DocBook Examples working with dblatex

COLLABORATORS

	TITLE : DocBook Examples working with dblatex	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 13, 2009	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

List of Figures

List of Tables

List of Examples

Chapter 1

Structuring Text

1.1 Inlined Elements

You can explicitely modify the rendering of portions of text with `emphasis` that can take several roles:

- *Default emphasis like this one is often slanted.*
- **The role attribute can be set to bold.**
- The role attribute you can be used to underline text.

In DocBook it is wiser to use the elements defining the nature of the text instead of an explicit rendering. It makes the document more portable and free from the processing tool.

Here are some examples of inlined elements: a `filename`, an application, a **command**, a *`replaceable`* field. A portion of `code`, a `literal` element.

1.2 Cross References

It is quite useful to be able to refer to other parts of the document. `xref` and `link` can be used for this purpose.

You can refer to a chapter: Chapter 4, a section: Section 5.2.1, to any kind of floating objects: Table 4.2, Example 2.1, Figure 5.1.

You can define the cross-reference text with `link` instead of the automatic formatting done by `xref`: The Table Chapter, a section about callouts, a colored table, a listing example, a figure with callouts.

The referenced text can come from another reference specified with the `endterm` attribute: Tables .

You can also cite a bibliographic document: [R1] or refer to any other fancy part of the document: This is line 4.

A glossary term can appear like that: *XML* and *SGML*.

1.3 Lists

You can use `itemizedlist`:

- item 1
- item 2
  - item a

  - item b
  - item c
- item 3

Here is a compact list:

- item 1
- item 2
- item 3

You can use `orderedlist`:

1. item 1
2. (a) item a
   (b) item b
   (c) item c
3. item 3

You can use change how the `orderedlist` numbers are formatted by playing with the `numeration` attribute:

i. Numeration is set to 'lowerroman'.

ii. I. Numeration is set to 'upperroman'.
II. item II
III. item III

iii. item iii

The following section is an included file. We used the following syntax with **xinclude** and **xpointer** :

```
<xi:include xmlns:xi="http://www.w3.org/2001/XInclude" href="example_unicode.xml" xpointer ↩
    ="unicode"/>
```

1.4 Unicode Support

Dblatex currently does not have a full support of Unicode, but this situtation will change in future releases.

Nevertheless, you can already insert most of the ISO latin-1 characters as show by Example 1.1.

You should use the following syntax *&#ddd;* where `ddd` stands for the decimal value of the character you wish to insert. You could also use *&#xhhh;* where `hhh` stands for the hexadecimal value of the character you wish to insert. The decimal or hexadecimal value must correspond to the encoding charset defined in the first line of your XML-file. For example ASCII tables can be found at http://www.table-ascii.com

Example 1.1 Insert ISO-8859-1 characters

To display @ you can use @
To display ® you can use ®
To display ± you can use ±
To display ¾ you can use ¾
To display ¿ you can use ¿
To display Ø you can use Ø
To display ß you can use ß

1.5 Admonitions and Sidebars

To emphasize some blocks of text you can use the following admonitions: `note`, `warning`, `caution`, `tip`, `important`. The admonitions can be titled or not. Here are how they appear.

Note Title
This is a titled note.

Warning Title
This is a titled warning.

Tip Title
This is a titled tip.

Important Title
This is a titled important admonition.

Some text outside the current flow can be put in a `sidebar` like the following:

Sidebar Title

This is a sidebar that talks about something not directly connected to the current text flow.

Chapter 2

Listings

2.1 Embedded Listing

You can directly put the listing content in your document by using the `programlisting` element.

Example 2.1 works with **dblatex** and with any XSLT processor with the DocBook XSL stylesheets (including xsltproc).

With dblatex you can specify the program listing language, and whether the the lines should be numbered. In the example Example 2.1 the language attribute is set to "c" and and the linenumbering attribute is set to "numbered".

Example 2.1 A simple C program

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  double *a;

  a = malloc (10*sizeof *a);

  a[10]=-999;

  free(a);
  a = NULL;

  return EXIT_SUCCESS;
}
```

2.2 Using External Files

You can refer to external listing files instead of embedding the listing in the document. It makes your document more modular, and easier to maintain.

Several methods are available to include external files, as show by the following sections.

2.2.1 Using XInclude

Instead of the listing content, use `xi:include` in `programlisting` to point to the listing file. See Example 2.2.

It works for dblatex and for the XSL stylesheets with any XSLT processor aware of XInclude.

Example 2.2 A Python program

```
import sys
from os import listdir
from os.path import isdir, isfile, join

def display_tree(directory='.', prefix=''):
    """Filesystem tree view"""
    files = listdir(directory)
    files.sort()
    for f in files:
        print prefix + '|-- ' + f
        fullname = join(directory, f)
        if isdir(fullname):
            display_tree(fullname, prefix + '|   ')

if __name__ == "__main__":
    if len(sys.argv) > 1:
        display_tree(directory=sys.argv[1])
    else:
        display_tree()
```

2.2.2 Using `textdata`

You can use `textdata` in a `programlisting` to refer to a listing file. To produce HTML or FO output with the DocBook XSL stylesheets this method requires some XSLT extensions provided by Saxon or Xalan. dblatex requires no extension and thus works with xsltproc.

Example 2.3 is written like this.

```
<programlisting language="python" linenumbering="numbered"><textobject><textdata
fileref="tree.py"/></textobject></programlisting>
```

Note
Every space or newline in a `programlisting` is actually shown. So, just include the `textobject` and `textdata` elements with no extra space.

Example 2.3 A Python program

```
import sys
from os import listdir
from os.path import isdir, isfile, join

def display_tree(directory='.', prefix=''):
    """Filesystem tree view"""
    files = listdir(directory)
    files.sort()
    for f in files:
        print prefix + '|-- ' + f
        fullname = join(directory, f)
        if isdir(fullname):
            display_tree(fullname, prefix + '|   ')

if __name__ == "__main__":
    if len(sys.argv) > 1:
        display_tree(directory=sys.argv[1])
    else:
        display_tree()
```

2.2.3 Using `inlinegraphic` or `inlinemediaobject`

Here are some alternative combinations to achieve the same goal. I believe they are obsoleted by the `textdata` use. There are the same constraints than with `textdata` to produce HTML or FO output.

Example 2.4 is written like this.

```
<programlisting language="python"><inlinemediaobject>
<imageobject><imagedata format="linespecific" fileref="tree.py"/></imageobject>
</inlinemediaobject></programlisting>
```

Example 2.5 is written like this.

```
<programlisting linenumbering="numbered"><inlinegraphic
  format="linespecific" fileref="tree.py"/></programlisting>
```

Example 2.4 A Python program

```
import sys
from os import listdir
from os.path import isdir, isfile, join

def display_tree(directory='.', prefix=''):
    """Filesystem tree view"""
    files = listdir(directory)
    files.sort()
    for f in files:
        print prefix + '|-- ' + f
        fullname = join(directory, f)
        if isdir(fullname):
            display_tree(fullname, prefix + '|   ')

if __name__ == "__main__":
    if len(sys.argv) > 1:
        display_tree(directory=sys.argv[1])
    else:
        display_tree()
```

Example 2.5 A Python program

```
1  import sys
2  from os import listdir
3  from os.path import isdir, isfile, join
4
5  def display_tree(directory='.', prefix=''):
6      """Filesystem tree view"""
7      files = listdir(directory)
8      files.sort()
9      for f in files:
10         print prefix + '|-- ' + f
11         fullname = join(directory, f)
12         if isdir(fullname):
13             display_tree(fullname, prefix + '|   ')
14
15 if __name__ == "__main__":
16     if len(sys.argv) > 1:
17         display_tree(directory=sys.argv[1])
18     else:
19         display_tree()
```

Chapter 3

Images

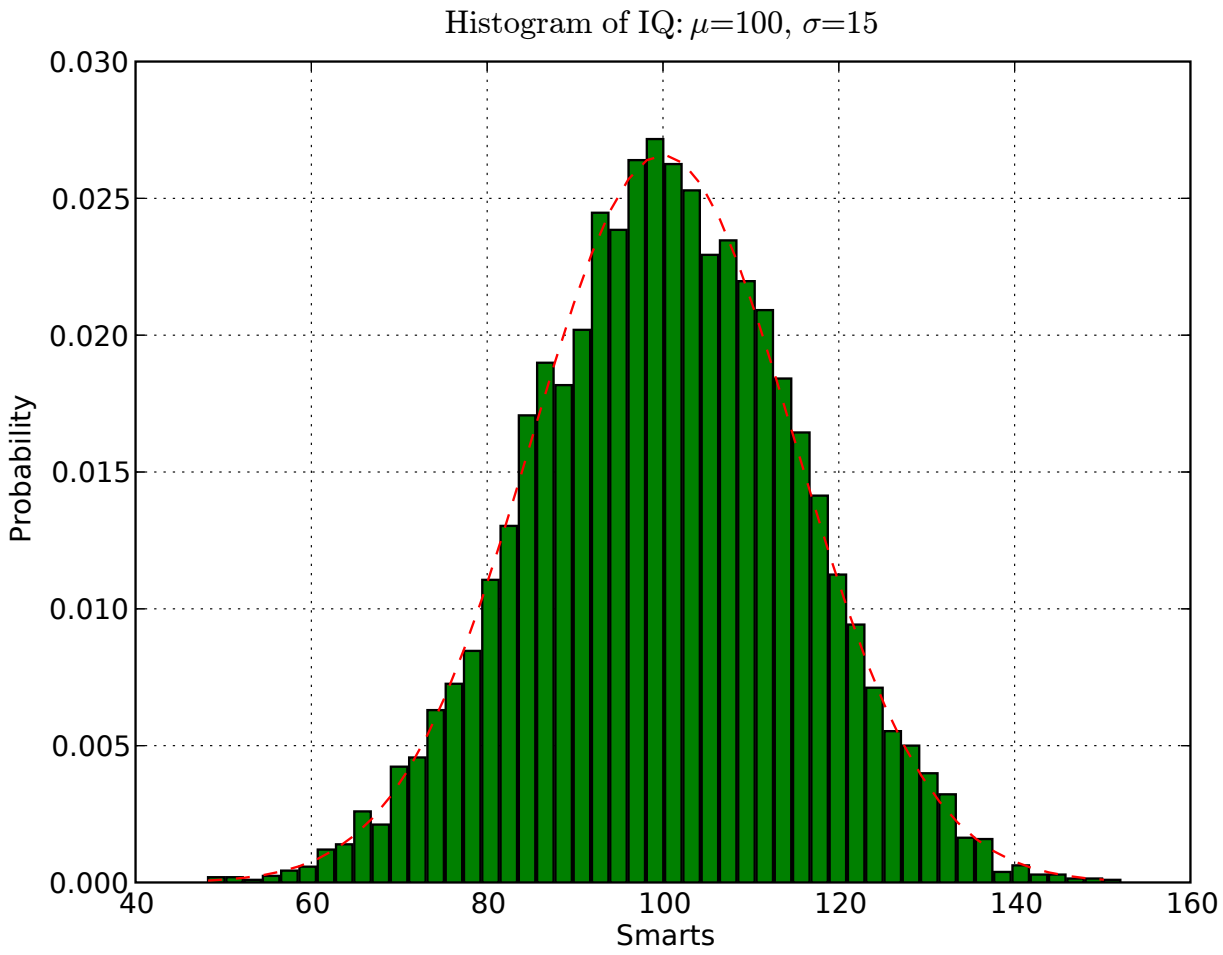


Figure 3.1: Original image

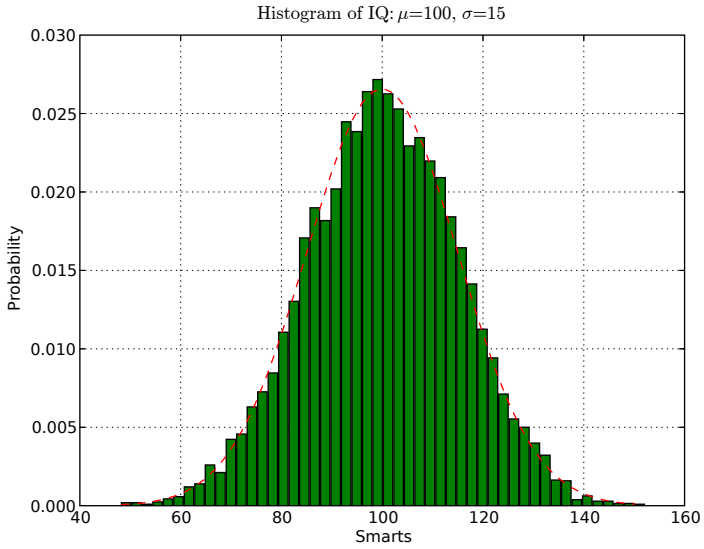


Figure 3.2: Image resized (50%)

Chapter 4

Tables

4.1 Normal Table

Horizontal Span		a3	a4	a5
b1	b2	b3	b4	Vertical Span
c1	Span Both		c4	
d1			d4	d5
f1	f2	f3	f4	f5

Table 4.1: Sample Table

4.2 Colored Table

The table cells can be colored. You need to use a specific Processing Instruction to specify the color, and place the PI in the element that must be colored (`entry`, `row`, or `colspec`), like this:

```
<row>
  <?dblatex bgcolor="[gray]{0.8}" ?>
  <entry>Name</entry>
  <entry><?dblatex bgcolor="[rgb]{0.8,1,0.6}" ?>Value</entry>
  <entry>Status</entry>
  <entry xreflabel="This is line 4" id="line4">Comment</entry>
</row>
```

As shown by the source excerpt or by Table 4.2, the color syntax must be directly understandable by latex.

Name	Value	Status	Comment
6541 EFR 75	150	Nominal	Standard
3478 PQA 65	50	Critical	Exhaust anomaly
3798 JHR 19	100	Alarm	Slight fluctuation
7412 NRV 12	None	Out of order	Lack of fuel

Table 4.2: Colored Table

4.3 Sub Table

You can use an `entrytbl` instead of `entry` to insert a subtable in a table cell. Table 4.3 contains a subtable entry.

a1	b1			c1
a2	b2a1	b2b1	b2c1	c2
	b2a2	b2b2	b2c2	
	b2a3	b2b3	b2c3	
a3	b3			c3

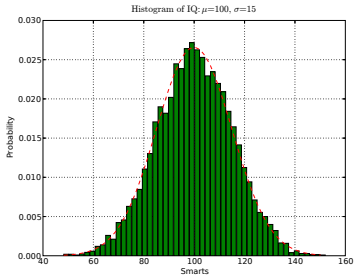
Table 4.3: Table with an `entrytbl`

4.4 Tables and Images in a Table

You can nest the tables, and/or include some images to have some nice output.

a1	a2	a3
b1	b2	b3
c1	c2	c3

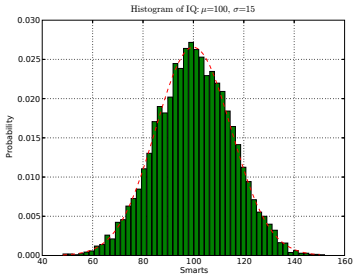
a2



b1

a1	a2	a3
b1	b2	b3
c1	c2	c3

b3



c2

a1	a2	a3
b1	b2	b3
c1	c2	c3

Table 4.4: Tables and Images in Table

4.5 Landscape Table

Table 4.5 show a table in a landscape mode. In addition its role attribute is set to "tiny" that sets a smaller font size.

ID	Type	Name	Trigge	Sigma	% Failure Limit	Unit	Valid	Nb Failures	% Failure	Value Min	Value Max	Mean [+/- sigma]	Abs Mrg Min	Abs Mrg Mean [+/- sigma]	Rel Mrg Min	Rel Mrg Mean	Abs Dep Max	Abs Dep Mean [+/- sigma]	Rel Dep Max	Rel Dep Mean
CATEGORY 1																				
123_IDENTITY_C1	INFO	Isolation Test	1	ras	ras	-	100	0	0.0	1.000	1.000	1.000	-	-	-	-	-	-	-	-
123_IDENTITY_C1	INFO	Isolation Test	1	ras	ras	-	100	0	0.0	1.000	1.000	1.000	-	-	-	-	-	-	-	-
123_IDENTITY_C1	INFO	Isolation Test	1	ras	ras	-	100	0	0.0	1.000	1.000	1.000	-	-	-	-	-	-	-	-
123_IDENTITY_C1	INFO	Isolation Test	1	ras	ras	-	100	0	0.0	1.000	1.000	1.000	-	-	-	-	-	-	-	-
123_IDENTITY_C1	INFO	Isolation Test	1	ras	ras	-	100	0	0.0	1.000	1.000	1.000	-	-	-	-	-	-	-	-
123_IDENTITY_C1	INFO	Isolation Test	1	ras	ras	-	100	0	0.0	1.000	1.000	1.000	-	-	-	-	-	-	-	-
123_IDENTITY_C1	INFO	Isolation Test	1	ras	ras	-	100	0	0.0	1.000	1.000	1.000	-	-	-	-	-	-	-	-
123_IDENTITY_C1	INFO	Isolation Test	1	ras	ras	-	100	0	0.0	1.000	1.000	1.000	-	-	-	-	-	-	-	-
123_IDENTITY_C1	INFO	Isolation Test	1	ras	ras	-	100	0	0.0	1.000	1.000	1.000	-	-	-	-	-	-	-	-

Table 4.5: Landscape Table

Chapter 5

Callouts

Callouts are useful to annotate screenshots, program listings and images to give further explanations.

5.1 Callouts on Images

Figure 5.1 is an image with some callouts. As for now, dblatex seems to be the only processor that can handle callouts on images.

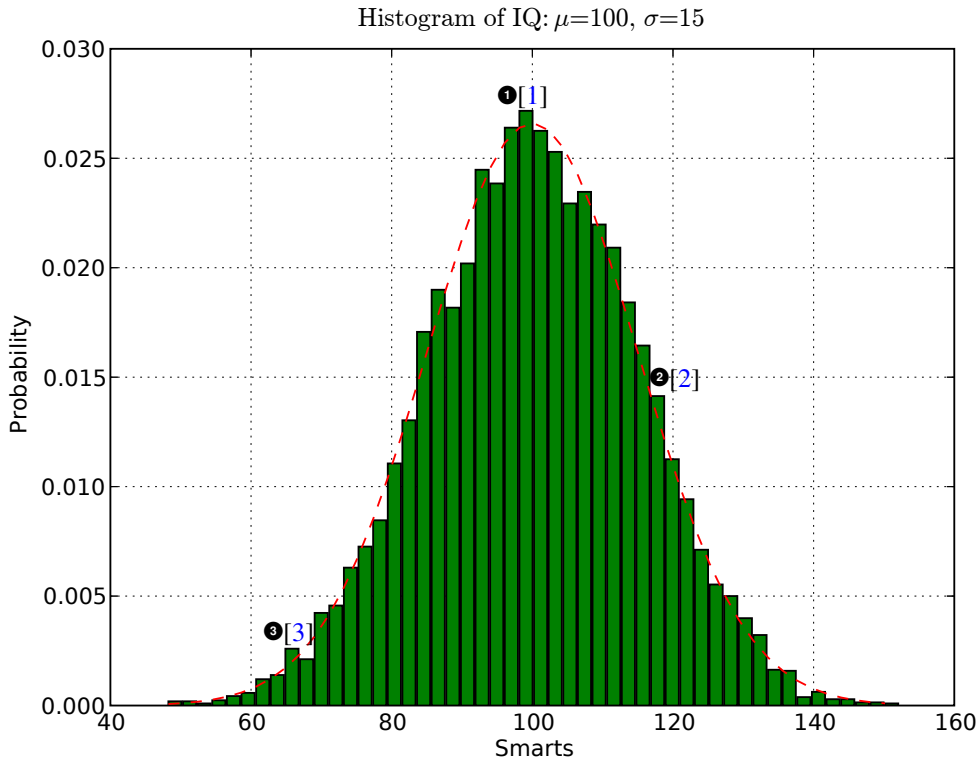


Figure 5.1: Image with Callouts

1. Here is the mean

❷ Here is the upper part

❸ Here is the lower part

5.2 Callouts on Listings

5.2.1 Callouts Embedded in the Listings

Example 5.1 shows some callouts embbeded in a listing.

Example 5.1 Callout in the Listing

```
#include <stdio.h>
#include <stdlib.h> ❶[1, 3]

int main(void) ❷
{
  double *a;

  a = ❸ malloc (10*sizeof *a); ❹

  a[10]=-999; ❺

  free(a);
  a = NULL; ❻

  return EXIT_SUCCESS;
}
```

❶ You have to include `stdlib.h` to get `malloc` prototype otherwise `malloc` is defaulted to 'int'

❷ `main` has to return an error code

❸ Note that you shouldn't cast malloc except if you want to be compatible with C++

❹ Note that you shouldn't explicitely specify the type in sizeof

❺ Segfault ! Unfortunately, array indices begin at 0 in C

❻ It is a good idea to nullify pointers when they are no more needed

5.2.2 Callouts on External Files

Example 5.2 uses an `areaspec` to define the callouts to apply on the external file listing. Of course, the methods to refer to the external file are the same than those described in Section 2.2

This example works with dblatex, but some XSLT processors with some extensions (Saxon, Xalan) are required to work with the DocBook stylesheets.

Example 5.2 Callout on External File

```
#include <stdio.h>
#include <stdlib.h>(1)[1, 3]

int main(void)(2)
{
    double *a;

    a = (3) malloc (10*sizeof *a);(4)

    a[10]=-999;(5)

    free(a);
    a = NULL;(6)

    return EXIT_SUCCESS;
}
```

(1) You have to call 'stdlib' to get malloc prototype otherwise malloc is defaulted to 'int'

(2) main has to return an error code

(3) Note that you shouldn't cast malloc except if you want to be compatible with C++

(4) Note that you shouldn't explicitely specify the type in sizeof

(5) Segfault! Unfortunately, array indices begin at 0 in C

(6) It is a good idea to nullify pointers when they are no more needed

Chapter 6

Bibliography

[R1] John Smith, *First document*, ABCXXXXXX, 23.02.2006.

[R2] John Smith, *Second document*, ABCXXXXXX, 23.02.2006.

Chapter 7

Glossary

This is not a real glossary, it's just an example.

E

Extensible Markup Language (XML)
: Some reasonable definition here. See Also "SGML".

S

SGML
: See "Standard Generalized Markup Language".

Standard Generalized Markup Language (SGML) [ISO 8879:1986]
: Some reasonable definition here. See Also "XML".

Chapter 8

Index