref | choice-pe-ref )+ ) >
```

```
<!ATTLIST choice-pe
    %name.att;>
```

Description

Parameter entity for a choice of elements. The name normally end with `.class`.

Example

The following example defines the class of inline elements.

```
<choice-pe name="inline.class"/>
  <desc>This parameter entity contains the inline
    elements.</desc>
  <pcdata/>
  <element-ref name="em"/>
  <element-ref name="tt"/>
  <element-ref name="nl"/>
  <element-ref name="gt"/>
```

```
  <element-ref name="lt"/>
  <element-ref name="amp"/>
</choice-pe>
```

It generates the following parameter entity in the DTD:

```
<!ENTITY % inline.class " #PCDATA |em |tt |nl |gt |lt |amp ">
```

2.5.2 dtd-esc

```
<!ELEMENT dtd-esc
    ( #PCDATA ) >
```

Description

The content of this element is verbatim reproduced in the dtd (and in the documentation). It offers the possibility to make markup declarations that are not supported by `dtddoc`. Please observe that the content should not contain special characters that are interpreted by TEX, otherwise the documentation file may be incomplete or false.

2.5.3 pcdata

```
<!ELEMENT pcdata
    ( #PCDATA ) >
```

```
<!ATTLIST pcdata
    %repeat.att;>
```

Description

Element pcdata stands for `PCDATA` in mixed content elements.

2.5.4 element-ref

```
<!ELEMENT element-ref EMPTY >
```

```
<!ATTLIST element-ref
    %name.att;
    %repeat.att;>
```

Description

A reference to the definition of an element.

2.5.5 choice-pe-ref

```
<!ELEMENT choice-pe-ref EMPTY >
```

```
<!ATTLIST choice-pe-ref
    %name.att;>
```

Description

A reference to a parameter entity.

2.5.6 enum-pe

```
<!ELEMENT enum-pe
    ( desc?, enum+ ) >
```

```
<!ATTLIST enum-pe
    %name.att;>
```

Description

A parameter entity which enumerates values of attributes.

Example

The possible values of the attribute default of an attribute are described in an enum-pe element:

```
<enum-pe name="default.enum">
  <desc>This parameter entity defines the
    mode of the attribute</desc>
  <enum value="req" desc="The attribute is required"/>
  <enum value="impl" desc="The attribute is implied"/>
  <enum value="fixval" desc="The attribute has a fixed value"/>
  <enum value="val" desc=" The attribute has a default value"/>
</enum-pe>
```

And this is the resulting entity in the DTD:

```
<!ENTITY % default.enum '( req |impl |fixval |val )'>
```

2.5.7 enum

```
<!ELEMENT enum EMPTY >
```

```
<!ATTLIST enum
    value CDATA #REQUIRED
    desc CDATA #IMPLIED>
```

Description

A value of an enumeration.

Attributes

value	The value
desc	The description of the value

2.5.8 attribute-pe

```
<!ELEMENT attribute-pe
    ( desc?, (attribute-pe-ref | attribute )+ ) >
```

```
<!ATTLIST attribute-pe
    %name.att;>
```

Description

A parameter entity for attributes. The name normally ends with `.att`.

Example

The definition

```
<attribute-pe name="name.att">
  <attribute name="name" type="NMTOKEN" default="req"
    desc="name of element"/>
</attribute-pe>
```

produces

```
<!ENTITY % name.att ' name NMTOKEN #REQUIRED '>
```

2.5.9 attribute-pe-ref

```
<!ELEMENT attribute-pe-ref EMPTY >
```

```
<!ATTLIST attribute-pe-ref
    %name.att;>
```

Description

A reference to a parameter entity.

2.5.10 attribute

```
<!ELEMENT attribute EMPTY >
```

```
<!ATTLIST attribute
    %name.att;
    type %type.enum; #REQUIRED
    enum-pe-ref CDATA #IMPLIED
    default %default.enum; #IMPLIED
    val CDATA #IMPLIED
    desc CDATA #IMPLIED>
```

Description

Definition of an attribute. There is a somewhat tricky dependence of attributes of this element.

Attributes

type	Type of attribute
enum-pe-ref	Enumeration for values, only used if type is enum
default	default may be req or impl corresponding to REQUIRED or IMPLIED, respectively. For a FIXED value use fixval and for a default value use val
val	The value, if default is fixval or val
desc	Description of attribute for documentation, for sake of simplicity rich text is not allowed.

2.5.11 element

```
<!ELEMENT element
    ( desc?, (empty | any | mixed | sequence | choice ),
    attlist?, example* ) >
```

```
<!ATTLIST element
    %name.att;>
```

Description

Definition of element with attribute list. The type of element: empty, any, mixed, sequence or choice is the main child element.

2.5.12 example

```
<!ELEMENT example
    ( %inline.class; )* >
```

Description

Paragraph in the documentation of an element, may contain rich text, use <tt> for typewriter text style.

2.5.13 empty

```
<!ELEMENT empty EMPTY >
```

Description

Defines an empty element.

Example

Definition

```
<element name="nl">
  <desc>newline</desc>
  <empty/>
</element>
```

Output

```
<!ELEMENT nl EMPTY>
```

2.5.14 any

```
<!ELEMENT any EMPTY >
```

Description

Defines an element with type ANY.

Example

Definition

```
<element name="any">
  <desc>example of an any element</desc>
  <any/>
</element>
```

Output

```
<!ELEMENT any ANY>
```

2.5.15 mixed

```
<!ELEMENT mixed
    ( pcdata?, element-ref*, choice-pe-ref* ) >
```

Description

Defines an element with mixed content.

Example

Definition

```
<element name="author">
</example>
  <desc>Author</desc>
  <mixed>
    <choice-pe-ref name="inline.class"/>
  </mixed>
</element>
```

Output

```
<!ELEMENT author ( %inline.class; )*>
```

2.5.16 sequence

```
<!ELEMENT sequence
    (element-ref | choice)+ >
```

Description

Defines an (sub)element which is the sequence of elements or choices of elements.

Example

Definition

```
<element name="docinfo">
  <desc>Header with document information: author, subject,
    revision and keywords.</desc>
  <sequence>
    <element-ref name="author"/>
    <element-ref name="subject" repeat="?"/>
```

```
    <element-ref name="revision"/>
    <element-ref name="keywords" repeat="?"/>
  </sequence>
</element>
```

Output

```
<!ELEMENT docinfo (
  author, subject?, revision, keywords?)>
```

2.5.17 choice

```
<!ELEMENT choice
    (element-ref | choice-pe-ref | sequence)+ >
```

```
<!ATTLIST choice
    %repeat.att;>
```

Description

Defines an (sub)element which is the choice of elements.

Example

Definition

```
<element name="element">
  <sequence>
    <element-ref name="desc" repeat="?"/>
    <choice>
      <element-ref name="empty"/>
      <element-ref name="any"/>
      <element-ref name="mixed"/>
      <element-ref name="sequence"/>
      <element-ref name="choice"/>
    </choice>
    <element-ref name="attlist" repeat="?"/>
    <element-ref name="example" repeat="*"/>
    </sequence>
</element>
```

Output

```
<!ELEMENT element (
```

```
  desc?,
  (empty
  |any
  |mixed
  |sequence
  |choice),
  attlist?,
  example*)>
```

2.5.18 attlist

```
<!ELEMENT attlist
    (attribute-pe-ref | attribute)+ >
```

Description

Defines the attribute list of an element.

Example

Definition

```
<element name="attribute">
  <empty/>
  <attlist>
    <attribute-pe-ref name="name.att"/>
    <attribute name="type" type="enum"
      enum-pe-ref="type.enum" default="impl"/>
    <attribute name="enum-pe-ref" type="CDATA" default="impl"/>
    <attribute name="default" type="enum"
      enum-pe-ref="default.enum" default="impl"/>
    <attribute name="val" type="CDATA" default="impl"/>
    <attribute name="desc" type="CDATA" default="impl"/>
  </attlist>
</element>
```

Output

```
<!ELEMENT attribute EMPTY>
<!ATTLIST attribute
  %name.att;
  type %type.enum; #IMPLIED
  enum-pe-ref CDATA #IMPLIED
  default %default.enum; #IMPLIED
```

```
val CDATA #IMPLIED
desc CDATA #IMPLIED>
```

2.6 Inline elements in descriptions

The following elements are inline elements in the elements needed for the documentation of the DTD.

2.6.1 em

```
<!ELEMENT em
    ( %inline.class; )* >
```

Description

emphasize

2.6.2 q

```
<!ELEMENT q
    ( %inline.class; )* >
```

Description

quote

2.6.3 tt

```
<!ELEMENT tt
    ( %inline.class; )* >
```

Description

typewriter style

2.6.4 v

```
<!ELEMENT v
    ( #PCDATA ) >
```

Description

tex' verbatim

2.6.5 ul

```
<!ELEMENT ul
    ( li+ ) >
```

Description

unordered list

2.6.6 ol

```
<!ELEMENT ol
    ( li+ ) >
```

Description

ordered list

2.6.7 li

```
<!ELEMENT li
    ( %inline.class; )* >
```

Description

list item

2.6.8 img

```
<!ELEMENT img EMPTY >
```

```
<!ATTLIST img
    file CDATA #REQUIRED
    ext %imgext.enum; "pdf"
    width CDATA #IMPLIED
    left CDATA #IMPLIED
    hasBox %bool; "false">
```

Description

image

Attributes

file	name of graphic file, with (relative) path, but without extension.
ext	extension of graphic file, default: pdf
width	output width in points (tex: bp), without specification of width the original width of the image file is used
left	left indentation in points (tex:bp), 0 if omitted
hasBox	true if the image should be surrounded by a box

2.6.9 nl

```
<!ELEMENT nl EMPTY >
```

Description

newline

2.6.10 gt

```
<!ELEMENT gt EMPTY >
```

Description

greater than

2.6.11 lt

```
<!ELEMENT lt EMPTY >
```

Description

less than

2.6.12 amp

```
<!ELEMENT amp EMPTY >
```

Description

ampersand

2.6.13 ding

```
<!ELEMENT ding
    ( #PCDATA ) >
```

Description

Glyph from Zapf Dingbats. The content of the element is an integer that specifies the glyph to be typeset according to the character table. The character table of the Postscript font `ZapfDingbats` can be found on p.336 of "The LaTeX Companion".

Example

The element <ding>52</ding> gives ✔.

3 Index

3.1 Parameter entities for elements

3.2 Parameter entities for enumerations

3.3 Parameter entities for attributes

3.4 Elements and their attributes