Beschreibung des Rumex Baukastens

Hier findest du die Beschreibung des rumex Baukastens. Zur Zeit ist hier noch vieles durcheinander da ich mehr am System als an der Beschreibung bastle. Du kannst aber schon mal gerne rein schauen.

Vorwort

Die Beweggründe mit einem eigenen System eine Internet Seite zu Verwalten sind mit der Zeit in mir gewachsen.

Es gibt immer mehr Systeme die zum Erstellen und Verwalten von Internetseiten eingesetzt werden. Probiert habe ich schon viele und mit den meisten war ich auch sehr zufrieden, wenn da nicht immer der Hacken mit dem Updates wäre. Irgendwie, zumindest kommt es mir so vor, ist der Aufwand das System der Seite auf dem aktuellen Stand zu halten größer geworden als den Inhalt der eigentlichen Seite zu pflegen.

Ich wünschte mir wieder eine Internetseite wie in früheren Zeiten. Eine Seite die aus einfachen HTML Seiten besteht.

Irgendwann bin ich dann auf `markdown` und `pandoc` gestoßen und die Idee dieses Home Page Baukastens ist entstanden.

Rumex?

Rumex ist die lateinische Bezeichnung für den Ampfer und dieser taucht in der Natur dann auf, wenn der Boden Überdüngt, Verdichtung und Beschädigt ist. Rumex gehört zu den sogenannten Pionier Pflanzen. Er ist ein Lückenfüller.

Genau das soll Rumex auch sein “ein Lückenfüller” für alle die ... Systeme satt haben.

Installation und Einrichten des Rumex Baukastens

Vorbereitung des Rechners

Rumex ist auf ein *nix System ausgerichtet. Auf diesem sollten folgende Programme installiert sein:

- bash
- make
- perl
- git
- pandoc
- imagemagick
- wget
- ~~wput~~
- sitecopy
- vim (g)vim
- texlive

Wobei `bash`, `make` und `perl` eigentlich bei jeder `*nix` Installation bereits vorhanden sein dürft. Die restlichen Programme müssen nach installiert werden.

Wer mit dem Editor `vim` zurecht kommt sollte sich auch `(g)vim` installieren. Rumex besitzt eine `gvim` Erweiterung die, die Arbeit bzw. die Suche nach dem richtigen Befehl am Anfang um einiges erleichtert.

`texlive`[1] wird nur gebraucht wenn man auch PDF Dateien erstellen möchte.

~~`wput`~~ `sitecopy` wird nur gebraucht wenn man die Daten per FTP hoch laden möchte.

```
sudo apt-get install make perl

sudo apt-get install git-core wget sitecopy pandoc imagemagick

sudo apt-get install gvim

sudo apt-get install texlive
```

[1] Die Funktion der PDF Erstellung ist zur Zeit noch nicht enthalten.

Installation von Rumex auf dem lokalen Arbeitsplatz

Für die Installation auf deinem Rechner musst du dir zu erst das ZIP bzw. das tar.gz Archiv vom github Server holen und entpacken

```
wget https://github.com/itbayer/rumex/archive/gh-pages.zip
unzip gh-pages.zip
```

oder

```
wget https://github.com/itbayer/rumex/archive/gh-pages.tar.gz
tar -xzvf gh-pages.tar.gz
```

Jetzt wechsle in das Verzeichnis `rumex-gh-pages/.rx` und starte die Befehle `make install` und `make show`.

```
cd rumex-gh-pages/.rx/
make install
make show
```

Fertig :-)

Rumex auf dich einstellen

Nach der Installation muss Rumex noch auf dich eingestellt werden. Genauer gesagt sollten folgende Angabe für deine neuen Seite angepasst werden.

- URL deiner Seite
- Impressum
- Kopf / Fusszeile
- Logo

Eine Kurzbeschreibung findest du, nach der Installation, auf der Startseite von Rumex oder hier.

moot.it

(**???**): Hier fehlt noch die Beschreibung

- ein/auschalten
- Kontoname einstellen

gvim Unterstützung einrichten

(???): Hier fehlt noch die Beschreibung@

- Erstellen der Startdatei
- Einrichten

AutoSave Funktion

Mit dem Plugin AutoSave kann man Rumex auch dazu bringen Dateien nicht nur automatisch zu speichern sondern auch gleich die HTML Datei zu erstellen.

Jedoch ist eine kleine Änderung am Plugin notwendig.

```
diff --git a/plugin/AutoSave.vim b/plugin/AutoSave.vim
index f09a904..ec2f463 100644
--- a/plugin/AutoSave.vim
+++ b/plugin/AutoSave.vim
@@ -11,6 +11,12 @@ else
   let g:auto_save_loaded = 1
 endif

+if exists("g:rumex")
+  finish
+else
+  let g:rumex = 0
+endif
+
 let s:save_cpo = &cpo
 set cpo&vim

@@ -33,10 +39,14 @@ function! AutoSave()
   if g:auto_save >= 1
     let was_modified = &modified
     silent! wa
-    if was_modified && !&modified
-      echo "(AutoSaved at " . strftime("%T") . ")"
-    endif
-  endif
+   if was_modified && !&modified
+       if g:rumex == 1
+           silent! make html
+           echo "html datei wurde erstellt"
+       endif
+       echo "(AutoSaved at " . strftime("%T") . ")"
```

```
+   endif
+endif
 endfunction

 function! AutoSaveToggle()
```

Mit dem Befehl `:let rumex=1` bzw. `let rumex=0` kann das Erstellen der HTML Datei ein bzw. ausgeschaltet werden.

Installiert man sich dann noch im FireFox dann noch die Erweiterung. “Tab Auto Reload” werden die Änderungen im Browser immer gleich angezeigt.

Über die Nutzbarkeit kann man sich streiten da nach einem HTML Lauf der Cursor immer an den Anfang der Zeile, in der man sich gerade befindet, springt. Mir ist die Variante mit `<F5>` lieber. Die “Tab Auto Reload” Funktion im Firefox verwende ich aber schon sehr gerne.

Tipps zur Benutzung gvim

vim Kurztasten

Eine Übersicht der Rumex Kurztasten für den Editor vim findest du auf der Seite VIM-Kurztasten.

gvim Menü

Eine Übersicht des Rumex Menüs für den Editor gvim findest du auf der Seite GVIM-Menü.

Dateiname ergänzen Will man in die Datei einen Dateinamen einbauen, weiß aber nicht mehr genau wie er heißt, kann man folgenden Trick verwenden. In diesem Beispiel wird ein Bildname gesucht. Im Text schreibe man `../bilder/tw` und drückt dann die Tastenkombination `C-X + C-F`, gvim öffnet ein Dialogfeld in dem alle Dateien die auf dieses Muster übereinstimmen geöffnet. Gibt es nur einen Treffer wird dieser gleich eingefügt.

Wort innerhalb des Dokumentes suchen Sucht man ein Wort das man im Dokument schon einmal verwendet hat, um zum Beispiel darauf zu verweisen. Schreibt man den Wortanfang und drückt dann `C-P`. Es öffnet sich ein Dialogfeld in dem alle Wörter die auf dieses Muster passen angezeigt werden. Gibt es nur einen Treffer wird dieser gleich eingefügt.

Nützliche Erweiterungen Vim bietet ein paar nützliche Erweiterungen in Form von plugins an. Hier eine Liste, der plugins, die ich gerne verwende.

pathogen.vim Diese Erweiterung macht das Installieren weitere Erweiterungen einfach. Dabei ist die Installation von `pathogen` schnell erledigt.

```
mkdir -p ~/.vim/autoload ~/.vim/bundle
wget -O ~/.vim/autoload/pathogen.vim //
http://www.vim.org/scripts/download_script.php?src_id=16224
```

In die `.vimrc` muss dann noch nachfolgende Zeile eingebaut werden.

```
call pathogen#infect()
```

eingebaut werden.

Jetzt braucht man die Erweiterungen nur mehr in das Verzeichnis `.vim/bundle` zu kopieren und vim neu starten. In Verbindung mit `git` wieder eine einfache Sache.

```
cd ~/.vim/bundle
git clone https://github.com/vim-pandoc/vim-pandoc.git
```

vim-pandoc Erweiterung rund um pandoc.

FuzzyFinder Dateien schnell zum editieren öffnen. Installiert ist diese Erweiterung, vorausgesetzt man verwendet pathogen, mit dem Befehl:

```
wget -O /tmp/vim-fuzzyfinder.zip http://www.vim.org/scripts/download_script.php?src_id=
mkdir ~/.vim/bundle/vim-fuzzyfinder
unzip /tmp/vim-fuzzyfinder.zip -d ~/.vim/bundle/vim-fuzzyfinder/
```

Für das öffnen dies Datei Dialogs sollte man sich dann noch eine Kurztaste konfigurieren.

```
" ---------------------------------------------------------
" FuzzyFinder File Suche auf <F12> binden
map <F12> :FuzzyFinderFile <CR>
```

Seite auf einem github.com Server einrichten

(???)

- Einrichten eines github.com Zugangs

- Arbeits- Repository auf den AP[2] holen
- Die Rumex ZIP Datei
- Die Dateien des ZIP Archives in das Arbeits- Repository kopieren
- Grund Dateien anpassen
  - (**???**): Angabe welche Dateien fehlt noch@
- Erste Änderungen vornehme
- `make online` - Fertig.

Seite auf einem nicht github.com Server einrichten

Datei upload per git

(**???**)

Auf der Seite rumex.it-bayer.de findet man eine Beschreibung wie man den rumex Baukasten auf einen **nicht** `github.com` Server installiert.

Datei upload per ftp

(**???**)

Aufbau des rumex Baukastens

(**???**)

root Verzeichnis

Im `root` Verzeichnis findet man alle `HTML` Dateien der Seite. Diese werde vom Baukasten erstellt und müssen nicht von Hand verändert werden. Zusätzlich findet man noch ein folgende Systemdateien:

rss.xml News Feed Datei, wird vom System erstelle

readme.md Beschreibungsdatei die von github.com gebraucht wird

robots.txt Datei für die Suchmaschinen

favicon.ico Icon für den Browser

.htaccess Konfiguration für den Apache Server

CNAME (**???**): Beschreibung@

[2] Mit AP ist der **A**rbeits**P**latz Rechner gemeint.

Unterverzeichnisse

.rumex/ (**???**): Beschreibung@

.rx/ (**???**): Beschreibung@

rxtpl/ Standard Template Verzeichnis. In diesem Verzeichnis befinden sich die Dateien die für das Aussehen der Seite verantwortlich sind.

Folgende Dateinen und Verzeichnisse sind hier zu finden

- index.html (Weiterleitung zum root Verzeichnis)
- css/
- js/
- img/

bilder/ In diesem Verzeichnis werden alle Bilder der Seite abgelegt.

Steuerung durch Dateiendung

Das Aussehen der Dateien bezüglich des Inhaltsverzeichnisses könnte auch durch die Dateiendung gesteuert werden.

Beispiel

datei.rx0s Würde eine HTML Datei ohne Inhaltsverzeichnis erstellen.

datei.rx1s Erstellt die HTML Datei mit dem Inhaltsverzeichnis aus den Einträgen der H1 Überschriften.

datei.rx2s Erstellt eine HTML Datei mit dem Inhaltsverzeichnis aud den Einträgen der H1 und H2 Überschriften.

Zusätzlich könnte die Dateiendung auch eine unterschiedliche Verwendung der Dateien ermöglichen.

datei.rx0s, datei.rx1s ... Standard Datei.

datei.rx0x, datei.rx1x ... Datei die nicht in die Liste auf `index.html` eingebunden wird.

datei.rx0v, datie.rx1v ... Versteckte Datei. Diese taucht weder in der `index.html` noch in der `sitemap.xml` auf.

datei.rx0w Datei mit einer Weiterleitung. Die Weiterleitung wird dabei mittels `javascript` realisiert, da github keine `.htaccess` Weiterleitung unterstützt.

```
% Weiterleitung nach beschreibung.html
%
%

<script language="javascript">
<!--
window.location.href="beschreibung.html";
// -->
</script>
```

Bei der Änderung der Dateiendung bleibt der eigentliche `html` Name gleich. Nur die Funktion der Einbindung ändert sich.

Sonderseiten

Im Verzeichnis `pandoc` befinden sich Sonderseiten.

rumex/index.rx0x Diese Datei wird vom Programm `.bin/make_index.pl` erstellt, muss somit nicht vorhanden sein.

rumex/start.rx0s Die `start.rx0s` wird als Vortext in die `index.rx0x` eingebunden. Mit ihr kann man oberhalb der Seiten Liste einen extra Text in die `index.html` eingebunden werden.
Diese Datei ist erforderlich und muss vorhanden sein.

rumex/rss.rx0x Datei für die RSS Feed Funktion.

rumex/impressum.rx0x Datei für die Impressumsangaben.
Ist nicht zwingend erforderlich. Jedoch muss die `.inc/fuss.html` entsprechende bearbeitet, der Link muss raus genommen, werden.

rumex/Makefile Sie `make` Steuerdatei.

Aufbau der Startseite

Die Startseite `markdown/start.rx0s` muss vorhanden sein. Es reicht auch ein ‘touch markdown/start.rx0s.

Der normale Aufbau könnte so ausschauen. Die `pandoc` Kopfzeile sind nicht zwingend erforderlich.

```
% start.rx0s
%
%

Hier kommt auch schon der Vortext für die index.html
```

Aufbau der Einzelseiten

Die Einzelseiten liegen alle im Verzeichnis `rumex` und zwar in der Sprache `markdown` bzw. der Erweiterung von `pandoc`.

Diese Einzelseiten werden in chronologischer Reihenfolge in die Startseite `index.html` eingebunden und bilden sozusagen das Inhaltsverzeichnis der Seite. In jeder Einzelseite wird dazu ein sogenannter *"Vortext"* hinterlegt. Die Seite bzw. der Kopf der Seite hat dabei folgenden Aufbau.

```
% Seiten Überschrift 1
% Seiten Überschrift 2
% Seiten Überschrift 3

<!--


# Listen-Überschrift

Überschrift und Text der in der Listenübersicht
auf index.html angezeigt wird.

Alles was innerhalb der HTML Kommentar Marken
steht wird nur auf der index.html Seite angezeigt.

-->

Alles was sich außerhalb der HTML Marken
befindet wird auch auf der eigentlichen Seite angezeigt.

Durch das Schlüsselwort "schnipp", das auch in HTML
Kommentar Marken stehen muss, wird der Vortext beendet.
Auf der index.html erscheint an dieser Stelle der Link
"... weiter lesen".

<!-- schnipp -->

Ab hier geht dann der Inhlat der eigentlichen Seite los.
```

Template

(**???**)

Änderung gegenüber des Original `pandoc` Templates.

```
5c5,8
<   <meta name="generator" content="pandoc">
---
>   <!-- rumex Version -->
>   <meta name="generator" content="$meta_generator$">
>   <!-- rumex Suchmaschine -->
>   <meta name="robots" content="$meta_robots$">
30a34,35
>   <!-- rumex RSS -->
>   <link rel="alternate" type="application/rss+xml" title="$rsstitel$" href="$rssfile$" />
41c46
< <h1 class="title">$title$</h1>
---
> <h1 class="title"><a title="zur Startseite" href="index.html">$title$</a></h1>
43c48
< <h2 class="author">$author$</h2>
---
> <h2 class="author"><a title="zur Startseite" href="index.html">$author$</a></h2>
46c51
< <h3 class="date">$date$</h3>
---
> <h3 class="date"><a title="zur Startseite" href="index.html">$date$</a></h3>
54a60
> <div id=seite>
55a62
> </div>
```

RSS Feed Funktion

(???)

Wird nicht aus den einzelnen Dateien erstellt sondern muss manuell editiert werden, Datei `.rx/rss.rx0x`.

Jede Überschrift eines Eintrags muss mit einem `{.nn1}` enden.

Danach kommen die Angaben zu:

Link: *Verweis zur Seite mit weiteren Informationen*
Autor: *Autor der den Eintrag geschrieben hat*
Kategorie: *Kategoie des Eintrags*
Datum: *Datum des Eintrags.* Das richtige Format bekommt man mit dem Befehl `date -R`.

die in HTML Kommentar Marker eingeschlossen sind.

Anschließend folgt die Meldung. Zur Zeit werden folgende `pandoc` Formatierungen unterstützt.

- Überschriften ab der Stufe 3 `###`
- Aufzählungen `-`
- Aufzählungen `*`
- Zitate `>`
- Links `[Link](http://muster.tdl)`. Diese dürfen nicht am Anfang einer Zeile stehen.
- Bilder `![Bild](../bilder/muster.png "muster.png")`, Diese dürfen nicht am Anfang einer Zeile stehen.
- Code ‘Code‘.

Beispiel einer RSS Feed Seite

```
% Abo Seite - Newsletter - RSS Feed
%
%

<!--
| Titel: Rumex - Ein Home Page Baukasten
| Beschreibung: Aktuelles und Neuigkeiten vom rumex Home Page Baukasten
| KLink: http://www.it-bayer.de/rumex/
| Lang: de-DE
| BildTitel: IT-Bayer rumex Meldungen
| BildURL: http://www.itbayer.de/rumex/bilder/404.png
| BildLink: http://www.it-bayer.de/rumex/
| BildBeschreibung: Aktuelles und Neuigkeiten von der rumex Seite.
-->

# rumex bekommt die RSS Feed Funktion{.rssfeed1}

<!--
| Link: http://www.it-bayer.de/rumex/
| Autor: IT-Bayer
| Kategorie: Neues
| Datum: Thu, 04 Jul 2013 10:07:06 +0200
-->

In den rumex Baukasten wurde eine RSS Feed Funktion eingebaut.
Die Einträge werden dabei auch aus einem `pandoc` Dokument gelesen
und nach `rss.xml` geschrieben.

Zusätzlich ist diese Datei auch als `rss.html` unter
<http://www.it-bayer.de/rumex/rss.html> verfügbar.
```

RSS Dateiname

Der Dateiname ist mit `rss.xml` vorbelegt und kann über die Variable `RSS_FILE` in der `config.md` geändert werden.

RSS Titel

Der Title des RSS Feed wird durch die Variable `RSS_TITEL` angepasst.

```
RSS_TITEL = "Neuigkeiten von rumex Baukasten"
```

RSS auslagern

Den RSS Link kann man auch auslagern so dass dieser auf eine andere Seite zeigt. Dazu setzt man die Variable `RSS_EXTERN` mit dem entsprechenden Link. Die Variabel `RSS_FILE` wird dadurch nicht mehr verwendet. Auch der RSS Lauf wird dadurch ausgeschaltet und durch eine Meldung ersetzt.

RSS Kurztaste

Für die einzelnen Einträge steht auch eine Kurztaste `.rnn` zur Verfügung.
In Gvim unter

```
Rumex -> TextBausteine -> NeuerNews Eintrag
```

Eingefügt wird dann folgende Vorgabe. Der Wert hinter Datum wird von System ausgelesen und entsprechende gesetzt.

```
 # Neue Nachricht{.nn1}

<!--
| Link: http://www.it-bayer.de/rumex/
| Autor: IT-Bayer
| Kategorie: Neues
| Datum: Mon, 28 Oct 2013 07:36:56 +0100
-->

Ab hier geht die neue Nachricht los.
```

Interna

Durch die beschriebenen RSS Variablen wird die Erstellung des RSS Feed gesteuert. Es wird in jede HTML Datei nachfolgender Header Abschnitt eingebaut wenn die `RSS_TITLE` Variable gesetzt wurde. `RSS_FILE` bzw. `RSS_EXTERN` steuern den `href` Eintrag.

```
<!-- rumex RSS -->
<link rel="alternate" type="application/rss+xml" title="Neuigkeiten von rumex Baukasten" hre
```

Rumex auf einem USB Stick

Rumex kann auch auf einem USB Stich installiert werden. Der Stick muss aber ein Dateiformat besitzt
welches mit Dateirechten und Symbolische Links umgehen kann.

USB Sticks im VFAT Format funktionieren nicht. Man kann zwar die Daten darauf ablegen. Das Arbeiten über den Stick funktioniert nicht wirklich. Auch wenn man die Daten, von einem VFAT Stick, auf ein *nix System kopiert werden muss händisch nach gebessert werden.

LinkTipp: USB Stick unter Linux verschlüsseln

make Steuerung

(**???**)

Gesteuert wird der Baukasten mittels `make` im Unterverzeichnis `rumex`. Folgende `make` Befehle stehen dabei zur Verfügung.

make html Erstellt die einzelnen html Dateien. Hier kann auch nur `make` verwendet werden.

make index Erstellt die `index.md` Datei aus der dann die `index.html` Datei erstellt wird.

make all Eine Zusammenstellung aus `make index` und `make html`.

make online Daten auf github hoch laden.

make bilder Erstellt Bilder in verschiedenen Auflösungen.

Update

(**???**)

Homepage Änderung, schnell und immer aktuell

Wer wünscht es sich nicht einen schnellen Zugriff auf seine Homepage. Sei es um schnell was zu ändern oder genauso schnell was neues einzustellen. Rumex bietet in Verbindung mit `tilda` eine schöne, einfache und auch schnelle Lösung.

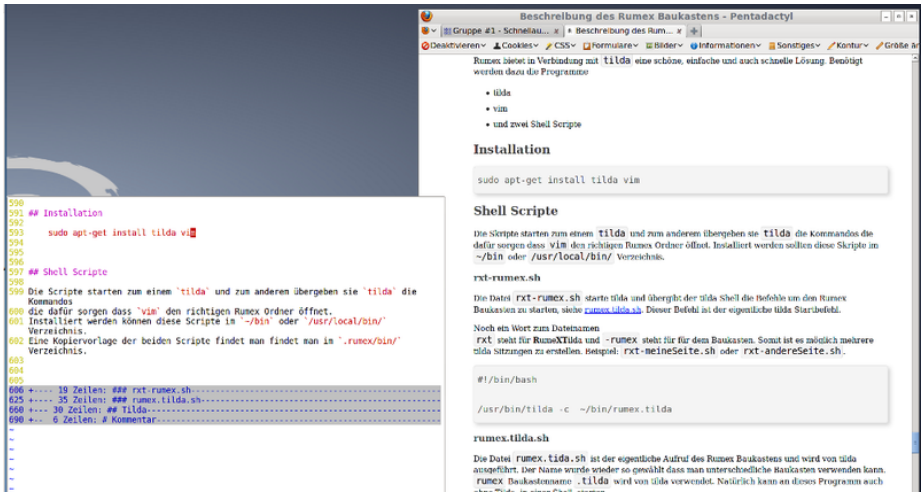

Abbildung 1: Rumex im tilda Fenster. Mit einem Tastendruck öffnet sch das tilda Fenster und man kann die Texte eintippen. Ein erneuter Tastendruck schließt das tilda Fenster wieder und der Bildschirm ist wieder frei. Hat man seine Änderung abgeschossen kann man mit den Rumex vim Kurztasten die Änderung schnell online stellen.

Installation der tilda Unterstützung

Die Installation ist nicht umfangreich. Man braucht vim und tilda und dann noch zwei bash Script. Eine Kopiervorlage der beiden Scripte findet man findet man im `.rumex/bin/` Verzeichnis.

```
sudo apt-get install tilda vim
cp .../rumex/.rumex/bin/rxt-rumex.sh ~/bin/.
cp .../rumex/.rumex/bin/rumex-tilda.sh ~/bin/.
```

Diese beiden bash Scripte müssen anschließend noch angepasst werden.

Shell Scripte

Die Scripte starten zum einem `tilda` und zum anderem übergeben sie `tilda` die Kommandos die dafür sorgen `vim` im richtigen Rumex Ordner zu öffnet. Installiert

werden können diese Scripte im `~/bin` oder `/usr/local/bin/` Verzeichnis.

rumex-tilda.sh Die Datei `rumex-tilda.sh` starte tilda und übergibt der tilda Shell die Befehle um den Rumex Baukasten zu starten, siehe rumex.vim.sh.

```
#!/bin/bash

/usr/bin/tilda -c  ~/bin/rumex-vim.sh
```

rumex-vim.sh Mit dem Befehl `rumex-vim.sh` wird der Rumex Baukastens aufgerufen. Dieser Befehl wird unter anderem auch von `rumex-tilda.sh` verwendet. `rumex-vim.sh` kann natürlich auch in einem Shellfenster ausgeführt werden.

rumex-gvim.sh Mit dem Befehl `rumex-gvim.sh` wird der Rumex Baukasten mit dem Editor gvim gestartet.

Tilda einrichten

Nach dem ersten Start wird Tilda in linken oberen Bildschirm Bereich eingeblendet. Man sollte Tilda nun noch an seine Bedürfnissen anpassen. Dazu in das Tilda Fenster mit der rechten Maustaste klicken und `Eigenschaften` aus wählen.

U̇brigens: Man kann `tilda` mehrfach starten. Somit kann auf mehreren Rumex Installationen parallel über diese Weiße zugegriffen werden. Man sollte nur jede `tilda` Sitzung ein wenig anders konfigurieren.

Nachteil: Ein Nachteil von `tilda` darf man aber nicht verschweigen. Bei wechseln zwischen den Fenstern kann man die Tastenkombination `<ALT>+<TAB>` nicht verwenden bzw. man kommt mit dieser Kombination nicht mehr zurück nach `tilda`. Schließt und öffnet man `tilda` mit der definierten Taste bekommt man aber den Fokus wieder in das Fenster.

Will man die `<ALT>+<TAB>` Kombination doch verwenden muss man die Standardeinstellung von tilda ändern. Den erforderlichen Schalter findet man in der Konfiguration, Reiter *Allgemein* -> Schalter *Nicht in der Taskleiste anzeigen.*