Vim Structured Text

Mikolaj Machowski

16 Mar 2005

Author: Mikolaj Machowski

Title: Vim Structured Tex - HTML and LaTeX output

Keywords: Vim, LaTeX, PDF, HTML, XML

Version: 1.0-beta2

Date: 16 Mar 2005

For a long time Vim users were asking for "real" export to HTML. This is, I believe, first real try to achieve this effect. This is dumbed down version of reStructuredText, popular Python language documentation tool (so I borrowed minor parts of its documentation).

Contents

1 Introduction

Structured means Vim script recognizes some patterns and translates them into form recognizable by WWW browsers. In fact, Vim creates quasi-XML form which in future can be transferred into almost any markup language, like LaTeX, RTF or even OpenDocument (OpenOffice.org, KOffice) or MS-Office XML.

Unfortunately, currently VST has one major drawback: no nesting of elements. It prevents creation of very sophisticated documents but as you can see from this document it also gives nice possibilities.

2 Installation

Put file `vst.vim` into your global plugin directory or create special filetype and use it as filetype plugin.

3 Usage

Transformation is called with command:

```
:Vst [{format}]
```

Where {format} is format of exported file. Format argument is optional and without it default value (HTML) will be used. Argument name is case-insensitive: HTML, html, Html are equivalents. Formatted file will be opened in new buffer without name.

3.1 Export to HTML

These commands will create XTHML 1.0 Transitional file:

```
:Vst html
:Vst
```

From special characters/entities VST handles at the moment:

```
<, >, &, (c)
```

Results in: ¡, ¿, &, (c).
List can be extended on request.
By default VST uses internal stylesheet for formatting. You can define `g:vst_external_css` variable and use your own stylesheet with name of `stylesheet.css`.

3.2 Export to LaTeX

These commands will LaTeX version of VST file:

```
:Vst latex
:Vst tex
```

Document will use `hyperref` package and it will be fully hyperlinked. It is better to compile it with `pdflatex`, directly to PDF than use pure `LaTeX`.
Current differences between HTML and LaTeX export:

- Frames around note, tip and warning directives are all black (and thin)
- Preformatted paragraphs inside tip, note, warning are breaking compilation (conflict of verbatim and minipage)
- Striking of text is ignored
- Definition lists have different format - term is in the same line as definition and is bolded (TODO - fix)
- 2html is treated as ordinary preformatted text

3.3 Export to XML

This command will produce XML-like code:

```
:Vst xml
```

This is mainly for debugging purposes.

4 Structure

4.1 Paragraphs

Base unit of text is **paragraph**, text separated by at least one blank line. All lines in paragraphs must have the same indentation. Paragraph indented will be displayed as quotation (blockquote). It is possible to embed any type and any number of elements inside blockquote - respecting their own rules of embedding. Example:

```
This is example of paragraph. This
is continuation of paragraph.

    This is indented paragraph. Looking
    like quoted text.

        This is quoted quoted text. Indented two times.

This is another one.
```

Results in:
This is example of paragraph. This is continuation of paragraph.

> This is indented paragraph. Looking like quoted text.
>
> > This is quoted quoted text. Indented two times.

This is another one.

Embedding of elements is supported for paragraphs, blockquotes, ordered lists, unordered lists and tip, note, warning directives. In all of them can be embedded the rest of one-level elements.

4.2 Text styles

Inside of paragraph (and other text elements you can use another markup - *italics* with "`*italics*`", **bold** with "`**bold**`", "`‘‘double back-quotes‘‘`" for typewriter text and ~~strike~~ "`=strike=`" to cross out the text (useful for dealing with TODOs).

This text is *italicised*.

This text is **strongly emphasised**.

This text is `a code`.

This text was *deleted*.

Only few combinations are supported and support for them may be removed from future versions:

italicised bold:

```
***italicised bold***
```

``italicised literal``:

```
*``italicised literal``*
```

`bold literal` (Note: this is not simple combination):

```
**`bold literal`**
```

italicised strike:

```
*=italicised strike=*
```

bold strike:

```
**=bold strike=**
```

Order of marks is important.

If you find that you want to use one of the "special" characters in text, it should be OK - VST can deal with most typical situations. For example, this * asterisk is handled just fine. If you actually want text *surrounded by asterisks* to **not** be italicised, then you need to indicate that the asterisk is not special. You do this by placing a backslash just before it, like so "". Remember: special treatment of these few characters is entering inline literals - even there you have to escape it with double backslash:

```
``\*``
```

For another method of manipulating font check Roles.

4.3 Special characters

Sometimes it is necessary to escape special treatment of some characters (or give that meaning). Then you have to put backslash
before them.

Remove space:

```
this that
```

Result: thisthat

Do not italicise word:

```
not *this* word
```

Result: not *this* word
Do not strike word:

```
not =strike= word
```

Result: not =strike= word
Insert non-breaking space:

```
non-breaking-space
```

Result: non breaking space (` ` in HTML)

4.4 Roles

VST supports additional methods of text manipulation. They are called *roles*. Usual form is:

```
:name:‘text‘
```

Roles are requiring white spaces or non-word characters around them. You can escape them so they will be:

```
H :sub:‘2‘ O
```

Result: H_2O
It looks awkwardly, especially if repeated many times in text. You can help it with Replacement:

```
|H2O|

.. |H2O| H :sub:‘2‘ O
```

Result is: H_2O

4.4.1 Subscript

This role will show $_{smalltextbelow}$ normal line of text:

```
:sub:‘small text below‘
```

4.4.2 Superscript

This role will show ^{smalltextover} normal line of text:

```
:sup:‘small text over‘
```

4.4.3 Big

This role will show some bigger text:

```
:big:‘bigger text‘
```

4.4.4 Small

This role will show some smaller text:

```
:small:‘smaller text‘
```

4.5 Lists

Lists of items come in three main flavours: **enumerated**, **bulleted** and **definitions**. List element can contain only one paragraph.

Lists must always start a new paragraph – that is, they must appear after a blank line.

4.5.1 Enumerated lists

Start a line off with a number or letter followed by a period ".", or right bracket ")". Following forms are recognised:

```
1. numbers

A. upper-case letters
   and it goes over many lines but **not** paragraphs or other
   elements like sub-lists

a. lower-case letters

1) numbers again
```

Results in:

1. numbers

A. upper-case letters and it goes over many lines but **not** paragraphs or other elements like sub-lists

a. lower-case letters

1. numbers again

VST is taking numeration of first element as numeration of whole list. This code:

```
a. roman list
2. decimal list
```

Will be rendered as:

a. roman list

b. decimal list

Two lists of the same type must have separator between them. In other cause they will be rendered as one:

```
1. List1 Elem1
2. List1 Elem2

1. List2 Elem1
2. List2 Elem2
```

Results in:

1. List1 Elem1
2. List2 Elem2
3. List2 Elem1
4. List2 Elem2

Placing between them empty comment is enough. Anyway, short description is always good thing.

4.5.2 Bulleted lists

Just like enumerated lists, start the line off with a bullet point character - either "-", "+" or "*":

```
* a bullet point using "*"

- list using "-"

+ yet another list
```

Results in:

- a bullet point using "*"
- list using "-"
- yet another list

These elements are connected. `*` bulleted list always will be `circle`, `-` will be `disc` and `+` will be `square`.

4.5.3 Embedding of lists

Enumerated and bulleted lists can contain many elements and can be nested. This code will be rendered:

```
1. This is description how to make lists embeddable
   one into other.

       - start one list
       - insert blank line and bigger indentation
       - start another list
       - insert blank line before next element

2. It is possible to embed paragraphs into list (and blockquotes)
   also.

   Paragraphs and blockquotes have to be separated by blank line and
   start where list begins: indentation of list "leader" plus
   "leader", punctuation sign and space.

        That is memorable quote.

    Those features are not implemented for other types of elements.
    Only: plain paragraphs, blockquotes, ordered lists, bulleted
    lists.

    Indentation level is *very* important. One space can break
    things.
```

Results in:

1. This is description how to make lists embeddable one into other.

   - start one list
   - insert blank line and bigger indentation
   - start another list
   - insert blank line before next element

2. It is possible to embed paragraphs into list (and blockquotes) also.

   Paragraphs and blockquotes have to be separated by blank line and start where list begins: indentation of list ”leader” plus ”leader”, punctuation sign and space.

   > That is memorable quote.

   Those features are not implemented for other types of elements. Only: plain paragraphs, blockquotes, ordered lists, bulleted lists.

   Indentation level is *very* important. One space can break things.

4.5.4 Definition lists

Unlike the other two, the definition lists consist of a term, and the definition of that term. The format of a definition list is:

```
what
    Definition lists associate a term with a definition.

*how*
    The term is a one-line phrase, and the definition is one
    paragraph, indented relative to the term.  Blank lines are not
    allowed between term and definition.
```

Results in:

what Definition lists associate a term with a definition.

how The term is a one-line phrase, and the definition is one paragraph, indented relative to the term. Blank lines are not allowed between term and definition.

4.6 Field list

Special kind of list designed for headers of files or highlighting important information. Paragraph in form:

```
:Author: Mikolaj Machowski
:Something: Somewhere
```

Results in:

Author: Mikolaj Machowski

Something: Somewhere

Vim recognizes some names of field list as special and places them in meta info of created documents:

- Author
- Title
- Date
- Subject
- Keywords

4.7 Preformatting (code samples)

To just include a chunk of preformatted, text, finish the prior paragraph with "`::`". The preformatted block is finished when the text falls back to the same indentation level as a paragraph prior to the preformatted block. For example:

```
An example::

      Whitespace, newlines, blank lines, and all kinds of markup
        (like *this* or this) is preserved by literal blocks.
    Lookie here, I've dropped an indentation level
    (but not far enough)

no more example
```

Results in:
An example:

```
  Whitespace, newlines, blank lines, and all kinds of markup
    (like *this* or this) is preserved by literal blocks.
Lookie here, I've dropped an indentation level
(but not far enough)
```

no more example
Note that if a paragraph consists only of "`::`", then it's removed from the output:

```
::

   This is preformatted text, and the
   last "::" paragraph is removed
```

Results in:

```
This is preformatted text, and the
last "::" paragraph is removed
```

4.8 Sections

To break longer text up into sections, you use **section headers**. These are a single line of text (one or more words) with adornment: an underline, in dashes "`-----`", equals "`======`", tildes "`~~~~~~`" or any of the non-alphanumeric characters `= - ` ~ ^ * + #` that you feel comfortable with. The underline must be at least as long as the title text. Be consistent, since all sections marked with the same adornment style are deemed to be at the same level:

```
Chapter 1 Title
===============

Section 1.1 Title
-----------------

Subsection 1.1.1 Title
~~~~~~~~~~~~~~~~~~~~~~

Section 1.2 Title
-----------------

Chapter 2 Title
===============
```

To indicate the document title, use a unique adornment style at the beginning of the document. To indicate the document subtitle, use another unique adornment style immediately after the document title. For example:

```
 Document Title
================

 Subtitle
----------

Section Title
~~~~~~~~~~~~~

...
```

4.9 Links

Links are important part of modern document. VST allows to create external and internal links.

4.9.1 Starting point

Starting point looks always identical:

```
We explained ‘starting point‘_ somewhere else

The same for ‘start‘_
```

Note: even when start is single word it has to be enclosed into backticks.

4.9.2 External links

Definition of external target:

```
.. _starting point: http://www.vim.org
.. _start: http://skawina.eu.org/mikolaj
```

Note: lack of backticks around titles.

4.9.3 Internal links

Definition of internal target can be done in two ways.
First is to put definition in text:

```
some text about _‘starting point‘ explaing this term
```

Backtics are obligatory.
Second way is anonymous target:

```
.. _starting point:
```

Very similar to external target just pointing nowhere so here.
Very important: link definitions are case sensitive. They have to be always the same:

```
‘start‘_
_‘start‘
```

will be connected but:

```
‘Starting point‘_
.. _starting point: http://www.vim.org
```

no.

4.9.4 Standalone links

Links can be put directly into text when written explicitly in text:

```
This link to http://www.smith.info page by mailto:john@smith.info .
```

Results in:
This link to http://www.smith.info page by john@smith.info .
Check Special characters how to escape necessary space at the end of example.

4.10 Transitions

It is a nice touch to separate some paragraphs and parts of text visually. In some old-fashioned books it is done with small graphics, in newer with eg. row of asterisks `* * *`.

In VST you can do this with line of letters, preferred are characters used for sections underscoring:

```
==============================================

----------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**********************************************
```

etc. It have to be separated from other elements with blank lines. In exported file they will look like straight line:

---

5 Directives

You can achieve special treating of some paragraphs by using directives. They have always a form of:

```
.. directive:
```

You can include in your text images, comments or footnotes. At the moment only one paragraph of text can be contained in directive.

5.1 Images

To include an image in your document, you use the the `image` directive. For example:

```
.. image: test.png
```

Spaces in filename should be avoided, but can work properly on standard settings.

You can supply additional information about image with sub-directives:

```
.. image: test.png
   width: 200
   height: 100
   alt: Alternative text
   title: Title of image
```

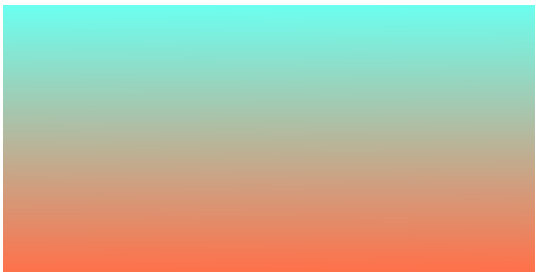

Figure 1: Title of image

Results in:

Getting info about image dimensions is boring. You can use special subdirective `identify:` which uses program from ImageMagick suite of programs (available on most OS where Vim is available):

```
.. image: test.png
   identify:
   alt: Alternative text
   title: Title of image
```

`identify:` can handle argument which will serve as scale factor. 100 is scale 1:1, scale will decrease size of image **only in document**. Real size of image will not change:

```
.. image: test.png
   identify: 50
   alt: Alternative text
   title: Title of image
```

It is possible to make image a link with subdirective `target:`:

```
.. image: test.png
   target: http://www.vst.info/test.png
```

It will make image a link to other image. When you are scaling image view it is a good idea to make it clickable and point to full scale version - possible with special argument `self`:

```
.. image: test.png
   identify: 50
   target: self
```

Results in:

Figure 2: Title of image

5.2 Comments

Sometimes you would like to insert some text which should not be visible in exported document, but can be useful in text version like Vim modeline:

```
.. comment: vim:set tw=72:
```

or hiding folding markers:

```
.. comment: {{{
.. comment: }}}
```

These lines will be in exported format, just not visible.

5.3 Footnotes

You can include in text special links to fragments which don't match into current paragraph, and place those fragments wherever you want in document. Example:

```
This doesn't belong here[1]_.

.. [1] I will describe it here.
```

Results in:
This doesn't belong here[1].
There are numbered footnotes, try to keep them in order to not disonteriate readers. VST will not warn about duplicate footnotes.

5.4 Table of contents

For longer text it is good idea to put in document table of contents. In VST you can place table of contents at desired position with directive:

```
.. toc:
```

In exported document it will be replaced by unordered list with elements indented to present structure of document.

This directive automatically places also title. Default is 'Table of Contents'. If you want give title in your language add this to directive (for Polish):

[1] I will describe it here.

```
.. toc: Spis tresci
```

I have omitted Polish diacritics to avoid encoding problems.

Directive `toc` accepts subdirective `depth:` which argument is level of headers shown in table of contents:

```
.. toc:
   depth: 3
```

Will show in table of contents only headers down to 3rd level.

5.5 Replacement

This is an exception from general format of directive:

```
.. |from| into
```

It consists of three parts: leading commas, source part enclosed in — and the rest of line which will replace source in file. Beware: `from` and `into` elements are processed through `substitute()` Vim function and have to be proper Vim regexps. Three characters will be escaped automatically: `&` `~`.

5.6 Tip

This directive can contain many various elements:

```
.. tip:

   First paragraph.

   1. List element
   2. List element

   Second paragraph
```

Results in:

Tip

First paragraph.

1. List element
2. List element

Second paragraph

All elements must have bigger indentation than directive. These elements will be placed in grey frame and with title 'Tip'. You can give your title to frame as argument to "`.. tip:`". Empty line between directive and first element is obligatory.

5.7 Note

This directive can contain many various elements:

```
.. note:

    First paragraph.

        Remember these noble words.

    Second paragraph
```

Results in:

> **Note**
> First paragraph.
>
> Remember these noble words.
>
> Second paragraph

All elements must have bigger indentation than directive. These elements will be placed in black frame and with title 'Note'. You can give your title to frame as argument to "`.. note:`". Empty line between directive and first element is obligatory.

5.8 Warning

This directive can contain many various elements:

```
.. warning:

    First paragraph.

    - List element
    - List element

    Second paragraph
```

Results in:

> **Warning**
> First paragraph.
>
> - List element
> - List element
>
> Second paragraph

All elements must have bigger indentation than directive. These elements will be placed in red frame and with title 'Warning'. You can give your title to frame as argument to "`..  warning:`". Empty line between directive and first element is obligatory.

5.9 2html

This feature will work only in gVim. In other cases directive will be treated as "::".

Directive designed to make use from `2html.vim` script for syntax highlighting of code:

```
.. 2html: [{filetype}] [{colorscheme}]
```

`filetype` Argument will set proper highlighting for the following fragment of code (usually those snippets will be too short for automatic recognition). May be omitted, following paragraphs will be treated as normal code snippets - without coloring. Obligatory if you want to declare `colorscheme`.

`colorscheme` Will set colorscheme used for syntax highlighting. If omitted current colorscheme will be used. If you started without colorscheme and declared `colorscheme` argument it will be set. There is no way to return automatically apart from saving and restoring all settings. Maybe in future.

Following paragraphs have to be separated by blank line and have bigger indentation.

5.9.1 Examples

Fragment of `vst.vim` in blue colorscheme:

```
.. 2html: vim blue

    if exists('depth') && depth != ''
        let hdepth = strpart(g:ptype[j], '1')
        if hdepth > depth
            let j += 1
            continue
        endif
    endif
```

Result:

```
if exists('depth') && depth != ''
    let hdepth = strpart(g:ptype[j], '1')
    if hdepth > depth
        let j += 1
```

```
        continue
    endif
endif
```

The same fragment in manxome:

```
if exists('depth') && depth != ''
    let hdepth = strpart(g:ptype[j], '1')
    if hdepth > depth
        let j += 1
        continue
    endif
endif
```

And fragment of Vim itself - ex_cmds.c by murphy. Declaration:

```
.. 2html: c murphy

if (fp_out != NULL)
{
(void)mch_setperm(tempname,
          (long)((st_old.st_mode & 0777) | 0600));
/* this only works for root: */
(void)chown((char *)tempname, st_old.st_uid, st_old.st_gid);
}
```

6 ChangeLog

- 11 Mar 2005 - Initial announcement on vim@vim.org
- 11 Mar 2005 - added `depth:` subdirective to Table of contents directive
- 11 Mar 2005 - added `target:` subdirective to Images directive
- 11 Mar 2005 - added Replacement directive
- 12 Mar 2005 - added Big role
- 12 Mar 2005 - added Small role
- 12 Mar 2005 - added Superscript role
- 12 Mar 2005 - added Subscript role
- 12 Mar 2005 - added more Text styles
- 12 Mar 2005 - added way to escape Special characters
- 13 Mar 2005 - added multi element directives: Tip, Note, Warning

- 13 Mar 2005 - 2html directive for coloring of code
- 13 Mar 2005 - recursive nesting inside blockquotes possible!
- 14 Mar 2005 - recursive nesting in ordered and unordered lists possible!
- 15 Mar 2005 - initial LaTeX support
- 16 Mar 2005 - polishing LaTeX support
- 16 Mar 2005 - take info from Field list and add meta info for HTML and LaTeX (PDF)