The Scite – TEX integration

Abstract
Editors are a sensitive, often emotional subject. Some editors have exactly the properties a software designer or a writer desires and one gets attached to it. Still, most computer experts such as TEX users often are use three or more different editors each day. Scite is a modern programmers editor which is very flexible, very configurable, and easily extended. We integrated Scite with TEX, CONTEXT, LATEX, METAPOST and viewer and succeeded in that it is now possible to design and write your texts, manuscripts, reports, manuals and books with the Scite editor without having to leave the editor to compile and view your work. The article describes what is available and what you need with special emphasis on highlighting commands with lexers.

About Scite

Scite is a source code editor written by Neil Hodgson. After playing with several editors we found that this editor is quite configurable and extendible. At PRAGMA ADE we use TEXEDIT, an editor written long ago in Niklaus Wirth's MODULA as well as a platform independent reimplementation of it called TEXWORK written in PERL/TK. Although our editors possess some functionality that is not (yet) present in Scite, we decided to use Scite because it frees us from the editor maintenance chore.

Installing Scite

Installing Scite is straightforward. We assume below that you use MS WINDOWS but for other operating systems installation is not much different.

First you need to fetch the archive from:

```
www.scintilla.org
```

The MS WINDOWS binaries are in `wscite.zip`, and you can unzip this in any directory as long as the binary executable ends up in your `PATH` or as shortcut icon on your desktop.

The TEX lexer

Scite provides several so called 'lexers'. A lexer does the syntax highlighting of your document. The TEX commands that are used in instructing the compiler on how to typeset your document can be given a color. And the various brackets that need to be balanced, such as '`{ } [ ] ( )`' also get a color. The way a TEX file is treated is configurable and can be found in the file:

```
tex.properties
```

You can edit this file to your needs using the menu entry under `options` in the top bar of Scite. In this file, the following settings apply to the TEX lexer:

```
lexer.tex.interface.default=0
lexer.tex.use.keywords=1
lexer.tex.comment.process=0
lexer.tex.auto.if=1
```

The option `lexer.tex.interface.default` for example determines the way TEX keywords are highlighted. You can control the interface from your document as well by placing the line below as the first line in it. This often makes more sense than editing the configuration file to your incidental needs.

```
% interface=all|tex|nl|en|de|cz|it|ro|latex
```

The values in the properties file and the keywords in the preamble line above have the following meaning:

0	`all`	all commands (preceded by a backslash)
1	`tex`	TEX, ε-TEX, PDFTEX, OMEGA primitives (and macros)
2	`nl`	the dutch CONTEXT interface
3	`en`	the english CONTEXT interface
4	`de`	the german CONTEXT interface
5	`cz`	the czech CONTEXT interface
6	`it`	the italian CONTEXT interface
7	`ro`	the romanian CONTEXT interface
8	`latex`	LATEX (apart from packages)

The configuration file is set up in such a way that you can easily add more keywords to the lists. The keywords for the second and higher CONTEXT command language and LATEX interfaces are defined in their own properties files:

```
cont-nl-scite.properties
...
cont-en-scite.properties
latex-scite.properties
```

The CONTEXT distribution comes with a file:

```
context.properties
```

as well as the interface specific files. There are two ways to make sure that the extra CONTEXT and LATEX keywords are loaded.

Under the menu entry 'options', you will find a long list of property filenames. By opening one of them, you can determine the location of these files by looking at your windows banner (at the top). You should copy the following files to that path, perhaps named `c:\scite\wscite`.

```
cont-*-scite.properties
latex-scite.properties
```

The first line `*-scite.properties` files define the CONTEXT keywords, and the second line configures Scite for LATEX. If you need another brand of TEX file to be processed, you can tweak the properties, or safer: redefine some of them in your `SciTEUser.properties`. For example plain TEX users may want:

```
file.patterns.tex=*.tex;*.sty;
filter.tex=TeX|$(file.patterns.tex)|
lexer.$(file.patterns.tex)=tex
command.compile.$(file.patterns.tex)=
command.build.$(file.patterns.tex)=tex $(FileNameExt)
command.go.$(file.patterns.tex)=gv $(FileName).pdf
```

LATEX users may want:

```
file.patterns.latex=*.tex;*.sty;*.aux;*.toc;*.idx;
filter.latex=LaTeX|$(file.patterns.latex)|
lexer.$(file.patterns.latex)=tex
command.compile.$(file.patterns.latex)=
command.build.$(file.patterns.latex)=pdflatex $(FileNameExt)
command.go.$(file.patterns.latex)=gv $(FileName).pdf
```

A CONTEXT user needs:

```
file.patterns.context=*.tex;*.tui;*.tuo;*.sty;
filter.context=ConTeXt|$(file.patterns.context)|
lexer.$(file.patterns.context)=tex
command.compile.$(file.patterns.context)=
command.build.$(file.patterns.context)=texexec --pdf $(FileNameExt)
command.go.$(file.patterns.context)=gv $(FileName).pdf
```

For CONTEXT users these definitions are already assembled in a special properties file:

```
context.properties
```

Put it in the same location as your `SciTEUser.properties`, and in this user file add the following line:

```
import context
```

Now you have many more commands available. Familiarize yourself with these new options and take a look into this file. Beware: this setup assumes that you have the Latin Modern Typewriter font on your system and that your operating system is aware of that. If Windows is unaware then locate the Latin Modern Fonts in the `texmf-extra` tree on the latest TEX Live distribution of the TEX user groups. Install the Latin Modern files on Windows by copying the `pfb` and `pfm` files to the fonts path on your system, in most cases this is:

```
c:\windows\fonts
```

You can add or locally change options after the line that loads `context.properties`. If you did not copy the `cont-*.properties` files to the Scite properties path, you can put them in the same path as `SciTEUser.properties`, in which case you have to add:

```
import latex-scite
import context
```

Try it and find out that Scite can easily be tuned to your preferences.
The instructions for context users are:

fetch `wscite.zip` from `www.scintilla.org`
create a path `c:\scite`
unzip `wscite.zip` in `c:\scite`
copy `context.properties` to `c:\wscite`
open `c:\scite\wscite\SciTEUser.properties` (using Scite)
at the end, add the line `import context.properties`
if needed, add `c:\scite\wscite` to your path
if needed, create shortcut to `c:\scite\wscite\scite.exe`

The CONTEXT related properties files are not included in the Scite distribution but instead are part of the CONTEXT distribution which can be found in one of the following places:

```
../tex/texmf/context/data
../tex/texmf-local/context/data
```

We generate the interface specific property files automatically from the CONTEXT interface definition files, while the xx file (in the CONTEXT zip file) is hand–crafted and contains missing or very special bits and pieces.
Let us return to the powerful properties options in `tex.properties`. For testing purposes you can disable keyword coloring with:

```
lexer.tex.use.keywords=0
```

You can also influence the way comment is treated with:

```
lexer.tex.comment.process=0
```

When set to zero, comment is not interpreted as TEX code and it will come out in a uniform color. But, when set to one, you will get as much color as a TEX source. The lexer tries to cope with the TEX syntax as well as possible and takes care of the `^^` notation. A special treatment gets `\if`:

```
lexer.tex.auto.if=1
```

This is the default setting. When set to one, all `\ifwhatever`'s will be seen as a command. When set to zero, only the primitive `\if` will be treatedas such. When this property is set to one, the lexer will not color an `\ifwhatever` that follows an `\newif`.

The METAPOST lexer

The METAPOST lexer is set up slightly differently from its TEX counterpart, because METAPOST is more a true language than TEX is although, as with the TEX lexer, we can control the interpretation of identifiers. The METAPOST specific configuration file is:

```
metapost.properties
```

Here are located properties like:

```
lexer.metapost.interface.default=1
```

Instead of editing the configuration file you can control the lexer with the first line in your document:

```
% interface=none|metapost|mp|metafun
```

The numbers and keywords have the following meaning:

0	`none`	no highlighting of identifiers
1	`metapost or mp`	METAPOST primitives and macros
2	`metafun`	MetaFun macros

Similar to the TEX lexer, you can influence the way comments are handled:

```
lexer.metapost.comment.process=1
```

This will interpret comment as METAPOST code, which is not that useful (opposite to TEX, where documentation is often coded in TEX).

The lexer will color the METAPOST keywords and — when enabled — additional keywords (like those of MetaFun) also, and shown in a slanted font. These MetaFun keywords are defined in yet another separate file:

```
metafun-scite.properties
```

You can either copy this file to the path where your global properties files is located, or put a copy in the path of your user properties file. In that case you again need to add an entry to in file `SciTEUser.properties`:

```
import metafun-scite
```

The lexer recognizes `btex ... etex` pairs and will treat anything in between as just text. The same happens with strings (placed between `"`). Both act on a per line basis.

Epilogue. This completes our special TEX installation instructions for the Scite integrating editor. We believe Scite to be a well designed, flexible editor allowing the perfect integration of TEX and METAPOST systems. An hour or less suffices to discover the convenience of this system. The next hour shows some minor weak points on which we are working to reinforce them.

Hans Hagen
pragma@wxs.nl