Latex-Suite – die ultimative Anleitung

Marco Strehler

23.07.2014

1 Einführung

In der Referenz zur Latex-Suite ist zwar alles vorhanden, aber vieles wird nur sehr knapp erwähnt und mit verschiedenen Problemen wird der User alleine gelassen.

Diese Anleitung soll es dem Einsteiger in die Latex-Suite einfach machen und ist aus den Notizen entstanden, die ich mir – zur Ergänzung zur Reference – selbst gemacht habe.

Konventionen in diesem Dokument:

- vim-Hilfe: immer wenn auf das Hilfe-System von vim hingewiesen wird (aufgerufen aus dem Normal-Modus mit dem Befehl `h: help`, wobei `help` hier das gesuchte Hilfethema ist). Wird genau diese Notation gewählt: also z.B. `h: latex-suite` oder `h: tutorial`.
- Die Einstellungen werden jeweils als kleines Listing dargestellt.

2 Konfigurations-Dateien

Um die Sache nicht allzu einfach zu machen, muss man seine Konfigurationen in verschiedenen Dateien vornehmen.

vimrc Konfigurationsfile für vim (`h: vimrc`).

tex.vim Konfigurationsfile für globale Variablen der Latex-Suite. Pfad: `~/.vim/ftplugin/tex.vim`. **Datei existiert nicht, muss von Hand erzeugt werden.**

tex_macros.vim Definitionen für Macros (IMAP). Pfad: `~/.vim/after/ftplugin/tex_macros.vim`. **Datei existiert nicht, muss von Hand erzeugt werden.**

bib.vim Konfiguraitonsfile für bib-Files. Pfad: `~/.vim/ftplugin/bib.vim`.

3 Templates

3.1 Templates einfügen

Siehe `h: latex-suite-templates`.

Ein neues LaTeX-Dokument beginnt man am besten mit einem Dokument-Gerüst (Template). Das Vorgehen ist dabei folgendermassen:

1. In vim ein neues Dokument erstellen. Wichtig ist dabei die Endung `.tex`, damit die Latex-Suite aktiviert wird.
2. mit `:TTemplate<CR>` eine Liste mit den vorhandenen Vorlagen im oben erwähnten Template-Ordner anzeigen lassen. Mit der Eingabe der entsprechenden Nummer und <CR> wird das Template an der Cursorstelle eingefügt. Alternativ kann der Template-Namen gleich angegeben werden. Also z.B. `:TTemplate meinTemplate`. Gut zu wissen: die Auto-Vervollständigung mit der Tabulator-Taste funktioniert hier ebenfalls. Mit `:TTe<Tab>` sowie `m<Tab>` bekommt man ebenfalls `:TTemplate meinTemplate`.

Bei mir funktioniert das Template-Menü in MacVim nicht korrekt. Genauer: die Nummern der Templates wird angezeigt, allerdings nicht der Template-Name, was das Menü praktisch unbrauchbar macht. Da ich aber sowieso der Meinung bin, dass vim möglichst über die Tastatur bedient werden soll, ist das ein verschmerzbarer Bug.

3.2 Templates anpassen, eigene Templates einfügen

Die LS-Referenz gibt an, dass die Templates im Verzeichnis `$VIM/ftplugin/latex-suite/templates` zu finden sind. Das ist aber ein Verzeichnis, dass es so nicht gibt. Dabei ist `$VIM` das Kürzel für das Benutzerverzeichnis, das auf unterschiedlichen Betriebssystemen unterschiedliche Pfade aufweist.

Auf meinem Mac gelange ich aus dem Terminal mit dem Befehl

```
cd ~/.vim/ftplugin/latex-suite/templates/
```

dorthin. Wobei das Zeichen ~ (Tilde), wiederum eine Abkürzung für das Userverzeichnis ist. Erzeugt wird es mit cmd-N.

Die Referenz erwähnt weiter, dass Templates *dynamic elements* enthalten können in der Form von Vim-Ausdrücken. Leider schweigt sich die Referenz darüber aus, wie das zu bewerkstelligen ist. Die vorhandenen Beispiele scheinen Befehle zwischen zwei `!comp!`-Ausdrücken zu platzieren. Siehe Listening 1.

Weitere Informationen zu vim-Expressionen holt man sich mit der internen Hilfe mit den Befehlen `:h expression` oder `:h usr_41.txt`.

Das bereits vorhandene Beispiel im Ordner `templates`:

Listing 1: Template report.tex

```
1 <+       +>       !comp!   !exe!
2 %          File: !comp!expand("%")!comp!
3 %       Created: !comp!strftime("%a %b %d %I:00 %p %Y ").substitute(strftime('%Z
      '), '\<\(\w\)\(\w*\)\>\(\W\/$\)', '\1', 'g')!comp!
4 % Last Change: !comp!strftime("%a %b %d %I:00 %p %Y ").substitute(strftime('%Z
      '), '\<\(\w\)\(\w*\)\>\(\W\/$\)', '\1', 'g')!comp!
```

Listing 2: Template-Header, wie er als Makro verwendet werden kann.

```
<+        +>        !comp!   !exe!
% Datei: !comp!expand("%")!comp!
% Begonnen: !comp!strftime("%d.%m.%Y ")!comp!
% Letzte Aenderung: !comp!strftime("%d.%m.%Y ")!comp!


%
\documentclass[a4paper]{report}
\begin{document}
<++>
\end{document}
```

Wenn beim Templete Zeileneinzüge verwendet werden, dann kommt es zu einem hässlchen Treppeneffekt.

3.3 Hinweise zu eigenen Templates

Beim Erstellen von eigenen Templates stellen sich m.E. zwei grundsätzliche Fragen:

- Wie viel soll ich in ein Template hineinpacken? Das meint, wie viele Strukturen sollen schon fertig für den Gebrauch in so einer Vorlage vorhanden sein (z.B. schon einige Frame-Umgebungen bei der Beamer-Klasse).
- Soll ich für eine Dokumentklasse verschiedene Templates mit verschiedenen Optionen als separate Dateien abspeichern oder soll ein Template mit verschiedenen Optionen (zum Beispiel mit der Möglichkeit des Auskommentieren) zur Verfügung gestellt werden?

Ich empfehle *wenig* Strukturen (also kurze Dateien). Insbesondere weil für mich repetivie Codes in ein Makro gehören.

Aus ähnlichen Gründen versuche ich möglichst, Optionen als Auskommentierte Alternativen zu gestalten. Das reduziert die vorhandenen, und zu pflegenden Templates.

Also möglichst *kurze* und *wenig* Templates erzeugen.

Am besten speichert man sich ein persönlicher Template-Header als Makro ab (Listing 2) und kann ihn so in schon vorhandene Templates bequem einfügen. In das Templates können dann noch die von der Latex-Suite unterstützten Platzhalter `<+Platzhalter+>` eingebaut werden. Mit Ctrl-J kann dann bequem von Platzhalter zu Platzhalter gesprungen und die wichtigsten Daten eingegeben werden.

In vim können alle Templates zur Bearbeitung mit dem Befehl

```
:arga  /.vim/ftplugin/latex-suite/templates/*.tex
```

geöffnet und in die Argumenten-Liste geladen werden. So können alle (schon vorhandenenen Templates) an die Latex-Suite angepasst werden. Dabei hilft ein Template-Header, der in ein Makro gepackt wird (Listing 2).

Siehe dazu auch `:h vim-arguments`.

Noch offen: Unterschiede vim, gvim? Wieso werden in MacVim die Menüs nicht angezeigt? Namenskonvention?

4 Makros

4.1 Spezielles Mapping IMAP

Die LS kommt mit einer speziellen Mapping-Technik, die sich vom normalen Mapping im Einfüge-Modus unterscheidet. Grundsätzliches dazu findet sich mit `:h map-overviw` oder im File `~/.vim/plugin/imaps.v`

4.2 Platzhalter

Die Makros der LS benutzen ein geniales Platzhalter-System. Die Platzhalter sind dort, wo bei einem eingefügten Makro das nächste, relevante Text eingefügt werden muss. Der Platzhalter besteht aus `<+` als «öffnende» Klammer und `+>` als «schliessende» Klammer. Dazwischen kann eine kurzer Memotext stehen, der dem Platzhalter noch einen Sinn gibt (`<+Titel+>`). Mit der Tastenkombination `Ctrl-J` kann dann vorwärts von Platzhalter zu Platzhalter gehüpft werden. Mit einem aktivierten Platzhalter ist man im Select-Modus (`:h select-mode`).

Im Select-Modus kann gleich des gewünschte eigetippt werden. Der Select-Modus kommt einem als Mac- oder Windows-User am bekanntesten vor: anlog zur Auswahl, die mit dem nächsten Tastendruck «überschrieben» wird.

Eine geniale Einrichtung, die einem unheimlich viel Bewegungsbefehle spart.

4.3 Makros ein- und ausschalten

Über die Variable `Imap_FreezeImap` lässt sich die Makro-Funktion ein- und ausschalten.

- `:let b:Imap_FreezeImap=1` schaltet die Makros im aktuellen Buffer aus
- `:let b:Imap_FreezeImap=0` schaltet die Makros im aktuellen Buffer ein
- `:let b:Imap_FreezeImap` zeigt den aktuellen Status an. Allerdings erst, wenn die Variable erstmalig initialisiert (d.h. auf 0 oder 1 gesetzt) wurde. Ansonsten erfolgt eine Fehlermeldung (E121: Undefined variable: b:Imap_FreezeImap).

Mit `g:Imap_FreezeImap` können die Makros systemweit ein- und ausgeschaltet werden.

4.4 Makros für LaTeX-Umgebung

4.4.1 Umgebung oder Paket einfügen mit F5

Die Taste `F5` – gedrückt im Einfüge- oder Normal-Modus – hat abängig davon, ob sie auf einer leeren oder mit einem Umgebungs-Schlüsselwort versehen Zeile gedrückt wird unterschiedliche Auswirkung. Des weiteren spiel es eine Rolle ob man sich in der Preambel oder im Dokumentteil des LaTeX-File befindet.

Mit dem Cursor auf einer *leeren* Zeile wird mit `F5` eine Liste mit Schlüsselwörter angezeigt. Diese stellen entweder Paketnamen (in der Präambel) oder LaTeX-Umgebungen (im Dokumentteil) dar. Die entsprechende Nummer wird gewählt und mit `<CR>` quittiert. Der entsprechende Code wird eingefügt.

	in Präambel	im Dokumentteil
leere Zeile	Auswahl Paket	Auswahl Umgebung
Schlüsselwort	verwende `usepackage`	verwende `begin ...end`
Visual-Mode	nichts	schliesst ausgewählter Text in die Umgebung ein
Line Visual-Mode	nichts	schliesst ausgewählten Text in die Umgebung ein (zeilenweise)

Tabelle 1: Funktion der Taste F5, abhängig von der Position der Eingabe

Listing 3: Beispiel für Variable `g:Tex_HotKeyMappings` in der Datei `tex.vim`

```
1 " Nur die ersten 4 Umgebungen werden verwendet
2 " Shift-F1 bis Shift-F4
3 let g:Tex_HotKeyMappings = 'itemize,enumerate,frame,verbatim'
```

Enthält die Zeile bereits ein Schlüsselwort wird dieses als Paketname oder als Umgebungsname verwendet und mit dem entsprechende LaTeX-Code ergänzt (Tabelle 1).

4.4.2 Umgebung einfügen mit «Hotkeys» Shift-F1 bis Shift-F4

Mit den Tastenkombinationen Shift-F1 bis Shift-F4 können die wichtigsten Umgebungen abgespeichert werden.

Das geschieht in der Datei `tex.vim` mit der Variablen `g:Tex_HotKeyMappings`. Ein Beispiel findet sich im Listing 3.

Im Gegensatz zum Einfügen einer Umgebung mit der F5-Taste funktionieren die Hotkeys nur im Einfüge-Modus.

4.4.3 Drei-Buchstaben-Kürzel

Umgebungen können im Einfüge-Modus mit einem Drei-Buchstaben-Kürzel eingesetzt werden. Diese sind in Grossbuchstaben und beginnen mit `E`, gefolgt von zwei weiteren Grossbuchstaben die

- bei Umgebungsbezeichnungen, die aus zwei Wörtern bestehen, jeweils die Anfangsbuchstaben darstellen (`flushleft` wird zu E**FL**).
- bei Umgebungsbezeichnungen, die aus einem Wort bestehen, den ersten und zweiten Buchstaben darstellen (`equation` wird zu EEQ).
- bei Konflikten wird nicht der zweite Buchstabe, sondern der *letzte* Buchstabe genommen (`quote` wird zu EQE, `quotation` wird zu EQN).

4.4.4 Umgebungen einsetzen im Visual-Modus: mit F5

Die Latex-Suitebietet die Möglichkeit, ein im Visual-Modus ausgewählten Text automatisch in eine LaTeX-Umgebung zu fassen. Die angezeigte Liste kann mit der Variable `g:Tex_PromptedEnvironments` angepasst werden.

Listing 4: Beispiel für Variable `g:Tex_PromptedEnvironments` in der Datei `tex.vim`

```
1 " Beliebig viele LaTeX–Umgebungen koennen angegeben werden.
2 let g:Tex_PromptedEnvironments = 'frame,center,lstlisting,itemize'
```

4.4.5 Umgebung einsetzen im Visual-Modus mit Drei-Buchstaben-Kürzel

Der Text kann auch ausgewählt werden und einem Drei-Buchstaben-Kürzel ausgewählt werden. Die drei Buchstaben bestehen aus , (Komma) und den oben beschriebener Abkürzung, aber in *Kleinbuchstaben*.

Aus `ETT` wird im Visual-Mode also `,tt`. Danach ist man im Normal-Modus.

4.4.6 Umgebung wechseln mit Shift-F5

Wird im Normal-Modus die Shift-F5 gedrückt, kontrolliert Latex-Suite, in welcher Umgebung der Cursor aktuell ist. Es kann so die Umgebung gewechselt werden. Bei geschachtelten Umgebungen wird die innerste ersetzt. Mit u kommt man wieder in die alte Umgebung.

Die mit Shift-F5 angezeigten, voreingestellten Umgebungen verändert werden können wird im Abschnitt 4.4.1 dargestellt.

Bug? Bei mir wird bei Anwendung von Shift-F5 jeweils korrekterweise die innerste Umgebung ersetzt. Allerdings werden die Backlash-Zeichen vor begin und end verdoppelt.

4.4.7 Makros für LaTeX-Befehle anpassen

Die Variable `g:Tex_PromptedEnvironments` bestimmt, welche Umgebungen mit `F5` angezeigt wird. Beispiel siehe Listing 4.

5 Diverse Probleme

5.1 Folding ein- und ausschalten

Siehe: http://stackoverflow.com/questions/3322453/how-can-i-disable-code-folding-in-vim-with-vim-latex

5.2 Viewer in MacVim

5.2.1 Skim

5.2.2 Preview

5.2.3 PDF Reader (Acrobat

http://www.iskysoft.de/edit-pdf/pdf-reader-for-mac.html