The SCons Docbook DTD

Dirk Baechle

Table of Contents

1. Purpose 1
2. Identifiers 1
3. Usage for SCons examples 1
4. Additional elements 2
    4.1. Top-level 2
    4.2. sconstruct 3
    4.3. scons_example 4
    4.4. scons_example_file 5
    4.5. scons_output/scons_output_command 6
5. Configurations for visual editing 6
    5.1. XMLMind editor 6
    5.2. Syntext Serna Free 7

1. Purpose

Based on Docbook v4.5, this DTD extends the standard by a few elements. They are used for automatic processing of SCons examples in the User's guide.

2. Identifiers

The public and system identifiers of this DTD are:

```
"-//SCONS//DTD DocBook V4.5-Based extension V1.0//EN"
"http://www.scons.org/dtd/scons.dtd"
```

3. Usage for SCons examples

An "SCons example" looks like this, and essentially describes a set of input files (program source files as well as SConscript files):

```
<scons_example name="ex1">
  <file name="SConstruct" printme="1">
    env = Environment()
    env.Program('foo')
  </file>
  <file name="foo.c">
    int main() { printf("foo.c\n"); }
  </file>
</scons_example>
```

The `<file>` contents within the `<scons_example>` tag will get written into a temporary directory whenever example output needs to be generated. By default, the `<file>` contents are not inserted into

text directly, unless you set the "printme" attribute on one or more files, in which case they will get inserted within a <programlisting> tag. This makes it easy to define the example at the appropriate point in the text where you intend to show the SConstruct file.

Note that you should usually give the <scons_example> a "name" attribute so that you can refer to the example configuration later to run SCons and generate output.

If you just want to show a file's contents without worry about running SCons, there's a shorter <sconstruct> tag:

```
<sconstruct>
  env = Environment()
  env.Program('foo')
</sconstruct>
```

This is essentially equivalent to <scons_example><file printme="1">, but it's more straightforward.

SCons output is generated from the following sort of tag:

```
<scons_output example="ex1" os="posix">
  <scons_output_command>scons -Q foo</scons_output_command>
  <scons_output_command>scons -Q foo</scons_output_command>
</scons_output>
```

You tell it which example to use with the "example" attribute, and then give it a list of <scons_output_command> tags to execute. You can also supply an "os" tag, which specifies the type of operating system this example is intended to show; if you omit this, default value is "posix".

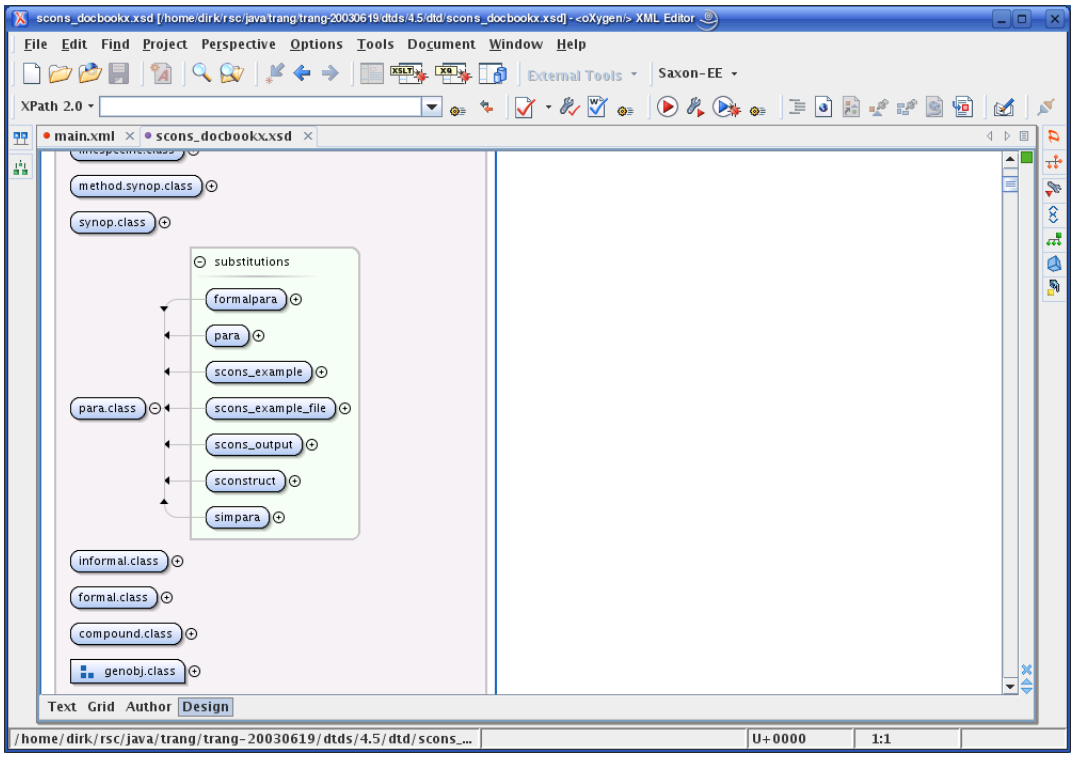
The generated SGML will show the command line (with the appropriate command-line prompt for the operating system), execute the command in a temporary directory with the example files, capture the standard output from SCons, and insert it into the text as appropriate. Error output gets passed through to your error output so you can see if there are any problems executing the command.

4. Additional elements

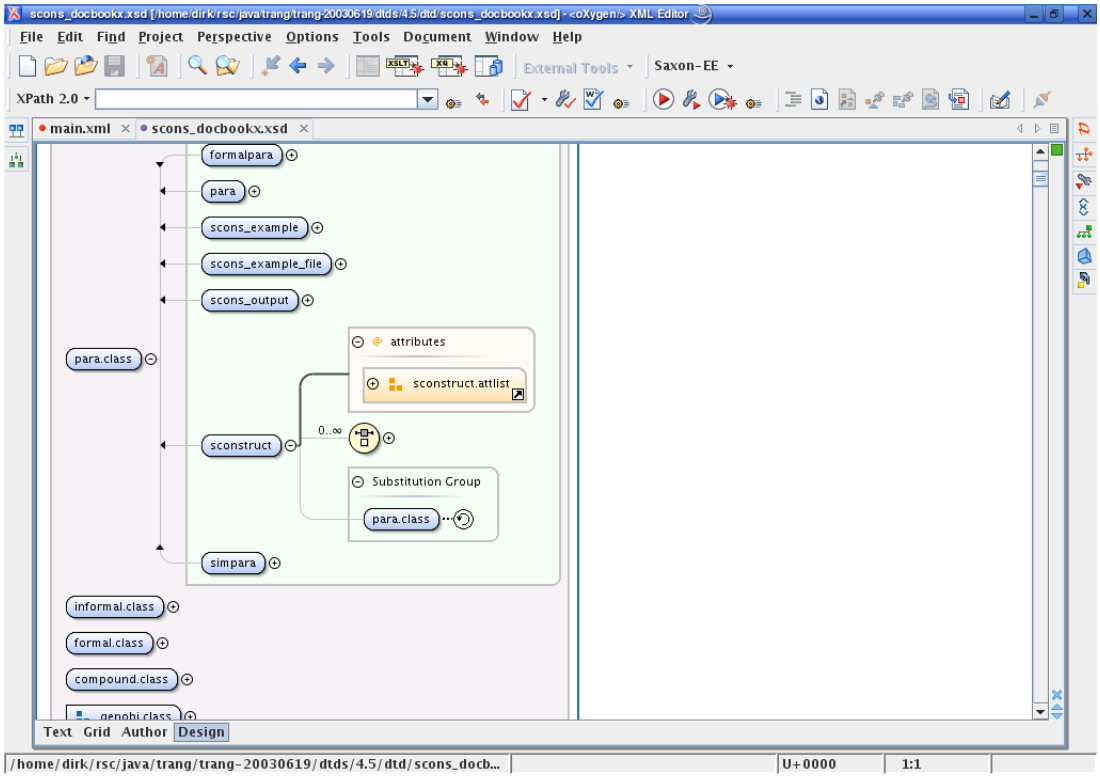
In the following, we shortly list the added elements (or tags) for reference. The screenshots were taken while preparing the DTD with a trial version of *oXygen* 11.0.

4.1. Top-level

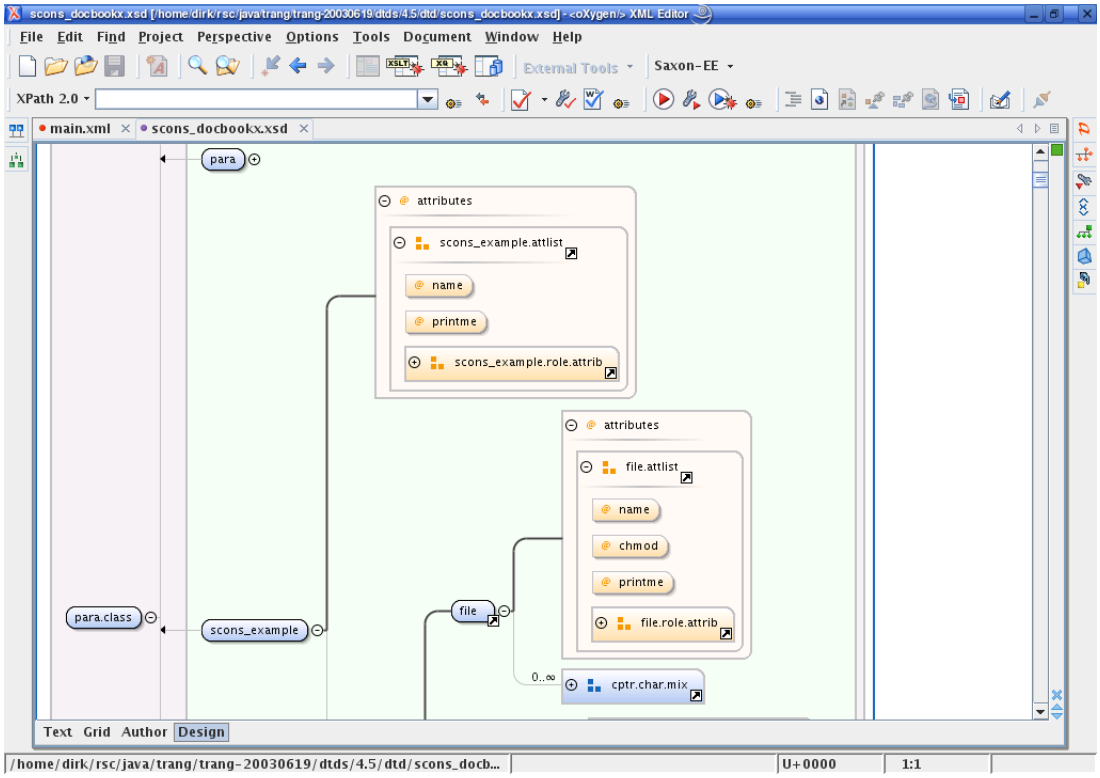
The top-level tags scons_example, scons_example_file, sconstruct and scons_output are added to the para.class, alongside para and simpara. This means that these tags can be inserted in any place, where a paragraph would be valid too.



4.2. sconstruct

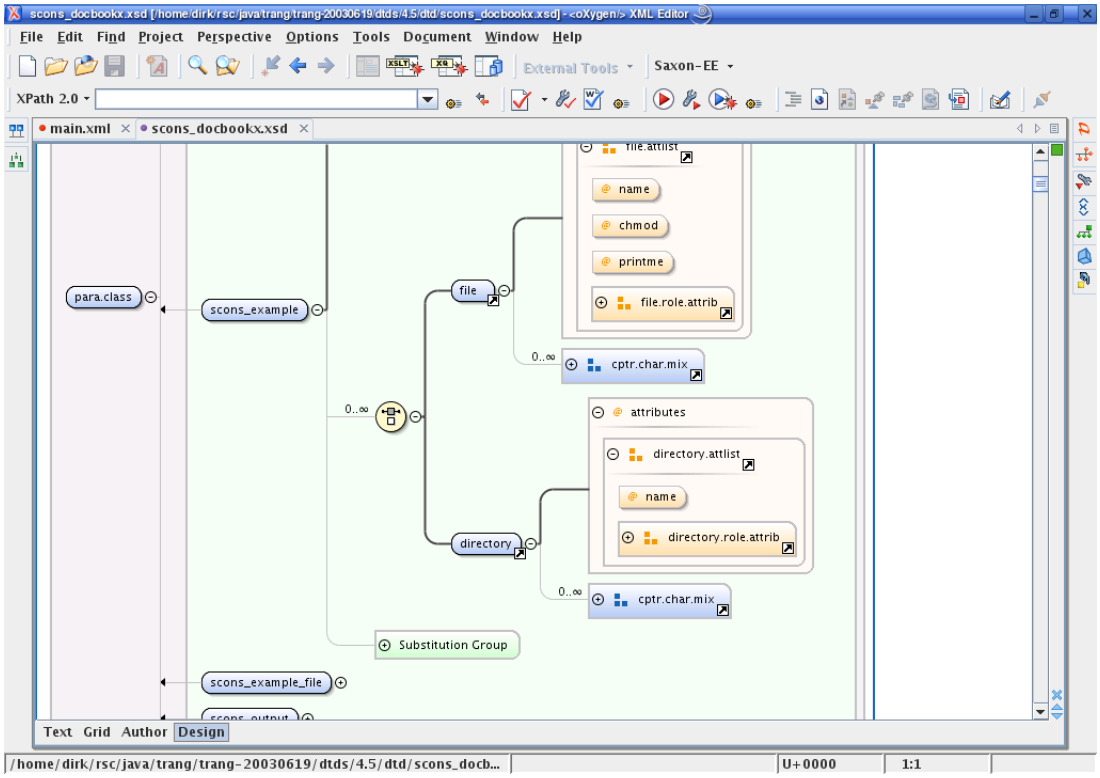


4.3. scons_example

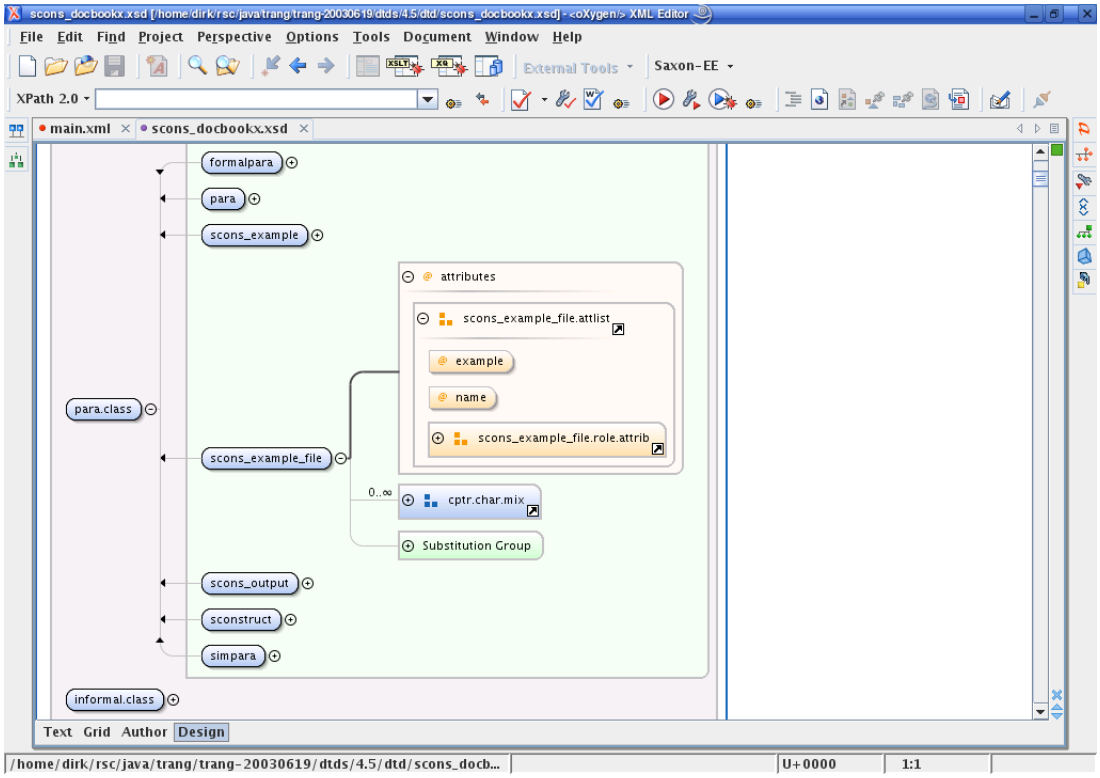


An `scons_example` can hold an arbitrary number of `file` and `directory` entries.

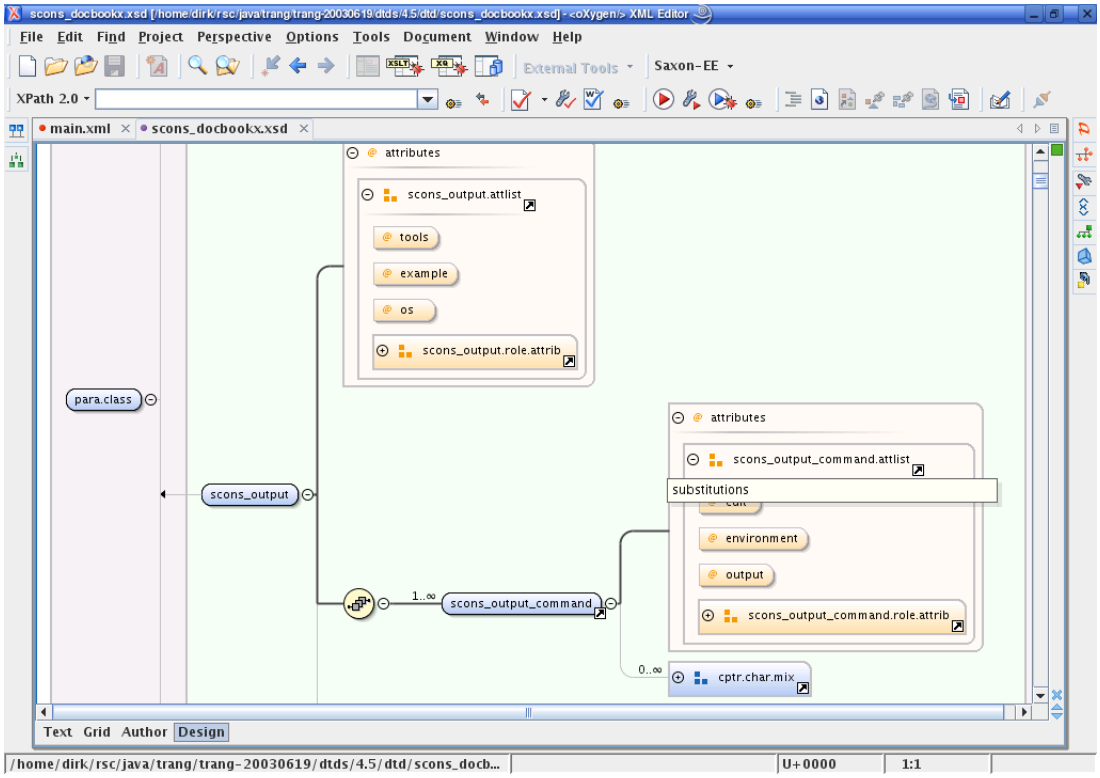
4.3.1. file/directory



4.4. scons_example_file



4.5. scons_output/scons_output_command



5. Configurations for visual editing

Configuration sets for two visual XML editors have been prepared. These are *XmlMind* and *Syntext Serna*, both available in free versions that offer good WYSIWYG like editing. Other, more text-oriented solutions like Editix or Quanta may be used with success too (check the list of DocBook Authoring tools [http://wiki.docbook.org/topic/DocBookAuthoringTools] for an exhaustive list).

Finally, I also found Howto create a visual Docbook Editor in 10 Minutes [http://relation.to/Bloggers/HowToCreateAVisualDocBookEditorIn10Minutes], a short description about how to use the VPE/JBOSS plugin of Eclipse to preview XML documents. But I did not follow this route any further...any volunteers out there?

5.1. XMLMind editor

Homepage	http://www.xmlmind.com/xmleditor
Tested version	Personal Edition 4.5.2

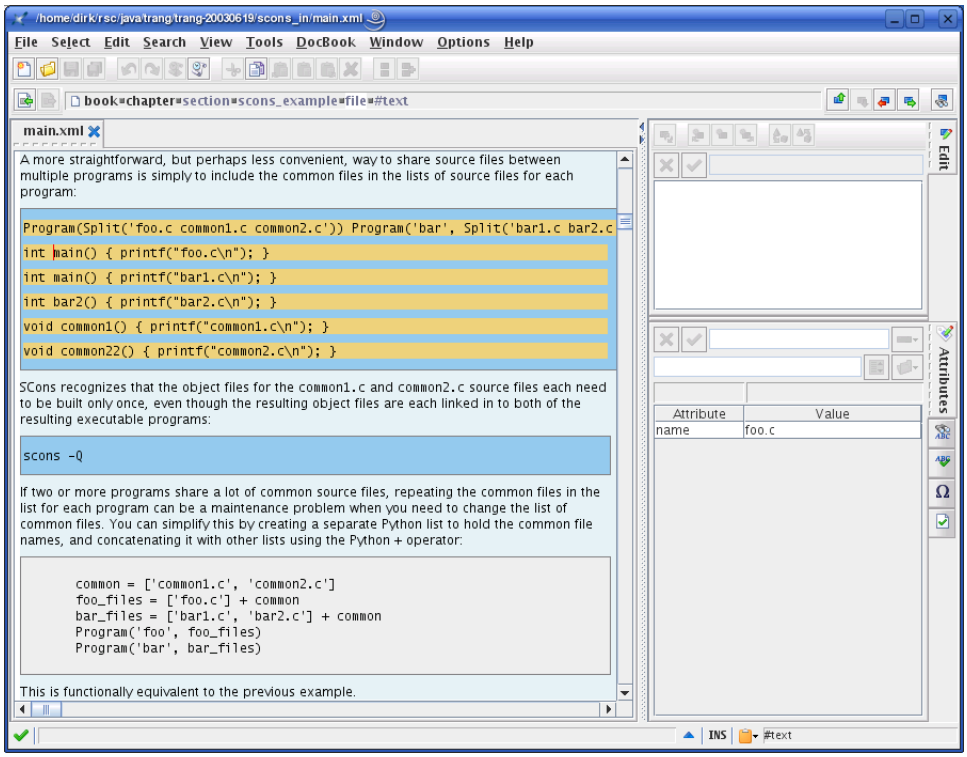
This editor comes in a free version, containing all the basic features that you will need for editing Docbook documents. So don't hesitate to give it a try.

Copy the contents of the "`xxe4`" folder in the archive to your user's XXE config directory. This is located at

- "`$HOME/.xxe4/`" on Linux.
- "`$HOME/Library/Application Support/XMLmind/XMLEditor4/`" on the Mac.

- `%APPDATA%\XMLmind\XMLEditor4\` on Windows 2000, XP, Vista.

After a restart of the program, the SCons DTD is picked up automatically, and you can even select it when inserting new Books, Articles,... whatever.

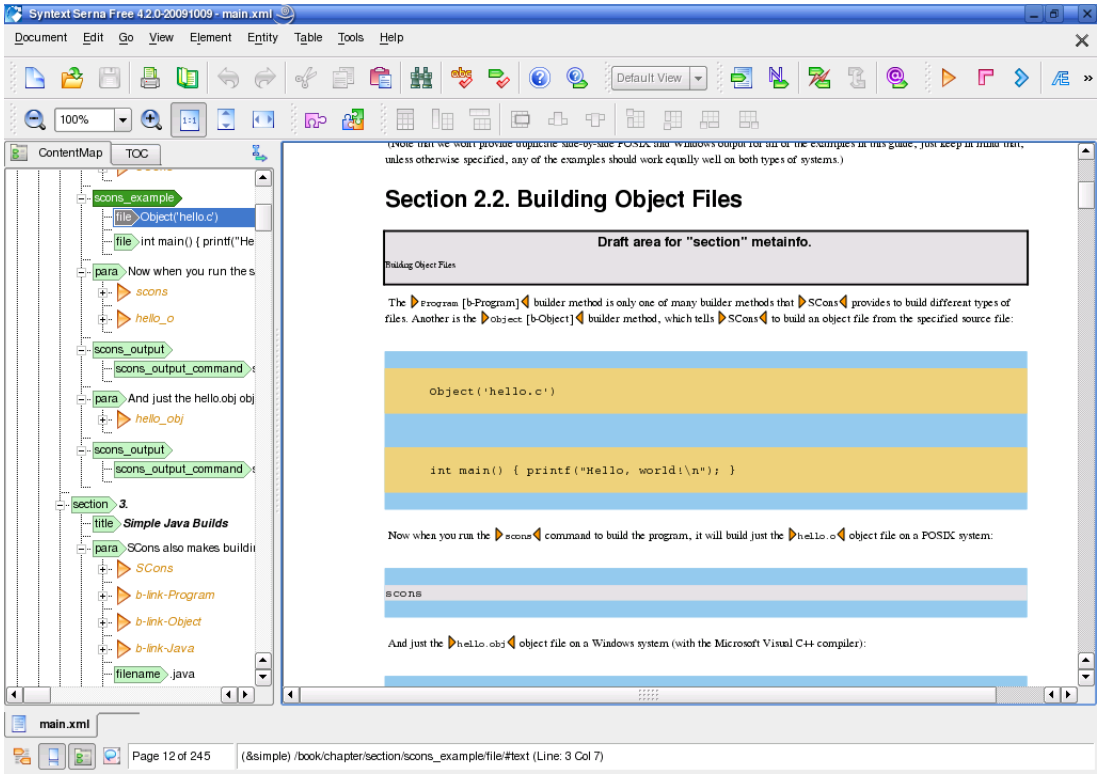


5.2. Syntext Serna Free

Homepage	http://www.syntext.com/downloads/serna-free/
Tested version	4.2.0

This is a good alternative to the XmlMind editor, however its "Insert tag" philosophy did not convince me that much. But your mileage may vary...

Copy the prepared "`scons`" addon folder to the "`plugins`" directory of your *Serna* installation. Depending on where and how you installed the program, you might need root access for this step.


Syntext Serna Free 4.2.0-20091009 - main.xml
Document Edit Go View Element Entity Table Tools Help
Default View
100%
ContentMap TOC
scons_example
file Object('hello.c')
file int main() { printf("He
para Now when you run the s
scons
hello_o
scons_output
scons_output_command
para And just the hello.obj obj
hello_obj
scons_output
scons_output_command
section 3.
title Simple Java Builds
para SCons also makes buildi
SCons
b-link-Program
b-link-Object
b-link-Java
filename java
unless otherwise specified, any of the examples should work equally well on both types of systems.)
Section 2.2. Building Object Files
Draft area for "section" metainfo.
Building Object Files
The Program [b-Program] builder method is only one of many builder methods that SCons provides to build different types of files. Another is the Object [b-Object] builder method, which tells SCons to build an object file from the specified source file:
Object('hello.c')
int main() { printf("Hello, world!\n"); }
Now when you run the scons command to build the program, it will build just the hello.o object file on a POSIX system:
scons
And just the hello.obj object file on a Windows system (with the Microsoft Visual C++ compiler):
main.xml
Page 12 of 245
(&simple) /book/chapter/section/scons_example/file/#text (Line: 3 Col 7)