Maturaarbeit an der Kantonsschule am Burggraben

wiki2xhtml – Einfach Internetseiten erstellen

Ein in Java geschriebenes Programm
zum Erstellen von kompletten Internetseiten
aus Textdateien in der Wiki-Syntax

Vorgelegt am 5.2.2007 durch Simon Eugster (4eNP) bei Jürg Strassmann

Inhaltsverzeichnis

Einleitung

Eine Maturaarbeit ist nicht etwas Alltägliches; wenn man eine schreibt, dann meistens nur ein Mal im Leben. Ich habe mir darum das Thema sorgfältig ausgesucht. Was habe ich dabei berücksichtigt?

Ich habe mir vor der Themenauswahl ein paar Maturaarbeiten und Maturaarbeitspräsentationen angeschaut. Die einen Themen waren durchaus interessant, die anderen weniger. Etwas hatten aber die meisten Arbeiten gemeinsam, etwas, was mir recht zu denken gab: Sie verstauben, werden nach ein, zwei Jahren vergessen sein und niemanden interessiert nachträglich noch, was in der Arbeit steht. Unter anderem weil eine Arbeit nach zwei Jahren von der Bibliothek der Kanti ins Archiv der Kanti wandert. Auch wenn sich jemand für die Arbeit interessieren würde: Die Chancen, auf die Arbeit zu stossen, sind recht gering, vor allem, wenn man nicht an der Kanti arbeitet. Dafür die empfohlenen 100 Stunden zu investieren war mir zu schade; eine solche Arbeit, nur damit ich eine Note im Maturazeugnis habe.

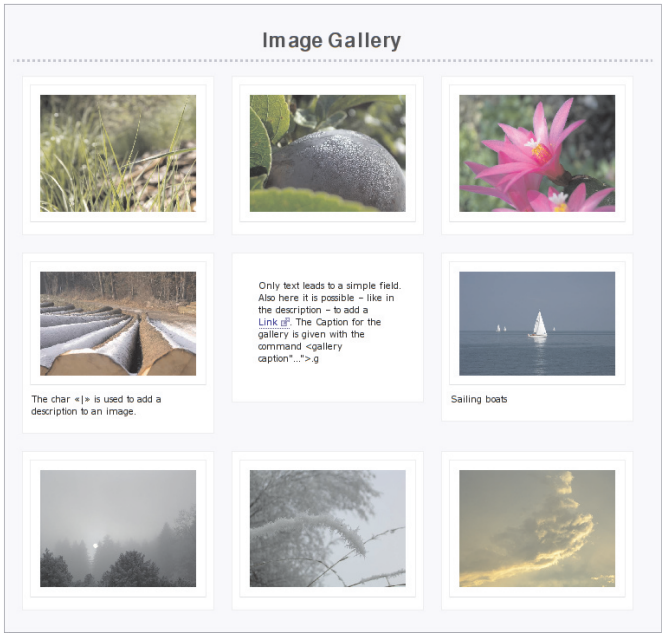
Für mich war eigentlich schon von vornherein klar, dass ich etwas Praktisches machen würde. Etwas, wovon auch andere Leute profitieren können, etwas, das man also nicht einfach in die Ecke stellt und vergisst oder im Archiv der Kanti vor interessierten Blicken versteckt. Schliesslich habe ich mich dazu entschlossen, ein Programm zu schreiben, mit dessen Hilfe ein Anfänger *mit wenig Aufwand eine eigene Internetseite erstellen kann, die gut aussieht, den Standards entspricht und barrierefrei ist.* Es soll aber durch die Flexibilität auch von Profis verwendet werden können. Kein einfaches Ziel also, da auch schon unzählige Programme, meist WYSIWYG-Editoren[1], im Internet kursieren. Diese WYSIWYG-Editoren stellen jedoch in vielen Fällen das eigentliche Problem dar. Oft generieren sie unsauberen Quellcode (wie das aktuelle Frontpage von Microsoft), und vor allem sehen die

1 WYSIWYG steht für «What you see is what you get». Mit solchen Programmen können Internetseiten ähnlich wie mit einem DTP- oder einem Office-Programm (OpenOffice.org, Word) zusammengeklickt werden, man sieht das Resultat schon während dem Erstellen. Ein Beispiel für ein soches Programm wäre NVU (http://nvu.com).

erstellten Seiten oft sehr nach einem Gebastel, total veraltet oder einfach schrecklich aus (hier ist allerdings meist nicht das Programm schuld). Darum will ich wiki2xhtml[2] – mein Programm – möglichst viel selber machen lassen. Der Benutzer soll also nur den Text liefern, der dann in eine Internetseite umgewandelt wird, nicht aber teilweise heikle Dinge wie z. B. die Schriftgrösse oder Zeilenabstände festlegen. Gewisse Dinge müssen aber vom Benutzer vorgegeben werden, etwa Textstellen, die fett geschrieben werden sollen. Hierfür eignet sich die Wiki-Syntax ideal, die zudem einfach zu erlernen (und für das Programm schwierig zu verarbeiten) ist.

Der Benutzer liefert also den Text, in dem er bereits festgelegt hat, was fett werden soll, wo eine Aufzählung steht, wie ein Link beschriftet werden soll. Erfahrene Anwender können diesen Text bereits mit weiterem Code vollpflastern, etwa wenn wiki2xhtml eine bestimmte Funktion (noch) nicht unterstützt oder wenn ein Textteil speziell formatiert werden soll. Nun wird dieser Text in (X)HTML[3] umgewandelt und per CSS[4]-Datei – die natürlich anpassbar ist – gestaltet. Fortgeschrittene Benutzer können diese nach Belieben verändern oder eine eigene Datei verwenden.

Ein weiterer Anwendungsbereich: Das Programm kann das Grundgerüst für HTML-Dateien (dazu gehören unter anderem die Tags html, head und body) weglassen und somit Code generieren, der in anderen Programmen weiterverwendet werden kann. PostNuke – ein CMS[5] – etwa verlangt Reintext oder HTML-Code. Für eine Liste benötigt man HTML-Code, der von Hand aber recht mühsam zu schreiben ist. Hier ist man viel schneller und bequemer am Ziel, wenn man den Code mit meinem Programm generiert.



Mit wiki2xhtml erstellte Bildergalerie

2 Der Name meines Programmes hat sich etwas durch Zufall ergeben; zuerst nannte ich es wiki2html, bei Sourceforge.net existierte aber schon ein solches Programm, und so wurde es wiki2xhtml – denn es konvertiert Text von der Wiki-Syntax in XHTML-Dateien.

3 XHTML hat im Gegensatz zu HTML strenge Vorschriften: Die Elemente werden etwa immer klein geschrieben (ein <BODY> ist also nicht mehr erlaubt). Weitere Informationen: http://de.wikipedia.org/wiki/XHTML

4 CSS wird unter dem Abschnitt «Design» noch näher betrachtet.

5 Ein CMS (kurz für «Content Management System») ist ein Programm, das es erlaubt, im Internet direkt Inhalte einer Internetseite zu bearbeiten, die Dateien müssen also nicht mehr hochgeladen werden. Diese Systeme vereinfachen es enorm, eine Seite zu aktualisieren oder neue Inhalte hinzuzufügen, deswegen werden sie immer populärer. Ein Beispiel für eine CMS-basierte Intenetseite ist übrigens die Wikipedia.

wiki2xhtml

Programmierung

Ich schreibe das gesamte Programm in Java 1.5[6]. Java hat mindestens einen Vorteil gegenüber anderen Programmiersprachen: Es ist plattformunabhängig. Das heisst, man kann das Programm überall verwenden, wo eine Java Runtime zur Verfügung steht. Das betrifft vor allem die drei gängigsten Plattformen Linux, Mac und Windows. In anderen Sprachen muss der Quelltext für jede Plattform neu kompiliert – also in für Maschinen lesbaren Code umgewandelt – werden, wobei man eine ausführbare Datei erhält. Meist müssen auch Teile des Quellcodes selber angepasst werden. Das ist zum Teil nicht einfach, vor allem wenn man die gewünschte Plattform gar nicht hat. Dann kann man selber nur für die Plattformen, die man selber benutzt, programmieren oder muss sich jemanden suchen, der sich für das Projekt interessiert und andere Plattformen verfügbar hat.

Bei Java reicht eine Datei und eine «Quellcode-Auführung».[7] Die sogenannten jar-Dateien können dann z. B. sowohl unter Windows als auch Mac ausgeführt werden. Dies hat mehrere Vorteile: Einerseits wird es für den Programmierer einfacher, da weniger programmiert und abgeändert werden muss – natürlich muss er aber auch plattformunabhängig programmieren. Aber wie geht das? Das Paradebeispiel hierfür sind die Dateiseparatoren: Windows verwendet dazu den Backslash (Beispiel: C:\Dokumente und Einstellungen\User\ für das «home-Verzeichnis») und die beiden von Unix abstammenden Betriebssyteme den normalen Schrägstrich (Home-Verzeichnis unter Linux: /home/user/). Sollen nun in einem bestimmten Verzeichnis Einstellungen des Programmes gespeichert werden, kann man diese nicht einfach in «C:\Dokumente und Einstellungen\User\Programm\» speichern, denn unter Unix wird hier einfach eine Datei mit diesem Namen angelegt; dort ist der Backslash in Dateinamen erlaubt. Sondern Java stellt dazu Variablen zur Verfügung. In diesem Beispiel soll im jeweiligen «home-Verzeichnis» – also das Verzeichnis, wo der Benutzer seine Daten speichert – ein Ordner «Programm» erstellt werden, wo die betreffenden (Konfigurations-)Dateien gespeichert werden. Der absolute Pfadname zu diesem Ordner lautet nun in Java so (File.separatorChar ist gleichbedeutend mit java.lang.System.getProperty("file.separator")):

---

6 Zuerst verwendete ich Java 1.4; Gegen das aktuellste Java – damals 1.5 – sprach, dass nicht alle Systeme aktuell gehalten werden und oft noch die alte Java-Runtime installiert ist. Als ich dann aber die Eingabedateien in eine Liste variabler Länge speichern wollte, musste ich aufgrund besserer Unterstützung solcher Listen auf 1.5 umsteigen. Inzwischen ist die Version 1.6 erschienen.

7 Der Grund, warum das funktioniert, ist, dass man zum Ausführen von Java-Programmen eine Runtime braucht. Diese interpretiert den Java-Code dann für jedes Betriebssystem. Der Nachteil an diesem System ist, dass Java daduch recht langsam wird.

```
java.lang.System.getProperty("user.dir") + File.separatorChar +
"Programm" + File.separatorChar
```

Abhängig vom Betriebssystem werden in diese Variablen nun die richtigen Inhalte eingefügt. Unter Windows enthält der String nachher «C:\Dokumente und Einstellungen\User\Programm\», und unter Linux «/home/user/Programm/». Mehr braucht es nicht, um für mehrere Plattformen zu programmieren – dank dieser Variablen.

Ein weiterer Vorteil ist die Verbreitung. Würde das Programm nur unter Apple laufen, könnte ungefähr 95 % der Computer nichts damit anfangen. So aber kann eigentlich jeder mithelfen, das Programm testen, Fehler korrigieren und schlussendlich auch nutzen. Ausserdem ist Java gratis zu bekommen. Die benötigte Java-Runtime hat momentan eine Grösse von etwa 15 MB. Sun Microsystems hat 1995 Java geschaffen und wird jetzt – nach langen Diskussionen – Java in OpenSource unter der GNU General Public License (siehe Wikipedia: http://de.wikipedia.org/wiki/GPL) veröffentlichen.

Der letzte Vorteil, den ich hier aufzählen möchte, ist der Quellcode. Java ist recht leicht und gut lesbar, oft (für meinen Geschmack) besser als C/C++ (wovon Java ja abstammt), und sieht – ein einigermassen guter Viewer vorausgesetzt – auch sauber aus. Und solange der Quellcode eines Programmes übersichtlich wirkt, ist es auch einfacher, sich darin zurechtzufinden oder allenfalls sich darin einzuarbeiten. Natürlich ist es auch Geschmacks- und Gewöhnungssache, ob man nun eine Sprache «schön» findet.

Zum Schreiben des Codes verwende ich Eclipse Callisto[8] 3.2.1, eine frei erhältliche Programmierumgebung. Eclipse wurde von IBM entwickelt und 2001 für jedermann freigegeben. Der Wert des Quellcodes wurde auf ungefähr 40 Millionen US-Dollar geschätzt. Eclipse selber ist ebenfalls in Java geschrieben und sehr benutzerfreundlich gestaltet. Sowohl bei der Eingabe des Codes als auch bei der Fehlersuche hilft es massgeblich.

Die Autovervollständigung erspart einem mit etwas Erfahrung einige Tastendrücke. Wenn ich zum Beispiel eine Variable namens «anyName» definiert habe, reicht schon die Eingabe von «anNa», und Anna wird per Ctrl-Leertaste automatisch zu anyName vervollständigt. Auch bei Funktionen hilft sie: Nach Eingabe von «System.» werden alle möglichen Befehle aufgelistet. Das ist recht nützlich, vor allem, wenn man noch ungefähr weiss, was man mit der Funktion machen konnte, aber nicht mehr, wie sie genau hiess. Etwa bei einer Variable des Typs String: Hier kann man im String nach einer bestimmten Zeichenfolge suchen. Ctrl-Leertaste nach einem String zeigt die verfügbaren Funktionen und auch dessen Rückgabewert. Bei der Funktion «charAt» etwa wird nach einem Zeichen innerhalb des Strings gesucht. Der Rückgabewert ist ein Integer, also eine Ganzzahl, welcher mit den Werten $\geqslant 0$ angibt, an welcher Stelle das Zeichen gefunden wurde, oder

---

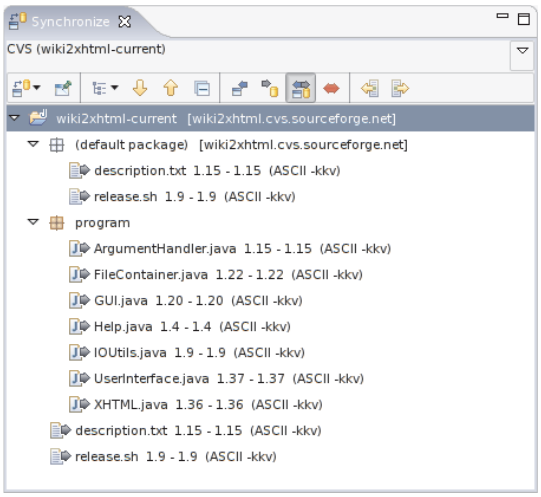
8 Ab der Version 3.2 (Codename Callisto) unterstützt Eclipse Java 1.6. Ausserdem wurden einige Funktionen wie das Plugin-Management verbessert. Eclipse, mit dem momentan ungefähr 2.3 Millionen Entwickler arbeiten, kann unter http://www.eclipse.org heruntergeladen werden.
Nebenbei: Der Name Callisto stammt von einem der Monde Jupiters.

aber -1 ist, falls das Zeichen nicht enthalten ist. Der Aufwand per Autovervollständigung lässt sich so etwas verringern.

Eine Stärke von Eclipse, die schon recht früh implementiert wurde, ist die Anzeige von Syntaxfehlern während der Eingabe und die Angabe von Lösungsvorschlägen. Wenn man beispielsweise bei einer for-Schleife die verschiedenen Argumente statt durch Semikola fälschlicherweise durch Kommata trennt, wird die entsprechende Textstelle rot markiert mit der Meldung, dass ein Semikolon erwartet wird. Variablen oder importierte Packages, die nicht verwendet werden, werden gelb angestrichen, so dass der Quellcode nicht zu viele «Variablenleichen» enthält und übersichtlicher wird.

Aber auch wenn man eine als final (unveränderbar) definierte Variable während der Programmausführung verändern will oder ein Package für bestimmte Funktionen fehlt, bringt Eclipse Lösungsvorschläge. Vor allem am Anfang ist das recht wichtig, wenn man programmieren lernt, da man so vieles lernen kann. Aber auch später wird man so immer wieder kluger ...

Eclipse verfügt noch über weitere Fertigkeiten. Zwei, die ich regelmässig verwende, sind die Debug-Ansicht und die CVS-Verwaltung. Im Debugger kann ich, wenn ein Programmteil nicht ordnungsgemäss funktioniert, Variablen überprüfen, während das Programm läuft – beispielsweise ob eine einen falschen Wert enthält – oder den Code schrittweise durchlaufen, um Ursachen für Fehler auf den Grund zu gehen, falls die Fehlermeldung selber nicht genug aussagt. Vor allem bei komplizierteren Situationen ist das oft von Vorteil, da ich zum Beispiel in Schleifen sehe, wie sich die Werte schrittweise verändern.



Eclipse beim Abgleichen der Dateien über CVS

Die CVS-Ansicht ist dazu gedacht, mit einem CVS-Server zu kommunizieren, das heisst Daten auszutauschen. Ich bin per Zufall zu einem Account auf dem Server der ETH gekommen. Ich konnte dort etwas experimentieren und mir einen Überblick über die verschiedenen Funktionen verschaffen. Da alle Daten auf einem Server gespeichert werden, kann man von jedem Computer aus oder natürlich auch vom selben PC aus mit verschiedenen Betriebssystemen darauf zugreifen. CVS[9] bietet aber noch weit mehr

9 Im Internet ist unter http://cvsbook.red-bean.com das unter der GPL lizenzierte Buch «Open Source Development with CVS» frei verfügbar, wo Interessierte weitere Informationen über CVS finden können.
CVS *(Concurrent Versions System)* wird allmählich von Subversion (SVN), einer modernen Neuentwicklung, abgelöst. CVS weist einige Schwachstellen auf, beispielsweise ist es nicht möglich, ein Verzeichnis umzubenennen, sondern das alte als gelöscht markiert und das neue hochgeladen werden. Für Eclipse existieren bereits Plugins für Subersion, in der Standardversion unterstützt es nur CVS.

Vorteile, denn es ist nicht nur für einen, sondern für mehrere Benutzer bzw. Entwickler gedacht. Das heisst, an einem Projekt können mehrere Entwickler gleichzeitig arbeiten. Damit das funktioniert, müssen einige Dinge gewährleistet sein. Das Wichtigste ist eine History, damit man – wie bei einem Wiki – alte Versionen wiederherstellen kann. So kann man schauen, wie sich das Projekt entwickelt, und natürlich auch Fehler suchen. Manchmal baut jemand einen Fehler ein, der im schlimmsten Fall dazu führt, dass das Programm nicht mehr richtig oder sogar gar nicht mehr funktioniert. Dann muss man zu älteren, noch funktionierenden Versionen zurückgreifen können.

Wenn man lokal etwas am Quelltext des Programmes ändert, muss man die geänderte Version «committen», das heisst auf dem Server aktualisieren. Der Quelltext bekommt nun eine neue Versionsnummer. Wenn eine andere Person etwas geändert hat, dann kann die lokale Kopie über «update» aktualisiert werden (und prüfen, ob noch alles funktioniert …). Dabei wird die lokale alte Version überschrieben. Falls zwei Personen gleichzeitig etwas ändern, muss die zweite Person, die die Änderungen hochladen will, die Konflikte manuell lösen. Dies kann teilweise recht anspruchsvoll sein, da man die Änderungen des ersten Programmierers verstehen muss, um sie in die eigene Version einbauen zu können. Eclipse bietet bei Konflikten ein Vergleichsfenster, wo beide Dateien nebeneinander verglichen werden und die unterschiedlichen Textstellen hervorgehoben werden.

CVS lässt sich also durchaus mit einem Wiki – wie beispielsweise der Wikipedia – vergleichen. Die Idee dahinter ist die selbe: Mehrere Autoren oder Programmierer arbeiten zusammen an einem Text (bzw. Quellcode). Es spielt keine Rolle, wo man sich befindet, und, da die Computerwelt sehr englischorientiert ist, auch nicht, welche Muttersprache man spricht. Kommentare (innerhalb des Quelltextes) werden meist in Englisch geschrieben, der Rest (Java selbst) ist nicht mehrsprachig. So kann jeder, der Lust hat, an einem Projekt mitarbeiten – wenn er es denn findet.

Jetzt habe ich mein Projekt bei sourceforge.net[10] gehostet. Dort stehen mir nicht nur ein CVS-Server zur Verfügung, ich kann etwa auch eine Internetseite zum Projekt erstellen oder neue Versionen raufladen. Weiter stehen zum Beispiel ein News-System, RSS-Feeds, ein Forum für Diskussionen, ein Bereich zur Meldung von Bugs oder Wünschen zur Verfügung.

Design

Die Gestaltung der Internetseite erfolgt mit CSS. CSS, Abkürzung für Casading Style Sheet, ist eine Sprache, die nur dazu dient, das Aussehen und die Anordnung einzelner

10 Sourceforge.net ist die grösste Seite für OpenSource-Entwicklung im Internet. Momentan sind über 100 000 Projekte gehostet (unter anderem FileZilla, Gaim, Inkscape und phpMyAdmin) und über eine Million Entwickler registriert. Mein Projekt kann man unter http://sf.net/projects/wiki2xhtml finden.

Elemente zu beschreiben, nicht aber deren Inhalt. So muss zum Beispiel der Titel einer Überschrift in der HTML-Datei stehen. Welche Schriftart verwendet werden soll, ob der Text fett sein soll oder gar durch ein Bild ersetzt, wird alles mit Hilfe von CSS beschrieben. Natürlich kann man das auch teilweise mit HTML machen. Will man aber für eine Überschrift die Schriftart ändern, benötigt man (für jede einzelne Überschrift) folgenden Code:

```
<font family="Verdana"><h2>Überschrift</h2></font>
```

Bei mehreren Überschriften (auch der selben Ordnung) muss das jedes Mal separat angegeben werden. Dies hat eine erhöhte Dateigrösse (und auch weniger Übersicht) zur Folge und der Traffic (Datenverkehr) steigt an – was schlussendlich auch Geld kostet. Mit CSS kann man nun einfach sagen «Schreibe alle h2-Überschriften in Verdana». Der Code kommt dabei vorzugsweise in eine separate Datei (es ist auch möglich, den CSS-Code direkt in die Dateien zu schreiben, was aber nur in speziellen Fällen sinnvoll ist). Will man nun irgend ein kleines Detail ändern (die Schrift überall ein Punkt kleiner machen), braucht man so nur noch diese eine CSS-Datei abzuändern und – je nach Grösse der Seite – nicht jede einzelne der womöglich hunderten bis tausenden HTML-Dateien. Der Code für obiges Beispiel würde übrigens so ausschauen:

```
h2 { font-family: Verdana; }
```

Damit kann man im HTML-Code das veraltete font-Tag weglassen:

```
<h2>Überschrift</h2>
```

Erstetzt man im CSS-Code das «h2» durch «h1, h2, h3, h4, h5, h6», gilt die dafür angegebene Gestaltung für alle (der sechs erlaubten) Überschriftenebenen. Die CSS-Datei selber kann natürlich auch recht gross werden, im Extremfall sogar grösser als eine HTML-Seite, falls diese sehr wenig Text enthält und wenige der im CSS beschriebenen Elemente bennutzt. Trotzdem müssen weniger Daten übertragen werden, denn die CSS-Datei bleibt – zumindest für ein paar Tage, je nach Einstellung – im Cache des Browsers und muss nicht bei jedem Klick auf eine andere Unterseite neu geladen werden – falls diese die selbe Datei verwendet.

Ein weiterer Vorteil dieser Dateiauslagerung: Man kann ganz einfach ein Design durch ein anderes ersetzen. Da die dazu benötigten Dateien meist – wie auch in meinem Programm – in einem separaten Ordner gespeichert werden, braucht man nur den Ordner auszutauschen, um zu schauen, wie ein anderes Design auf die Seite wirkt oder sogar, wie die Seite unformatiert aussähe, wenn man die Dateien ganz weglässt. Das ist eine extreme Erleichterung: So kann jeder Designs für mein Programm bzw. die davon generierte Internetseite schreiben und es weiter verteilen, so dass es jeder nutzen kann. Müsste man dazu den Quelltext der Seiten ändern, wäre dies beinahe unmöglich. Eine Datenbank aus verschiedenen CSS-Dateien (und dazugehörigen Bildern etc.) ist also durchaus denkbar.

Eine weitere Möglichkeit von CSS ist die gerätespezifische Formatierung. Es wird zum Beispiel unterschieden zwischen Bildschirm, Screenreader, Drucker und Handy. Je nach Seiteninhalt wird eine Seite vielleicht sehr oft ausgedruckt. Wenn hier noch das seitenlange Menu, der unansehnliche und störende Banner, das störende Hintergrundbild mitgedruckt werden, gibt das natürlich allen Anlass zur Freude. Aus Liebe zur Natur und dem Benutzer lassen sich der Papierabfall und die Tonerkosten reduzieren, indem man für den Drucker die unerwünschten Teile einfach weglässt. Dafür kann man vielleicht auf eine bei längeren Texten besser lesbare Serifenschrift wechseln oder am Schluss eine Bemerkung wie ein «First Edition» oder meine Kontonummer sichtbar machen.

Arbeitsweise des Programmes

Ich finde es immer wieder erstaunlich, wie das menschliche Gehirn Zusammenhänge sieht. Die Beschreibungen, die in der Wikipedia-Syntax verwendet werden – zum Beispiel für Listen – sind von Hand ganz einfach in HTML-Code umformbar, wenn man die Regeln kennt. Gerade bei diesem Beispiel: Hier reicht ein Überfliegen der eigentlich sehr simplen Regeln, wie Listen verschachtelt werden. Um dies dem Computer beizubringen, dauert es ungleich länger (dafür ist er später auch viel schneller bei deren Umsetzung – sonst wäre die Arbeit an einem solchen Programm recht sinnlos). Erstaunlich, dass man dabei manchmal Neues dazulernt, denn der Computer muss jedes Detail kennen und richtig umzusetzen wissen. Einige oder sogar die Mehrzahl dieser Details findet man meist erst dann, wenn man das Programm auf eine möglichst unmöglich komplizierte und sinnlose Liste loslässt, da so die meisten Spezialfälle auftreten. Durch die eventuell fehlerhafte Ausgabe kann der Quelltext dann korrigiert werden.

Für Listen muss mein Programm jede einzelne Zeile durchgehen und sich die Listentiefe wie auch die Listenstruktur merken. Denn je nach Art des Listenpunktes (geordnet/ungeordnet) muss sich das Programm anders verhalten und dabei auch berücksichtigen, dass eventuell gar kein Listenpunkt geöffnet oder geschlossen werden muss, wenn man eine Ebene überspringt. Ein Beispiel:

```
* Dem ersten Listeneintrag folgt direkt ...
*** ... ein Listenpunkt auf der dritten Ebene.
```

Im HTML-Code würde dies folgendermassen aussehen:

```
<ul><li>Erster Listeneintrag wird geöffnet
    <ul>
        <ul><li>Dritter Listeneintrag</li></ul>
    </ul>
</li></ul>
```

Hier sieht man sehr schön, dass für die zweite Ebene kein <li> geöffnet werden darf. Man muss also wirklich jede Zeile separat verarbeiten und kann nicht einfach für jede tiefere Ebene ein ul- und ein li-Tag anhängen.

Glücklicherweise gibt es auch einfachere Dinge im Wiki-Code. Ein typisches Beispiel ist die Kursivschrift: In einem solchen Fall kann man mit regulären Ausdrücken arbeiten, also mit bestimmten Buchstabenmustern, nach denen gesucht werden soll. In diesem Fall – da kursiver Text von je zwei Apostrophen («''kursiv''») umschlossen werden muss – würde das Muster lauten: «Zwei Apostrophe, gefolgt von irgendwelchen Zeichen ausser Apostrophen, nach denen wieder zwei Apostrophe kommen» (Diese Regel könnte man natürlich noch verfeinern). In einem regulären Ausdruck ausgedrückt wäre das «''(^')+''». In Java bekommt man mit folgendem Befehl das «Pattern», das Muster, mit dessen Hilfe man dann im Text suchen und ersetzen kann:

```
Pattern.compile("''(^')+''");
```

Der Ausdruck innerhalb der Klammer steht hier für irgendein Zeichen ausser dem Apostroph: Ein «^» am Anfang der Klammer negiert den Inhalt. Würde es fehlen, träfe der Ausdruck nur auf einen Apostroph zu und kein anderes Zeichen. Da aber nicht ein einziges Zeichen gesucht wird, das kursiv sein soll, muss noch eine Mengenangabe erfolgen, was hier mit dem «+» geschieht: Es bedeutet, dass das vorherige Zeichen mindestens ein Mal vorkommen muss. Eine Grenze nach oben ist nicht gegeben. Ohne dieses Zeichen würde man mit diesem Ausdruck nur einzelne kursive Zeichen erkennen (Der von den Wildcards[11] bekannte Stern («*») würde hier bedeuten, dass das Zeichen mindestens ein Mal oder auch gar nicht vorkommen kann, ist also nicht brauchbar). Findet das Programm nun eine solche Stelle, schreibt es an den Anfang des Fundortes das öffnende Tag für Kursivschrift (je nachdem ein «<i>» oder ein «»), löscht dann die zwei darauf folgenden und die letzen beiden Zeichen (also die Apostrophe) und hängt das schliessende Tag hinten an.

Was man hier schlussendlich erhält, ist zwar HTML-Code, aber nicht eine fertige und brauchbare HTML-Datei. Dazu fehlen noch einige Angaben, unter anderem auch das HTML-Tag und der Header, wo zum Beispiel der Seitentitel angegeben wird. Hier kommt nun der nächste Schritt meines Programms, der erlaubt, abhängig vom gewählten Design verschiedene Dateien zu erzeugen. Dazu benögigt es ein «Gerüst» der HTML-Datei, die ich in meinem Beispiel «reck.html» genannt habe. Hier ersetzt das Programm bestimmte Stellen durch Text oder Code.

Will man zum Beispiel mit CSS den Inhalt und das Menu separat formatieren, schreibt man davor ein und dahinter ein abschliessendes Div-Tag ( ), mit dem die jeweiligen Abschnitte etwa eine verschiedene Hintergrundfarbe erhalten oder Listenpunkte im Menu gar nicht angezeigt werden. Ein Ausschnitt aus dem Gerüst:

---

11 Wildcards werden oft für Dateiauswahlen benutzt: Ein `*.txt` steht zum Beispiel für alle Dateien, die mit .txt enden, *.* für alle Dateien, die einen Punkt enthalten.

wiki2xhtml im Textmodus unter Linux: Die Datei help.html wird aus help.txt generiert

```
<div id="menu">
 [MENU]
</div><!-- end menu -->
<div id="content">
 [CONTENT]
</div><!-- end content -->
```

Hier setzt das Programm nun das Menu und den Inhalt an der jeweiligen Position ein. Dies ist möglich, weil das Menu und der Inhalt separat generiert werden. Diese Teilung ermöglicht es nun, dem Text im Menu und im Inhalt verschiedene Formatierungen zuzuweisen, indem man für die jeweilige ID (z. B. menu) mit CSS diese Eigenschaften beschreibt. Die ID ist einmalig; es ist nicht erlaubt, innerhalb einer Seite zwei Elemente mit der selben ID zu belegen. Denn sie erfüllen ausserdem noch eine weitere Funktion: Sie sind Sprungmarken. Das heisst, ich kann zum Beispiel einer Überschrift die ID «#1» geben. Gebe ich diesen Anker nun in einem Link an, indem ich die ID («#1») an den Dateinamen anhänge, springt der Browser automatisch zur gewünschten Überschrift.

Will ich nun im Menu z. B. statt Punkten ein Bild einsetzen, das vor jedem Menupunkt erscheint – denn das Menu ist als Liste definiert –, sieht das so aus:

```
#menu ul {
    list-style-image: url(list.png);
}
```

Damit wird der Listenpunkt durch das Bild «list.png» ersetzt. Ohne diese speziellen div-Tags wäre dies nur recht umständlich möglich, da man das Menu kaum vom Inhalt trennen könnte. So kann man aber, falls gewünscht, auch die Struktur der Datei verändern oder das Menu (vorzugsweise wenn es horizontal ausgerichtet ist) ein zweites Mal aufführen, etwa am Ende der Seite, damit der Benutzer nicht immer nach oben scrollen muss. Diese Flexibilität soll auch helfen, wenn man neue Designs schreibt, denn so ist die Datei nicht nur auf ein einziges Stylesheet angepasst.

Die Einteilung in verschiedene Abschnitte – title, header, menu, content, footer – ist nicht der einzige Grund, warum dieses HTML-Gerüst nicht schon im Code des Programms steht, sondern selber geschrieben und geändert werden kann. Dies hat auch weitere Gründe:

Einbindung von Dateien: Man kann mehr als eine CSS-Datei in die Seite einbinden. Oft genutzt wird das separate CSS-File für den Druck (was auch sinnvoll ist), wo der Text vielleicht eine Serifenschrift bekommt – die bei längeren Texten besser lesbar ist – oder das Menu unsichtbar gemacht wird. Natürlich kann man auch für andere Medien CSS-Dateien schreiben: Vom Braille-Terminal übers Handy bis zum Screenreader. Nicht jeder will das aber oder hat die Zeit dafür, darum werden diese Dateien manuell angegeben. So kann man nicht nur Dateien weglassen, sondern bei Bedarf auch weitere hinzufügen. Vielleicht will jemand eine Seite speziell auch für Handys entwickeln und will darum eine weitere Datei einbinden. JavaScript-Dateien können eine weitere Möglichkeit darstellen. Hier muss man aber eventuell etwas Handarbeit anlegen, damit sie auch funktionieren, falls das Programm Anweisungen falsch versteht.

Codierungsangabe: Normalerweise sollte hier UTF-8[12] verwendet werden, da dies mit Sonderzeichen am wenigsten Probleme macht. In UTF-8 ist eine Codierung für Unicode-Zeichen. Die ersten 128 Zeichen werden hier immer noch mit einem Byte angegeben, Sonderzeichen, spezielle Schriftzeichen und weitere Spezialzeichen können bis vier Byte belegen. Daraus ergibt sich dann einen theoretischen Zeichenumfang von über zwei Millionen Zeichen. Bisher gibt es aber noch keine Schriftart, die den gesamten Umfang nutzt. Gibt man in einem UTF-8-Kodierten Text eine ISO-Norm an, beispielsweise die hier gebräuchliche ISO-8859-1, werden die Sonderzeichen falsch ausgegeben. In diesen ISO-Codierungen haben alle Zeichen 8 bit (1 Byte). Somit werden zwei Byte grosse Unicode-Zeichen in der ISO-Codierung als zwei Zeichen aufgefasst. Ein ä wird hier zum Beispiel zu einem «Ã¤», ein ö zu einem «Ã¶», ein ü zu «Ã¼», und das Anführungszeichen « zu einem «Â«». In speziellen Fällen will man aber eine ISO-Norm verwenden – vielleicht ganz einfach darum, weil man keine Unicode-Dateien schreiben kann. Auch hier kann man, wie bei einem Stylesheet, mit einer kleinen Modifikation alle Seiten ändern. Natürlich muss man hier den Wiki-Code noch «parsen», damit die Änderungen auch für alle HTML-Dateien übernommen werden.

---

12 Die Wikipedia weiss zu diesem Thema etwas mehr. Weiterführende Informationen für Interessierte findet man unter http://de.wikipedia.org/wiki/UTF-8.

13 Das «World Wide Web Consortium», abgekürzt W3C, hält zu diesem Thema weitere Informationen und Richtlinien unter http://www.w3.org/WAI/ bereit. Da das W3C laut Wikipedia keine «zwischenstaatlich anerkannte Organisation» ist, kann sie nur Empfehlungen bekanntgeben, aber keine Standards vorschreiben. Das W3C wurde von Tim Berners-Lee, der als Begründer des Internet gilt, gegründet.

Ein Überblick über die verschiedenen Punkte steht unter http://www.w3.org/WAI/WCAG20/quickref/ bereit.

Barrierefrei?

«Barrierefreiheit»[13↑], das ist ein Schlagwort, das man in letzter Zeit immer öfters hört und liest. Das Ziel ist, dass jeder auf eine Internetseite zugreifen kann. Das klingt vielleicht etwas banal, ist es aber ganz und gar nicht, denn man muss eine ganze Menge Dinge berücksichtigen. Einige davon habe ich bereits schon unter dem Abschnitt «CSS» angesprochen. Damit aus dieser Arbeit nicht ein Buch wird, will ich nur ein paar Beispiele zeigen.

Bilder: Nicht alle können Bilder sehen, sei es, weil man sehbehindert ist oder weil man mit einem Textbrowser, zum Beispiel Lynx, unterwegs ist. Darum sollte bei Grafiken immer ein Alternativtext angegeben werden. Man kann an Stelle des Bildes einfach eine kurze Beschreibung setzen lassen (einen Alternativtext, der dann erscheint, wenn die Grafik nicht angezeigt oder geladen werden kann) oder einen Link auf eine Beschreibungsseite einfügen. Letzteres eignet sich eher bei Bildern oder Grafiken, die zum Verständnis des Inhaltes einer Seite beitragen, ersteres wird zum Beispiel als Ersatz für grafische Links verwendet. Es ist besonders ärgerlich, wenn auf einer Seite nur Grafiken für die Navigation vorhanden sind, aber kein Alternativtext. Denn so ist es eine Zumutung, bei abgeschalteten oder nicht sehbaren Grafiken zu navigieren.

Navigation: Die Navigation sollte im Quellcode grundsätzlich an erster Stelle, also vor dem eigentlichen Inhalt, stehen. Für eine Übersicht über die Seitenstruktur will man sich nicht zuerst den ganzen Seiteninhalt vorlesen lassen, bis man zum Menu kommt. Auch dies betrifft vor allem Blinde. Ebenfalls sinnvoll ist ein Link zum Anfang des Inhaltes, den man vor dem Menu platziert, damit man das Menu überspringen kann: Es ist genauso ermüdend, wenn man auf jeder Seite immer zuerst das Menu vorgelesen bekommt, wenn man sich doch eigentlich für den Inhalt interessiert.

Farben: Gewisse Farbkombinationen sind von Grund auf sehr mühsam für das Auge, ein Beispiel ist die Interdiscount-Seite mit schwarzer Schrift auf rotem Hintergrund. Am besten lässt sich meist dunkle Schtift auf hellem Hintergrund lesen. Weitere Farbkombinationen, die man unterlassen sollte, sind solche, die Menschen mit Farbenblindheit nicht unterscheiden können. Ich kann zum Beispiel orange Schrift auf hellgrünem Hintergrund kaum lesen. Eine Internetseite, die mit diesen Farben arbeitet, werde ich kaum näher betrachten, egal wie spannend der Inhalt ist.

Kompatiblität: Eine Internetseite sollte in allen Browsern mehr oder weniger gleich aussehen. Mit dem Erscheinen des IE 7 hat Microsoft endlich eine der grössten Schwachstellen aus dem Weg/Web geräumt; dessen Umsetzung von CSS lässt aber oft immer noch zu wünschen übrig. Ausserdem nutzen immer noch ein grosser Teil der PC-Besitzer den IE 6. Eine Seite sollte also zumindest für die zweitneuste Version mehr oder weniger korrekt angezeigt werden. Eine Optimierung für einen bestimmten Browser geschieht normalerweise nur aus Bequemlichkeit, damit man sich bei den schlechteren Browsern nicht mit diversen Hacks und Spezialfällen herumschlagen muss, wie es vor allem beim

IE 6 der Fall war, aber wenn die Seite ein breites Publikum ansprechen soll, dann muss man sogar das auf sich nehmen. Eine Internetseite sollte ausserdem auch dann angezeigt werden, wenn man keine Plugins (z. B. für Flash) installiert oder JavaScript deaktiviert hat. Da gewisse Dinge ohne JavaScript nicht gut funktionieren, ist ein Hinweis sinnvoll, wenn man dies deaktiviert hat.

Internetanschluss: Noch lange nicht jeder hat ADSL und muss darum eventuell sehr lange warten, bis eine Seite aufgebaut wird. Je mehr Grafiken man verwendet, desto grösser ist die Gefahr, dass die Ladezeiten zu gross sind und der Benutzer die Seite verlässt. Bilder sollte man fürs Internet optimieren, indem man zum Beispiel bei Vorschaubildern eine angemessene Kompression wählt und so die Dateigrösse minimal hält.

Man kann natürlich noch viel mehr beachten als diese Punkte. Es stellt sich dann aber auch die Frage, ob das überhaupt sinnvoll ist, wenn man nur eine kleine Website für Kollegen betreibt, die weniger für die Öffentlichkeit bestimmt ist. Bei «wichtigeren» Seiten wie beispielsweise die einer Bank muss man selbstverständlich etwas mehr darauf achten. In meinem Programm konnte ich bisher nur einen Punkt wirklich beachten: Die Position des Menus. Die restlichen Dinge sind oft auch abhängig von der verwendeten CSS-Datei.

Schlusswort

Ich bin zufrieden mit dem Ergebnis und hoffe, dass es in Zukunft noch vielen Anwendern helfen wird und etwas Zeit ersparen kann. Es hat Spass gemacht, am Programm zu schreiben, und ich habe viel dazugelernt – nicht nur in Java. Mittlerweile hatte ich mich in den unzähligen Stunden endlich in Linux eingelebt, Windows starte ich nur noch für Spiele oder um zu schauen, ob das Programm auch dort ordnungsgemäss funktioniert.

Es ist schön zu sehen, dass wiki2xhtml mehr kann, als ich ursprünglich vorgesehen habe, etwa das Navigationsmenu, Tabellen, die Bildergalerie oder das GUI waren nicht geplant. Überhaupt funktioniert das Programm besser als erwartet, besser sogar als einige teure kommerzielle Produkte, was mich natürlich freut – und auch etwas nachdenklich macht …

Eine Funktion, die ich nochmals hervorheben möchte, ist die, nur den HTML-Code ohne das Gerüst (also ohne die Tags html, head, body etc.) zu erstellen. Es kann also nicht nur zum Erstellen von Internetseiten verwendet werden, sondern kann auch bei anderen Programmen eine Hilfe sein.

Da wiki2xhtml nun bei sourceforge.net gehostet ist, kann nach Abschluss meiner Arbeit jeder Interessierte daran mitschreiben, und es ist weiterhin für jeden abrufbar und nicht im Archiv versteckt ;) Ich habe vor, es noch weiter zu entwickeln, noch ein paar zusätzliche Designs hinzuzufügen und vielleicht auch mal eine abgeänderte Version zu schreiben, die statt HTML-Dateien fertige LaTeX-Dateien erstellt. Damit könnte man das Erstellen von WikiReadern[14] über LaTeX etwas vereinfachen.

Nebenbei habe ich beim Schreiben meiner Arbeit auch Scribus kennen gelernt. Scribus ist ein OpenSource-DTP-Programm, das sich zum Beispiel mit Adobe InDesign vergleichen lässt. Mit Desktop-Publishing-Programmen werden unter anderem auch Plakate, Magazine oder Bücher erstellt, darum war diese Erfahrung zusätzlich spannend.

Bei meinem Projekt haben mir viele Leute geholfen, darum gehe ich über zum …

Dank

Ich möchte hier noch einigen Personen danken, die mir bei meiner Arbeit geholfen haben. Dies sind:

- **Tsoots, mrdocs, avox** und **a_l_e** aus dem IRC-Kanal #scribus bei Freenode.org, für die Tipps zu Scribus
- **Daniel-S-P** aus dem IRC-Kanal #inkscape, der mich auf Scribus gebracht hat
- **rcjsuen** aus dem IRC-Kanal #eclipse (auch bei Freenode.org), für die Hilfe zu CVS und Sourceforge.net
- **Gerrit** (aka[15] Nil18; aus dem Diabolo-Forum, http://www.diabolo666-board.com) aus Wien und **Nicole Graf** aus meiner Klasse, für die Korrekturlesung
- **Matthias** (aka Murray; aus dem Java-Forum, http://java-forum.org), für die Tipps zur Unterscheidung von Binär- und Textdateien
- **RoEn, slam** und **devil** aus dem IRC-Kanal #sidux bei oftc.net, die mich für die Funktion zum Weglassen des HTML-Gerüstes angeregt haben
- **Michael Gasperl** (aka Migas bei der deutschen Wikipedia), für die Tipps beim GUI und die Korrekturlesung
- Natürlich **Jürg Strassmann,** für die vielen Tipps zum Text und die Anregungen und die Betreuung meiner Arbeit
- Mein Vater **Markus Eugster,** für den Ausdruck der Arbeit
- **Theophil Graf** aus Windisch, für die Materialien und Tipps zum Binden der Arbeit
- Alle Mithelfenden von **http://dict.leo.org**, die mit dieser Internetseite zahlreiche Übersetzungen von Deutsch nach Englisch geliefert haben

14 Siehe http://de.wikipedia.org/wiki/Wikipedia:WikiReader

15 «aka» steht für «also known as», auf Deutsch «auch bekannt als». Oft verwendet im Internet.

Quellen

- Björn Seibert, Manuela Hoffmann: Professionelles Webdesign mit (X)HTML und CSS. Galileo Comoputing, Bonn, 2006.
  Dieses Buch hat mich quer durch das Thema Webdesign begleitet. Tiefgehend und interessant, auch weil auf verschiedene Aspekte eingegangen werden: Barrierefreies Web, gutes XHTML und CSS, Typografie etc.
- http://java.sun.com/docs/books/tutorial/essential/regex/ – «Lesson: Regular Expressions (The Java™ Tutorials > Essential Classes)»
  Ein Tutorial von Sun zu regulären Ausdrücken, die ich für diverse Befehle verwendet habe. Für Interessierte empfehlenswert.
- http://java.sun.com/j2se/1.5.0/docs/index.html – «JDK 5 Documentation»
  Die Dokumentation zu Java 5 selber – ein riesiges Nachschlagewerk, das alle möglichen Befehle und Funktionen etc. beschreibt, in dem man aber trotz der Grösse noch einige Dinge finden kann.
- http://www.galileocomputing.de/openbook/javainsel6/ – «Galileo Computing : Buch : Java ist auch eine Insel»
  Das Buch «Java ist auch eine Insel» umfasst momentan über 1000 Seiten, ist aber auch kostenlos als «Openbook» (eBook) erhältlich. Ein ideales Nachschlagewerk, das vieles näher erklärt und gute Tipps auf Lager hat. Nebenbei: Java ist – neben der Programmiersprache und der Insel – auch eine starke Kaffeebohne (daher das Java-Logo) oder ein Tanz aus den 20er-Jahren.
- http://www.css4you.de/ – «CSS 4 You - The Finest in Stylesheets»
  Diese CSS-Referenz enthält zu allen CSS-Eigenschaften die möglichen Werte und zeigt auch eventuelle Inkompatiblitäten mit verschiedenen Browsern auf.
- http://www.w3schools.com/ – «W3Schools Online Web Tutorials»
  Auch hier habe ich wieder ein Nachschlagewerk verwendet: http://www.w3schools.com/tags/. Hier sind alle HTML-Tags aufgelistet, aber auch die Entities wie – (das steht für den n-dash, den Gedankenstrich), die man zur Eingabe von Sonderzeichen verwenden kann.

Bestätigung der Eigentätigkeit

«Der Unterzeichnete bestätigt mit seiner Unterschrift, dass die Arbeit selbstständig verfasst und in schriftliche Form gebracht worden ist, dass sich die Mitwirkung anderer Personen auf Beratung und Korrekturlesen beschränkt hat und dass alle verwendeten Unterlagen und Gewährspersonen aufgeführt sind. Er weiss, dass die erstellte Arbeit Eigentum der Schule ist und dass eine Veröffentlichung oder Weitergabe der Zustimmung von Autor, Betreuer und Schulleitung bedarf.»

Simon Eugster

Kurzzusammenfassung

Ich hatte das Ziel, ein Programm zu schreiben, das für Anfänger leicht zu bedienen ist und mit wenig Aufwand barrierefreie, den Standards entsprechende und gut aussehende Internetseiten erstellen kann. Als Programmiersprache verwendete ich Java, hauptsächlich aus dem Grund, weil es die einzige «wirkliche» Programmiersprache ist, die ich mehr oder weniger beherrsche, aber auch wegen der Plattformunabhängigkeit; Ich arbeite recht viel mit Linux, wollte aber nicht nur für ein Betriebssystem programmieren.

Eine aktuelle Version meines Programmes «wiki2xhtml» ist im Internet unter http://sourceforge.net/projects/wiki2xhtml zu finden. Es verlangt Textdateien als Eingabe und generiert daraus (X)HTML-Dateien, die mit CSS formatiert werden. Komplexere Seiten mit einem Menu zur Navigation, Galerien, Listen und Tabellen sind kein Problem; die mitgelieferte Hilfedatei erklärt (beinahe) alle Möglichkeiten.

Betreuer: Jürg Strassmann
Verfasser: Simon Eugster