Projektarbeit:
“htmLaTeX” - Konvertierungsoftware
–Ausarbeitung–

von:
Kaiser, Björn
Mühlendamm 6
24937 Flensburg
bjoern-kaiser@versanet.de
Matrikel-Nr.: 371658

und

Baß, Björn
Ritterstraße 28
24939 Flensburg
b-bass@versanet.de
Matrikel-Nr.: 341125

Betreuer: Prof. Dr. Hans Werner Lang
SoSe 2011-I
Fachbereich Technik
Fachhochschule Flensburg

Abgabetermin: 23.03.2011

Inhaltsverzeichnis

1. Einleitung - Motivation

Das Ziel dieses Projektes ist eine möglichst flexible Anwendung zu schaffen, die aus einem Markup in eine andere per XML[1]-definierbare Syntax konvertieren kann.

Dies wurde am Beispiel von der Konvertierung von JavaDoc - generiertem HTML-Code und einer anschließenden Umwandlung zu LaTeX[2] verfolgt.

Um möglichst menschenlesbar weitere Konvertierungsszenarien umsetzen zu können wurden folgende Festlegungen getroffen:

- Die Eingabesemantik wird in einer XML-Datei beschrieben und soll die Umsetzung des Ausgangscodes in eine (pseudo)-HTML Semantik beschreiben.
- Die Ausgabesemantik wird ebenso beschrieben und definiert die Konvertierung in das Zielformat.
- Diese beiden Konfigurationsdateitypen müssen im Rahmen ihrer Syntax in der jeweils inline verfassten DTD [3] frei beschrieben werden können.

2. Durchführung der Projektarbeit

2.1. Rahmenbedingungen und Tools

Betriebssystem Es wurde parallel unter *Windows 7*[4] und *Linux*[5] entwickelt

Sprache Um einerseits hohe Plattformunabhängigkeit und andererseits Performanz zu erreichen wurde das Projekt in der C++ Klassenbibliothek *Qt*[6] erstellt.

IDE Es wurde die Entwicklungsumgebung[7] *Qt Creator*[8] benutzt.

Lokalisierung Es wurden Übersetzungen in den Sprachen Englisch und Deutsch erstellt, die zur Laufzeit gewechselt werden können.

Interface Das Programm ist sowohl über ein grafisches Interface als auch als per Skript bedienbar und verwaltet die Programmeinstellungen in einer XML-Datei.

[1] **X**tensible **M**arkup **L**anguage
[2] **La**mport **TeX** - umfangreiche Sammlung von TeX-Makros
[3] **D**okument**typ****d**efinition
[4] Microsoft® und Windows® sind eingetragene Marken der Microsoft Corporation.
[5] Linux® ist ein eingetragenes Markenzeichen von Linus Torvalds
[6] Qt® ist ein eingetragenes Markenzeichen der Nokia Corporation. http://qt.nokia.com/
[7] engl. **I**ntegrated **D**evelopment **E**nvironment
[8] http://qt.nokia.com/products/developer-tools

Dokumentation Die Dokumentation der Implementierungsdetails wurde aus dem Quellcode mithilfe des freien Dokumentationswerkzeuges *Doxygen*[9] generiert.

Versionierung Für die Versionierung wurde das verteilte Versionskontrollsystem *Git*[10] verwandt. Als Hoster diente der spezialisierte Webhosting-Dienst *GitHub*[11].

Unit Tests Nach einigen Problemen mit der Qt-eigenen Komponente für Unit Tests (QTestLib) haben wir auf Unit Tests vorerst verzichtet.

Webpräsenz Auf http://opus4711.github.com/htmlatex/ können der Quelltext, die Entwicklerdokumentation, das Pflichtenheft und die Ausarbeitung (dieses Dokument) in einem zip oder tar-Archiv heruntergeladen werden.

Eingabehilfen Als Screenreader wurde *Dolphin*[12] eingesetzt.

Issues Als Issue-Tracking-System wurde die Issues-Komponente von *GitHub* eingesetzt.

2.2. Implementierungsphase

Das Projekt wurde mithilfe der Extreme Programming-Methode teils in paralleler Einzelarbeit auf Personen- und Aufgaben(Issue)-bezogenen Entwicklungszweigen, teils gemeinsam umgesetzt.

Zu Begin wurde auf dem Hauptentwicklungszweig “master” der noch funktionslose Prototyp der Anwendung erstellt und die grafische Oberfläche entworfen. Im Folgenden wurden Teilfunktionalitäten in getrennten Branches in kurzen Zyklen (zwischen 30 Min und 5 Std. Programmierzeit) implementiert. Nach gegenseitiger Absprache wurde der Branch dann auf den beiden Betriebssystemen getestet und mit dem Hauptentwicklungzweig verschmolzen (“merge”).

Kodierung Die Kodierungsfunktionen von Qt lieferten entgegen der Dokumentation nicht auf beiden Systemen die angestrebten Ergebnisse.

Wurde ein `QTextStream` mit `setCodec("UTF-8")` festgelegt, konnte er UTF-8 Text mit und ohne BOM[13] nicht korrekt verarbeiten. Erst die auf einen `QString` angewandte Methode `toLatin1()` konnte plattformübergreifend die von uns genutzte UTF-8 Auswahl korrekt darstellen. Uns ist bewusst, dass die Latin1-Codierung in diesem Bereich weitestgehend deckungsgleich ist, dies aber für größtmögliche Flexibiltät wohl nur ein Workaround ist. Bei der Verarbeitung der

[9] http://www.stack.nl/~dimitri/doxygen/index.html
[10] http://git-scm.com/
[11] http://github.com/
[12] Dolphin® ist ein eingetragenes Markenzeichen der Dolphin Computer Access Ltd. http://www.yourdolphin.com/
[13] Byte Order Mark

Übersetzungsdateien für dieses Projekt mittels `QTextCodec` und der Methode `codecForName("utf8")` trat dieses seltsame Verhalten nicht auf.

Interface Bei der Gestaltung der Kommandozeilenversion haben wir uns in vielen Fällen an den GNU Coding Standards[14] orientiert.

```
bjoern@lotos:~/projekt/htmlatex> ./htmlatex
usage: htmlatex INPUTFILE FORMAT INPUTDEFINITION OUTPUTFILE FORMAT OUTPUTDEFINITION [-g|--gui]
  e.g. htmlatex index.html javadoc input_javadoc.xml manual.tex tex output_tex.xml -g
       -g, --gui       Launch the GUI
       -h, --help      Show some examples
       -v, --verbose   Verbose mode
       -lang=de, -lang=en      Sets the language to German or English

       See the "README" file for further information.
```

Abbildung 1: GUI

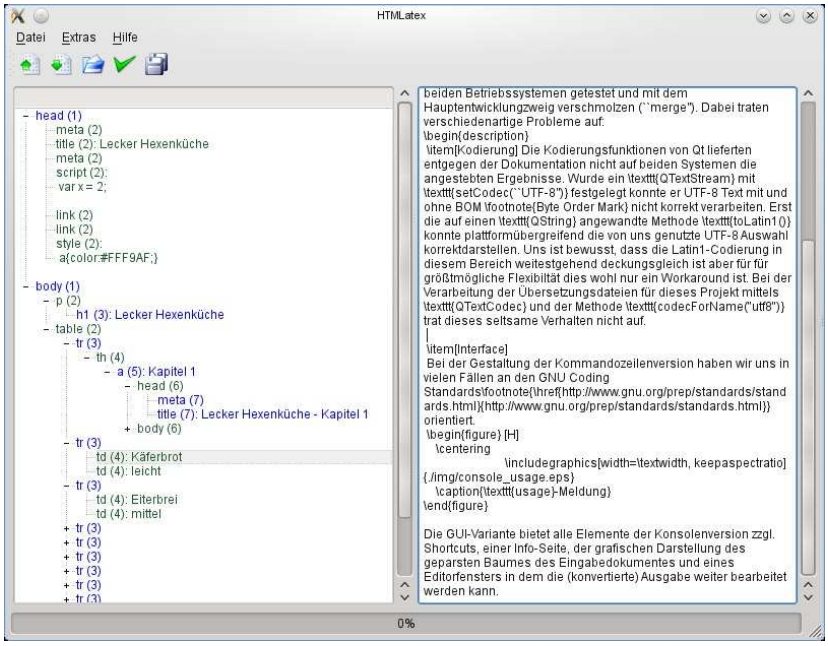


Abbildung 2: GUI-Variante

Die GUI-Variante bietet alle Elemente der Konsolenversion zzgl. Shortcuts, einer Info-Seite, einem Fortschrittsbalken, der grafischen Darstellung des geparsten Baumes des Eingabedokumentes und eines Editorfensters in dem die (konvertierte) Ausgabe weiter bearbeitet werden kann.

[14] http://www.gnu.org/prep/standards/standards.html

Dank des Dokument-Generators Doxygen ist eine Einbindung der Entwicklerdokumentation dieses Projektes als “Hilfe”-Komponente kein großer Aufwand mehr, da direkt Qt Compressed Help (.qch) ausgegeben werden kann.

Einstellungen Die Einstellungen werden in der Datei `htmlatex_settings.dat` binär serialisiert.

Lokalisierung Zu übersetzende Strings wurden mittels der Qt-Funktion `tr("str")` markiert, mit dem Programm `lupdate` in ein XML-Zwischenformat (.ts) extrahiert und mithilfe des `QtLinguist` für deutsch und englisch übersetzt. Die Übersetzungen werden dann mittels des Programms `lrelease` in das binäre .qm-Format (Qt Message File Format) überführt, das auf hohe lookup-Geschwindigkeit optimiert ist und von der Programmdatei direkt genutzt wird. Fehlt die gewünschte Sprachdatei, werden die im Quelltext direkt geschriebenen Strings ausgegeben.

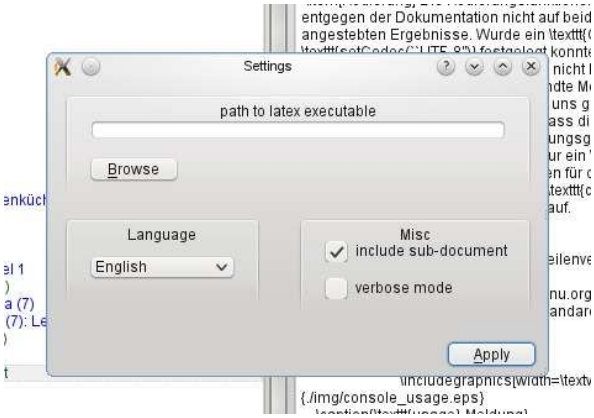


Abbildung 3: Einstellungen

externer Programmaufruf Der Pfad zu LaTeXProgrammdatei kann ebenfalls in den Einstellungen eingegeben werden. Für Linux könnte auch ein Aufruf des Kommandos `which` Voreinstellungen liefern.

2.3. Reflektion

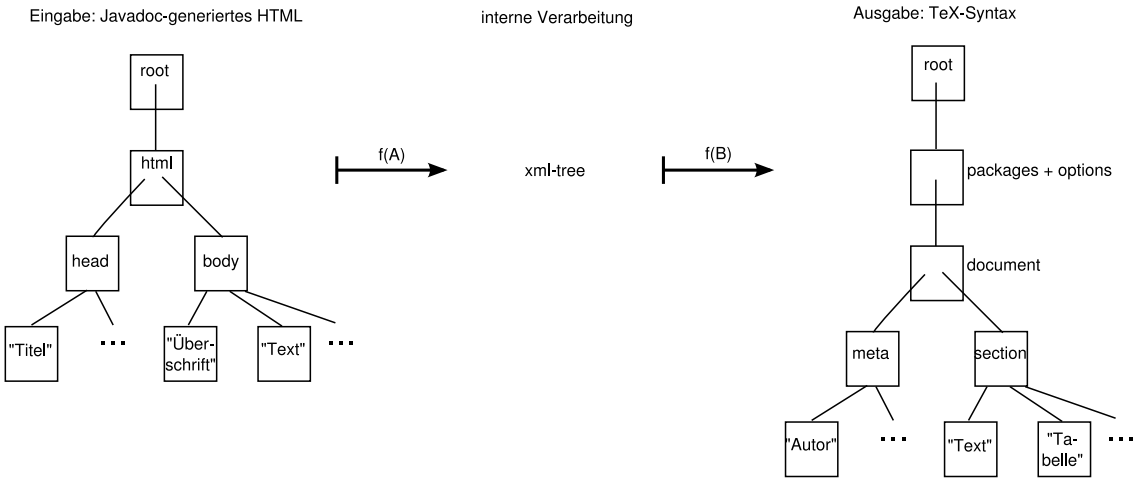
3. Implementierung der Konvertierung

Es wurde die im Pflichtenheft visualisierte Basis für die Konvertierung angestrebt. Hierbei haben wir uns für einen XML-DOM[15]-Baum für die interne Verarbeitung festgelegt.

[15] Document Object Model

1. Das Eingabeformat steht – im Falle von JavaDoc, wie es z.B. für Java unter http://download.oracle.com/javase/6/docs/api/overview-summary.html zu finden ist – leider nicht in einem XML konformen Format bereit. Um es hierin zu überführen, nimmt ein Präprozessor die nötigen Umformungen vor. Er wird in einer Hook-Methode vor dem Einlesevorgang ausgeführt und ist leicht zu ersetzen.

2. Nun beginnt der Einlesevorgang mithilfe der Eingabe-Definitionsdatei im XML-Format.



4. Ausblick

4.1. Funktionsumfang

4.1.1. Konsolenanwendung

4.1.2. Grafische Oberfläche

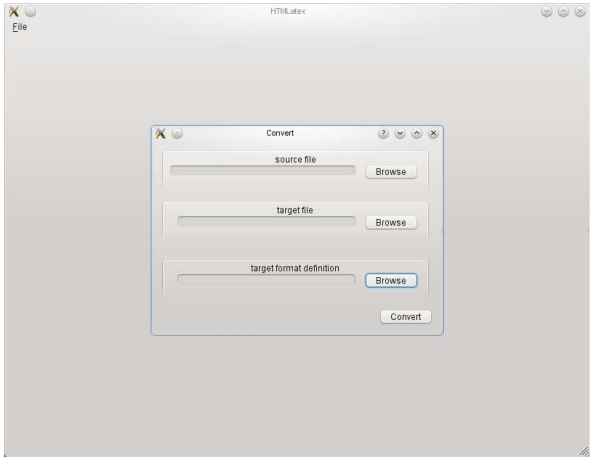


Abbildung 4: BILDUNTERSCHRIFT

Neben Verbesserungen am Interface ist es nun interessant zu überprüfen, ob das Programm in der Lage ist in, der Praxis mit mehr Formaten erfolgreich zu arbeiten und Aussenstehende zu gewinnen, die Formatdefinitonen in XML-Form anpassen.

Zur Senkung des Aufwandes wäre es wohl sehr wünschenswert, wenn dieses z.B. mittels einer Eingabemaske zum Bearbeiten und Erstellen dieser Beschreibungsdateien geschähe. Darüber hinaus würde ein Syntaxhighlighting Tippfehler weiter minimieren.

Der die Anpassungen, die der Präprozessor vorzunehmen hat, damit das Eingabeformat XML-Konform wird könnten auch anpassbar in XML definiert werden.

Der Editor für die Ausgabedateien sollte neben “Suchen&Ersetzen” und einer Schriftgrößen-Einstellung die üblichen Textbearbeitungswerkzeuge zum *Ausschneiden*, *Kopieren* und *Einfügen*, so wie für *Rückgängig* und *Wiederholen* bekommen. `undo()`, `redo()`, `cut()`, `copy()`, `paste()` und `setFontPointSize()` sind im *QTextEdit*-Widget schon enthalten und deshalb auch leicht nachzurüsten.

Die Baumansicht des Eingabedokumentes sollte per Kontextmenü einen speziellen Knoten oder Unterbaum vor dem Übersetzungsvorgang entfernen können, um in Dokumentenstrukturen mit vielen Unterdokumenten die Auswahl begrenzen zu können.

Ebenso ist es erstrebenswert, die maximal zu verfolgende Linktiefe begrenzen zu können und Linkschleifen zu vermeiden.

A. Erklärung

Hiermit erklären wir, dass das Projekt htmLaTeXvon uns selbständig und ohne Hilfe Dritter erarbeitet und realisiert wurde.

Björn Kaiser

Björn Baß