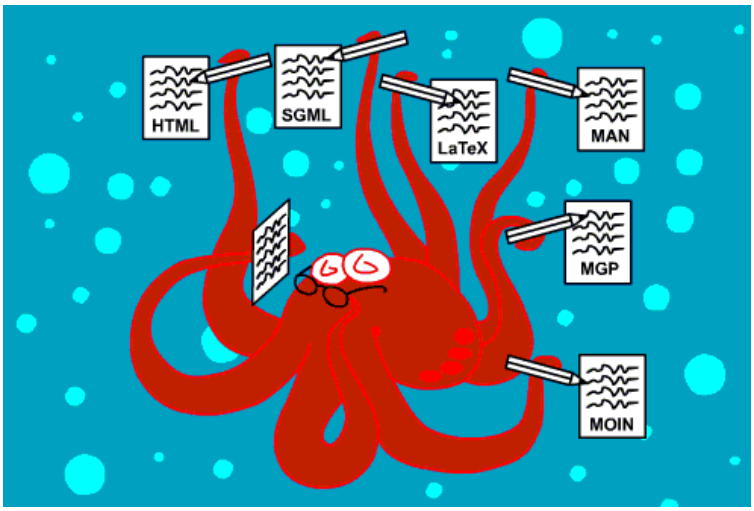
Txt2tags Benutzerhandbuch

http://txt2tags.sf.net



Aurélio Marinho Jargas v2.3 (May 2005)

Übersetzung von Andreas Deininger (Sep 2005)

Inhaltsverzeichnis

Inhaltsverzeichnis

Teil I − Einführung in txt2tags

Die ersten Fragen, die sich ihnen stellen

Dieses Kapitel bietet einen Überblick über txt2tags, in dem das Programm, dessen Zweck und dessen Möglichkeiten vorgestellt werden

Worum handelt es sich?

Txt2tags ist ein Werkzeug zur Textformatierung und −konvertierung.

Txt2tags konvertiert ein ASCII−Text Datei mit einigen wenigen Auszeichnungen zu jedem der unterstützten Ausgabeformate:

- HTML Dokument
- XHTML Dokument
- SGML Dokument
- LaTeX Dokument
- UNIX man page
- Magic Point Präsentation
- MoinMoin Seite
- PageMaker 6.0 Dokument
- ASCII−Text (keine Auszeichnungen)

Wieso sollte ich txt2tags benutzen?

txt2tags kann für sie sehr nützlich sein, wenn sie:

- Dokumente in verschiedenen Formaten veröffentlichen müssen
- für das Update von Dokumenten in verschiedenen Formaten verantwortlich sind
- technische Dokumentationen oder Handbücher verfassen müssen
- nicht wissen, wie sie ein Dokument in einem bestimmten Format schreiben sollen
- keinen speziellen Editor für ein bestimmtes Format besitzen
- mit einem einfachen Texteditor ihre Dokumente aktualisieren wollen

Die hauptsächliche Motivation hinter all dem ist:

- Sparen sie Zeit, schreiben sie **Inhalte** und vergessen Sie die **Formatierung**

Was zeichnet txt2tags gegenüber ähnlichen Werkzeugen aus?

Txt2tags wird sehr geradlinig erweitert, basierend auf grundlegenden Konzepten. Das sind die wichtigsten Merkmale:

Quellcode gut lesbar	Txt2tags Auszeichnungen sind sehr einfach und in natürlicher Weise gehalten.
Erzeugtes Dokument lesbar	Ebenso wie der Quellcode ist auch das erzeugte Dokument gut lesbar, mit Einrückungen und kurzen Zeilenlängen.
Konsistente Auszeichnungen	Txt2tags Auszeichnungen sind sehr spezifisch, sie eignen sich für alle Arten von Dokumenten und können dennoch nicht mit den eigentlichen Inhalten des Dokuments verwechselt werden.
Konsistente Regeln	Ebenso einfach wie die Auszeichnungen selbst sind auch die Regeln, wie die Auszeichnungen miteinander verknüpft sind, es gibt keine

	"Ausnahmen" oder "Spezialfälle".
Einfache Strukturen	Alle unterstützten Formatierungen sind **einfach**, ohne zusätzlich Optionen oder sonstige komplizierte Schalter. Eine Auszeichnung ist lediglich eine Auszeichnung, ohne jegliche Optionen.
Einfach zu erlernen	Aufgrund der einfachen Auszeichnungen und da der Quellcode lesbar ist, weist txt2tags eine flache, benutzerfreundliche Lernkurve auf.
Hübsche Beispiele	Die **Bespieldateien**, welche in dem Paket enthalten sind, geben reale Beispiele für einfache und auch komplexe Dokumente, die für und mit txt2tags geschrieben wurden.
Wertvolle Werkzeuge	Die **Syntax−Dateien**, welche in dem Paket enthalten sind (für die Editoren vim, emacs, nano and kate), unterstützen sie beim Verfassen fehlerfreier Dokumente.
Drei Benutzerschnittstellen	Es existieren eine sehr benutzerfreundliche **grafische Tk−Oberfläche**, eine **Webschnittstelle**, um das Programm im Inter− oder Intranet zu nutzen und eine **Kommandozeilen Version** des Programms für fortgeschrittene Benutzer sowie für das Skripting.
Skripting	Über den mächtigen Kommandozeilen−Modus können erfahrene Benutzer bestimmte Aufgaben **automatisieren** sowie konvertierte Dateien **nachträglich verändern**.
Download und Aufruf / mehrere Plattformen	Txt2tags ist ein einfaches **Python Skript**. Es braucht nicht kompiliert zu werden und benötigt keine zusätzlichen Module. Insofern läuft es problemlos auf *NIX, Linux, Windows und Macintosh Computern.
Regelmäßige Updates	Das Programm hat eine aktive Mailingliste, über die Benutzer Korrekturen und Verbesserungen vorschlagen können. Der Autor selbst nutzt das Programm ausgiebig sowohl beruflich als auch privat, insofern wird die Entwicklung nicht kurzfristig eingestellt werden.

Muss ich für das Programm bezahlen?

Definitiv NEIN!

Es ist frei, GPL, Open Source, Public Domain, *<setzen sie hier ihr Lieblings−Schlagwort ein>*.

Sie können das Programm kopieren, benutzen, verändern, verkaufen und es sogar als das ihrige veröffentlichen. Fragen des Copyrights und der Softwarelizenzierung sind für den Autor von nachrangiger Bedeutung.

Unterstützte Auszeichnungsstrukturen

Nachfolgend eine Aufzählung aller von txt2tags unterstützten Strukturen.

- Kopfbereich (Titel des Dokuments, Name des Autors, Datum)
- Abschnittstitel (nummeriert oder nicht nummeriert)
- Absätze
- Schriftarten
    - ♦ fett
    - ♦ kursiv
    - ♦ unterstrichen
- Schreibmaschinenschrift (für Originaltext)
    - ♦ Originaltext innerhalb eines Absatzes
    - ♦ einzelne Originaltext−Zeile

    - ♦ Originaltext−Bereich (mehrere Zeilen umfassend)
- Bereich für ein Zitat
- Verknüpfungen
    - ♦ URL/Internet Verknüpfungen
    - ♦ E−Mail Verknüpfungen
    - ♦ lokale Verknüpfungen
    - ♦ benannte Verknüpfungen
- Listen
    - ♦ Aufzählungslisten
    - ♦ nummerierte Listen
    - ♦ Definitionslisten
- horizontale Trennlinien
- Bilder (ausgerichtet)
- Tabellen (mit oder ohne Rahmen, ausgerichtet, über mehrere Spalten hinweg)
- Spezielle Auszeichnung für rohen Text (wird nicht geparst)
- Spezielles Makro für das aktuelle Datum (mit flexibler Formatierung)
- Kommentare (für Notizen, Aufgabenlisten, Fehlerbeschreibungen etc.)

Unterstützte Ausgabeformate

HTML

Jeder weiß was HTML ist. (Hinweis: Internet)

Txt2tags erzeugt standardkonforme HTML Dokumente, die ansprechend aussehen und deren Quelltext gut lesbar ist. Es nutzt KEIN Javascript, keine Frames und auch keine sonstigen überflüssigen Formatierungstechniken, die für einfache, technische orientierte Dokumente auch gar nicht gebraucht werden. Falls gewünscht, kann allerdings eine getrennte CSS−Datei genutzt werden. Txt2tags erzeugt Code konform zum "*HTML 4.0 Transitional*"−Standard.

Seit Version 2.0 ist der von txt2tags erzeugte HTML−Code 100% standardkonform, dies kann mit dem W3C−Validator überprüft werden.

XHTML

Dies ist die neue HTML−Generation mit einem strikteren Regelwerk, so müssen etwa alle Tags, die geöffnet wurden, auch wieder geschlossen werden. Dadurch kann der Code einfacher geparst und verstanden werden. Von Aufgabenzweck her gesehen kann es grob generalisierend auch als HTML angesehen werden. Txt2tags erzeugt Code konform zum "*XHTML 1.0 Transitional*"−Standard.

Seit Version 2.0 ist der von txt2tags erzeugte XHTML−Code 100% standardkonform, dies kann mit dem W3C−Validator überprüft werden.

SGML

Dies ist ein verbreitetes Dokumentenformat für welches mit den sgmltools mächtige Anwendungen zur Konvertierung bereitstehen. Aus einer SGML−Datei können HTML−, PDF−, Postscript−, info, LaTeX, Lyx, RTF− sowie XML−Dokumente erzeugt werden. Die sgml2*−Werkzeuge erzeugen dabei automatisch Inhaltsverzeichnisse und können einzelnen Abschnitte auf getrennten Seiten anordnen (sgml2html).

Txt2tags erzeugt SGML Dateien die vom Systemtyp her linuxdoc Quellen gleichen, diese sind dafür geeignet, unmittelbar mit sgml2* Werkzeugen konvertiert zu werden, hierfür werden keine zusätzlichen Katalogdateien benötigt und auch sonst sind keine weiteren Anforderungen seitens des SGML−Standards vonnöten.

LATEX

Dieses im akademischem Bereich bevorzugte Dokumentenformat ist leistungsfähiger als sie es zu träumen wagen. Komplette Bücher, komplizierte Formeln und jeder noch so komplexe Text kann in LaTeX verfasst werden. Aber seien sie darauf gefasst sich ihre Haare auszuraufen, wenn sie versuchen, die Auszeichnungen per Hand zu schreiben ...

Txt2tags erzeugt kompilierfähige LaTeX Quelldateien und übernimmt dabei all die komplexen Maskierungsaufgaben und die sonstigen Ausnahmen. Sie als Autor müssen sich nur noch um den Text selbst kümmern.

LOUT

Ähnlich leistungsstark wie LaTeX, aber mit einer einfacheren Syntax, welche ein "@"−Zeichen anstelle des "\"−Zeichens verwendet und in den meisten Fällen ohne das Setzen von Klammern auskommt. Der Ansatz "Alles ist ein Objekt" macht die Auszeichnung sehr viel weniger komplex.

Txt2tags erzeugt gebrauchsfertige Lout Dateien, welche mittels des "lout"−Kommandos zu Postscript− oder PDF−Dateien konvertiert werden können.

MAN

UNIX man pages blieben über all die Jahre bestehen. Dokumentenformate kommen und gehen, man pages bleiben im Gegensatz dazu in Gebrauch, sie sind nicht zu verdrängen.

Es gibt andere Werkzeuge um man Dokumente zu erzeugen, aber txt2tags hat einen entscheidenden Vorteil: ein einziger Quelltext, vielerlei Zielformate. So können die Inhalte der man−page genau so gut in eine HTML−Seite, eine Magic Point− Präsentation, etc. konvertiert werden.

MGP

Magic Point ist ein sehr praktisches Präsentationswerkzeug (ähnlich Microsoft PowerPoint), die einzelnen Folien werden dabei mittels einer Auszeichnungssprache definiert. Auf diese Weise können komplexe Präsentationen in einem Editor erzeugt werden (etwa mittels vi/emacs/notepad).

Txt2tags erzeugt gebrauchsfertige .mgp Dateien, in denen bereits alle notwendigen Kopfzeilen für Schriften und für das generelle Erscheinungsbild definiert sind, Unterstützung für Akzente nach ISO−8859 inbegriffen.

BONUS 1: von txt2tags erzeugte .mgp Dateien nutzen XFree86 Type1 Schriftarten! Dies bedeutet, dass sie keine zusätzlichen TrueType Schriftart−Dateien für ihre Präsentation benötigen.

BONUS 2: die Farbdefinitionen für die Schriftarten sind so gewählt, dass selbst auf Systemen, auf denen nur eine geringe Farbtiefe zur Verfügung steht (gestartet mit `startx -- -bpp 8`), die Präsentation dennoch ansprechend aussieht!

Das Schlüsselprinzip: Konvertieren und benutzen. Keine aufwändigen Anpassungen oder sonstigen Anforderungen sind vonnöten.

MOIN

Sie wissen nicht was MoinMoin ist? Es ist ein WikiWiki!

Die Moin Syntax ist nicht sonderlich adrett, insbesondere da ständig `{{{'''''Klammern und Hochzeichen'''''}}}`, hinzugefügt werden müssen. Auch hier bietet txt2tags mit seinen einfachen Auszeichnungen eine universelle Lösung: ein Quelltext, vielerlei Zielformate.

PM6

I wette sie wussten das noch nicht, aber tatsächlich hat Adobe PageMaker 6.0 eine eigene Auszeichnungssprache! Schriftarten und −stile, Farbtabellen und die meisten anderen, über Mausklick auswählbaren Eigenschaften sind in dieser Auszeichnungssprache verfügbar. Der Import eines solchen Texts erfolgt über den Menüpunkt "Import tagged text". Ein Hinweis am Rande: es handelt sich um ein <HTML "ähnliches"> Auszeichnungsformat.

Txt2tags erzeugt all diese Auszeichnungen und definiert zusätzlich einen umfangreichen, funktionierenden Kopfbereich, in dem Absatzformate und −formatierungen gesetzt werden. Dies ist dabei der aufwändigere Teil. **ACHTUNG:** Keine Zeilenumbrüche! Jeder Abschnitt muss eine einzelne Zeile umfassen.

Anmerkung des Autors: *Mein gesamtes, in portugiesisch verfasstes Buch über reguläre Ausdrücke wurde im Editor VI geschrieben, mit txt2tags in das PageMaker−Format konvertiert und ging anschließend in Druck.*

TXT

TXT ist Text. Der einzig wahre Formatierungs−Typ.

Obwohl die txt2tags Auszeichnungen sehr intuitiv und unauffällig sind, können sie entfernt werden, indem die Quelldatei in reinen ASCII−Text konvertiert wird.

Titelzeilen sind dabei unterstrichen, und der Text ist mehr oder weniger der gleiche wie in der Quelltext−Datei.

Übersicht über die je nach Zielformaten unterstützten Strukturen

Struktur	html	xhtml	sgml	tex	lout	man	mgp	moin	pm6	txt
Kopfzeile	J	J	J	J	J	J	J	N	N	J
Abschnittstitel	J	J	J	J	J	J	J	J	J	J
Absätze	J	J	J	J	J	J	J	J	J	J
fett	J	J	J	J	J	J	J	J	J	–
kursiv	J	J	J	J	J	J	J	J	J	–
unterstrichen	J	J	–	J	J	–	J	J	J	–
Schreibmaschine	J	J	J	J	J	–	J	J	J	–
Originaltextzeile	J	J	J	J	J	J	J	J	J	–
Originaltextbereich	J	J	J	J	J	J	J	J	J	–
Zitate	J	J	J	J	J	J	J	J	J	J
Internet Verknüpfungen	J	J	J	–	–	–	–	J	–	–
E−Mail Verknüpfung	J	J	J	–	–	–	–	J	–	–
lokale Verknüpfungen	J	J	J	N	N	–	–	J	–	–
benannte Verknüpfungen	J	J	J	–	–	–	–	J	–	–
Aufzählungslisten	J	J	J	J	J	J	J	J	J	J
Nummerierte Listen	J	J	J	J	J	J	J	J	J	J
Definitionslisten	J	J	J	J	J	J	N	N	N	J
Trennlinien	J	J	–	J	J	–	J	J	N	J
Bilder	J	J	J	J	J	–	J	J	N	–
Tabellen	J	J	J	J	N	J	N	J	N	N
Extras	**html**	**xhtml**	**sgml**	**tex**	**lout**	**man**	**mgp**	**moin**	**pm6**	**txt**
Bilder ausrichten	J	J	N	N	J	–	J	N	N	–
Tabellenzellen ausrichten	J	J	J	J	N	J	N	J	N	N
mehrspaltige Zellen	J	J	N	N	N	N	N	N	N	N

Legende

J *Unterstützt*

N *Nicht unterstützt (eventuell in zukünftigen Versionen)*

− *Nicht unterstützt (in diesem Zielformat nicht möglich)*

Die drei Benutzerschnittstellen: GUI, Web und die Kommandozeile

Da die unterschiedlichen Nutzergruppen verschiedene Bedürfnisse haben und in verschiedenen Umgebungen arbeiten, ist txt2tags sehr flexibel bezüglich der Art und Weise, wie es aufgerufen werden kann.

Es existieren drei Benutzerschnittstellen für das Programm, jede davon mit mit ihren eigenen Möglichkeiten und jeweils für einen anderen Zweck gedacht.

- **GUI**: in Tk verfasst, eine Fensteroberfläche zum Anklicken für txt2tags .
- **Web**: in PHP verfasst, gestattet es den Benutzers txt2tags in einem Webbrowser ablaufen zu lassen,wodurch die clientseitige Installation von txt2tags überflüssig wird.
- **Kommandozeile**: in Python verfasst, ist dies das Herzstück des Programms. Alle Optionen sind über die Optionsschalter der Kommandozeile konfigurierbar.

Graphische Tk Schnittstelle

Seit Version 1.0 steht eine hübsche graphische Oberfläche zur Verfügung, welche unter Linux, Windows, Mac und anderen Systeme lauffähig ist.

Das Programm erkennt automatisch, ob ihr System zur Anzeige der graphischen Schnittstelle fähig ist, und startet diese, falls der Aufruf ohne jegliches Argument erfolgt. Der Aufruf der graphischen Schnittstelle kann auch über den Aufruf mit der `−−gui`−Option erzwungen werden. Falls irgendwelche Ressourcen fehlen sollten, teilt das Programm ihnen dies mit.

> **Hinweis:** Das Tkinter−Modul ist zwingend erforderlich. Da dieses jedoch bereits standardmäßig in der Python−Distribution enthalten ist, besitzen sie es eventuell schon.

Die Schnittstelle ist ziemlich einfach und intuitiv:

1. Sie lokaliseren die Quelldatei auf ihrem Datenträger und deren Optionen werden geladen.
2. Falls die Quelle dann noch leer sein sollte, müssen sie eine auswählen.
3. Zusätzlich gibt es noch Optionen, die sie gegebenenfalls auswählen können, wobei jedoch keine dieser Optionen zwingend benötigt wird.
4. Zuletzt, drücken Sie die Schaltfläche "Umwandlung starten!".

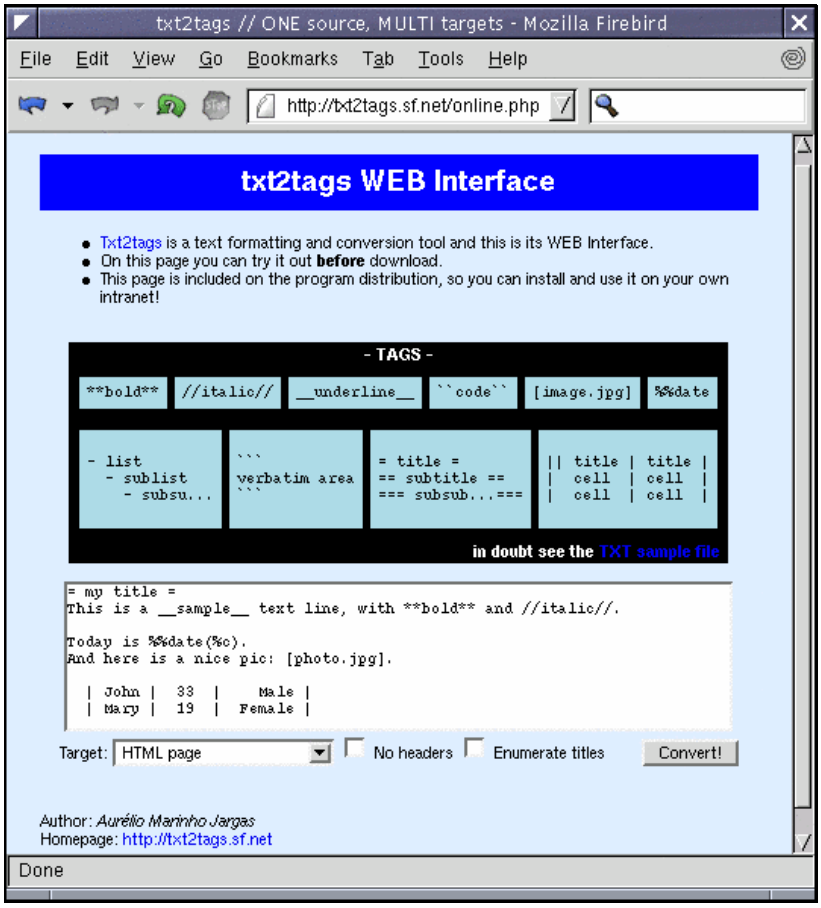
Sehr praktisch kann es sein, die Option "*Ausgabe an Bildschirm leiten*" auszuwählen. Hierdurch können sie den erzeugten Code in einem separatem Fenster überprüfen, dabei wird zunächst noch keine Datei gespeichert. Wenn der erzeugte Code dann ihren Vorstellungen entspricht, können sie diese Option wieder abwählen und die Datei wird gespeichert.

Die voreingestellten Farben der Schnittstelle können in der Datei `~/.txt2tagsrc` geändert werden, zur Einstellung muss dabei der Parameter `%!guicolors` gesetzt werden. Ein Beispiel:

```
% nutze benutzerdefinierte Farben für die graphische Schnittstelle
% (Hintergrund1, Vordergrund1, Hintergrund2, Vordergrund2)
%!guicolors: blue white brown yellow
```

Webschnittstelle

Die Webschnittstelle ist im Internet verfügbar über die Adresse http://txt2tags.sf.net/online.php. Dort können sie dass Programm sofort austesten und nutzen, noch bevor sie es dann eventuell herunterladen.



Sie können die Webschnittstelle auch in ihrem Intranet zur Verfügung stellen, dadurch brauchen sie dann txt2tags nicht lokal auf allen Rechnern in ihrem Intranet installieren.

Kommandozeile als Schnittstelle

Für erfahrene Kommandozeilenbenutzer sollte die Ausgabe von −−help genügen:

```
Usage: txt2tags [OPTIONEN] [Quelldateiname.t2t ...]

  -t, --target        Typ des Zieldokuments festlegen. Derzeit unterstützt:
                      html, xhtml, sgml, tex, lout, man, mgp, moin, pm6, txt
  -i, --infile=DATEI  DATEI als Quellcodedatei festlegen ('-' für STDIN)
  -o, --outfile=DATEI DATEI as the Ausgabedatei festlegen ('-' für STDOUT)
  -n, --enum-title    Titelzeilen fortlaufend nummerieren (1, 1.1, 1.1.1, etc)
  -H, --no-headers    unterdrücke Kopfzeile, Titel und Fußzeile
      --headers       Kopf-, Fuß- und Titelzeilen anzeigen (Voreinstellung: EIN)
      --encoding      Zeichensatz des Zieldokuments (utf-8, iso-8859-1, etc)
      --style=DATEI   DATEI zur Definition des Dokumentenstils benutzen (ähnlich HTML CSS)
      --css-sugar     CSS-freundliche Auszeichnungen für HTML and XHTML-Zieldokumente einfügen
      --css-inside    CSS-Stilinformationen innerhalb des HTML/XHTML Kopfbereichs einfügen
      --mask-email    E-Mail-Adressen verbergen. x@y.z wird umgewandelt in <x (a) y z>
      --toc           Inhaltsverzeichnis hinzufügen
      --toc-only      lediglich das Inhaltsverzeichnis ausgeben
      --toc-level=N   maximale Nummerierungstiefe im Inhaltsverzeichnis einstellen
      --rc            benutzerdefinierte Einstellungsdatei ~/.txt2tagsrc laden (Voreinstellung: EIN)
      --gui           Aufruf der graphischen Tk-Schnittstelle
  -q, --quiet         stiller Modus, jegliche Ausgaben werden unterdrückt (außer Fehlermeldungen)
  -v, --verbose       Statusmeldungen während der Konvertierung ausgeben
  -h, --help          diese Hilfeinformation anzeigen und Programm beenden
  -V, --version       Programmversion anzeigen und Programm beenden
      --dump-config   alle gefundenen Konfigurationsquellen ausgeben
```

```
Abschalten von einzelnen Optionen:
     −−no−outfile, −−no−infile, −−no−style, −−no−encoding, −−no−headers
     −−no−toc, −−no−toc−only, −−no−mask−email, −−no−enum−title, −−no−rc
     −−no−css−sugar, −−no−css−inside, −−no−quiet

Beispiel:
     txt2tags −t html −−toc meineDatei.t2t

Per Voreinstellung wird die Ausgabe der Konvertierung in der Datei 'meineDatei.<target>' gespeichert.
Benutzen sie −−outfile um den Namen der Ausgabedatei festzulegen.
Wird als Eingabedatei '−' angeben, liest das Programm von STDIN.
Wird als Eingabedatei '−' angegeben, erfolgt die Ausgabe nach STDOUT.
```

Beispiele

Vorausgesetzt, sie haben eine Quelldatei `Datei.t2t`, los geht's mit der Konvertierung!.

Konvertierung nach HTML	`$ txt2tags −t html Datei.t2t`
Das gleiche, mit Umleitung	`$ txt2tags −t html −o − Datei.t2t > Datei.html`
	.
inkl. Inhaltsverzeichnis	`$ txt2tags −t html −−toc Datei.t2t`
und die Titel nummeriert	`$ txt2tags −t html −−toc −−enum−title Datei.t2t`
	.
kurze Inhaltsübersicht	`$ txt2tags −−toc−only Datei.t2t`
diese dann noch nummeriert?	`$ txt2tags −−toc−only −−enum−title Datei.t2t`
	.
Einzeiler gelesen von STDIN	`$ echo −e "\n**bold**" | txt2tags −t html −−no−headers −`
Test der E−Mail Maskierung	`$ echo −e "\njohn.wayne@farwest.com" | txt2tags −t txt −−mask−email −−no−headers −`
Nachbearbeitung mittels sed	`$ txt2tags −t html −o− Datei.t2t | sed "s/<BODY .*/<BODY BGCOLOR=green>/" > datei.html`

Anmerkung

Seit Version 1.6 kann mit Hilfe der `%!preproc` and `%!postproc` Konfigurationsfilter eine Vor− und Nachbearbeitung vorgenommen werden.

Teil II − Ok, I möchte das Programm. Wie geht's weiter?

Laden sie das Programm einfach herunter und installieren sie es auf ihrem Rechner.

Herunterladen & Installieren von Python

Zunächst müssen sie den Python−Interpreter herunterladen und auf Ihrem System installieren. Sollte Python bereits installiert sein, überspringen sie diesen Schritt.

Python ist eine der tollsten Programmiersprachen die derzeit verfügbar sind, und ist für Windows, Linux, UNIX, Macintosh, und andere Systeme verfügbar. Sie können es sich von der Python Webseite. herunterladen. Auf derselben Seite finden sie auch Hinweise zur Installation. Txt2tags läuft mit Python ab der Version 1.5 (oder neuer).

Falls sie sich nicht sicher sind, ob auf ihrem System Python installiert ist, öffnen sie eine Konsole (tty, xterm, MSDOS) und geben sie am Kommandoprompt `python` ein. Falls Python nicht installiert sein sollte, wird sie ihre System darauf aufmerksam machen.

Download von txt2tags

Der offizielle Ort für die txt2tags−Distribution ist die Website des Programms, aufrufbar via http://txt2tags.sf.net/src.

Alle zum Programm gehörigen Dateien finden sich in einem tarball (.tgz Datei), die Dateien können mit den meisten gebräuchlichen Pack−Programmen (einschließlich Winzip) aus dem tarball entpackt werden

Besorgen sie sich stets die **neueste** Version (wobei die neuste Version stets auch die höchste Versionsnummer aufweist). Die früheren Versionen verbleiben lediglich zu Dokumentationszwecken auf der Seite.

Installation von txt2tags

Da es sich bei txt2tags um ein gewöhnliches Python−Skript handelt, muss txt2tags nicht gesondert installiert werden.

Die einzige Datei, die sie brauchen, um das Programm zu nutzen ist, das txt2tags−Skript selbst. Bei allen anderen Dateien in der Download−Archivdatei handelt es sich entweder um Dokumentationsdateien, Werkzeuge oder Beispieldateien.

Die sicherste Art und Weise, txt2tags ablaufen zu lassen besteht darin, es über den Python−Compiler aufzurufen:

```
Eingabeprompt$ python txt2tags
```

Wenn sie txt2tags auf ihrem Rechner als einzeln aufrufbares Programm "installieren" wollen, kopieren (oder verknüpfen) sie das txt2tags−Skript in ein Verzeichnis, welches in der Umgebungsvariable PATH enthalten ist und stellen sie sicher, dass ihr System weiß, mit welchem Programm ihre Datei aufzurufen ist.

UNIX/Linux

Markieren sie das Skript als ausführbar (`chmod a+x txt2tags`) und kopieren sie das Programm in ein Verzeichnis, das Bestandteil ihrer Umgebungsvariable PATH ist (`cp txt2tags /usr/local/bin`)

Windows

Benennen sie das Skript txt2tags nach txt2tags.py um (`ren txt2tags txt2tags.py`) und kopieren sie es in ein Verzeichnis, das Bestandteil ihrer Umgebungsvariable PATH ist (`copy txt2tags.py C:\WINNT`)

Falls sie die graphische Benutzerschnittstelle des Programms nutzen wollen, können sie für das Programm anschließend ein Symbol auf ihrem Desktop anlegen.

Spezielle Pakete für Windows−Benutzer

Für txt2tags sind auch zwei .EXE−Dateien verfügbar, mit denen das Programm auf Windows−Rechnern mit nur wenigen Mausklicks installiert werden kann:

- Das bloße txt2tag−Skript für all diejenigen, die den Python−Interpreter bereits installiert haben
- Die 'Stand−alone'−Version, welche keinen Python−Interpreter benötigt (eine abgespeckte Python−Installationsdatei ist in diese Version integriert)

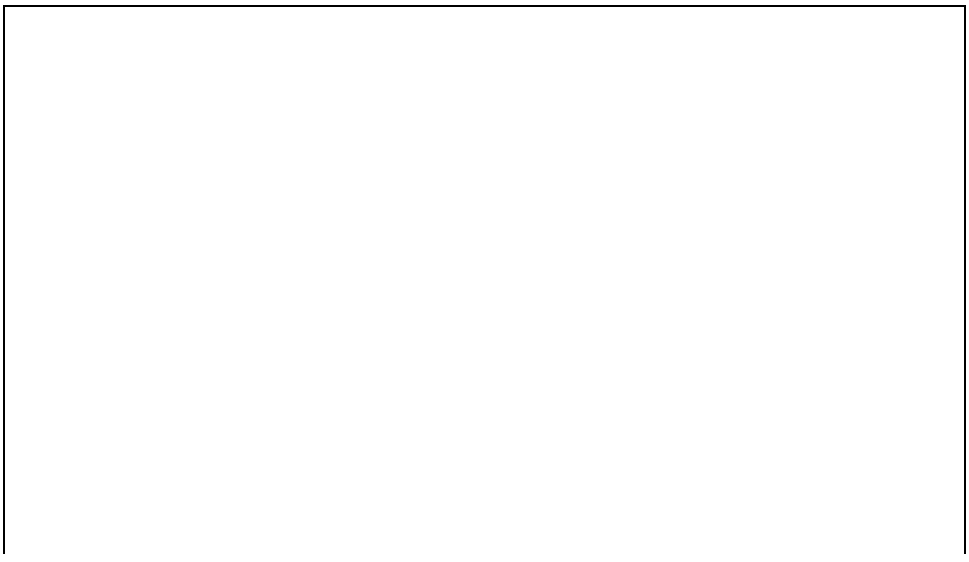
Besuchen Sie bitte die *txt2tags−Windows* Webseite um diese Pakete herunterzuladen:
http://txt2tags−win.sf.net

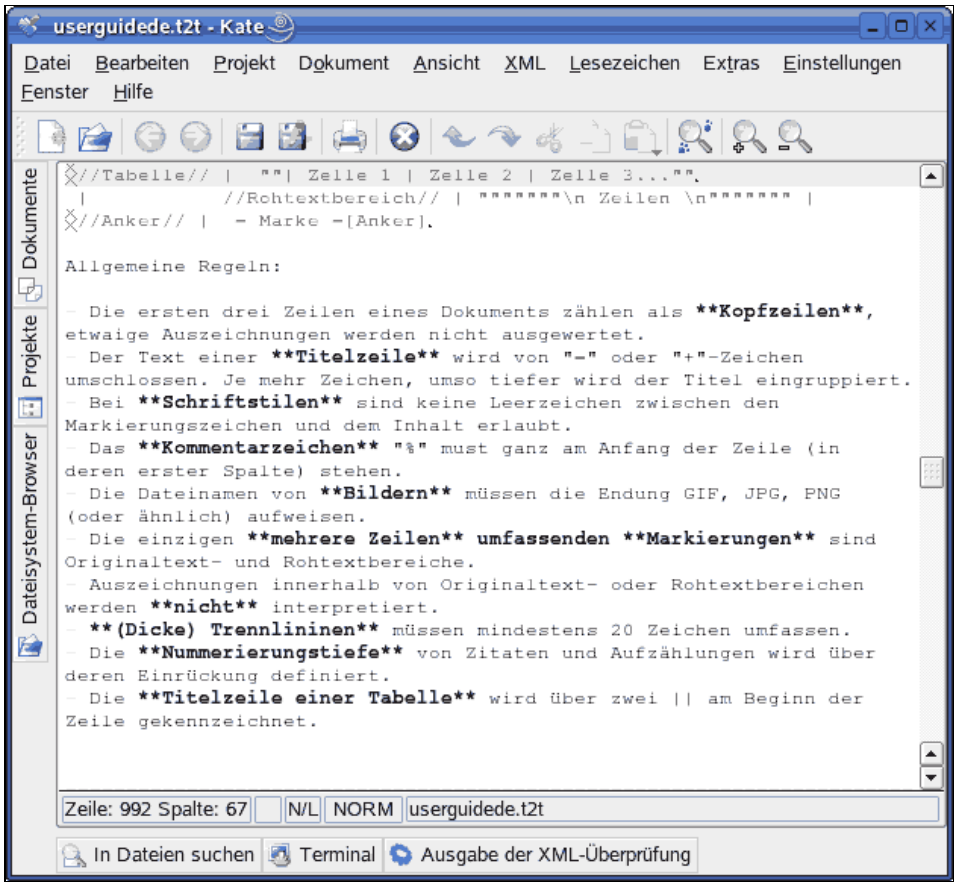
Installation von Dateien zum Hervorheben der Programmsyntax für Texteditoren

Txt2tags wird mit sehr nützlichen Dateien zum Hervorheben der Programmsyntax ausgeliefert, solche Dateien sind dabei für die folgenden Texteditoren verfügbar:

- Vim (www.vim.org)
- Emacs (www.emacs.org)
- Nano (www.nano−editor.org)
- Kate (http://kate.kde.org)

Über diese Dateien zum Hervorheben der Programmsyntax wird der Editor über alle txt2tags−Regeln und Auszeichnungen informiert, dies unterstützt den Benutzer dabei, fehlerfreie Dokumente zu verfassen. Da die Auszeichnungen farbig hervorgehoben werden, sehen sie unmittelbar, ob sie die Auszeichnungsmarken in ihren Texten korrekt eingesetzt haben.





Beispieldatei, geöffnet im Editor Kate

Für jeden Editor erfolgt die Installation dieser Dateien zum Hervorheben der Programmsyntax unterschiedlich, lesen sie vor der Installation bitte die Kopfzeilen der jeweiligen Syntax−Datei sowie die Dokumentation zu ihrem Editor.

Teil III − Verfassen und Konvertieren ihres ersten Dokuments

Überprüfung der Werkzeuge

Um ihre erste Textumwandlung erfolgreich vornehmen zu können, brauchen sie drei Dinge: txt2tags, einen Texteditor und einen Webbrowser.

1. Stellen Sie sicher, dass txt2tags auf ihrem Rechner installiert und funktionsfähig ist.

    - **Kommandozeilenschnittstelle:** Rufen Sie "txt2tags" auf der Kommandozeile auf, und sie sollten eine Meldung erhalten, die eine "Fehlende Eingabedatei" anmahnt. Falls der Aufruf von txt2tags fehlschlägt, versuchen sie "python /Pfad/zu/txt2tags" oder auch "/Pfad/zu/Python /Pfad/zu/txt2tags" (falls Python nicht Bestandteil ihrer Umgebungsvariable PATH ist).
    - **Graphische Schnittstelle:** Klicken sie auf das Programmsymbol um die graphische Benutzerschnittstelle zu starten.
2. Starten sie einen Texteditor mit dem sie gut vertraut sind. Hierbei kann **jeder** Texteditor genutzt werden, angefangen vom guten alten vi über MS−Word bis hin zum Texteditor aus der OpenOffice−Suite. Erstellen sie ein neues, leeres Dokument, das sie dann als ihr erstes txt2tags−Dokument verfassen.
3. Starten Sie den von ihnen bevorzugten Webbrowser, sie brauchen ihn, um das Resultat der Konvertierung des von ihnen verfassten Dokuments in eine HTML−Seite zu überprüfen.

Schreiben sie den Kopfbereich des Dokuments

1. Wechseln sie zu ihrem Texteditor und tippen sie auf der ersten Zeile den Titel ihres Dokuments: *Mein erstes Dokument*
2. Fügen sie nun einen Untertitel hinzu, indem sie auf der zweiten Zeile den folgenden Text einfügen: *Ein Test von txt2tags*
3. Anschließend können sie auf der dritten Zeile das Datum der Abfassung vermerken, wie etwa: *Freitag, 26. August 2005*

   Falls alles glatt gelaufen ist, sollten sie jetzt ein dreizeiliges Dokument mit folgendem Inhalt vor sich sehen:

   ```
   Mein erstes Dokument
   Ein Test von txt2tags
   Freitag, 26. August 2005
   ```

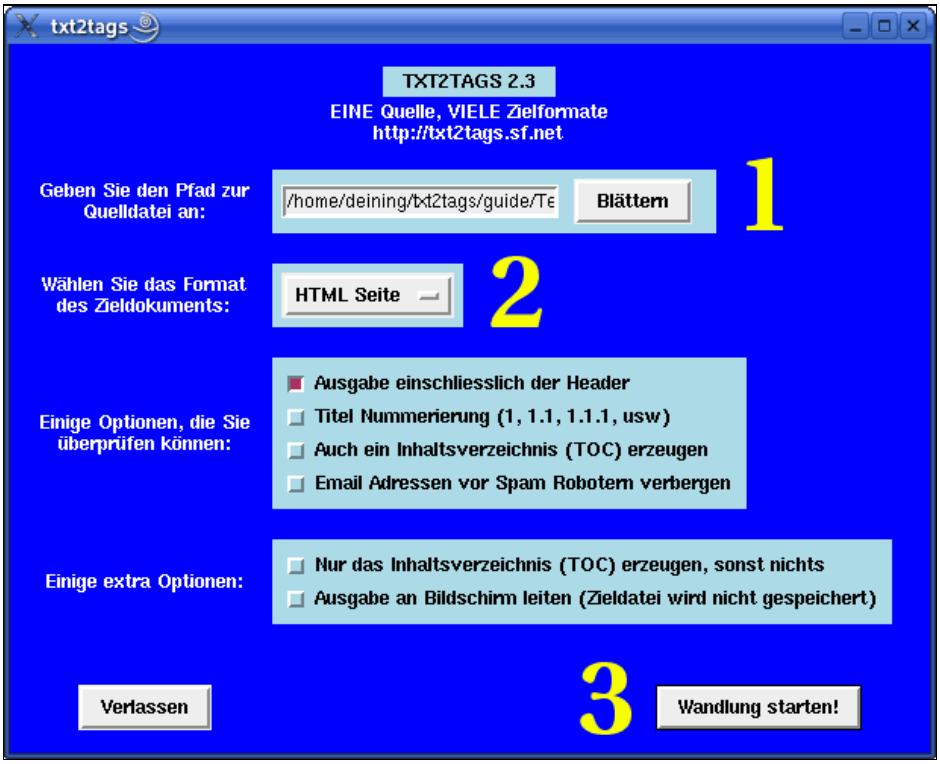
   Dies ist zwar nur ein kleiner Teil des gesamten Dokuments, wir können diesen Teil jedoch bereits konvertieren und anschließend die Ergebnisse überprüfen.

   Speichern sie dazu das Dokument unter dem Namen `Test.txt`. Merken sie sich dabei, in welchen Ordner sie die Datei abspeichern, sie benötigen diese Information in Kürze wieder.

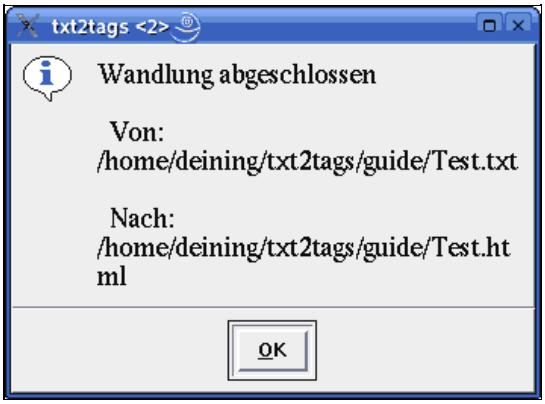
Die erste Umwandlung eines Dokuments − GUI Schnittstelle

Falls sie txt2tags über die Kommandozeilenschnittstelle bedienen wollen, überspringen sie bitte diesen Schritt und befolgen sie die Anweisungen im nächsten Abschnitt.

Falls jedoch nach dem Aufruf von txt2tags die graphische Schnittstelle erscheint, folgen sie diesen Schritten:



1. Klicken sie auf die mit "Blättern" beschriftete Schaltfläche und wählen sie das Dokument `Test.txt`, welches sie soeben erstellt und gespeichert haben (sicher haben sie sich den Ordner gemerkt, in dem sie das Dokument abgespeichert haben!).
2. Zurück am Ausgangsschirm wählen sie "HTML Seite" in der mit "Wählen Sie das Format des Zieldokuments" beschrifteten Auswahlbox.
3. Klicken sie auf die Schaltfläche "Wandlung starten!".



Nun wird ein Fenster erscheinen, welches sie darüber informiert, dass die Datei erfolgreich konvertiert wurde. Beachten sie dabei, dass die erzeugte HTML−Seite in demselben Ordner gespeichert wurde wie die für die Konvertierung benutzte Quelldatei, die Datei weist dabei die Endung "html" auf.

Die erste Umwandlung eines Dokuments − Kommandozeile

Falls sie txt2tags über die graphische Schnittstelle bedienen wollen, befolgen sie bitte nicht diesen Schritt sondern folgen sie den Anweisungen im vorherigen Abschnitt.

Falls sie jedoch txt2tags über die Kommandozeile bedienen wollen, wechseln sie in den Ordner, in welchen sich das von ihnen erstellte Dokument abgespeichert haben und tippen dort das folgende Kommando ein:

```
txt2tags  −−target html  Test.txt
```

Beachten sie dabei bitte, dass Leerzeichen zwischen den einzelnen Bestandteilen des Kommandos, nicht aber innerhalb des Textpassage "−−target" erlaubt sind, da es sich bei letzterer um eine Option handelt. Dieser Option folgt der Text "html", hierüber wird dem Programm mitgeteilt, in welches Format ihre Quelldatei konvertiert werden soll. Der letzte Bestandteil des Kommandos schließlich ist der Dateiname des Quelldokuments.

Nach vollendeter Konvertierung wird die resultierende Datei unter dem Namen `Test.html` abgespeichert, wobei das Programm die folgende Mitteilung ausgibt: "*txt2tags erzeugte Test.html*". Sollte das nicht der Fall sein, erscheint eine entsprechende Meldung, der sie entnehmen können, welcher Fehler sich bei der Eingabe des Kommandos auf der Kommandozeile eingeschlichen hat. Lesen sie solche Meldungen sorgfältig durch!

Hier folgt ein Beispiel dafür, wie sich das Ganze am Bildschirm präsentiert:

```
prompt$ txt2tags −−target html Test.txt
txt2tags erzeugte Test.html
prompt$
```

Überprüfung des Resultats

Öffnen sie die Datei `Test.html` in einem Webbrowser um zu überprüfen, ob alles korrekt dargestellt wird.

Da ist das Resultat! Sie haben lediglich 3 kurze Textzeilen eingeben, und txt2tags hat den Rest erledigt und dabei sowohl den HTML−Kopfbereich (inklusive Dateiinformationen) erstellt als auch die Ausrichtung, Größe, Erscheinungsbild und Abstände für den Text adäquat festgelegt. Wie sie sehen können, erscheint der Titel ihres Texts auch in der Titelzeile des Browsers.

Sie schreiben den Text, txt2tags erledigt den Rest ;)

Hinweis: Sie können auf von txt2tags erzeugte Dokumente auch CSS−Stylesheets anwenden, dadurch kann das Erscheinungsbild der Seite 100%−ig von ihnen festgelegt werden.

Verfassen des Dokuments

Zurück im Texteditor ist nun der nächste Schritt die Eingabe der eigentlichen Inhalte des Dokuments. Sie können hierbei in Klartext schreiben, so wie sie das normalerweise beim Verfassen von E−Mail−Nachrichten auch tun. Sie werden sehen, dass txt2tags Abschnitte und Aufzählungen automatisch erkennt, sie müssen sie gar nicht besonders auszeichnen.

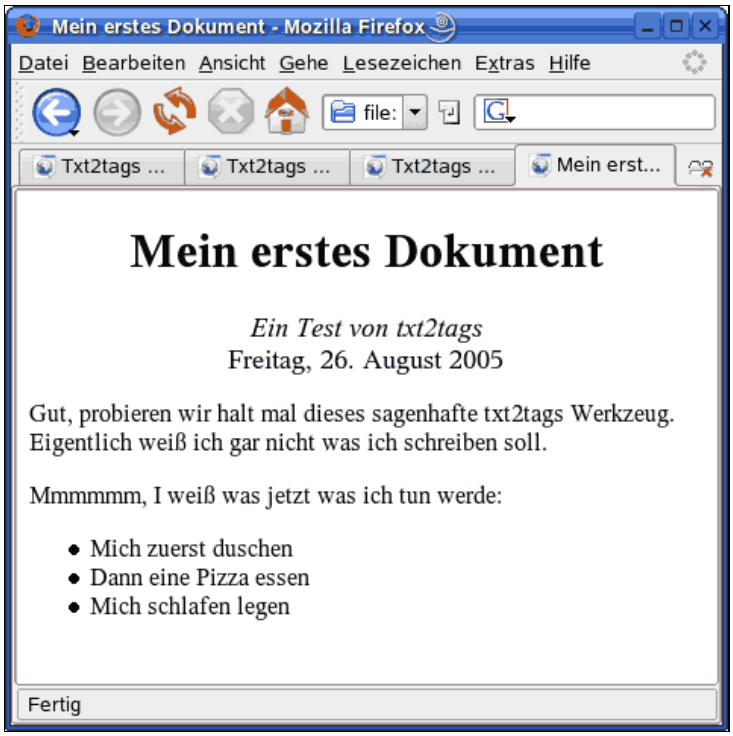
Und dann wieder: speichern sie ihr Dokument, wandeln es um und überprüfen sie das Ergebnis. Das ist der Entwicklungszyklus eines Dokuments in txt2tags. Sie konzentrieren sich ganz auf die Inhalte des Dokuments und sind mit ihrem Dokument schneller fertig als wenn sie ein mit Funktionen überladenes Textverarbeitungsprogramm benutzt hätten. Keine Mausklicks, keine Menüs, Fenster oder sonstige Ablenkungen.

Vergleichen sie doch einfach die folgenden Inhalte der Quelldatei `Test.txt`, welche in Klartext verfasst wurde, mit der bei der Konvertierung erzeugten HTML−Seite:

```
Mein erstes Dokument
Ein Test von txt2tags
Freitag, 26. August 2005

Gut, probieren wir halt mal dieses sagenhafte txt2tags Werkzeug.
Eigentlich weiß ich gar nicht was ich schreiben soll.

Mmmmmm, I weiß was jetzt was ich tun werde:
- Mich zuerst duschen
- Dann eine Pizza essen
- Mich schlafen legen
```



Sie können eine ganze Homepage erstellen, ohne auch nur das geringste Wissen über HTML zu besitzen! Sie brauchen keinerlei Auszeichnungsmarken einfügen. Und der Clou, dieselbe Quelldatei kann auch in jedes andere der von txt2tags unterstützten Formate umgewandelt werden.

Neben dem Klartext versteht txt2tags einige ganz einfache Auszeichnungsmarken, welche sie nutzen können, falls sie eine andere Textformatierung oder −struktur benötigen, wie etwa Fett− oder Kursivschrift, Titel, Bilder, Tabellen und anderes mehr. Hier zwei einfache Beispiele: `**zeichnet`

`fettgedruckte Passagen aus**` und `== kann für Titel genutzt werden ==`. Die Auszeichnungen lernen sie am besten mittels der Demonstration von txt2tags Auszeichnungen.

Teil IV − Die Konzepte von txt2tags beherrschen lernen

Die Bereiche eines .t2t Dokuments

Mit txt2tags ausgezeichnete Dateien sind in drei Bereiche aufgeteilt. Jeder Bereich hat einen anderen Zweck und besitzt seine eigenen Regeln. Es handelt sich dabei um:

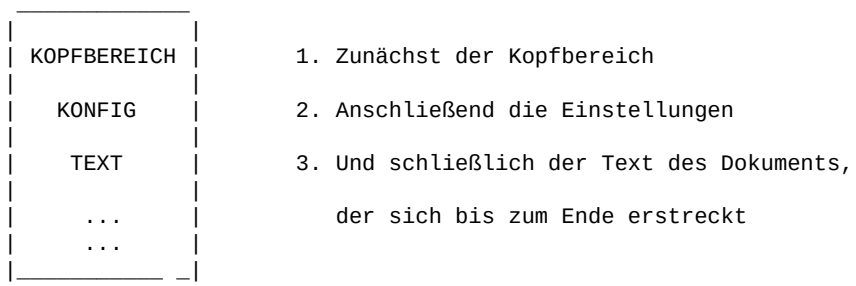
Kopfbereich
: Dies ist der Platz für den Titel des Dokuments, zusätzlich können hier die Autoren, die Version und der Zeitpunkt der Abfassung des Dokuments angegeben werden. (optional)

Konfigurationsbereich
: Dies ist der richtige Platz für im gesamten Dokument gültige Einstellungen sowie für das Verhalten des Parsers beeinflussende Anweisungen. (optional)

Textbereich
: Dies ist der Platz für den Inhalt des Dokuments. (wird benötigt)

Wie man aus der obigen Aufzählung ersehen kann, sind die ersten beiden Bereiche nicht zwingend erforderlich, insofern ist der *Textbereich* der einzige Bereich, für den Text eingegeben werden muss.

Die Abgrenzung der einzelnen Bereiche untereinander erfolgt nach speziellen Regeln, die wir im Detail im nächsten Kapitel kennenlernen werden. Zunächst wollen wir uns erstmal mit einer graphischen Darstellung der einzelnen Bereiche eines Dokuments zufrieden geben:



Hier bereits ein Kurzüberblick darüber, wie die einzelnen Bereiche definiert sind:

Kopfbereich	Die ersten drei Zeilen der Datei, oder eine Leerzeile als erste Zeile falls kein Kopfbereich erforderlich ist.
Konfigurationsbereich	Beginnt direkt im Anschluss an den Kopfbereich (zweite oder vierte Zeile) und endet mit Beginn des *Textkörpers*.
Textkörper	Die erste gültige Textzeile (keine Kommentar− oder Parameterzeile) nach dem *Kopfbereich*.

Komplettes Beispiel

```
Der Titel meines tollen Dokuments
Dr. Andreas Deininger
Letzte Aktualisierung: %%mtime(%c)

%! Target  : html
%! Style   : fancy.css
%! Encoding: iso−8859−1
%! Options : −−toc −−enum−title

Hallo! Dies ist mein Test−Dokument.
Es endet bereits hier.
```

Kopfbereich

Positionierung:

- Eindeutig festgelegte Position: Die **ersten 3 Zeilen** der Datei. Nicht mehr und nicht weniger
- Eindeutig festgelegte Position: Die **erste Zeile** der Datei falls es sich dabei um eine Leerzeile handelt. Dies bedeutet dann, dass auf den Kopfbereich verzichtet wird.

Der Kopfbereich ist der einzige Bereich, dessen Position eindeutig anhand der Zeilennummern festgelegt ist: es handelt sich dabei um die ersten drei Zeilen der Quelldatei.

Diese Zeilen können mit beliebigen Inhalten gefüllt werden, der Typ der Informationen ist nirgends statisch festgeschrieben. Dennoch wird sich für die meisten Dokumente ein Aufbau der Kopfbereichs in der folgenden Art anbieten:

- *1. Zeile*: Titel des Dokuments
- *2. Zeile*: Name des Autors und/oder seine E−Mail−Adresse
- *3. Zeile*: Version des Dokuments und/oder Zeitpunkt der Abfassung (hierfür bietet sich das `%%date`−Makro an)

Denken sie daran, dass die ersten drei Zeilen des Quelldokuments zugleich auch die ersten drei Zeilen des Zieldokuments darstellen, welche abgesetzt und gegenüber dem Textkörper stark hervorgehoben dargestellt werden (z. B. durch die Schriftgröße, Fettschrift). Falls das Dokument einen seitenorientierten Aufbau besitzt, wird der Kopfbereich alleinstehend zentriert auf der ersten Seite des Dokuments wiedergegeben.

Weniger (oder gar keine) Kopfzeilen

Gelegentlich wird der Fall auftreten, dass der Autor des Dokuments weniger als drei Zeilen für den Kopfbereich benötigt, etwa wenn er lediglich den Titel des Dokuments und/oder das Datum der Abfassung angeben möchte.

Lassen sie in solch einem Fall die zweite und/oder dritte Zeile ihres Dokuments einfach leer (Leerzeile) und es wird entsprechend auch kein Text im Kopf des Zieldokuments angegeben werden. Denken sie jedoch daran, dass diese Leerzeilen, auch wenn sie keinen Inhalt tragen, dennoch als ein Teil des Kopfbereich zählen, so dass der der Textkörper des Dokuments nach wie vor frühestens **nach** der dritten Zeile beginnen kann.

Der Titel (die erste Zeile) ist das einzige Element im Kopfbereich des Dokuments, das zwingend angegeben werden muss. Falls sie die erste Zeile des Dokuments leer lassen und damit den Titel quasi auslassen, teilen sie txt2tags mit, dass das Dokument **keinen Kopfbereich** besitzt. Dies bedeutet dann zugleich, dass der *Textbereich* direkt anschließend beginnen kann, in diesem speziellen Fall also bereits auf der zweiten Zeile des Dokuments.

Der Verzicht auf die Angabe von Kopfzeilen ist besonders dann nützlich, wenn sie dem Dokument nach der Konvertierung eine eigene, von ihnen angepasst gestaltete Titelseite hinzufügen möchten. Gewöhnlich wird die Kommandozeilenoption `−−no−headers` für diese Art der angepassten Dokumentgestaltung verwendet.

Auf den Punkt gebracht

In Kurzform: "Kopfzeilen sind lediglich durch ihre Position definiert, nicht über ihren Inhalt"

Platzieren sie beliebigen Text auf der ersten Zeile ihres Dokuments, und er wird auch im konvertierten Dokument auf der ersten Zeile erscheinen. Dasselbe gilt auch für die auf der zweiten und der dritten Dokumentenzeile platzierten Kopfzeilen.

Konfigurationsbereich

Positionierung:

- Beginnt direkt im Anschluss an den Kopfbereich
  - ♦ Beginnt auf der **vierten Zeile** der Datei falls ein **Kopfbereich** angegeben wurde
  - ♦ Beginnt auf der **zweiten Zeile** der Datei falls **kein Kopfbereich** angegeben wurde
- Endet mit dem Beginn des Textbereichs
  - ♦ Ein Leerzeile, eine Kommentarzeile oder eine Zeile in der keine Einstellung festgelegt wird beendet automatisch den Konfigurationsbereich

Der Konfigurationsbereich ist optional. Ein gewöhnlicher Benutzer kann viele txt2tags−Dateien verfassen ohne dass er überhaupt weiß, dass ein Konfigurationsbereich existiert, ein erfahrener Benutzer aber wird dessen Mächtigkeit und die über seine Nutzung gebotene Flexibilität zu schätzen wissen.

Der Konfigurationsbereich kann dazu genutzt werden, dokumentenspezifische Einstellungen zu speichern, so dass diese bei der Konvertierung des Dokuments nicht jedesmal über die Kommandozeile explizit als Kommandozeilenoptionen mit angegeben werden müssen. Auf diese Art und Weise können zum Beispiel das Zielformat für die Konvertierung sowie die Kodierung der Datei festgelegt werden.

Für weitergehende Informationen zu diesem Thema lesen sie bitte den Abschnitt Einstellungen .

Textbereich

Positionierung:

- Beginnt mit der ersten gültigen Textzeile innerhalb der Datei
  - ♦ Zeilen innerhalb des Kopfbereichs, Zeilen welche Einstellungen enthalten sowie Kommentarzeilen sind **keine** gültigen Textzeilen
- Endet zugleich mit dem Ende der Datei (EOF)

Jeglicher Text außerhalb des Kopfbereichs und des Konfigurationsbereichs zählt zum Textbereich.

Der Textkörper enthält die Inhalte des Dokuments sowie alle Auszeichnungen und −strukturen, welche von txt2tags erkannt werden. Innerhalb des Textkörpers können auch Kommentare, etwa für *Aufgabenlisten (TODOs)* oder für sonstige, nur für den Verfasser bestimmte Notizen niedergeschrieben werden.

Sie können die Kommandozeilenoption `--no-headers` benutzen, um nur den Textkörper des Dokuments zu konvertieren, der Kopfbereich wird dann unterdrückt. Dies ist nützlich, wenn die Informationen des Kopfbereichs in einer separaten Datei hintergelegt sind, an diese Datei kann dann der konvertierte Textkörper angehängt werden.

Einstellungen

Einstellungen sind spezielle Konfigurationsangaben, über welche das Ergebnis der Konvertierung beeinflusst werden kann. Solche Einstellungen müssen im Konfigurationsbereich des Quelldokuments platziert werden. Ihre Syntax lautet:

%! Schlüsselwort : Wert

Liste aller gültigen Schlüsselwörter:

Schlüsselwort	Beschreibung
Target	Definiert das Zielformat in welches das Dokument beim Fehlen von Kommandozeilenoptionen konvertiert wird.
Options	Definiert voreingestellt Optionen, welche bei der Konvertierung genutzt werden. Das Format ist dasselbe wie dasjenige von Kommandozeilenoptionen.
Style	Definiert das Aussehen des Dokuments. Damit kann einer HTML/XHTML−Datei eine CSS−Datei zugeordnet oder in einem LaTeX−Dokument ein Paket geladen werden.
Encoding	Definiert die Kodierung bzw. den Zeichensatz des Dokument. Findet immer dann Anwendung, wenn das Dokument Umlaute, mit Akzenten versehene Buchstaben oder andere nicht ASCII−Zeichen enthält.
PreProc	Eingabefilter. Definiert Regeln für das "Suchen und Ersetzen" welche auf das Quelldokument angewendet werden.
PostProc	Ausgabefilter. Definiert Regeln für das "Suchen und Ersetzen" welche auf das bereits konvertierte Dokument angewendet werden.

Beispiel:

```
%! Target  : html
%! Options : --toc --toc-level 3
%! Style   : fancy.css
%! Encoding: iso-8859-1
%! PreProc : "HPM"        "Hans-Peter Müller"
%! PostProc: '<BODY.*?>'  '<BODY bgcolor="yellow">'
```

Kommandozeilenoptionen

Der schnellste Weg, das in txt2tags vordefinierte Verhalten abzuändern besteht darin, die verfügbaren Kommandozeilenoptionen zu nutzen. Diese Optionen sind dabei allerdings nur dann verfügbar, wenn txt2tags auf der Kommandozeile aufgerufen wird, nicht aber wenn die graphische Schnittstelle oder die Webschnittstelle benutzt wird.

Wie andere Systemwerkzeuge auch, akzeptiert das Programm eine Reihe von vordefinierten Optionen. Eine Option wird angegeben entweder über einen Trennstrich gefolgt von einem Buchstaben oder über zwei Trennstriche gefolgt von einem oder mehreren Wörtern, Beispiele hierfür sind `-t` bzw. `--target`. Und da wir schon bei der Option target sind: dies ist die einzige Option, deren Angabe zwingend erforderlich ist, alle anderen sind optional.

Zu den am häufigsten genutzen Optionen zählen `--outfile`, über welche der Name der Ausgabedatei bestimmt werden kann, `--toc`, welche die automatische Erzeugung eines Inhaltsverzeichnisses bewirkt und `--encoding`, über welche die Kodierung bzw. der Zeichensatz des Dokuments bestimmt werden kann. Die meisten Optionen können abgeschaltet werden, indem ihrem Name die Zeichenkette "no−" vorangestellt wird, zum Beispiel: `--no-encoding` and `--no-toc`.

Alternativ können sie die von ihnen gewünschten Optionen für eine bestimmte Quelldatei auch innerhalb des Konfigurationsbereichs dieser Datei angeben, hierzu müssen sie dort die `%!options`−Einstellungen nutzen. Bei diesem Vorgehen brauchen sie dann die Optionen nicht nochmals auf der Kommandozeile eingeben. Beispiel: `%!options: --toc -o MeinDokument.html`. Eine Ausnahme stellt die Angabe des Zielformats dar, bei der Einstellung dieses Parameters innerhalb der Quelldatei kommt eine andere Zeichenkette zum Einsatz: `%!target: html`.

Verwenden sie die Option `--help`, um eine komplette Liste aller in txt2tags verfügbaren Optionen angezeigt zu bekommen.

Benutzerdefinierte Konfigurationsdatei (RC−Datei)

Die benutzerdefinierte Konfigurationsdatei (auch rc−Datei genannt, abgeleitet von resource control) dient als zentraler Ort zur Speicherung von Einstellungen, die bei ALLEN Konvertierungen Berücksichtigung finden. Anstelle dieselben Einstellungen in jeder von ihnen verfassten .t2t−Datei wiederholen, schreiben sie diese einfach einmal in ihre benutzerdefinierte Konfigurationsdatei. Die dort notierten Einstellungen werden dann zukünftig bei jeder Konvertierung angewandt, egal ob sie mit txt2tags bereits bestehende oder neu erstellte Quelldateien bearbeiten.

Abhängig vom verwendeten Betriebssystem sind sowohl für den Dateinamen als auch für das Verzeichnis, in welchem die benutzerdefinierte Konfigurationsdatei abgespeichert wird, unterschiedliche Werte zu wählen. Alternativ kann der Speicherort dieser Datei vom Benutzer auch über eine Umgebungsvariable deklariert werden.

	Speicherort für die RC Datei
Windows:	`%HOMEPATH%\_t2trc`
Linux und andere:	`$HOME/.txt2tagsrc`
benutzerdefiniert:	Umgebungsvariable `T2TCONFIG`

Das Format der Einstellungen ist dabei identisch mit denjenigen, welche sie bereits zur Angabe von Einstellungen innerhalb des Konfigurationsbereichs von .t2t−Dateien kennengelernt haben. Ein Beispiel für eine solche benutzerdefinierte Konfigurationsdatei findet sich im txt2tags−Tarball im Verzeichnis `doc/txt2tagsrc`.

Beispiel:

```
% Meine Konfigurationsparameter

%%% HTML−Tags stets CSS−freundlich auszeichnen
%!options(html): −−css−sugar

%%% Voreingestellte Nummerierungstiefe für Inhaltsverzeichnisse anpassen (für alle Zielformate)
%!options: −−toc−level 4

%%% ISO−8859−1 als Standardkodierung von txt2tags−Dokumenten festlegen
%!options: −−encoding iso−8859−1
```

Jede nicht leere Zeile, die keinen Kommentar und auch keine gültige Einstellungsoptionen enthält, führt zur Ausgabe einer Fehlermeldung, wenn txt2tags aufgerufen wird. Seien sie insofern bitte vorsichtig, wenn sie diese Datei bearbeiten.

Txt2tags wendet beim Konvertierungsvorgang für jegliche Quelldateien automatisch die Inhalte der benutzerdefinierten Konfigurationsdatei an. Um bei der Konvertierung einer einzelnen Datei dieses Verhalten außer Kraft zu setzen, existiert die Kommandozeilenoption `−−no−rc`.

Konfigurationsquellen und die Reihenfolge ihrer Anwendung

Es existieren drei Möglichkeiten, txt2tags darüber zu informieren, welche Optionen und Einstellungen bei der Konvertierung Berücksichtigung finden sollen. Diese werden in der folgenden Reihenfolge gelesen und angewandt:

1. Einstellungen, welche in der benutzerdefinierten Konfigurationsdatei vorgenommen wurden (RC)
2. Einstellungen im Konfigurationsbereich des Quelldokuments
3. Kommandozeilenoptionen

Txt2tags liest zunächst die benutzerdefinierte Konfigurationsdatei ein (sofern vorhanden) und wendet die dort gefundenen Einstellungen auf das Quelldokument an. Danach wird der Konfigurationsbereich der Quelldatei nach dort getätigten Einstellungen durchsucht. Falls dort solche Einstellungen gefunden werden, werden diese ebenfalls angewandt, wobei ggf. bereits in der benutzerdefinierten Konfigurationsdatei gemachte Einstellungen überschrieben werden. Schließlich finden auch noch die auf der Kommandozeile angegebenen Einstellungsparameter Anwendung, wobei diesen Einstellungen nochmals höhere Priorität eingeräumt wird.

Wurde also etwa die Kodierung für das Quelldokument in allen drei Konfigurationsquellen definiert, so wird der an der Kommandozeile für die Dokumentenkodierung angegebene Wert benutzt werden.

Das %!include Kommando

Das `include`−Kommando kann dazu genutzt werden, die Inhalte einer externen Datei in den Textbereich des Quelldokuments einzufügen. Es handelt sich hierbei nicht um eine Konfigurationseinstellung sondern um ein Kommando, welches nur innerhalb des Textbereichs des Dokuments gültig ist.

Mit Hilfe des `include`−Kommandos kann ein großes Dokument in mehrere kleinere Teildokumente (etwa als Kapitel innerhalb eines Buchs) aufgeteilt werden, oder aber es kann der gesamte Inhalt einer externen Datei in das Quelldokument eingefügt werden. Ein Beispiel:

```
Mein erstes Buch
Dr. Andreas Deininger
Erste Auflage

%!include: Einleitung.t2t
%!include: Kapitel1.t2t
%!include: Kapitel2.t2t
...
%!include: Kapitel9.t2t
%!include: Nachwort.t2t
```

An die `%!include`−Zeichenkette wird einfach der Dateiname der einzufügenden Datei angehängt. Die (optionale) explizite Beschränkung der Einfügeaktion auf ein spezifisches Ausgabeformat wird ebenfalls unterstützt, so dass auch dieses Kommando gültig ist:

```
%!include(html): Datei.t2t
```

Beachten sie, dass über die include−Anweisung der Textbereich einer Datei in das Quelldokument eingefügt wird. Sowohl Kopf− als auch Konfigurationsbereich der einzufügenden Datei werden ignoriert. Dadurch können sie die eingefügte Datei sowohl alleinstehend als auch im Gesamtzusammenhang des Hauptdokuments konvertieren.

Zusätzlich gibt es noch drei weitere Arten, Dokumente zu integrieren:

- Einfügung als Originaltext
- Einfügung als Rohtext
- Einfügung als Auszeichnungen enthaltender Text

Wird die externe Datei als **Originaltext** eingefügt, so bleibt die Originalformatierung (incl. Abstände der Zeichen untereinander) erhalten, genauso, wie wenn der Text innerhalb von txt2tags als Originaltextbereich ausgezeichnet worden wäre (mittels ```). Möchten sie den Text auf diese Art und Weise einfügen, umschließen sie die externe Textdatei mit zwei einfachen, rückwärts geneigten Hochzeichen:

```
%!include: ``/etc/fstab``
```

Wird die externe Datei als **Rohtext** eingefügt, so wird die Textdatei gleichsam unbesehen eingefügt, das heißt, der Text wird nicht auf spezifische txt2tags−Auszeichnungen untersucht (geparst), genauso, wie wenn der Text innerhalb von txt2tags als Rohtextbereich ausgezeichnet worden wäre (mittels """). Möchten sie den Text auf diese Art und Weise einfügen, umschließen sie die externe Textdatei mit zwei geraden hochgestellten Anführungszeichen:

```
%!include: ""Klasse_Text.txt""
```

Und schließlich gibt es noch die Möglichkeit Text als **Auszeichnungen enthaltenden Text** einzufügen, der Inhalt des einzufügenden Dokuments wird dabei direkt in das resultierende Dokument durchgereicht, dabei wird der Text weder auf spezifische txt2tags−Auszeichnungen hin untersucht (geparst) noch werden irgendwelche Ersetzungen vorgenommen. Somit können zusätzliche in txt2tags oder anderweitiger Syntax ausgezeichnete Textteile zu ihrem Dokument hinzugefügt werden. Als nützlich kann sich das etwa erweisen wenn sie Informationen im Kopf−oder Fußbereich hinzufügen wollen oder falls sie Text mit einer komplizierten Syntax einfügen wollen, welche nicht von txt2tags unterstützt wird:

```
%!include(html): ''Fussbereich.html''
```

Beachten Sie, dass der Dateiname mit zwei einfachen geraden Hochzeichen umschlossen ist. Da der Text bereits geparst ist, müssen sie explizit das Zielformat angeben, um Fehler zu vermeiden (hier: html).

Das %!includeconf Kommando

Das `includeconf`−Kommando dient dazu, Konfigurationen, welche in einer externen Datei definiert wurden, in das aktuelle Dokument mit aufzunehmen. Dieses Kommando ist nur innerhalb des Konfigurationsbereichs des Quelldokuments gültig.

Dieses Kommando ist etwa dann nützlich, wenn dieselben Konfigurationseinstellungen von mehreren Dateien geteilt werden sollen, sie können die Konfigurationsoptionen gleichsam zentralisieren. Bei jeder Datei, innerhalb welcher diese zentral abgelegten Konfigurationseinstellungen Anwendung finden sollen, ist dann innerhalb des Konfigurationsbereichs `includeconf` aufzurufen. Ein Beispiel:

```
Mein erstes Dokument
Andreas Deininger
September, 2005

%!includeconf: config.t2t

Hallo, dies ist mein erstes Dokument.
```

Das Format für Einstellungen innerhalb der externen Konfigurationsdatei ist dasselbe wie dasjenige das bei benutzerdefinierten Konfigurationsdateien Anwendung findet.

Teil V − Auszeichnungen beherrschen lernen

Übersicht über alle txt2tags Auszeichnungen:

Grundlegende Auszeichnungen	**. . . . . .**	**Schriftstile**	**. . . . . .**
Kopfbereich	Erste 3 Zeilen	*Fett*	**Wörter**
Titel	= Wörter =	*Kursiv*	//Wörter//
Nummerierte Titel	+ Wörter +	*Unterstrichen*	__Wörter__
Abschnitt	Wörter	*Schreibmaschine*	``Wörter``
Text Blocks	**. . . . . .**	**Andere**	**. . . . . .**
Zitat	<TAB>Wörter	*Trennlinie*	−−−−−−−−−−...
Aufzählung	− Wörter	*Dicke Linie*	==========...
Nummerierte Aufzählung	+ Wörter	*Verknüpfungen*	[Angabe URL]
Definitionsliste	: Wörter	*Bild*	[Bilddatei.jpg]
Originaltextzeile	``` Wörter	*Kommentar*	% Kommentare
Originaltextbereich	```\n Zeilen \n```	*Rohtext*	""Wörter""
Rohtextzeile	""" Wörter	*Tabelle*	| Zelle 1 | Zelle 2 | Zelle 3...
Rohtextbereich	"""\n Zeilen \n"""	*Anker*	= Marke =[Anker]

Allgemeine Regeln:

- Die ersten drei Zeilen eines Dokuments zählen als **Kopfzeilen**, etwaige Auszeichnungen werden nicht ausgewertet.
- Der Text einer **Titelzeile** wird von "=" oder "+"−Zeichen umschlossen. Je mehr Zeichen, umso tiefer wird der Titel eingruppiert.
- Bei **Schriftstilen** sind keine Leerzeichen zwischen den Markierungszeichen und dem Inhalt erlaubt.
- Das **Kommentarzeichen** "%" muss ganz am Anfang der Zeile (in deren erster Spalte) stehen.
- Die Dateinamen von **Bildern** müssen die Endung GIF, JPG, PNG (oder ähnlich) aufweisen.
- Die einzigen **mehrere Zeilen** umfassenden **Markierungen** sind Originaltext− und Rohtextbereiche.
- Auszeichnungen innerhalb von Originaltext− oder Rohtextbereichen werden **nicht** interpretiert.
- **(Dicke) Trennlininen** müssen mindestens 20 Zeichen umfassen.
- Die **Nummerierungstiefe** von Zitaten und Aufzählungen wird über deren Einrückung definiert.
- Die **Titelzeile einer Tabelle** wird über zwei || am Beginn der Zeile gekennzeichnet.

Kopfzeilen

- **Beschreibung:** Zeichnet den Kopfbereich des Dokuments aus
- **Eigenschaften:** Mehrzeilig, Leerzeichen, !Ausrichtung, !Verschachtelt
- **Enthält:** Makros
- **Syntax:**
    - Die ersten 3 Zeilen der Quelldatei.
    - Falls sie gar keinen Kopfbereich angeben möchten, lassen sie die erste Zeile des Quelldokuments leer. Passend für Einzeiler auf der Komandozeile oder bei selbstdefiniertem Kopfbereich.

    - Lassen sie die zweite und dritte Zeile des Quelldokuments leer, falls sie diese Teile des Kopfbereichs weglassen möchten.
- **Details:**
    - Markierungen werden NICHT ausgewertet
    - Die ersten drei Zeilen des Quelldokuments sind auch die ersten drei Zeilen im Zieldokument, wobei diese Zeichenketten stark vom sich anschließenden Textkörper abgehoben werden und ggf. sogar alleine auf einer separaten Titelseite platziert werden (falls das Zielformat dies zulässt).
    - Generell können beliebige Inhalte für den Kopfbereich gewählt werden, auf den einzelnen Zeilen wird wird keine spezifische, statisch festgelegte Information benötigt. Für die meisten Dokumente wird sich dennoch folgendes anbieten:
        - Zeile 1: Titel des Dokuments
        - Zeile 2: Name des Autor und/oder dessen E−Mail Adresse
        - Zeile 3: Erstellungsdatum und/oder Version des Dokuments (der richtige Platz für %%mtime)

Titelzeilen, Nummerierte Titelzeilen

- **Beschreibung:** Zeichnet Abschnittstitel aus (nummeriert oder nicht nummeriert)
- **Eigenschaften:** Mehrzeilig, Leerzeichen, !Ausrichtung, !Verschachtelt
- **Enthält:** Rohtext
- **Syntax:**
    - Hinweis: Falls die Abschnittstitel nummeriert werden sollen, ersetzen sie bei all den im folgenden aufgeführten Regeln das Zeichen "=" durch "+"
    - Jeweils gleich viele Gleichheitszeichen vor und hinter dem Abschnittstitel, `=wie hier in Abschnittstitel I=`
    - Je mehr Zeichen angeben werden, umso tiefer wird der Abschnitt eingeordnet: `=Titel=`, `==Untertitel==`, `===Titel unterhalb eines Untertitels===`, ...
    - Bei der Untergliederung sind maximal 5 Ebenen möglich, `=====wie hier=====`
    - Sind vor und hinter dem Titel nicht gleich viele "=" oder "+"−Zeichen angegeben, handelt es sich nicht um einen Abschnittstitel, `=wie hier===`
    - Innerhalb einer Auszeichnung sind beliebig viele Leerzeichen erlaubt, `= wie hier =`
    - Abschnittstiteln kann ein Textanker hinzugefügt werden `=wie hier=[Anker]`. Um auf einen solchen Anker zu verweisen, definieren sie eine `[lokale Verknüpfung #Anker]`
    - Der Name des Ankers darf ausschließlich Buchstaben und Zahlen sowie die Zeichen "−" und "_" enthalten (A−Za−z0−9_−). Sonderzeichen im Namen des Ankers sind **nicht** erlaubt.
- **Details:**
    - Markierungen werden NICHT ausgewertet
    - Makros werden NICHT ausgewertet

Abschnitt

- **Beschreibung:** Zeichnet einen Textabschnitt aus
- **Eigenschaften:** Mehrzeilig, Leerzeichen, !Ausrichtung, !Verschachtelt
- **Enthält:** Makros, Schriftstile, Rohtext, Verknüpfungen, Bilder, Kommentare
- **Syntax:**
    - Abschnitte sind Gruppen von Zeilen, die durch Leerzeilen voneinander getrennt sind
    - Ein Abschnitt kann auch durch andere Blockauszeichnungen, wie durch das Ende einer Aufzählung, eines Zitates, einer Tabelle oder eines Originaltextbereichs beendet werden

Kommentar

- **Beschreibung:** Wird zur Auszeichnung von Text benutzt, der nicht im Zieldokument erscheinen soll
- **Eigenschaften:** Mehrzeilig, !Leerzeichen, !Ausrichtung, !Verschachtelt
- **Enthält:** −
- **Syntax:**
  - Eine Kommentarzeile beginnt mit einem Prozentzeichen in der ersten Spalte, `% wie hier`
  - KEINE vorangehenden Leerzeichen erlaubt
- **Details:**
  - Als Kommentar ausgezeichneter Text taucht im umgewandelten Text nicht auf.
  - Keine Auszeichnung als Block möglich, jede einzelne Kommentarzeile muss mit einem %−Zeichen beginnen
  - Nützlich zur Erinnerung an noch anstehende Aufgaben (TODO), zum Markieren von Fehlern oder für Anmerkungen, etwa von einem Herausgeber

Fett, Kursiv, Unterstrichen

- **Beschreibung:** Wird genutzt um fetten/kursiven/unterstrichenen Text innerhalb eines Abschnitts, einer Tabelle, einer Aufzählung oder eines Zitats einzufügen
- **Eigenschaften:** Mehrzeilig, !Leerzeichen, !Ausrichtung, Verschachtelt
- **Enthält:** Makros, Schriftstile, Rohtext, Verknüpfungen, Bilder
- **Syntax:**
  - Zwei Sternchen umschließen fetten Text, `**wie hier**`
  - Zwei Schrägstriche umschließen kursiven Text, `//wie hier//`
  - Zwei Unterstriche umschließen unterstrichenen Text, `__wie hier__`
  - Zwischen den Auszeichnungen und den auszuzeichnenden Inhalten sind keine Leerzeichen erlaubt: `** so etwas ** ist ungültig`
- **Details:**
  - innerhalb einer mit einem Schriftstil ausgezeichneten Textphrase sind keine Zeilenumbrüche erlaubt, die Textpassage ist auf einer einzelnen Zeile der Quelldatei zu platzieren
  - Makros innerhalb einer Schriftstil−Auszeichnung sind erlaubt: `**%%date**`
  - Schriftstile können ineinander verschachtelt werden: `**__so__ //etwa//**`

Schreibmaschinenschrift

- **Beschreibung:** Fügt Text in Schreibmaschinenschrift innerhalb eines Abschnittes, einer Tabelle, einer Aufzählung oder eines Zitats ein
- **Eigenschaften:** Mehrzeilig, !Leerzeichen, !Ausrichtung, !Verschachtelt
- **Enthält:** −
- **Syntax:**
  - von zwei Hochzeichen umschlossen, ``` ``wie hier zu sehen`` ```
  - Die Hochzeichen müssen direkt an den Inhalt anschließen (keine Leerzeichen erlaubt!): ``` `` So etwas `` ist ungültig ```
- **Details:**
  - Markierungen werden NICHT ausgewertet
  - Makros werden NICHT ausgewertet
  - Der gesamte Text, der in Schreibmaschinenschrift gesetzt werden soll, muss sich auf einer einzigen Zeile befinden, es sind keine Zeilenumbrüche innerhalb der Textphrase erlaubt
  - In manchen Zielformaten bleiben die Abstände zwischen den Zeichen erhalten, in anderen werden aufeinanderfolgende Leerzeichen zu einem einzigen verschmolzen

- Sie können den Text in Schreibmaschinenschrift auch fett auszeichnen, indem sie ihn in die entsprechenden Zeichen einschließen: **``Schreibmaschine fett ``**. Dies ist genauso mit den anderen Schriftstilen möglich: etwa //``kursiv``// und __``unterstrichen``__.

Originaltext−Zeile, Originaltext−Bereich

- **Beschreibung:** Damit kann Programm−Code oder andersweitig formatierter Text eingefügt werden, hierbei bleiben die Wortabstände und die Zeilenumbrüche erhalten, die Ausgabe erfolgt in Schreibmaschinenschrift
- **Eigenschaften:** Mehrzeilig, !Leerzeichen, !Ausrichtung, !Verschachtelt
- **Enthält:** −
- **Syntax: Originaltext−Zeile:**
  - Eine Originaltext−Zeile beginnt mit drei aufeinanderfolgenden einfachen Anführungszeichen, gefolgt von einem Leerzeichen, an welches sich der Originaltext anschließt, ``` etwa so
  - Die einfachen Anführungszeichen müssen sich direkt am Zeilenanfang befinden, es sind keine vorangehenden Leerzeichen erlaubt
- **Syntax: Originaltext−Bereich:**
  - Eine Zeile mit genau drei aufeinanderfolgenden einfachen Anführungszeichen ```, gefolgt von Textzeilen, gefolgt von einer weiteren Zeile genau drei aufeinanderfolgenden einfachen Anführungszeichen ```
  - Vor und nach den Auszeichnungsmarken sind KEINE Leerzeichen erlaubt.
- **Details:**
  - Markierungen werden NICHT ausgewertet
  - Makros werden NICHT ausgewertet
  - Falls das Ende der Quelldatei (EOF) erreicht wird, wird ein ggf. geöffneter Originaltext−Bereich geschlossen.

Trennlinien, Dicke Trennlinien

- **Beschreibung:** Zeichnet eine Trennlinie oder eine dicke Trennlinie aus
- **Eigenschaften:** !Mehrzeilig, Leerzeichen, !Ausrichtung, !Verschachtelt
- **Enthält:** −
- **Syntax:**
  - Die Auszeichnung einer Trennlinie kann entweder mittels Strichen "−" oder Unterstrichen "_" vorgenommen werden
  - Für dicke Trennlinien wird das Gleichheitszeichen "=" verwendet
  - Für eine Linie sind mindestens 20 Striche/Unterstriche/Gleichheitszeichen erforderlich
  - Am Anfang und am Ende der Linie können Leerzeichen platziert werden
  - Jegliche andere Zeichen innerhalb der Zeile heben die Auszeichnung als Linie auf
- **Details:**
  - Falls das Zielformat keine Trennlinien unterstützt, wird stattdessen eine Kommentarzeile ausgegeben
  - Eine dicke Trennlinie führt bei unterschiedlichen Zielformaten zu unterschiedlichem Verhalten:
    - ◊ Eine dickere Trennlinie
    - ◊ Eine Pause bei Präsentationsformaten, wie etwa bei Magic Point
    - ◊ Ein Seitenumbruch in seitenorientierten Ausgabeformaten, wie etwa bei LaTeX

Verknüpfungen, Benannte Verknüpfungen

- **Beschreibung:** Dient zur Auszeichnung eines entfernten (Internet) Links oder einer lokalen Verknüpfung
- **Eigenschaften:** Mehrzeilig, !Leerzeichen, !Ausrichtung, !Verschachtelt
- **Enthält:** Makros, Rohtext, Bilder
- **Syntax:**
  - Jede gültige Internet URL, FTP−, News− oder E−Mail Adresse wird erkannt und automatisch umgewandelt.
  - Die Angabe des Protokolls (http, https, ftp) ist optional, `www.wiedies.com`
  - Für eine Verknüpfung kann ein Name angeben werden: `[Klicken sie hier www.url.com]`
  - Auch ein Bild kann auf eine Verknüpfung verweisen: `[[Bild.jpg] www.url.com]`
  - Makros innerhalb der Verknüpfungsadresse sind erlaubt: `[siehe Quelldatei %%infile]`
  - Makros innerhalb des Verknüpfungsnamens sind erlaubt: `[Spiegelung von %%outfile www.url.com]`
  - Alle Spezifikationen für einen einzelnen Link innerhalb der Quelldatei müssen sich auf einer Zeile befinden, Zeilenumbrüche sind hier nicht erlaubt
- **Details:**
  - Falls das Zielformat keine Verknüpfungen unterstützt, erfolgt die Auszeichnung mittels Unterstreichung

Zitate

- **Beschreibung:** Dient zur Auszeichnung eines Zitats (als eingerückte Zeile)
- **Eigenschaften:** Mehrzeilig, !Leerzeichen, !Ausrichtung, Verschachtelt
- **Enthält:** Makros, Schriftstile, Zitate, Rohtext, Trennlinien, Verknüpfungen, Bilder, Kommentare
- **Syntax:**
  - Eine Zeile, die mit einem Tabulatorzeichen (TAB) beginnt
  - Mehrere Tabulatorzeichen am Zeilenanfang erhöhen die Tiefe der Einrückung für das Zitat
  - Aufzählungen und Tabellen innerhalb von Zitaten sind nicht erlaubt
- **Details:**
  - Falls das Ende der Quelldatei (EOF) erreicht wird, wird ein ggf. geöffnetes Zitat abgeschlossen.
  - Manche Zielformate unterstützen die Verschachtelung von Zitaten nicht, in diesem Fall werden die untergeordneten Zitate entsprechend auf die Ebene verschoben, auf der sich das äußerste Zitat befindet.
  - Für die Tiefe der Verschachtelung von Zitaten gibt es kein Limit. Allerdings ist bei manchen Zielformaten die Verschachtelungstiefe beschränkt, Zitate unterhalb der tiefstmöglichen Ebene werden auf diese Ebene hochgestuft.

Aufzählungen, Nummerierte Aufzählungen, Definitionslisten

- **Beschreibung:** Dient dazu, den Beginn eines Aufzählungspunkts innerhalb einer Aufzählung auszuzeichnen
- **Eigenschaften:** Mehrzeilig, !Leerzeichen, !Ausrichtung, Verschachtelt
- **Enthält:** Makros, Schriftstile, Aufzählungen, Tabellen, Originaltext, Rohtext, Trennlinien, Verknüpfungen, Bilder, Kommentare
- **Syntax:**
  - Eine Zeile, welche mit einem Strich/Pluszeichen/Punkt beginnt, gefolgt von genau einem Leerzeichen
  - Das erste Zeichen einer Aufzählung kann KEIN Leerzeichen sein (Ausnahme:

Definitionslisten)
    - Über ggf. vorhandene Leerzeichen am Zeilenanfang (gewöhnliche Leerzeichen, keine Tabulatorzeichen) wird die Nummerierungstiefe der Aufzählung festgelegt (Verschachtelung)
    - Tiefer eingruppierte Aufzählungen werden beendet, falls ein höher angeordneter Aufzählungspunkt (von der Elternaufzählung aus gesehen) oder ein leerer Aufzählungspunkt folgt.
    - Mittels zweier aufeinanderfolgender Leerzeilen können alle geöffneten Aufzählungen geschlossen werden
- **Details:**
    - Auch falls das Ende der Quelldatei (EOF) erreicht wird, werden alle ggf. geöffneten Aufzählungen abgeschlossen.
    - Aufzählungen können auch gemischt werden, so ist etwa eine Definitionsliste innerhalb einer nummerierten Aufzählung problemlos möglich.
    - Manche Zielformate unterstützen die Verschachtelung von Aufzählungen nicht, in diesem Fall werden alle Unterpunkte der Aufzählung auf die erste (oberste) Aufzählungsebene hochgestuft.
    - Grundsätzlich besteht keine Beschränkung hinsichtlich der maximal möglichen Tiefe von Aufzählungen. Bei einzelnen Zielformaten können jedoch durchaus Beschränkungen vorliegen, in diesem Fall werden diejenigen Aufzählungen, welche tiefer als die maximal zulässige Tiefe ausgezeichnet sind, auf das tiefstmögliche Niveau hochgestuft.

Bilder

- **Beschreibung:** Markiert ein Bild
- **Eigenschaften:** Mehrzeilig, !Leerzeichen, Ausrichtung, !Verschachtelt
- **Enthält:** Makros
- **Syntax:**
    - Der Name der Bilddatei, von eckigen Klammern umgeben, `[meinbild.jpg]`
    - Dem Namen der Bilddatei muss als Erweiterung das Bildformat angehängt sein, etwa PNG, JPG, GIF, ... (sowohl Groß− als auch Kleinschreibung erlaubt)
    - Symbole im Namen der Bilddatei sind erlaubt, `[meinbild!~1.jpg]`
    - Makros im Namen der Bilddatei sind erlaubt, `[report−%%date(%Y−%m−%d).png]`
    - Leerzeichen im Namen der Bilddatei sind NICHT erlaubt, `[wie hier.jpg]`
    - Leerzeichen nach der öffnenden oder vor der schließenden Klammer sind NICHT erlaubt, `[ meinbild.jpg ]`
- **Details:**
    - Falls im Zielformat keine Bilder unterstützt werden, wird der Name der Bilddatei innerhalb (runder Klammern) wiedergegeben.
    - Über die Position der Markierung des Bildes auf der Zeile wird die Ausrichtung des Bildes definiert:
        - ◊ `[LINKS.jpg]` blablablabla
        - ◊ blablablabla `[ZENTRIERT.jpg]` blablablabla
        - ◊ blablablabla `[RECHTS.jpg]`

Tabellen

- **Beschreibung:** Dient zur Auszeichnung einer Tabellenzeile mit einer beliebigen Anzahl von Spalten
- **Eigenschaften:** Mehrzeilig, Leerzeichen, Ausrichtung, !Verschachtelt
- **Enthält:** Makros, Schriftstile, Rohtext, Verknüpfungen, Bilder, Kommentare
- **Syntax:**
    - Ein vorangestelltes "|"−Zeichen zeichnet eine Tabellenzeile aus
    - Zwei vorangestellte "||"−Zeichen zeichnen eine Zeile als Titelzeile einer Tabelle aus

- Dem ersten "|"−Zeichen vorangestellte Leerzeichen führen zu einer zentrierten Anordnung der Tabelle innerhalb des Textflusses
- Einzelne Zellen der Tabelle werden durch die Zeichenkette " | " (Leerzeichen "|"−Zeichen Leerzeichen) voneinander getrennt
- Wird am Ende der ersten Tabellenzeile das "|"−Zeichen angegeben, so wird für die Tabelle eine Umrandung ausgeben
- Abschließende "|"−Zeichen am Ende aller folgenden Tabellenzeilen werden ignoriert und dienen damit lediglich der besser optischen Darstellung der Tabelle
- Wird eine Tabellenzelle mit mehr als einem "|"−Zeichen abgeschlossen, so erstreckt sich diese Zelle über mehrere Spalten: "||" für 2 Spalten, "|||" für 3 Spalten (und so weiter)
- Die Anordnung der Leerzeichen innerhalb einer Tabellenzelle bestimmt deren horizontale Ausrichtung
- Beispiel: `| Zeile | einer | Tabelle | mit | sechs | Spalten |`

- **Details:**
  - Alle zu einer Zeile der Tabelle gehörigen Daten müssen auf einer einzigen Zeile der Quelldatei angeordnet sein, innerhalb einer Tabellenzeile sind keine Zeilenumbrüche erlaubt
  - Zielformate mit spaltenorientierter Ausrichtung (wie SGML und LaTeX) nutzen die Ausrichtung der ersten Tabellenzeile als Voreinstellung für alle sich anschließenden Tabellenzeilen
  - Jede nicht als Tabellenzeile ausgezeichnete Textzeile schließt eine geöffnete Tabelle ab, die einzige Ausnahme hiervor stellen Kommentarzeilen dar
  - Die Anzahl der Zellen ist variabel, verschiedene Zeilen einer Tabelle können eine unterschiedliche Anzahl an Zellen aufweisen
  - Derzeit gibt es keine Möglichkeit, innerhalb einzelner Spalten mehrere Zeilen zu einer einzigen Tabellenzelle zusammenzufassen.
  - Falls das Zielformat die Darstellung von Tabellen nicht unterstützt, werden die als Tabelle ausgezeichneten Zeilen als ein Originaltextbereich behandelt

Rohtext, Rohtextzeilen, Rohtextbereich

- **Beschreibung:** Wird genutzt um beliebigen Text vor dem Parsen zu "schützen", Auszeichnungen und Makros innerhalb des Rohtextes werden nicht expandiert.
- **Eigenschaften:** Mehrzeilig, !Leerzeichen, !Ausrichtung, !Verschachtelt
- **Enthält:** −
- **Syntax: Rohtext:**
  - Text umgeben von zwei Anführungszeichen, `""wie hier""`
  - Anführungszeichen und Text direkt aneinander (keine Leerzeichen)
- **Syntax: Rohtextzeile:**
  - Eine Zeile mit rohem Text beginnt mit drei aufeinanderfolgenden Anführungszeichen, `""" wie hier`
  - Die Anführungszeichen müssen sich unmittelbar am Zeilenanfang befinden, Leerzeichen davor sind nicht erlaubt
  - Ein Leerzeichen nach den Anführungszeichen trennt diese vom nachfolgenden Text ab
- **Syntax: Rohtextbereich:**
  - Ein Bereich mit rohem Text beginnt mit einer Zeile mit drei aufeinanderfolgenden Anführungszeichen, daran können sich beliebig viele weitere Textzeilen anschließen. Abgeschlossen wird der Bereich durch eine Zeile mit drei aufeinanderfolgenden Anführungszeichen
  - Leerzeichen vor oder nach den Markierungen sind NICHT erlaubt
- **Details:**
  - Markierungen werden NICHT ausgewertet
  - Makros werden NICHT ausgewertet
  - Falls das Ende der Quelldatei (EOF) erreicht wird, wird ein ggf. geöffneter

Rohtextbereich abgeschlossen.

Teil VI − Makros beherrschen lernen

Makros sind für spezielle Zwecke definierte Schlüsselwörter, die während der Konvertierung expandiert werden. Sie werden benutzt, um dynamische Informationen, wie etwas das aktuelle Datum oder Informationen über das Quelldokument in dieses einzufügen.

Ein Makro wird über die beiden Zeichen %% ausgezeichnet denen der Name des Makros folgt, ein Beispiel ist `%%date`. Einige Makros erkennen optionale Formatierungsanweisungen, welche innerhalb von Klammern unmittelbar nach dem Makronamen wiedergegeben werden, etwa in der Form `%%date(%Y−%m−%d)`. In solch einer Zeichenkette zur Formatierung kann herkömmlicher Text mit speziellen Formatierungsanweisungen gemischt werden, letztere werden über ein Prozentzeichen eingeleitet, dem ein einzelnes Zeichen als eigentliche Formatierungsanweisung folgt. Falls keine zusätzliche Formatierungsanweisung angeben wird, wird eine sinnvolle, vorgegebene Einstellung genutzt.

Makroname	Wird expandiert nach ...	Voreinstellung
%%date	Das aktuelle Datum	%Y%m%d
%%mtime	Der Zeitpunkt der letzten Änderung der Quelldatei	%Y%m%d
%%infile	Der Pfad des Quelldokuments	%f
%%outfile	Der Pfad der bei der Konvertierung erzeugten Datei	%f
%%toc	Das Inhaltsverzeichnis des Dokuments (Table of Contents)	−

Generelle Regeln:

- Beim Namen des Makros wird nicht zwischen Groß− und Kleinschreibung unterschieden, insofern sind `%%date`, `%%DaTe` und `%%DATE` identisch
- Makros sind sowohl im Kopfbereich als auch im Textbereich des Dokuments gültig. Eine Ausnahme stellt das Makro %%toc dar, welches lediglich im Textbereich erlaubt ist
- Wird im Konfigurationsbereich ein Makro angegeben, wird über dieses Makro automatisch der Textbereich eingeleitet
- Ein Makro kann an jeder beliebigen Stelle innerhalb einer Zeile platziert werden, und es sind beliebig viele Makros innerhalb einer Zeile erlaubt (eine Ausnahme stellt %%toc dar, das nur dann gültig ist, wenn es separat in einer Zeile angegeben wird)
- Makros können auch innerhalb der Auszeichnungen für Verknüpfungen und Bilder benutzt werden (außer %%toc)
- Innerhalb von Titeln, von Originaltext− und Rohtextbereichen auftretende Makros werden nicht expandiert

Komplettes Beispiel (expandierte Makros sind in Fettschrift wiedergegeben):

> Das ist das Benutzerhandbuch zu txt2tags, konvertiert ins Zielformat **html** mit Hilfe von txt2tags aus der Quelldatei **userguide−de.t2t**. Die Umwandlung wurde durchgeführt am **07. March 2006 um 10:27:22**, wobei die letzte Änderung am Quelldokument am **05. September 2005 um 12:08:55** vorgenommen wurde. Sowohl die Quelldatei als auch die erzeugte Datei befinden sich im Verzeichnis **userguide−de**.

%%date

Das `%%date` Makro wird zum aktuellen Datum und der aktuellen Uhrzeit expandiert. Es ist sehr nützlich um im Kopf− oder Fußbereich das Datum wiederzugeben, an dem das Dokument erstellt wurde. Um den Zeitpunkt der letzten Änderung des Quelldokuments wiederzugeben, benutzen sie bitte das %%mtime Makro.

Das %%date−Makro akzeptiert etliche Formatierungsanweisungen. Die komplette Liste kann auf der Python Webseite abgerufen werden. Die gebräuchlichsten Anweisungen sind nachfolgend aufgeführt:

Anweisung	Beschreibung
%a	Name des Wochentags (Kurzform), gemäß der aktuellen Lokale.
%A	Name des Wochentags (Langform), gemäß der aktuellen Lokale.
%b	Name des Monats (Kurzform), gemäß der aktuellen Lokale.
%B	Name des Monats (Langform), gemäß der aktuellen Lokale.
%c	Repräsentation von Datum und Uhrzeit gemäß der aktuellen Lokale.
%d	Tag des Monats angegeben als Dezimalzahl [01,31].
%H	Stunde, angegeben als Dezimalzahl (24−Stunden Anzeige) [00,23].
%I	Stunde, angegeben als Dezimalzahl (12−Stunden Anzeige) [01,12].
%m	Monat angegeben als Dezimalzahl [01,12].
%M	Minute angegeben als Dezimalzahl [00,59].
%p	Das Äquivalent zu AM bzw. PM, gemäß der aktuellen Lokale.
%S	Sekunde angegeben als Dezimalzahl [00,61]. (1)
%x	Repräsentation des Datums gemäß der aktuellen Lokale.
%X	Repräsentation der Uhrzeit gemäß der aktuellen Lokale.
%y	Jahresangabe als Dezimalzahl (zweistellig, ohne Angabe des Jahrhunderts).
%Y	Jahresangabe als Dezimalzahl (vierstellig, einschließlich Angabe des Jahrhunderts).
%%	Das "%"−Zeichen selbst.

Beispiele:

Makro	−−>	Resultate für den Zeitpunkt 07. March 2006 um 10:27
%%date(Konvertiert am: %c)	−−>	Konvertiert am: Tue Mar 7 10:27:22 2006
%%date(%d. %B %Y)	−−>	07. March 2006
%%date(%I:%M %p)	−−>	10:27 AM
%%date(Dieser Wochentag ist ein %A im Monat %B.)	−−>	Dieser Wochentag ist ein Tuesday im Monat March.

%%mtime

Das `%%mtime` Makro wird zum Zeitpunkt der letzten Änderung des Quelldokuments expandiert. Dieses Makro ist hilfreich um anzugeben, wann die Datei das letzte Mal geändert wurde. Dieses Makro ist eine "Schwester" des %%date−Makros und kann daher mit genau denselben Formatierungsanweisungen wie dieses aufgerufen werden.

Hier ein Beispiel: das Quelldokument für dieses Benutzerhandbuch wurde zuletzt geändert am **Mon Sep 5 12:08:55 2005**. Diese Datumsangabe wurde mittels des Makros `%%mtime(%c)` erzeugt.

%%infile

Das `%%infile` Makro wird expandiert zur Angabe des Speicherorts der Quelldatei im Dateisystem. Es ist sehr nützlich, um auf HTML−Seiten Verknüpfungen der Art "*siehe die Quelle dieser Datei*" zu erzeugen. Wenn sie solche Verknüpfungen erzeugen, können sie damit Anfänger unterstützen und ihnen Hilfestellung leisten, da diese dann ihren Quellcode als Vorlage für ihre eigene Seite verwenden können.

Dieses Makro akzeptiert die folgenden Formatierungsanweisungen:

%<Zeichen>	Beschreibung	Ausgabe für den Quelltext dieses Benutzerhandbuchs
%f	Dateiname	userguide−de.t2t
%F	Dateiname (ohne Erweiterung)	userguide−de
%e	Dateinamenerweiterung	t2t
%p	Absoluter Dateipfad	/a/txt2tags/src/doc/German/userguide−de/userguide−de.t2t
%d	Pfad der Datei (nur Verzeichnisse)	/a/txt2tags/src/doc/German/userguide−de
%D	Pfad der Datei (nur das Elternverzeichnis)	userguide−de
%%	Das "%"−Zeichen selbst	%

Beispiele:

Quelltext	−−>	Wird expandiert nach
Das Elternverzeichnis dieses Handbuchs ist %%infile(%D).	−−>	Das Elternverzeichnis dieses Handbuchs ist userguide−de.
Ich benutzte die Dateinamenerweiterung %%infile(%e).	−−>	Ich benutzte die Dateinamenerweiterung t2t.
[Siehe Quelldatei %%infile]	−−>	Siehe Quelldatei
Nach XHTML konvertiert trage ich den Namen %%infile(%F).xhtml	−−>	Nach XHTML konvertiert trage ich den Namen userguide−de.xhtml

Hinweis: Dieses Makro wird zur Zeichenkette "−" expandiert, falls von der Standardeingabe (STDIN) eingelesen wird.

%%outfile

Das `%%infile` Makro wird expandiert zur Angabe des Speicherorts der Zieldatei im Dateisystem. Es ist sehr nützlich, um den Dateinamen innerhalb des Kopf− oder Textbereichs des Dokuments wiederzugeben. Dieses Makro ist eine "Schwester" des %%infile−Makros und kann daher mit genau denselben Formatierungsanweisungen wie dieses aufgerufen werden.

Beispiele:

Quelltext	−−>	Wird expandiert nach
Sie lesen gerade die Datei %%outfile.	−−>	Sie lesen gerade die Datei userguide−pdf.html.
txt2tags −t %%outfile(%e) −i %%infile −o	−−>	txt2tags −t html −i userguide−de.t2t −o

%%outfile		userguide−pdf.html

Hinweis: Dieses Makro wird zur Zeichenkette "−" expandiert, falls auf die Standardausgabe (STDOUT) ausgegeben wird.

%%toc

Das `%%toc` Makro wird zu einem Inhaltsverzeichnis für das Dokument expandiert. Es ist sehr nützlich, um die genaue Stelle anzugeben, an der das Inhaltsverzeichnis wiedergegeben werden soll. Sie können dieses Makro auch mehrmals innerhalb eines Dokuments nutzen und dabei zum Beispiel das Inhaltsverzeichnis ein zweites Mal am Ende des Dokuments wiedergeben. Dieses Benutzerhandbuch nutzt das Makro %%toc um das Inhaltsverzeichnis wie gewünscht zu positionieren.

Im Unterschied zu allen anderen Makros erwartet das %%toc−Makro keine Formatierungsanweisungen und folgt auch sonst anderen Regeln:

- Es ist lediglich im Textbereich des Dokuments gültig
- Es muss alleinstehend auf einer einzelnen Zeile verwendet werden (dabei sind vorangehende und sich anschließende Leerzeichen erlaubt)
- Es erlangt nur Wirksamkeit in Verbindung mit der Kommandozeilenoption −−toc, wird diese nicht angeben, wird das Makro ignoriert
- Wird innerhalb des Dokuments das Makro %%toc angetroffen, wird die voreingestellte automatische Positionierung/Formatierung des Inhaltsverzeichnisses deaktiviert.

Teil VII − Einstellungen beherrschen lernen

Einstellungen sind spezielle, im Konfigurationsbereich des Quelldokuments aufgeführte Konfigurationsanweisungen, welche den Umwandlungsprozess beeinflussen können. Die Angabe solcher Einstellungen ist nicht zwingend erforderlich. Der gewöhnliche Benutzer kann sehr gut auch ohne sie auskommen. Aber Achtung: ihr Gebrauch kann süchtig machen, wenn man einmal angefangen hat, sie zu benutzen, kann man nicht mehr davon ablassen, dies immer wieder zu tun :)

Zeilen, in denen Einstellungen definiert werden sind *spezielle Kommentarzeilen*, welche zusätzlich durch ein vorangestelltes Ausrufezeichen ("!") ausgezeichnet sind, dadurch unterscheiden sie sich von herkömmlichen Kommentarzeilen. Die Syntax einer Einstellungen selbst ist zugleich einfach und doch variabel, sie setzt sich zusammen aus einem Schlüsselwort und einem Wert, wobei beide durch einen Doppelpunkt voneinander getrennt werden (":").

%! Schlüsselwort : Wert

Details der Syntax:

- Das Ausrufezeichen muss dem Kommentarzeichen unmittelbar folgen ("%!"), zwischen den beiden Zeichen ist kein Leerzeichen erlaubt.
- Die Leerzeichen vor und nach dem Schlüsselwort und dem Trennzeichen sind allesamt optional und können auch weggelassen werden.
- Weder bei der Angabe des Schlüsselwortes noch bei Definition des dazugehörigen Werts wird zwischen Groß− und Kleinschreibung unterschieden

Regeln:

- Einstellungen erlangen nur innerhalb des Konfigurationsbereichs Gültigkeit, werden sie dagegen im Textkörper des Dokuments angetroffen, werden sie als gewöhnliche Kommentare angesehen.
- Falls dasselbe Schlüsselwort mehr als einmal innerhalb des Konfigurationsbereichs auftaucht, so wird derjenige Wert benutzt, der als letztes angegeben wurde. Ausnahme: Eine Ausnahme stellen die Einstellungen 'options', 'preproc' und 'postproc' dar, diese sind kumulativ.
- Enthält eine Zeile mit einer Einstellung ein ungültiges Schlüsselwort, so wird die Zeile als gewöhnliche Kommentarzeile angesehen.
- Derart definierten Einstellungen wird Vorrang gegenüber Angaben aus einer benutzerdefinierten Konfigurationsdatei eingeräumt, sie selbst können jedoch mittels Kommandozeilenoptionen außer Kraft gesetzt werden.

%!Target

Über die Einstellung `target` kann das Zielformat für die Konvertierung des Dokuments voreingestellt werden:

```
%!target: html
```

Der Benutzer kann dann einfach den folgenden Befehl aufrufen:

```
$ txt2tags Datei.t2t
```

Dadurch wird die Konvertierung in das innerhalb des Dokuments spezifizierte Zielformat vorgenommen.

Für die Einstellung `target` kann kein optionales Zielformat angegeben werden, dies wäre auch völlig widersinnig, wie das folgende (ungültige!) Beispiel unmittelbar aufzeigt `%!target(tex): html`.

%!Options

Bei jeder Konvertierung eines Dokuments eine lange Anweisung auf der Kommandozeile anzugeben ist stupide und zugleich anfällig für Fehler. Die Einstellung `Options` erlaubt es dem Benutzer daher, alle für die Konvertierung genutzten Optionen zusammen mit dem Quelldokument abzuspeichern. Auf diese Art und Weise wird sichergestellt, dass das Dokument immer auf die gleiche Art und Weise, also immer mit denselben Optionen konvertiert wird.

Sie schreiben dazu die Optionen (ohne Fehler in ihrer Syntax!) genau so, wie sie sie auch an der Kommandozeile angeben würden, lassen aber den "txt2tags" Programmaufruf am Beginn der Anweisung, den Schalter für die Angabe des Zielformats und den Dateinamen für die Quelldatei am Ende der Anweisung einfach weg.

Ein Beispiel: sie haben bislang immer das folgende Kommando benutzt, um ihr Dokument umzuwandeln:

```
$ txt2tags −t html −−toc −−toc−level 2 −−enum−title Datei.t2t
```

Wenn sie jetzt die folgenden Einstellungen innerhalb ihres Quelldokuments platzieren, können sie sich zukünftig die Angabe all dieser Optionen auf der Kommandozeile sparen:

```
%!target: html
%!options(html): −−toc −−toc−level 2 −−enum−title
```

Da die benötigte Kommandozeilenanweisung jetzt nur noch "`txt2tags Datei.t2t`" lautet, brauchen sie jetzt zur Konvertierung den von ihnen bevorzugten Texteditor gar nicht mehr verlassen, sie können jetzt während der Bearbeitung des Quelltexts zwischendurch gelegentlich die Umwandlung anstoßen. In vi lautet die dafür erforderliche Befehlsfolge:

```
:!txt2tags %
```

%!Encoding

Die Einstellung "Encoding" wird bei der Verfassung von Texten in anderen Sprachen als in Englisch benötigt oder immer dann, wenn Umlaute, Buchstaben mit Akzenten oder andere länderspezifische Detailfunktionen genutzt werden sollen. In diesem Fall muss für das Zieldokument der *Zeichensatz* angepasst werden (falls erlaubt).

Die für die Einstellung "Encoding" gültigen Werte tragen denselben Namen wie diejenigen Werte, die als Zeichensatz für ein HTML Dokument gültig sind, wie etwa *iso−8859−1* oder *koi8−r*. Falls sie unschlüssig sind, welche Kodierung sie benutzen sollen, kann ihnen evtl. diese vollständige (und lange!) Liste Hilfestellung bieten.

LaTeX benutzt Aliasnamen für die Kodierung des Dokuments. Dies stellt jedoch für Benutzer von txt2tags kein Problem dar, da txt2tags die Namen intern umsetzt. Einige Beispiele:

txt2tags/HTML	>	LaTeX
windows−1250	>>>	cp1250
windows−1252	>>>	cp1252
ibm850	>>>	cp850
ibm852	>>>	cp852
iso−8859−1	>>>	latin1
iso−8859−2	>>>	latin2

koi8−r	>>>	koi8−r

Falls txt2tags den Wert für die Kodierung nicht kennt, wird er unverändert weitergegeben, insofern kann der Benutzer auch selbstdefinierte Kodierungen angeben und nutzen.

%!PreProc

Bei den PreProc−Einstellungen handelt es sich um einen Eingabefilter, der auf das Quelldokument angewandt wird. Es handelt sich dabei um einen Such− und Ersetzungsvorgang, der durchgeführt wird, direkt nachdem die Zeile von der Quelldatei eingelesen wurde, noch bevor die Datei auf jegliche txt2tags−spezifische Auszeichnungen untersucht wurde.

PreProc−Einstellungen sind nützlich, um Abkürzungen für immer wiederkehrend genutzten Text zu definieren, wie etwa:

```
%!preproc HPM                "Hans−Peter Müller"
%!preproc ERSCHEINUNGS_DATUM "12. August 2005"
%!preproc PÜNKTCHEN          "[Bilder/winzig/blauer_punkt.png]"
```

Der Benutzer kann dann eine Textzeile der folgenden Art und Weise schreiben:

```
Hallo, meine Name ist HPM. Heute ist der ERSCHEINUNGS_DATUM.
```

Und txt2tags wird diese Linie so "zu Gesicht bekommen":

```
Hallo, meine Name ist Hans−Peter Müller. Heute ist der 12. August 2005.
```

Der PreProc−Filter agiert dabei, nachdem der Autor das Dokument verfasst hat, aber noch bevor die eigentliche Konvertierung von txt2tags stattfindet. So gesehen ist es eine erste Konvertierung noch bevor die "richtige" Konvertierung stattfindet. Das Verhalten entspricht dabei demjenigen eines externen Sed− oder Perl−Filter, aufgerufen auf folgende Art und Weise:

```
$ cat Datei.t2t | preproc−script.sh | txt2tags −
```

Das von der txt2tags−Kernkomponente durchgeführte Parsen des Dokuments beginnt erst, nachdem alle mittels PreProc−Einstellungen definierten Ersetzungsvorgänge durchgeführt wurden.

%!PostProc

Bei den PostProc−Einstellungen handelt es sich um einen Ausgabefilter, der auf das bereits konvertierte Dokument angewandt wird. Es handelt sich dabei um einen Such− und Ersetzungsvorgang, der durchgeführt wird, nachdem txt2tags den Text auf spezifische Auszeichnungen hin untersucht und deren Verarbeitung abgeschlossen hat.

PreProc−Einstellungen sind nützlich, um kleinere Verbesserungen oder Korrekturen am erzeugten Dokument vorzunehmen, etwa indem erzeugte Auszeichnungen geändert oder ergänzt werden oder indem zusätzlicher Text hinzugefügt wird. Einige kleine Beispiele:

```
%!postproc(html): '<BODY.*?>' '<BODY BGCOLOR="green">'
%!postproc(tex) : "\\newpage" ""
```

Während über die Filter−Einstellungen in der ersten Zeile die Hintergrundfarbe eines HTML−Dokuments geändert wird, dient der zweite Filter dazu, Seitenumbrüche innerhalb eines LaTeX−Dokuments zu entfernen.

PostProc−Filter arbeiten dabei analog zu externen Sed− oder Perl−Filter, aufgerufen auf folgende Art und Weise:

```
$ txt2tags −t html −o− Datei.t2t | postproc−script.sh > Datei.html
```

Bevor Pre− und PostProc−Filter eingeführt wurden, war es üblich, kleine Skripte anzuwenden, um die Resultate der txt2tags−Konvertierung individuell anzupassen. Diese Skripte waren in der Tat eine Aneinanderreihung von vielen einzelnen sed− (oder sed−ähnlicher) Kommandos, um Aktionen der Art "Ersetze dies mit jenem" durchzuführen. Jetzt können all diese Zeichenketten zur Textersetzung zusammen mit dem Text des Dokuments abgespeichert werden. Als zusätzliches Plus können zur Mustersuche innerhalb der Filter Reguläre Ausdrücke verwendet werden, da Python die hierfür erforderliche Funktionalität zur Verfügung stellt.

%!Style

- Nützlich bei der Konvertierung in die Zielformate HTML oder XHTML, der jeweiligen Ausgabedatei wird dadurch eine CSS−Datei zur Definiton des Dokumentenstils zugeordnet
- Nützlich auch bei der Konvertierung in das Zielformat LaTeX, bestimmt in diesem Fall ein Paket, welches mittels des `\usepackage`−Kommandos geladen wird
- Derselbe Effekt kann auch über die Kommandozeilenoption `−−style` erzielt werden
- Die −−style Option hat höhere Priorität als %!style. Werden beide mit sich widersprechenden Optionen eingesetzt, erhält −−style den Vorrang

Einstellungen ausschließlich für ein spezifisches Zielformat

Alle Einstellungen (Ausnahme: %!target) können so definiert werden, dass sie nur bei der Konvertierung in ein spezifisches Zielformat Anwendung finden. Hierzu dient die Syntax `%!Schlüsselformat(Zielformat): Wert`. Damit kann der Benutzer für unterschiedliche Zielformate unterschiedliche Einstellungen vornehmen.

Als besonders nützlich erweist sich diese Art der Auszeichnung bei der Nutzung von Pre− und Postproc−Filtern. Gleichwohl können auch alle anderen Einstellungen zielformatspezifisch definiert werden. Derart können zum Beispiel unterschiedliche Dokumentstile für HTML und LaTeX−Dokumente definiert werden:

```
%!style(html): fancy.css
%!style(tex) : amssymb
```

Zielformatspezifische Einstellungen sind sehr nützlich, um das zu konvertierende Dokument individuell anzupassen:

```
%!target: sgml
%!options(sgml): −−toc
%!options(html): −−style foo.css
%!options(txt ): −−toc−only −−toc−level 2
```

In diesem Beispiel wurde SGML als Ausgabeformat voreingestellt, das erzeugte SGML−Dokument enthält dabei ein Inhaltsverzeichnis. Falls der Benutzer den Befehl `txt2tags −t html Datei.t2t` absetzt, finden lediglich die auf der dritten Zeile für das HTML−Zielformat definierten Einstellungen Anwendung, insofern wird die bei der Konvertierung erzeugte HTML−Datei mit der Datei "foo.css" assoziiert sein und kein Inhaltsverzeichnis besitzen.

Details für PreProc and PostProc Filter

- Filter bieten die Möglichkeit des "Suchens und Ersetzen" innerhalb der Datei (analog zum Programm sed)
- Filter folgen nicht dem Schema, dass (beim Auftreten mehrere gleichartiger Einstellungen) die zuletzt gefundene Einstellung genutzt wird, sondern sie können kumulativ angewendet werden. Folglich ist die Anzahl der Filter unbegrenzt hoch, sie können so viele Filter definieren, wie von ihnen benötigt werden. Die einzelnen Filter werden dabei genau in der Reihenfolge angewandt, in der sie angegeben wurden.
- Im Unterschied zu den anderen Einstellungen können daher sowohl Filter, die spezifisch für ein Zielformat definiert wurden als auch generische, für alle Zielformate definierte Filter bei einem Dokument Anwendung finden. Demonstriert wird dies durch das folgende Beispiel: bei der Konvertierung zu einem HTML−Dokument werden beide nachfolgend aufgeführten Filter Anwendung finden:

```
%!postproc      :   dieses   jenes
%!postproc(html):   dieses   anderes
```

- Ein Filter muss genau ZWEI Argumente erhalten
- Spezielle Escape−Sequenzen wie `\n` (Zeilenumbruch) und `\t` (Tabulatorzeichen) werden interpretiert
- Um Textpassagen zu löschen, ersetzen sie diese einfach mit einer leeren Zeichenkette

```
%!postproc: "unerwünschte Zeichenkette" ""
```

- Um Probleme zu vermeiden sollten sie immer dann, wenn sie PostProc−Filter nutzen um Tags auszutauschen, explizit ein Zielformat angeben: `%!PostProc(Zielformat): <dieses> <jenes>`
- PREproc wird angewandt direkt nachdem die Zeile gelesen wurde, und POSTproc kommt zum Tagen, nachdem alle Elemente geparst wurden. Dies entspricht der folgenden Befehlszeile (UUOC vorausgehend):

```
$ cat Datei.t2t | preproc.sh | txt2tags | postproc.sh
```

- Der erste Teil des Filters, über welchen das Ziel der Suche angegeben wird, wird nicht als gewöhnliche Zeichenkette eingelesen, sondern als Muster für einen regulären Ausdruck angesehen. Machen sie sich keine Sorgen, wenn sie nicht wissen was reguläre Ausdrucke sind, sie müssen dies nicht unbedingt wissen. Beachten sie aber in jedem Fall, dass sie, wenn sie bestimmte spezielle Zeichen innerhalb eines regulären Ausdrucks einsetzen wollen, sie diese "maskieren" müssen um sie nutzen zu können. Um ein Zeichen zu maskieren, müssen sie diesem einen umgekehrten Schrägstrich "\" voranstellen. Hier folgt eine Aufzählungen all derjenigen Zeichen, für die solch eine Maskierung vorgenommen werden muss:

```
\*  \+  \.  \^  \$  \?  \(  \)  \{  \[  \|  \\
```

- Pythons reguläre Ausdrücke sind verfügbar! Diese sind ähnlich zu denen, die in Perl Verwendung finden (PCRE). Beispiel: Verwandle bei einer HTML−Konvertierung alle öffnenden und schließenden "B" Auszeichnungen zu "STRONG"−Marken:

```
%!postproc(html):   '(</?)B>'   '\1STRONG>'
```

- Die Argumente für den Filter können auf dreierlei Art und Weise übergeben werden:
    1. Als einzelnes Word, wie etwa FOO (keine Leerzeichen)
    2. Als von doppelten Anführungszeichen umschlossene Zeichenkette, wie etwa "FOO"
    3. Als von einfachen Anführungszeichen umschlossene Zeichenkette, wie etwa 'FOO'
- Falls Ihr Muster doppelte Anführungszeichen enthält, schützen sie diese mit einfachen Anführungszeichen (und umgekehrt). Einige korrekte Beispiele:

```
%!postproc:   MUSTER    ERSETZUNG
%!postproc:  "MUSTER"  "ERSETZUNG"
%!postproc:  'MUSTER'  'ERSETZUNG'
%!postproc:   MUSTER   "ERSETZUNG"
%!postproc:  "MUSTER"  'ERSETZUNG'
```

Teil VIII − Schwarze Magie

Diese Kapitel kann Neulingen wirklich nicht empfohlen werden. Hier wird aufgezeigt, wie man mit txt2tags−Filtern unter Einsatz von komplexen Mustern und regulären Ausdrücken seltsam anmutende Dinge realisiert.

> **ACHTUNG!** Die im folgenden beschriebenen Praktiken werden NICHT empfohlen und können sogar Schäden hervorrufen. Im Extremfall kann es während des Konvertierungsprozesses sogar zum Verlust von Text aus dem Quelldokument kommen, welcher dann im Zieldokument nicht mehr auftaucht. Nutzen sie diese Dinge wirklich nur wenn sie sie unbedingt benötigen und nur dann, wenn sie wirklich wissen, was sie tun.

Ein Filter ist zwar ein sehr mächtiges, zugleich aber auch gefährliches Werkzeug!

Bei falsch definierten Filtern können sich unerwartete Resultate einstellen!

Behalten Sie das bitte immer im Gedächtnis.

Einfügen von mehreren Zeilen mittels %!PostProc (wie etwa von CSS−Regeln)

Soll in einem Filter das Muster zur Ersetzung mehrere Zeilen umfassen, so kann dies durch die Verwendung des Zeichen `\n`, welches einen Zeilenumbruch darstellt, realisiert wird.

Dies kann zum Beispiel nützlich sein, wenn nur einige wenige CSS_Regeln in ein HTML−Zieldokument eingefügt werden sollen, ohne dafür eine eigene Datei anzulegen:

```
%!postproc: <HEAD>      '<HEAD>\n<STYLE TYPE="text/css">\n</STYLE>'
%!postproc: (</STYLE>)  'body     { margin:3em              ;} \n\1'
%!postproc: (</STYLE>)  'a        { text−decoration:none    ;} \n\1'
%!postproc: (</STYLE>)  'pre,code { background−color:#ffffcc ;} \n\1'
%!postproc: (</STYLE>)  'th       { background−color:yellow  ;} \n\1'
```

Alle Filter sind dabei an den ersten Filter gebunden und ersetzen die Zeichenkette, welche dieser eingefügt hat. Auf diese Art und Weise wird die Zeichenkette "<HEAD>" verwandelt in:

```
<HEAD>
<STYLE TYPE="text/css">
body     { margin:3em              ;}
a        { text−decoration:none    ;}
pre,code { background−color:#ffffcc ;}
th       { background−color:yellow  ;}
</STYLE>
```

Nutzung von %!PreProc um Inhalte spezifisch nur in einem bestimmten Ausgabeformat zu erzeugen

Manchmal besteht die Notwendigkeit, eine Textpassage immer nur dann einzufügen, wenn in ein spezifisches Zielformat konvertiert wird und sie andernfalls wegzulassen. Dieses seltsam anmutende Verhalten kann über die Nutzung von PreProc mittels eines Tricks implementiert werden.

Der Ausgangspunkt besteht dabei darin, den zusätzlichen Text im Quelldokument zunächst als Kommentar auszuzeichnen. Die Auszeichnung erfolgt dabei aber in einer Weise, dass ein

zielformatspezifischer Filter diese Zeilen wieder in "normalen" Text zurück umwandelt.

Falls sie zum Beispiel ein zusätzlichen Abschnitt eingeben möchten, der allerdings nur in einem HTML−Zieldokument erscheint, geben sie diesen Text als "speziellen" Kommentar ein, und zwar auf folgende Art und Weise:

```
%html% Diese HTML−Seite wurde mit [txt2tags http://txt2tags.sf.net] erstellt.
%html% Die dazugehörige Quelltextdatei finden sie [here source.t2t].
```

Da diese beiden Zeilen mit dem Zeichen % beginnen, handelt es sich um Kommentarzeilen, die normalerweise ignoriert werden. Nun fügen sie der Quelldatei jedoch noch einen speziellen Filter hinzu:

```
%preproc(html): '^%html% ' ''
```

Dieser erkennt und entfernt die Kommentarauszeichnung am Zeilenanfang, dadurch werden Zeilen quasi "aktiviert", es handelt sich um keine Kommentarzeilen mehr. Da es sich um einen zielformatspezifischen Filter handelt, wird dieser nur bei der Konvertierung in ein HTML−Dokument angewandt.

Ändern von txt2tags−Auszeichnungen mittels %!PreProc

Falls sie ein Profi im Umgang mit regulären Ausdrücken sind, können sie sogar die Syntax für das Quelldokument anpassen und dabei die vordefinierten Auszeichnungen dergestalt modifizieren, dass sie diese in einer für sie möglicherweise einfacher erscheinenden Weise einzetzen können.

Hier ein Beispiel: das Tabulatorzeichen am Zeilenanfang wird von txt2tags als eine Auszeichnung für ein Zitat interpretiert. Falls der Benutzer dieses Verhalten nicht mag oder sein Texteditor nicht gut mit Tabulatorzeichen klarkommt, kann er die Auszeichnung für ein Zitat umdefinieren. Nehmen wir an, er möchte Zitate zukünftig durch die Zeichenkette ">>> " am Zeilenanfang auszeichnen, so kann er dies über einen einfachen Filter erreichen:

```
%!PreProc: '>>> ' '\t'
```

Im Quelldokument erscheinen jetzt Zitatpassagen in folgender Form:

```
>>> Dies ist ein Zitat.
>>> Der Benutzer hat hierfür diese eigenartigen Auszeichnung definiert.
>>> Diese Auszeichnung wird jedoch von Preproc−Filtern zu Tabulatorzeichen zurückkonvertiert.
```

Bevor das Parsen des Text beginnt, wird die seltsame ">>> "−Auszeichnung zu einem Tabulatorzeichen konvertiert und txt2tags erkennt diese dann als Auszeichnungen für eine Zitatpassage.

> **ACHTUNG!** Über den extremen Einsatz von PreProc Regeln kann letztlich die gesamte Syntax der Auszeichnungszeichen verändert werden, wobei sich durchaus auch Konflikte zwischen einzelnen Auszeichnungen ergeben können Seien sie daher wirklich sehr vorsichtig, wenn sie so etwas implementieren möchten.

Teil IX − Die Chronik von txt2tags

Im Juli 2001 erschien die erste öffentliche Version von txt2tags, sie trug die Versionsnummer 0.1. Aber die Ursprünge liegen mehr als ein Jahr zurück ...

Dieses Kapitel gibt einen Überblick über die Entwicklung des Werkzeugs, angefangen von ganz vorne bis hin zur aktuellen Version.

Januar 1999: Vorgeschichte

Vom Autor:

> *"Meine allerersten Versuche, ein Werkzeug für die Textkonvertierung zu erstellen begannen bereits 1999, in Form eines sehr einfachen und wenig leistungsfähigen Bourne Shell Skripts, das speziell ausgezeichneten Text in eine HTML−Seite umwandelt. Ja, ein weiteres txt2html−Werkzeug. Irgendwann muss doch irgendjemand sowas schon mal gemacht haben ... In aller Kürze, das Werkzeug erkannte nur einfache Auszeichnungen wie* `*bold*`*,* `/italic/`*,* `_under_`*, und maskierte die klassischen* `< & >` *Spezialzeichen von HTML. Nicht besonders eindrucksvoll, aber hey! ich war damals jung ;)"*

Juni 1999: Immer noch Vorgeschichte

Der Autor möchte noch etwas mehr sagen:

> *"Einige Monate vergingen, und eine große Hyphe um SGML hatte bei der Firma, für die ich arbeitete (Conectiva) um sich gegriffen. Daher wurde txt2html in ein txt2sgml−Skript umgewandelt. Zu dieser Zeit habe ich ernsthaft versucht alles über SED* zu lernen, insofern war txt2sgml ein 110−zeiliges Bourne Shell−Skript mit sehr viel SED−Code integriert."*
>
> * **SED:** UNIX Stream EDitor − ein automatisches Textbearbeitungswerkzeug

Diese verbesserte SGML−Version unterstützte bereits mehr Strukturen, wie etwa Aufzählungen oder die Auszeichnung als Originaltext. Anhand der folgenden Beispieldatei können sie den Ursprung der txt2tags−Aufzeichnungen erkennen:

```
* Das war eine fettgedruckte Zeile (FETT zeilenorientiert angewandt? Nun gut ...)

  ––
− Nicht nummerierte Aufzählungen waren den Aufzählungen in txt2tags bereits sehr ähnlich
− aber man benutzte –– um eine Liste anzufangen oder abzuschließen
  ––

=———————————————
Originaltext wurde durch das Muster =–– abgetrennt.
Die zusätzlichen Striche ——————— waren mehr kosmetischer Natur.
=———————————————
```

Noch nicht sehr eindrucksvoll, aber der große Schritt kommt jetzt ...

August 2000: Jetzt keine Vorgeschichte mehr

Wieder spricht der Autor:

> *"Ein Jahr verging, und zu diesem Zeitpunkt war ich wirklich in SED verliebt. Das txt2sgml.sh Shellskript wurde umgeschrieben und wurde dabei zu einem reinem, 350 Zeilen umfassenden SED−Skript. Etliche praktische Funktionalitäten wurden hinzugefügt wie etwa Unterabschnitte, Erkennung von URLs und Aufzählungen innerhalb von Aufzählungen. Ich habe dieses Skript viel benutzt und verbessert, fast ein ganzes Jahr lang."*

Eine Beispieldatei txt2sgml.sed:

```
* Sieh an, da sind diese drei ersten magischen Zeilen
* Der Titel des Dokuments / Autor / Datum
* Aber damals benötigten sie diese Sternchen am Anfang

HAUPTÜBERSCHRIFT

Kapitelangaben wurden über Zeilen komplett in Großbuchstaben gemacht.
Abschnitte wurden durch vorangestellte Leerzeichen gekennzeichnet.
Jedes einzelne Leerzeichen repräsentierte eine Gliederungsebene für das Kapitel.
Auszeichnung für Schriftstile: *fett*, **betont**, "kursiv" und `Schreibmaschine`.

− Aufzählung
  + Aufzählung eine Ebene tiefer
    = und eine Aufzählung noch eine Ebene tiefer
      (ausgezeichnet über das '=', nicht durch Einrückung)


Zwei Leerzeilen schlossen eine Aufzählung ab. Verknüpfungen wie www.beispiel.de
und E−Mail−Adressen wurden über reguläre Ausdrücke automagisch erkannt.
Und es gab eine seltsame Auszeichnung für Bilder:
%%image: path/to/image.jpg
```

Mai 2001: Umsetzung in Python und die Idee mehrerer Zielformate

Raten sie mal wer spricht:

> *"Ich habe angefangen, mein Buch über Reguläre Ausdrücke zu schreiben und verfasste den Text in einem Format, das für die Konvertierung mit txt2sgml.sed bestimmt war. Auf diese Art und Weise konnte ich es nach SGML konvertieren (und anschließend nach HTML, wofür ich das Programm sgml2html benutzte) und dann sofort im Browser überprüft, wie das Buch denn so aussah. Da 'sofort' und 'sgml2html' jedoch nicht so recht zusammenpassten, habe ich das SED−Skript in ein Werkzeug namens txt2html.sed umgewandelt, mit dem ich direkt HTML erzeugen konnte. [...] Der Verleger nutzte Adobe PageMaker Software um seine Bücher zu formatieren und das stellte für einen Linux−Fan, wie ich es bin ein Problem dar. Glücklicherweise habe ich festgestellt, dass PageMaker eine HTML−ähnliche Sprache für die Auszeichnung seiner Dokumente besitzt, insofern wandelte ich mein Skript sukzessive in ein Konvertierungswerkzeug namens txt2pagemaker.sed um. Letztendlich hatte ich dann drei ähnliche SED−Skripte um mein Dokument nach SGML, HTML und ins PageMaker−Format umzuwandeln. Und einige weitere Shellskripte waren für die Erzeugung des Inhaltsverzeichnisses und für die Nachbearbeitung des umgewandelten Texts zuständig. Mitten während ich das Buch schrieb, kam mir die Idee, alle diese Hilfsmittel in einem einzigen Werkzeug zu vereinen, und ich wählte mir Python als Programmiersprache dafür aus. TXT2TAGS war geboren."*

Juli 2001: 0.x Serie: Erstes öffentliches Erscheinen von txt2tags

Ja, er ists:

"Die Herausgabe des gedruckten Buchs (31. Juli) und die Veröffentlichung der allerersten Version 0.1 von txt2tags (26. Juli) lagen zeitlich sehr nahe beieinander, es handelte sich nur um Tage. Beide hingen sehr stark voneinander aber und wurden zusammen entwickelt. Neben SHML, HTML und PageMaker wurden noch weitere Zielformate für diese Version implementiert: MoinMoin, Magic Point und gewöhnlicher Text. Mehr als ein Jahr lang folgten weitere Versionen in der 0.x Series und das Programm begann zu wachsen: UNIX Man Page als neues Zielformat, das %%date−Makro, Unterstützung von Tabellen, eine Webschnittstelle, intelligente Ausrichtung von Bildern sowie die Erzeugung eines Inhaltsverzeichnisses. In Version 0.2 wurde ein Shell−Skript als Umhüllung hinzugefügt, mit welchem die Dateioperation durchgeführt und die Kommandozeilenoptionen verarbeitet wurden. Ich habe das so gemacht, da ich mich mit der Shell sehr gut auskannte, in Python aber völlig unbedarft war. Es sollte bis zur Version 0.9 dauern, ehe txt2tags wieder zu 100% aus Python Code bestand."

September 2002: 1.x Serie: Wachstum

Geben sie das Mikrophon jetzt nicht endlich ab?

"Die Idee von txt2tags hat sich als gut erwiesen. Ich habe mich entschieden, die Sache ernst zu nehmen. Der nächste Schritt bestand darin, das Programm weiterzuverbreiten und andere darüber zu informieren: Dokumentation! Die Webseite des Programm wurde gestartet, Mailing−Listen (Englisch und Portugiesisch) wurden eingerichtet und das Benutzerhandbuch wurde begonnen. Die Anzahl der Benutzer stieg und viele Anregungen kamen. Ja, ich denke es funktioniert. Hier die neuen Möglichkeiten, die zur 1.x−Serie hinzugefügt wurden: Graphische Schnittstelle (GUI), Kompatibilität zu Windows & Mac, LaTeX als Zielformat, %!style, include Kommandos sowie die mächtigen Pre− und Postproc−Filter zur Vor− und Nachbearbeitung des Dokuments."

Juli 2004: 2.x Serie: Reifeprozess

Ok, ich gebe das Mikrofon weiter:

"Wachstum ist schwierig und manchmal auch seltsam. Erinnern sie sich dass ich sagte, dass ich ein Python−Neuling war? Nun, ich wurde besser, aber es hatten sich viele alte Fehler angehäuft, insofern war ein größere Überarbeitung des Programmquelltexts unvermeidbar. Die neue Version war bei einigen Auszeichnungen nicht abwärtskompatibel, weswegen ein Skript für das Upgrade älterer Versionen erstellt wurde. Es dauerte eine ganze Weile, aber schließlich wurde Version 2.0 veröffentlicht. Diese brachte jede Menge neuer Funktionen, mit wie XHTML als Zielformat, HTML−Code der gemäß W3C−Standards validiert werden kann, i18n, und eine RC Datei. Ein Team von Übersetzern hat sich eingestellt und das Programm und auch die dazugehörige Dokumentation wurden in viele Sprachen übersetzt. Lout als Zielformat wurde hinzugefügt, und es kamen neue Makros hinzu: %%mtime, %%infile, %%outfile, and %%toc. Wird fortgesetzt ..."

The End.