

Submode for Documented L^AT_EX Source (DTX) for WinEdt 5.5

<http://www.winedt.org/Config/menus/DTX.php>

R Schlicht

w.m.l@gmx.net

Revision: 1.12

14th January 2008

Contents

Contents	1
1 Overview	2
2 Installation	2
3 Mode	2
4 Highlighting	2
5 Command Completion	3
6 Menus & Active Strings	3
7 Shortcuts	4
8 Tree directives	5
9 Regular Expression filter	6
10 Customization	6
11 Sample file	7
12 Uninstalling	7

1 Overview

The DTX submode contains highlighting settings, menu customisations, active strings and tree directives for DTX files. It will also make WinEdt acquainted with the structure of DTX files, so that some often-used shortcuts will work as expected. Furthermore, it will dynamically highlight all commands defined in the DTX file and make them available for command completion.

2 Installation

Extract the zip file to %b\Config\DTX (where %b is your local directory, e. g., C:\Documents and Settings\<you>\Application Data\WinEdt) and run the macro _install_DTX.edt.

The DTX submode requires WinEdt 5.5, build 20060814, or newer. Some components *might* work with builds as old as 20041213, however, if anything doesn't work – which is actually more than likely – you are on your own.

The DTX submode also requires the macro addOrUpdate.edt and the dynamicKeywords package, both available from www.winedt.org. Both have to be saved in %b\Macros\macro (or %B\Macros\macro).

3 Mode

Files with the extension .dtx (and .fdd) will automatically be recognised as being of the mode 'TeX:DTX'. That is, DTX is a submode of T_EX mode, inheriting its highlighting settings etc., only adding some features. Most notably, Wrapping will be turned off. (For files that have been opened before this package was installed, you might have to do this manually.)

In contrast to WinEdt's default settings, files with the extensions .sty, .ins and .fd will not be regarded as DTX file, since they, – well, they aren't. Instead, a 'STY' submode will be installed, which only differs from the normal T_EX mode in that wrapping and global switches are turned off.

4 Highlighting

The following switches will be highlighted (additionally to the default T_EX highlighting):

- documentation parts (i. e., lines starting with a single percent character)
- guards (e. g., '%<driver>', '%<*package>', '%</package>')
- '____\begin{macrocode}', '____\end{macrocode}'
- '|verbatim text|'
- '^A comments'

Furthermore, all commands that you define in the DTX file will be highlighted and made available for command completion. (You may need to call **Update Dynamic Keywords** in the Tools menu.)

5 Command Completion

In WinEdt's default settings, the \TeX primitives and basic \LaTeX commands are available for command completion (**CTRL-Enter**). The DTX mode will additionally make all macros that are defined in the DTX file (those enclosed in a macro environment) known to WinEdt.

After typing the beginning of a `\command` (at least the backslash), you can complete it by pressing **CTRL-Enter**. If there is more than one possible completion, you'll be prompted with a list to choose from. This also works for command names without backslash inside `\cs{...}`.

If no matching command is found, the default word completion will step in.

6 Menus & Active Strings

Since this mode will install the `dynamicKeywords` package, a new menu entry **Update Dynamic Keywords** will be added to the Tools menu if it doesn't already exist.

Using this command, you can update dynamic highlighting and command completion of the defined macros in the DTX file. (You can also use it for all other modes that take advantage of it. See www.winedt.org.)

A DTX submenu will be added to the **Insert** menu. All entries in the menu are also available as Active Strings (when typed at the beginning of the line).

- **Macrocode environment** (Active String: `%\`)`

will insert a code block:

```
% \begin{macrocode}
```

```
% \end{macrocode}
```

- **Documentation block** (Active String: `%%%`)`

will insert a documentation block:

```
% \end{macrocode}
```

% *

```
% \begin{macrocode}
```

- **Macro environment** (Active String: `%mm`)`

will insert a macro environment and ask you for its name. Afterwards, dynamic keywords will be updated (unless you change a line in `DTX.cfg`).

```
%\begin{macro}{*}
```

```
%\end{macro}
```

- Environment environment (Active String: ‘%ee’)
will insert a environment environment and ask you for its name:

```
%\begin{environment}{*}
```

```
%\end{environment}
```

- Close Guard (Active String: ‘%>>’)
will close the last opened guard.
- Changes (Active String: ‘%cc’)
will insert a change entry:

```
\changes{<next version>}{<current date>}{*}
```

The <next version> will be determined by (whichever is found first, in this order):

New (1.11)!

- a string ‘% !DTXversion:: "<next version>"’
(or the obsolete ‘% DTXversion: <next version>’)
- the version in the package identifier string (i. e., the second, bracketed argument to \ProvidesPackage), incremented by one
- the version defined in \fileversion, incremented
- the RCS revision (from ‘Revision:’ or ‘Id:’), incremented

You can alter this order in the configuration file DTX.cfg.

New (1.10)!

- CheckSum (also accessible by double-clicking on \Checksum)
will insert (or correct) the checksum (requires the log file).

7 Shortcuts

The following commands and shortcuts will be modified for DTX mode (and only for DTX mode):

Enter	Enter
Par Up	CTRL-Up
Par Down	CTRL-Down
Select Par Up	SHIFT-CTRL-Up
Select Par Down	SHIFT-CTRL-Down
Select Paragraph	SHIFT-CTRL-Ins
Format Paragraph	CTRL-ALT-Ins
Format Document	SHIFT-CTRL-ALT-Ins
Insert Comment	SHIFT-CTRL-ALT-Right
Remove Comment	SHIFT-CTRL-ALT-Left

In fact, they will behave (almost) as you would expect – that is, you can select or navigate through paragraphs, even though they are preceded by %. If you enter a new line and the first character of the current line is a percent character, it will be repeated in the next line with the indentation level retained. Insert/Remove Comment will insert/remove the DTX comment string ^A when inside a documentation block, the percent character otherwise.

Additionally, the active strings `\begin{?}` resp. `\end{ }` (to complete the current environment) will be adjusted for DTX mode.

Some random remarks:

- If you press Enter while at the very beginning of a line, the percent character will not be inserted in the empty line. That's a feature!
- **Format Paragraph** might have a different opinion about where the current paragraph starts and ends than you – in this case first select the lines you want to have formatted.
- The macros know nothing about special \LaTeX commands which would usually make WinEdt believe that a paragraph starts, like `\begin{something}`. Instead, the macros will decide on where a paragraph begins and ends solely by the level of indentation. You can take advantage of this to help the macros, e. g., by writing:

```
%\begin{macro}{something}
% This is the description of something
% which continues here.
```

Because of different indentation levels, the first line would be regarded as a different paragraph and hence would be exempt from being formatted.

- You should not try to format DTX paragraphs if they are not preceded by a comment character. Strange things might happen!
- I know it's not perfect, but that's still better than nothing ...

8 Tree directives

A TOC tree will be set up (cf. figure 1), which knows about the directives:

- `% \section`
- `% \subsection`
- `% \subsubsection`

(with zero to two spaces between percent and backslash character).

Furthermore, a tree of all `%<modules>` will be built. The ending guard will be the last child of the beginning guard. Modules may be nested. If the modules in the DTX file are improperly nested or not closed, you'll receive a warning.

You can select the respective text unit in the source file via the context menu for the section and module items.

Finally, all `\changes` entries will be gathered in the tree. You can change the sorting preferences by right-clicking on any item:

- Group by version (toggle)
- Sort by date, string or occurrence in the file

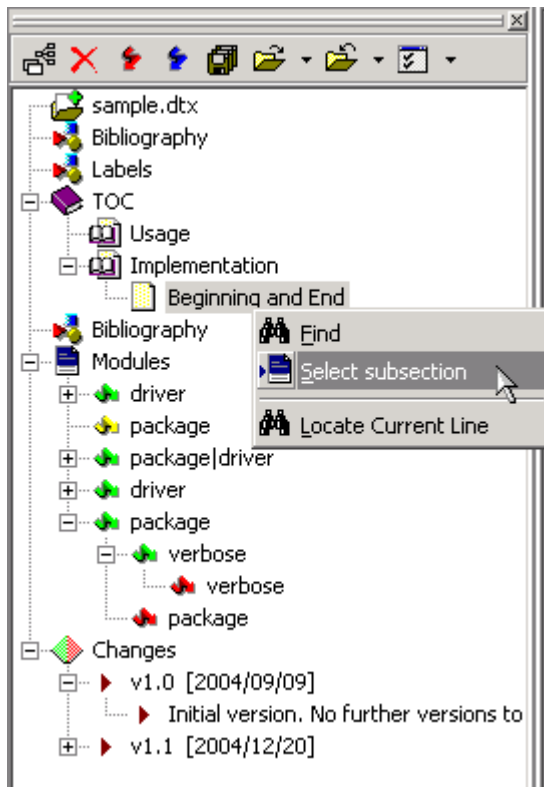


Figure 1:
WinEdt's Tree in DTX mode

9 Regular Expression filter

New (1.12)!

WinEdt 5.5 introduced the nifty Regular Expression `\E` and its opposite `\e`, which may be used to run a macro file after the search string has been found, e.g. to filter the results further. The DTX mode includes one such macro: `DTXdoc.edt`.

If you want to search for an expression (using the Find/Replace dialog), but only if it occurs in the documentation part of your DTX file, append `\E{DTXdoc}` to the expression. Conversely, to only find an expression if it occurs in code blocks, append `\e{DTXdoc}` to the expression. Note that you must check the **Regular Expression** switch in the dialog, which means that you either have to escape special characters with a backslash or enclose the complete main expression in `\"^...^"`, where `^` may be any character not appearing in the enclosed text.

For example, to search for the string `\my@macro{}`, but only if it occurs within code blocks, you could use either of the following expressions:

```
\my\@macro\{\}\e{DTXdoc}
\"+\my@macro{ }+\e{DTXdoc}
```

10 Customization

A number of options can be customised in the configuration file `DTX.cfg`:

- The right margin for formatting paragraphs (default: 80)

- Whether dynamic keywords should be automatically updated after inserting a new macro environment (default: yes)
- The default sorting criteria of the Changes tree
- The order for determining the next version for \changes, and the version prefix (default: v)
- The characters that a macro may consist of (default: \$Alpha+["@"]\$)

11 Sample file

The file `sample.dtx` in the DTX directory is a – surprise! – sample file for the DTX submode.

12 Uninstalling

Running the macro `_uninstall_DTX.edt` will uninstall the DTX submode.

Note that the Dynamic Keywords extensions will not be uninstalled as other packages might rely on them. If you are certain that this is not the case, you may remove the menu item Update Dynamic Keywords in Menu Setup | Main Menu | Tools; in the file `%b\Menus\Tools\Complete Word.edt`, you may remove the lines enclosed with:

```
// @BEGIN: Dynamic Completion
and
// @END: Dynamic Completion
```