LinuxFocus article number 201
http://linuxfocus.org

by Egon Willighagen
<egonw/at/linuxfocus.org>

About the author:

Got a masters degree in chemistry at the University of Nijmegen, and is doing his PhD research on molecular representation at the same University. Plays basketball and programs Java applications.

Editing DocBook XML Documents

Abstract:

This article describes the use of Kate and its XML plugin as a tool for editing DocBook XML documents.

Introduction

OpenSource XML editing has been long an issue. People need an editor that can produce valid XML, and preferably see the output directly. There are editors that can do this, but they are proprietary. In the OpenSource world there are a few programs that do not give you a nice graphical preview, but that can produce valid XML documents and that can advise you on the elements that are allowed at a specific position in the document.

There is of course Emacs with its DocBook minor mode which works fine (see previous DocBook article). This special mode makes it possible to choose child elements to be placed in the document based on the elements already in the document. And, that allows tab completion based on this knowledge.

But in this article I will not discuss this, but will introduce Kate as a Docbook XML editor instead.

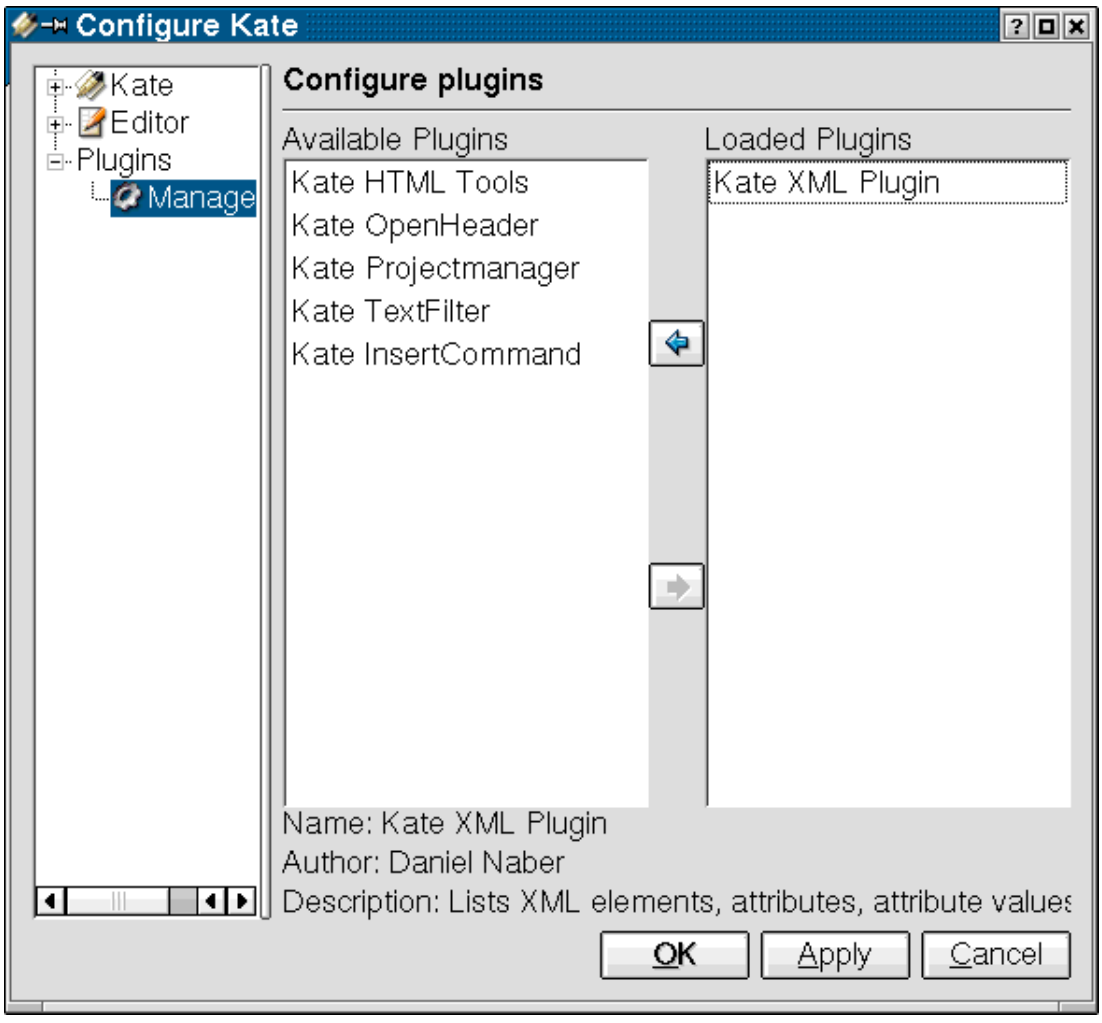
Kate

Kate is one of the editors from the KDE desktop environment. It is reasonably light-weight, supports multiple opened files and has syntax highlighting, also for XML. Moreover, last year an XML plugin was written by Daniel Naber that can help you write valid XML documents. In KDE 3.0 this

XML-plugin for Kate is included in the kde-addon package. If KDE 3.0's addon package is not yet installed, do that by installing one of the many binary distributions of by compiling the package from source:

```
./configure --prefix=/path/where/your/kde3/is/installed
make
sudo make install
```

When the plugin is installed you still need to make Kate aware of this. To do this, select "Configure Kate" in the "Settings" menu. In the "Plugins" manager you can place the XML plugin in the list of loaded plugins:



Building the Meta DTD

Meta DTDs are generated with Norman Walsh's dtdparser, of which packages can be downloaded from SourceForge.net.

I've used an slightly adapted version 2.0beta6. For example, I had to correct the path in the dtdparse

program in the first line to state the correct path to my Perl installation. Running the program would give output like:

```
> ./dtdparse /path/to/docbookx.dtd
Public ID: unknown
System ID: /usr/share/sgml/docbook/dtd/xml/4.1.2/docbookx.dtd
SGML declaration: unknown, using defaults for xml and namecase
Loading dbnotnx.mod
Loading dbcentx.mod
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOamsa.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOamsb.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOamsc.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOamsn.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOamso.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOamsr.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISObox.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOcyr1.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOcyr2.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOdia.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOgrk1.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOgrk2.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOgrk3.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOgrk4.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOlat1.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOlat2.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOnum.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOpub.ent
Loading /usr/share/sgml/entities/xml-iso-entities-8879.1986/ISOtech.ent
Loading dbpoolx.mod
Loading calstblx.dtd
Loading dbhierx.mod
Loading dbgenent.mod
Parse complete.
Writing docbookx.dtd.xml...
Done.
```

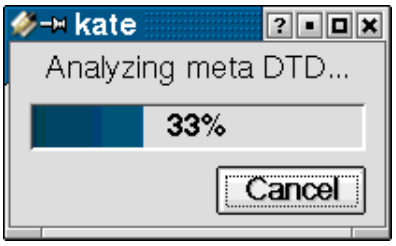
Note the large number of entities that are loaded. Partly because of this, the resulting Meta DTD is 1.63 Mibibytes large. But with an XSLT processor like xsltproc this can be sized down to 0.96 MiB with the simplify_dtd.xsl stylesheet from the kde-addon package:

```
> xsltproc simplify_dtd.xsl docbookx.dtd.xml > docbook-xml-4.1.2.dtd.xml
```

The resulting docbook-xml-4.1.2.dtd.xml can then be copied to /path/to/kde3/share/apps/katexmltools/ where /path/to is changed to the path where KDE3 is actually installed. But your homedir will do too, as the XML plugin will ask for a filename when assigning a new Meta DTD (see below).

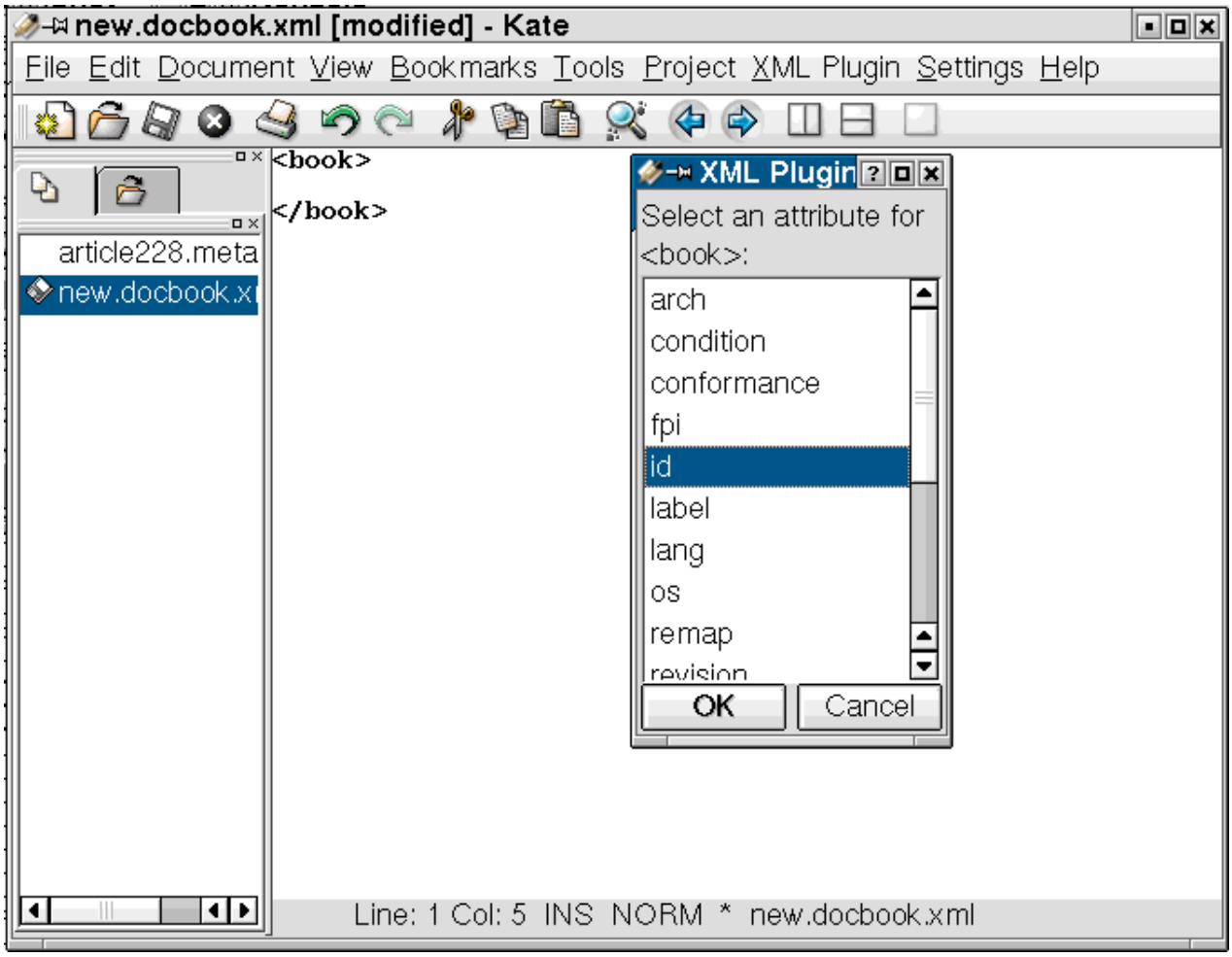
Editing

When starting a new DocBook XML document a Meta DTD needs to be assigned. This is done by choosing "Assign Meta DTD" in the "XML Plugin" menu. A file choose dialog will pop up and you can select the DTD you want to use. In this case, this is the DocBook XML 4.1.2 Meta DTD we just build. A new dialog will show stating the progress:



In the new, empty document we type "<book>". To close the element we type F11. If syntax highlighting is not turned on yet, you can do this manually by selecting "Highlight Mode" in the "Document" menu. XML syntax highlighting is in the "Markup" submenu.

Because we forgot which attributes the book element has, we place the cursor at the end of the word book in the starting tag and press Ctrl+Return. A pop up will show you a list of all possible attributes for the current element:

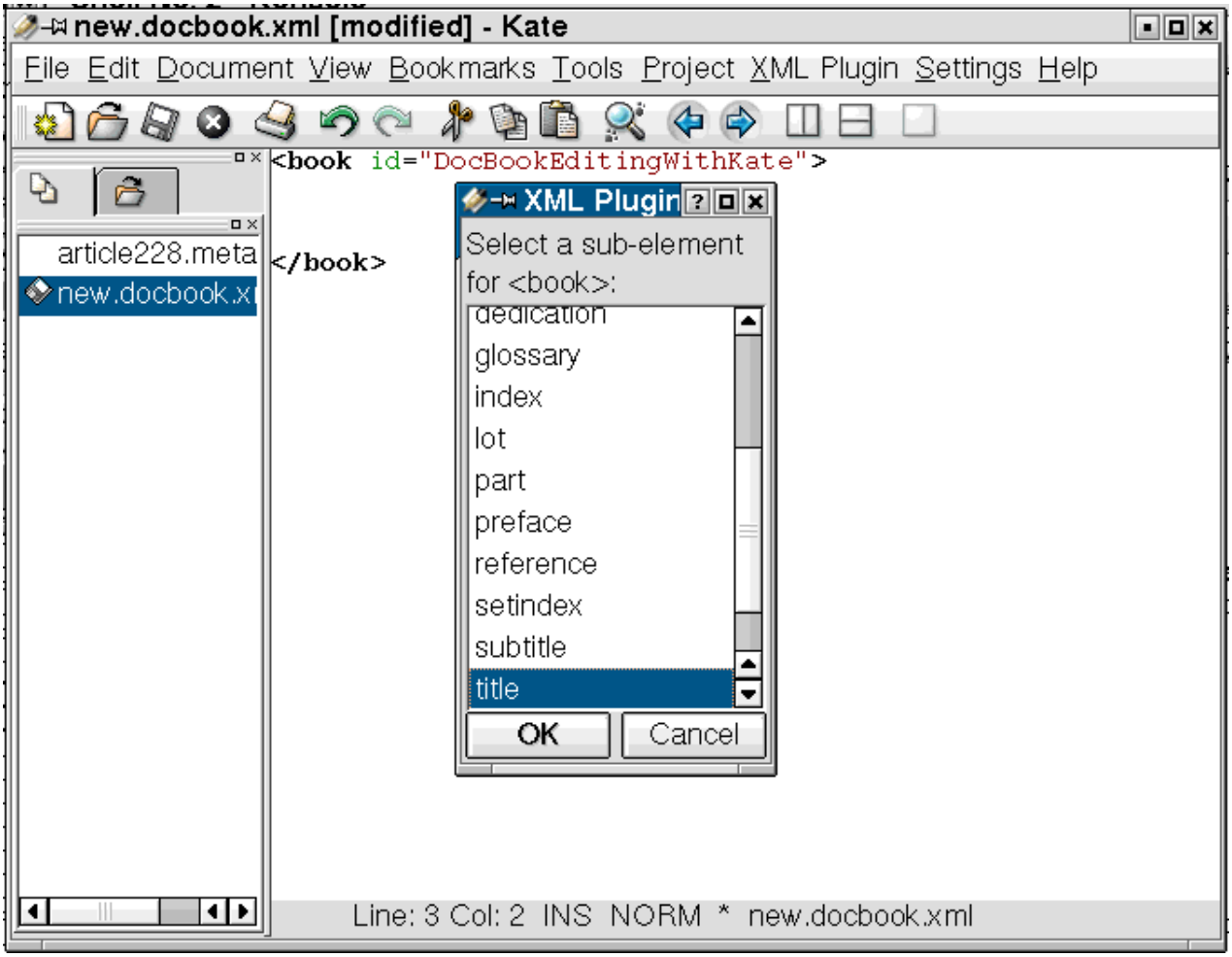


Placing the cursor between the "'s and pressing Ctrl+Return even gives you the possible values for the attribute if they are defined in the DTD. This is not the case for the id attribute.

After inserting the attribute we have this DocBook XML code:

```
<book id="SomeID">
</book>
```

By placing the cursor between the start and end tag of the book element and pressing Ctrl+Return at that position we get a new pop up list, but now with all possible child elements for the book element. We can select, for example, the title element:



One downside of this plugin is that it is only aware of child elements, and not about the order or times is is allowed to occur. For example, the DTD does not allow two or more title elements, but the XML plugin does not warn you about this. Since the plugin also does not have a validate option (yet), it is still very easy to get invalid documents.

A summary of the functionality of the plugin is given in the table below:

Task	Command
Insert XML (elements *and* attributes)	Ctrl+Return
Insert Entity	F10
Close Tag	F11

KDE DocBook

The KDE project itself also uses DocBook as a format for their documentation. As such the XML plugin comes by default with a KDE customized version of DocBook. Assigning that Meta DTD for DocBook is of course preferred over the Meta DTD for DocBook XML 4.1.2 if writing KDE documentation.

MathML and SVG

DocBook nowadays also supports MathML for including mathematical equations in documents. A mixed DTD for validating DocBook documents with MathML code can be found in this email message.

SVG's DTD has unfortunately not been written in such a way that it is possible to include it in DocBook. But efforts to make this possible are undergoing.

Use in Kate's XML plugin requires making a new Meta DTD based on the combined DTD with dtdparse.

Conclusion

The XML plugin for Kate can help you a lot when editing DocBook XML documents. It is not perfect, but then again, it was developed only recently. With KDE 3.0 it is available for very many people.

2005-01-14, generated by lfparser_pdf version 2.51