[Gastartikel] Cross-platform ToDo-Liste todo.txt

Was ist todo.txt?
Todo.txt ist eine ToDo-Liste, nicht mehr. Eine ToDo-Liste, die einfacher nicht sein könnte.
Eine Textdatei, die alle Aufgaben enthält, eine Konfiguration und eine Datei mit beendeten Aufgaben (quasi ein Archiv). Alle drei Dateien können mit einem Texteditor geöffnet und editiert werden – eine Eigenschaft, die für Gina Trapani, Entwicklerin von Todo.txt, sehr wichtig war und ist.

Wir schauen uns in diesem Artikel die Installation unter OSX, die Nutzung mit OSX und iOS und die Integration in Alfred.app an.
Todo.txt unterstützt jedoch ebenfalls Android, webOS, Windows Phone 7, Windows, Linux und mehr. Die Installation und Nutzung unter diesen Betriebssystemen ist vergleichbar mit denen, die in diesem Artikel im Mittelpunkt stehen.

Installation (OSX)
Die Installation unter OSX ist fast genauso einfach wie alles andere bei Todo.txt. Auf der Download Seite des GitHub Repositories findet sich die aktuelle Version im ZIP- und komprimiertem Tar-Format. Letztendlich ist es egal, welche Version man lädt, die meisten modernen Entpacker kommen mit beiden Formaten gut zurecht.
Derzeit ist die Version 2.9 die aktuellste, sodass wir diese herunterladen und entpacken.

```
$ wget https://github.com/downloads/ginatrapani/todo.txt-cli/todo.txt_cli-2.9.tar.gz
$ tar xfvz ./todo.txt_cli-2.9.tar.gz
```

Code 1: Download und Entpacken des CLI

Um das Benutzen von Todo.txt so einfach und schnell wie möglich zu machen, sollten wir Zugriff auf das Skript innerhalb der Pfade in unserer *$PATH* Variablen haben.

Der Rest der Installation ist optional und ein wenig komplexer als der Rest dieses Artikels. Er ist jedoch notwendig, um das später vorgestellte Skript und die Alfred.app Integration ausführen zu können.

Erst einmal eine Übersicht, was wir tun werden:

- Todo.txt in *~/opt/todotxt/* installieren
- Ein Skript zur Ausführung erstellen
- Das Skript in unsere *$PATH* Variable integrieren

Zuerst erstellen wir uns einen Ordner, der Todo.txt enthalten soll. Wir wählen *~/opt/todotxt/*.

```
$ mkdir -p ~/opt/todotxt/
```

Code 2: Todo.txt Ordner erstellen

In diesen Ordner schieben wir nun das eigentliche Todo.txt Skript.

```
$ mv ./todo.txt_cli-2.9/todo.sh ~/opt/todotxt/
$ mv ./todo.txt_cli-2.9/todo_completion ~/opt/todotxt/
```

Code 3: Todo.txt Skript in Installationsordner verschieben

Nun erstellen wir uns einen globalen Ordner für ausführbare Skripte, die automatisch alle in unsere *$PATH* Variable importiert werden sollen.

Code 4: Pfad zu ausführbaren Skripten erstellen

Nun fügen wir den Pfad zu *$PATH* hinzu und machen dies permanent, indem wir die Datei *.bash_profile* nutzen.

```
$ echo "export PATH=\$PATH\":~/opt/bin\"" >> ~/.bash_profile
```

Code 5: Neuen Pfad zu $PATH hinzufügen

Zuletzt laden wir ein Skript herunter und werden von hier an das Skript nutzen, um mit Todo.txt zu arbeiten.

```
$ curl -OL h
https://raw.github.com/NeQuissimus/Shortcuts/master/todo >
~/opt/bin/todo
$ chmod u+x ~/opt/bin/todo
```

Code 6: Skript herunterladen und ausführbar machen

Command-line interface

Todo.txt hat ein relativ einfach zu bedienendes Command-line interface (CLI). Man kann einfach *~/opt/todotxt/todo.sh* ausführen.

Der add Parameter, gefolgt von einem Text, fügt eine Aufgabe hinzu. Jede Aufgabe erhä automatisch eine ID.

Ist eine Aufgabe erledigt, ruft man das CLI mit dem done Parameter auf, gefolgt von der Aufgaben-ID.

Mit del oder rm können Aufgaben gelöscht werden. Der list Parameter listet alle aktuellen Aufgaben.

help listet alle möglichen Parameter auf, wir gehen auf die Weiteren hier nicht ein.

iOS App

Ein Todo.txt iOS App befindet sich im App Store.

Das App ist sehr einfach und intuitiv zu bedienen.

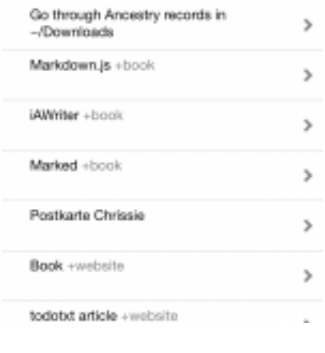

Bild 1: iOS Todo.txt

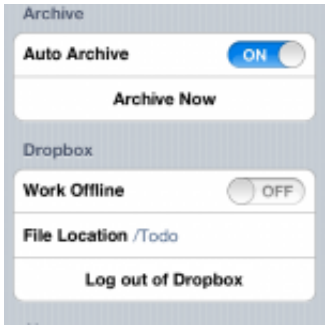

Bild 2: Einstellungen

Die iOS App unterstützt den Cloudservice Dropbox. Ist dies eingerichtet im iOS App, kann der Desktop-Client ebenfalls die Dropbox nutzen, sodass die Aufgaben immer synchron sind.

Ist die Auto-Archivierung aktiv, werden Aufgaben aus der todo.txt Datei in eine done.txt Datei bewegt. Ist sie inaktiv, so werden Aufgaben in der todo.txt Datei gespeichert und lediglich als “fertig” markiert.

Konfiguration

Um mit dem CLI die Dropbox zu unterstützen, gibt es zwei Möglichkeiten: Man kopiert alle Todo.txt Dateien in den Dropbox Ordner oder erstellt eine Konfigurationsdatei.

In der Regel ist der zweite Ansatz der sinnvollere, um Applikationen und Daten voneinander zu trennen.

Wir wollen unsere Dropboxdaten in *~/Dropbox/Todo* speichern.

Dort kopieren wir die *todo.cfg* hin, die bereits mit der Applikation kam.

```
$ mv ./todo.txt_cli-2.9/todo.cfg ~/Dropbox/Todo/
```

Code 7: Konfiguration in Dropbox bewegen

Alles, was wir in dieser Datei ändern müssen, steht im obersten Block. Es können ebenfalls Farben etc. geändert werden, das ist jedoch im Moment uninteressant.
Die Konfiguration sollte im oberen Teil wie folgt aussehen.

```
export TODO_DIR=~/Dropbox/Todo/
eval TODO_DIR=$TODO_DIR

export TODO_FILE="$TODO_DIR/todo.txt"
export DONE_FILE="$TODO_DIR/done.txt"
export REPORT_FILE="$TODO_DIR/report.txt"
```

Code 8: Konfiguration der Pfade

Diese Konfiguration kann nun von überall mit dem d Parameter aufgerufen werden. Das bedeutet, dass Todo.txt in unserem Pfad *~/opt/todotxt/* die Konfiguration in *~/Dropbox/Todo* nutzen kann.

```
$ ~/opt/todotxt/todo.sh -d ~/Dropbox/Todo/todo.cfg add "Testaufgabe"
```

Code 9: Beispielaufgabe in Dropbox hinzufügen

Ab diesem Zeitpunkt kann das Skript aus der Installation ebenfalls genutzt werden, wobei die Konfiguration nicht mehr explizit angegeben werden muss. Das Skript bildet praktisch ein Alias, das die Konfiguration automatisch mit angibt.

```
$ todo add "Testaufgabe"
```

Code 10: Äquivalent zu Code 9, mit Skript aus der Installationsanleitung

Alfred.app
Ich nutze Todo.txt unter OSX meist mit Alfred.app. Zu diesem Zweck habe ich zwei Extensions erstellt, eine fügt neue Aufgaben hinzu, die andere markiert sie als beendet.
Unter “Extensions” kann man einfach das + Zeichen anklicken und sich selbst die Extensions erstellen mit den Konfigurationen aus den Bildern unten.

Bild 3: Alfred Script zum Beenden von Aufgaben

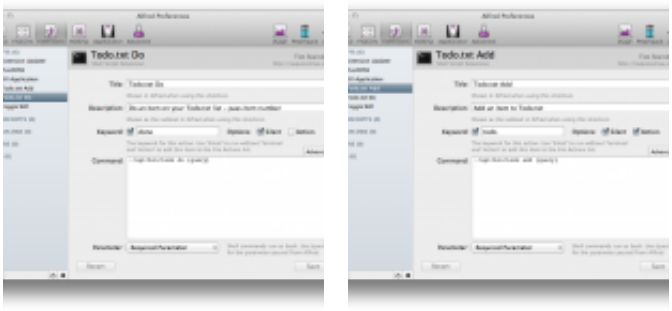

Bild 4: Alfred Script zum Hinzufügen von Aufgaben

Mithilfe der angegebenen Keywords “todo” und “done” können nun Aufgaben einfach in Todo.txt eingegeben werden.

GeekTool

Zuletzt empfehle ich, GeekTool aus dem App Store zu laden.
Mit GeekTool kann man dann ein Shell-Geeklet erstellen, dessen Befehl *~/opt/bin/todo ls* ist. Bei mir hat es einen Refreshwert von 600 Sekunden, sodass meine Aufgaben alle 10 Minuten auf dem Desktop aktualisiert werden.
Nun kann man die Aufgaben IDs jederzeit sehen und mit Alfred einfach und schnell die Aufgaben beenden.

Bild 5: GeekTool Todo.txt

Ich hoffe, dass der Artikel dem Einen oder Anderen eine einfache Alternative im Wald der ToDo-Applikationen vorgestellt hat.
Ich nutze das Setup seit einiger Zeit und bin sehr zufrieden.
Bei weiteren Fragen und/oder Anregungen bin ich bei Twitter zu finden, @Tim_Steinbach.