ZML

Das Zeitung-Markup-Language-System
Version 2.0

Torsten Bronger

Die Anleitung erläutert das ZML-System, mit dem man einfache Publikationen, die im weiteren Sinne Zeitungen darstellen, sowohl im Druck als auch online publizieren kann. Im speziellen wurde ZML für die Kármán Hochschulzeitung (http://www.karman-aachen.de/) der RWTH Aachen erstellt.

Inhaltsverzeichnis

1 Überblick

Eine ZML-Installation besteht aus zwei Verzeichnisbäumen: Zum einen benötigt man ZML selber. Man bekommt es auf Sourceforge (http://zeitung-ml.sourceforge.net/).

Ferner braucht man einen Verzeichnisbaum mit der Zeitung, das sogenannte „Zeitungs-Verzeichnis“. Ebenfalls auf Sourceforge kann man mit ‘karman-subset ’ eine Beispiel-Zeitung herunterladen. Die Struktur dieses Zeitungs-Verzeichnisbaum ist wichtig, daher sollte man sich das Beispiel genau anschauen.

Im folgenden gehe ich davon aus, daß beide Verzeichnisse dasselbe Mutter-Verzeichnis haben, also direkt nebeneinander liegen. Ist das nicht der Fall, muß man die relativen Pfade, mit denen man die ZML-Programme aufruft, entsprechend anpassen.

Theoretisch ist es möglich, ZML auch auf einem Windows-System zu benutzen. Allerdings dürfte das in einfacher Weise nur mit Cygwin möglich sein. Als Alternative kämen DJ Delories Kompilationen der GNU-Werkzeuge in Betracht. Trotzdem: Am besten funktioniert ZML unter Linux. Dort kann man (fast) draufkopieren und loslegen, see Kapitel 2 [Installation] auf Seite 9.

1.1 Die Zeitungs-Artikel

Neue Artikel werden für ZML im Verzeichnis ‘artikel/ ’ als XHTML-Dateien abgelegt. Ihr Aufbau ist recht simpel, trotzdem kann man das Anlegen eines neuen Artikels noch einfacher gestalten, indem man einen bestehenden Artikel als Ausgangspunkt nimmt.

Im Falle von Kármán werden die meisten Artikel in einem PHP-Wiki angelegt. Es existiert ein einfaches Umwandlungsprogramm, das die Wiki-Quellcodes in XHTML umwandelt. Allerdings ist Nachbearbeitung notwendig.

Wichtig: Die endgültigen Versionen der Zeitungsartikel sind die XHTML-Dateien, nicht die Wiki-Dateien. Die werden nach dem „Einfrieren“ weggeworfen, oder zumindest nicht mehr gebraucht. Weitere Änderung müssen von da ab in der XHTML-Version gemacht werden.

1.2 Der Generator für die Druckausgabe

Eine bestimmte Ausgabe der Zeitung wird gedruckt, indem eine ZML-Datei mit dem Namen ‘ausgabe- nummer.xml ’ im ZML-Wurzelverzeichnis angelegt wird und

```
../zml/tools/zmltopdf.py nummer
```

aufgerufen wird. Das Ergebnis ist eine PDF-Datei, die die aktuelle Ausgabe enthält. Dazu wird LaTeX angeworfen.
In der ZML-Datei wird das Drucklayout der Ausgabe festgelegt. Dabei sind alle Artikel Rechtecke, die auf die Kolumnen verteilt werden. Dieser Vorgang ist leider nicht automatisierbar. Das Erstellen der ZML-Datei ist ein wenig Übungssache. Natürlich geht man meist von der Datei für die vorherige Ausgabe aus und ändert sie.

1.3 Der Generator für die Webseiten

Die Webseiten einer ZML-Zeitung sind statisch. Das bedeutet, daß kein PHP oder ähnliches zum Einsatz kommt, sondern nur HTML-Dateien und Bilder. Das hat Vor- und Nachteile. Die Vorteile sind:

- Nahezu keinerlei Ansprüche müssen an den Web-Provider gestellt werden
- Man hat die volle Kontrolle über die Seiten, insbesondere über ihre Qualität in Bezug auf Standard-Konformität und Barrierefreiheit
- Die Programme, die die Seiten erstellen, sind verhältnismäßig einfach und alles kann offline getestet werden
- Der Verzicht auf ausgewachsene CMS-Systeme hält die Seiten einfach, was sowohl dem Leser als auch dem Webmaster zugute kommt.

Die Nachteile sind:

- Die vielen Zusatzfunktionen von CMS-Systemen können nicht genutzt werden.
- Die Layout-Flexibilität von CMS-System kann nur zum Teil mit ZML erreicht werden.
- Kollaboratives Arbeiten an der Webseite ist mit ZML nicht möglich. Nur diejenigen, die ZML bei sich installieren (und Zugang zum Webserver haben), können die Webseiten modifizieren.

Ist alles sauber installiert, ist das Erzeugen der Webseiten ein Kinderspiel. Man ruft lediglich aus dem Zeitungs-Wurzelverzeichnis

```
../zml/tools/make-karman-website.py
```

auf und wartet ab. Nach einiger Zeit ist die aktualisierte Webseite im Unterverzeichnis ‘webpages/ ’ verfügbar und kann hochgeladen werden, See Abschnitt 2.1.7 [Sitecopy] auf Seite 11. Letzteres geschieht automatisch, wenn man ‘make-karman-website.py ’ die Option ‘--upload ’ übergibt.
Die Webseite besteht im groben aus folgenden Teilen:

1. die Artikel, die die Zeitung ausmachen
2. die Bilder, die in den Artikeln gezeigt werden
3. die sogenannten *Snippets*, das sind alle Webseiten, die keine Artikel sind (z.B. Begrüßungsseite, Impressum etc)
4. der sogenannte *Pool*, in dem alle Bilder und sonstigen Dateien sind, die für die Webseite benötigt werden, aber nicht zu Artikeln gehören
5. die Newsfeeds (RSS und Atom)

2 Installation

Hier beschreibe ich nur die Installation unter Linux. Leute, die tatsächlich vorhaben, ZML unter Windows zu benutzen, werden die Informationen in diesem Kapitel sicherlich leicht auf Windows übertragen können. Es gibt keinen Grund, warum ZML nicht unter Windows laufen sollte, aber probiert hat's noch niemand.

2.1 Voraussetzungen

2.1.1 LaTeX

LaTeX wird genutzt, um die Zeitung zu drucken, also in ein PDF umzuwandeln. Dabei ist jedes nicht zu alte handelsübliche LaTeX-System ausreichend. Es sollten allerdings folgende Pakete nicht fehlen:

geometry, colortbl, ragged2e, amsmath, pifont, latexsym, nicefrac, soul, relsize, pdfcolmk und pdfcprot.

Die meisten dieser Pakete sind bei einer guten LaTeX-Installation bereits vorinstalliert. Im Prinzip bekommt man fehlende Pakete auf dem CTAN-Server (http://www.dante.de/software/ctan/), allerdings kann ich hier nicht erklären, wie man LaTeX-Pakete installiert.

2.1.1.1 Schriftarten

Schriftarten sind unter LaTeX leider ein besonderes Problem. ZML benötigt glücklicherweise nur zwei davon, nämlich eine serifenlose und eine serifenbehaftete Schriftart. Für Kármán ist das die Syntax und die Times. Die Times ist auf jedem Computer vorhanden, die Syntax leider nicht. Das installieren einer Schriftart geschieht in drei Schritten, die ich hier leider nur kurz anreißen kann, um nicht den Rahmen zu sprengen:

1. Man bekommt die Schriftart in Form eines Haufens von Dateien, die man einfach die die Verzeichnisse kopiert, wo schon Dateien mit derselben Dateiendung vorhanden sind.

2. Man meldet die Schriftarten bei pdfTeX an, indem man die '.map'-Datei in die Datei 'updmap.cfg' einträgt.

3. Man läßt 'updmap' und 'texhash' laufen.

2.1.2 Java

Java sollte in keiner Linux-Installation fehlen. Ich weise daher nur darauf hin, daß neue Java-Versionen (größer gleich 1.4) erheblich schneller sind. Allerdings ist auch 1.4 mittlerweile lange genug auf dem Markt, daß niemand mehr mit etwas älterem unterwegs sein sollte.

Java wird für den Saxon benötigt, der in Java geschrieben ist.

2.1.3 Saxon

Der Saxon ist *das* Werkzeug für ZML. Sowohl die Umwandlung in die Druckform, als auch die Erzeugung der Webseiten wird vom Saxon erledigt. Er ist ein XSLT-Interpreter, und praktisch der gesamte Quellcode von ZML liegt in XSLT vor.

Wichtig ist, *nicht* die Versionen 8.*x* oder 7.*x* zu benutzen, sondern eine Version aus der 6er-Reihe. Ganz wichtig. Man bekommt den Saxon, sofern er nicht bei der Linux-Distri deiner Wahl dabei ist, unter http://saxon.sf.net.

2.1.4 ImageMagick

ImageMagick (http://www.imagemagick.org) sollte ebenso bei jeder Linux-Distri mitgeliefert werden. Ohne es sicher zu wissen, gehe ich davon aus, daß die Version sehr unkritisch ist. ZML benötigt das ‘convert ’- und das ‘display ’-Programm.

2.1.5 Python

Einige wichtige Funktionen von ZML, insbesondere ‘phpwiki2html.py ’ und das Make-Skript ‘make-karman-website.py ’, benötigen Python 2.4 (http://www.python.org) oder neuer. Zusätzlich zur Standard-Distribution braucht man noch die Python Image Library (PIL) (http://www.pythonware.com/products/pil/) und ElementTree (http://effbot.org/zone/element-index.htm). Letzteres ist ab Python 2.5 Teil der Standard-Distribution.

2.1.6 HTML Tidy

HTML Tidy (http://www.w3.org/People/Raggett/tidy/) ist ein sehr schönes und beliebtes Programm. Es erfüllt zwei Aufgaben: Einerseits überprüft es die Webseiten von Kármán darauf, daß sie dem XHTML-Standard gehorchen, und es macht andererseits die Quellcodes der Webseiten lesbarer und Browser-kompatibler. Sollte eine Webseite nicht dem Standard genügen, wird eine Fehlermeldung auf dem Bildschirm ausgegeben.

Sollte man nicht über dieses Programm verfügen, dürfte es trivial sein, es nachzuinstallieren, da es nur aus einer einzelnen kleinen ausführbaren Datei besteht.

2.1.7 Sitecopy

Sitecopy kann benutzt werden, um die Kármán-Webseiten auf den WebDAV-Webserver hochzuladen. Selbstverständlich kann man dafür auch jedes andere geeignete Programm einsetzen.

Früher wurde Sitecopy vom Make-Skript, das die Webseiten zusammenbaut, implizit ganz am Ende des Prozesses gestartet. Das ist nun nicht mehr so.

Achtung: Sitecopy ist ein recht tückisches (oder sollte ich sagen schlechtes) Programm. Insbesondere muß man darauf achten, daß man mit dem Aufruf ‘sitecopy -u ’ keine Teile der Website versehentlich löscht, bloß weil sie lokal nicht vorhanden waren.

Ich empfehle folgenden Eintrag in ‘~/.sitecopyrc ’:

```
site karman
    server www-users.rwth-aachen.de
    username ********
    password ********
    remote /karman/
    url https://www-users.rwth-aachen.de/karman
    local ~/xml/rwth-zeitung/webpages/
    protocol webdav
    http secure
    exclude /Galerie
    exclude /wiki
    exclude /Veranstaltungen
```

Damit kann man einigermaßen sicherstellen, daß nur die automatisch generierten Webseiten vom Sitecopy-Aufruf betroffen sind. Allerdings sollte man trotzdem das Sitecopy-Manual gelesen und verstanden haben.

Wenn man Sitecopy implizit mit ‘make-karman-website.py ’ aufruft, indem man die Option ‘--upload ’ übergibt, wird der Name des Zeitungs-Verzeichnisses als Sitename für Sitecopy benutzt. Das ist in obigem Beispiel ‘karman ’, siehe erste Zeile ‘site karman ’. Man kann das auch auf der Kommandozeile überstimmen. Genaueres liefert die ‘--help ’-Option.

2.2 Auschecken der CVS-Repositories

Wenn es mit dem Auschecken der CVS-Verzeichnisse Probleme gibt, müßtest du mir eine Email schreiben, damit ich dir ZML als Tarball zuschicke.

Das ZML-System ist als öffentliches CVS-Repository auf Sourceforge realisiert. Man bekommt es mit dem Befehl

```
cvs -d:pserver:anonymous@zeitung-ml.cvs.sourceforge.net:/cvsroot/zeitung-ml login
cvs -d:pserver:anonymous@zeitung-ml.cvs.sourceforge.net:/cvsroot/zeitung-ml co -P zml
```

Bei der Paßwort-Abfrage drückt man einfach Return. Außerdem kann man mit

```
cvs -d:pserver:anonymous@zeitung-ml.cvs.sourceforge.net:/cvsroot/zeitung-ml co -P karman-subset
```

ein Zeitungs-Verzeichnis für eine Beispielzeitung auschecken. Es handelt sich dabei um die Zeitung Kármán, heruntergebrochen auf wenige Ausgaben. Das hat keine Copyright-Gründe (wer will, kann von mir die vollständige Version bekommen), sondern liegt einfach daran, daß wir die Dienste von Sourceforge nicht mißbrauchen möchten.

Wenn du offizieller Mitarbeiter von Kármán bist, brauchst du CVS-Zugriff auf das richtige Zeitungs-Verzeichnis von Kármán. Dazu mußt du mir zunächst den öffentlichen Teil deines SSH-Schlüssels zusenden. Es muß ein Protokoll-2-Schlüssel sein, vorzugsweise RSA mit mindestens 1024 Bit.

Dann kann man das Kármán-Repository mit

```
cvs -d bob.ipv.kfa-juelich.de:/usr/local/cvsroot co karman
```

auschecken.

3 Struktur der Verzeichnisse

Im folgenden gehe ich das Zeitungs-Verzeichnis und das ZML-Verzeichnisses durch und verliere jeweils ein paar Worte dazu. Wichtige Dateien werden dabei namentlich erwähnt und besprochen.

3.1 Struktur des Zeitungs-Verzeichnisses

‘.’ ‘ausgabe-nummer.xml’ sind alle ZML-Dateien, d.h. die Layoutbeschreibungen für die Printausgabe, see Kapitel 6 [ZML] auf Seite 35.

‘webseite.xml’ beschreibt die grundlegende Struktur der Webseiten, insbesondere alle Menüs, see Abschnitt 7.3 [webseite.xml] auf Seite 40.

‘extra-feed.xml’ enthält Artikel, die in keiner Ausgabe (oder zumindest nicht in der aktuellen) erschienen sind, aber dennoch in die Newsfeeds sollen. Die Datei ist so simpel aufgebaut, daß die Benutzung kein Problem sein sollte, see Abschnitt 7.6.2 [Extra-Einträge im Feed] auf Seite 47.

‘artikel’

Hier finden sich alle Zeitungs-Artikel im XHTML-Format, und nur die. Die Dateinamen haben die Form ‘name-jahr.xhtml’. Dabei sollte der *name* den Artikel einigermaßen gut beschreiben; er darf aber nur die 26 Kleinbuchstaben und den Bindestrich enthalten. Das Jahr ist das vierstellige Jahr, in dem der Artikel verfaßt wurde.

Sonderformen wie Impressum, KurzNotiert, Parties oder Kinoprogramm enden nicht mit dem Jahr, sondern mit dem Datum ihrer jeweiligen Ausgabe im JJJJ-MM-TT-Format.

Artikel, die gesplittet werden müssen, müssen sowohl gesplittet wie auch ungesplittet vorliegen. Den gesplitteten Dateien wird dann -a, -b etc. angefügt. Das bedeutet, daß die Dateinamen aller ungesplitteten Artikel unbedingt mit einer Ziffer enden!

Für umfassende Informationen dazu see Kapitel 4 [Artikel] auf Seite 17.

‘ausgaben’

‘ausgabe-nummer.pdf’ sind alle PDF-Dateien der Printausgaben. Sie werden erzeugt, an die Druckerei geschickt und dann in diesem Verzeichnis eingecheckt. Es ist aus zwei Gründen wichtig, die PDF-Dateien zu archivieren: Zum

einen werden sie für die Website gebraucht, zum anderen ändert sich die ZML-Software in inkompatibler Weise, so daß es unmöglich werden kann, eine ältere Ausgabe noch einmal zu setzen.

‘bilder ’

Hier finden sich alle Bilder, die in den Artikeln – und nur dort – benutzt werden. Auch Anzeigen gehören hier hinein. Photos haben im JPEG-Format abgelegt zu werden, Screenshots als PNGs und Vektorbilder als PDFs. Andere Formate sind nicht erlaubt. Die Breite der Bitmaps sollte 800 Pixel nicht überschreiten.

Bilder, die nur auf der Webseite gebraucht werden, Kármán-Logos etc. werden hier *nicht* abgelegt, sondern in ‘webpool/ ’.

‘pool ’ Dies ist ein Sammelverzeichnis, insbesondere für Bilder aller Art. Beispielsweise finden sich hier die Icons für die Printausgabe. Es können auch Experimente abgelegt werden.

Es ist allerdings zu beachten, daß der Inhalt dieses Verzeichnisse nicht zu den Webseiten gehört. Dafür ist ‘webpool/ ’ zuständig.

Besonders wichtig sind die Customisation-Layers, die hier abgelegt sind. Indem man sie modifiziert, kann man sowohl die Print-Ausgabe als auch die Webseiten an die eigene Zeitung anpassen. Die originalen ZML-Stylesheets, programmiert in XSLT, sind ja für die Kármán Hochschulzeitung geschrieben, insbesondere wird überall ihr Name und ihre Logos verwendet. Mit diesen Dateien kann man das modifizieren, see Abschnitt 7.8 [Customisation-Layers] auf Seite 49.

‘snippets ’

Hier befinden sich XHTML-Dateien, die einen sehr ähnlichen Aufbau zu Artikeln haben, jedoch die nicht-Artikel-Seiten der Webseite repräsentieren. Sie entsprechen ungefähr den Menüeinträgen in der Datei ‘webseite.xml ’, see Abschnitt 7.4 [Snippets] auf Seite 42.

‘webpool ’

Dieses Verzeichnis wird in Gänze unter dem Namen ‘pool/ ’ den Webseiten hinzugefügt. Es enthält daher vor allem Bilder und die CSS-Dateien für die Webseiten. Auch hier sind die einzigen erlaubten Bildformate JPEG, PNG und PDF.

‘wiki ’ Das PHP-Wiki. Es ist eine stabile Version des PHP-Wiki-Teams. Der Quellcode wurde einfach in dieses Verzeichnis gepackt, konfiguriert und eingecheckt. Es enthält mit „Kármaqua“ einen eigenen Stil. Daher sollte man ein wenig Sorgfalt walten lassen, wenn man eine neue PHP-Wiki-Version über diese Version drüberkopiert.

‘Galerie ’

enthält den Code für die Kármán-Galerien. Sven Burmeister weiß mehr darüber.

‘karman-dokumente ’
: Hier befinden sich die LaTeX-Quellcodes der Kármán-Vereinskorrespondenz. Die bestehenden Dokumente sollten selbsterklärend sein und durch Kopieren auch neue Dokumente einfach zu erstellen lassen.

3.2 Struktur des ZML-Verzeichnisses

‘tools ’
: Kleine Helferlein im Umgang mit ZML. Davon sind allerdings nur zwei wesentlich:

  ‘zmltopdf.py ’ ist ein Python-Skript, das aus einer Ausgabe deine PDF-Version macht. Der einzige Parameter ist die Nummer der Ausgabe. see Abschnitt 8.2 [Schritt 2 - das Print-Layout] auf Seite 52.

  ‘tbrplent ’ und ‘tbents.txt ’ sind das tbrplent-Hilfsprogramm, das bei der Umwandlung von XHTML nach LaTeX nötig ist. Es wird daher von ‘zmltopdf.py ’ aufgerufen.

  ‘phpwiki2html.py ’ wandelt (mehr schlecht als recht) PHP-Wiki-Artikel in ZMLs XHTML-Dateien um, see Abschnitt 8.1 [Schritt 1 - Erzeugen der XHTML-Dateien] auf Seite 51.

  ‘make-karman-website.py ’ erzeugt die Webseiten, See Abschnitt 7.5 [make-Skript] auf Seite 44.

‘doc’
: Die Dokumentation, die du gerade liest, im Texinfo (http://www.texinfo.org)-Format.

‘xslt ’
: Hier und in den Unterverzeichnissen befindet sich das Herz von ZML, programmiert in XSLT.

  ‘ausgaben-sammeln.xsl ’ liest die Datei ‘webseite.xml ’ (insbesondere die Liste der ‘ausgabe- ∗.xml ’-Dateien, die in Form der Entity &ausgaben; angehängt ist), um die (temporäre) Datei ‘ausgaben.xml ’ zu schreiben, in der alle Ausgaben und Artikel zusammengefaßt sind.

  ‘destilate-feed.xsl ’ erzeugt den aktuellen Newsfeed im Atom-1.0-Format. See Abschnitt 7.6 [Newsfeeds] auf Seite 47, um umfassende Informationen dazu zu bekommen.

  ‘atom2rss10.xsl ’ wandelt einen Atom-1.0-Feed in einen RSS-1.0-Feed um.

‘xslt/common ’
: Hier befindet sich Code, der in mehreren XSLT-Dateien eingebunden wird. Das vermeidet Code-Verdoppelung, allerdings ist dieses Verzeichnis noch leer, weil der Code verdoppelt ist.

‘xslt/zml2html ’

‘zml2html.xsl ’ wandelt sowohl Artikel als auch Snippets in Webseiten-fähige XHTML-Dateien um. Es handelt sich also um eine Umwandlung XHTML => XHTML. Dabei greift es auch auf ‘ausgaben.xml ’ und ‘webseite.xml ’ zu, um alle Querverbindungen korrekt zu erzeugen.

Es befinden sich außerdem noch einige Stil-Dateien in diesem Verzeichnis, welche für besondere Artikel-Typen zuständig sind.

‘xslt/zml2latex ’

‘zml2html.xsl ’ wandelt eine ‘ausgabe- nummer.xml ’-Datei in eine LaTeX-Datei um, die dann mit pdfLaTeX in den PDF umgewandelt werden kann. Das wird aber alles vom Skript ‘zmltopdf.py ’ erledigt.

Es befinden sich außerdem noch einige Stil-Dateien in diesem Verzeichnis, welche für besondere Artikel-Typen zuständig sind.

4 Artikel

Artikel werden als XHTML-1.1-Dateien im Verzeichnis ‘artikel/ ’ abgelegt. Eine solche Datei liegt in UTF-8-Kodierung[1] vor und beginnt folgendermaßen:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "../xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de">
<head>
  <title>Karlspreis 2005</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <meta name="generator" content="phpwiki2html.py, bronger@physik.rwth-aachen.de" />
</head>

<body>
  ...
```

Es ist wichtig, daß der Bezug auf die DTD lokal ist, weil sonst der XSLT-Prozeß sich mit dem Internet verbinden müßte und das natürlich Zeitverschwendung ist. Mit anderen Worten: Die Zeile

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "../xhtml11.dtd">
```

muß so und nicht anders aussehen.

Die Namensraumangabe mit `xmlns` darf nicht ausgelassen werden, ansonsten wird der Artikel nicht korrekt verarbeitet.

Das `<title>` -Element wird von ZML zwar nicht benutzt, man sollte es aber trotzdem auf den richtigen Inhalt setzen, nämlich denselben wie die `<h1>`-Überschrift, siehe unten.

Ich werde im folgenden erst etwas zu der Benennung von Artikel-Dateien sagen und dann auf das Format dieser Dateien eingehen weiter eingehen.

4.1 Der Dateiname eines Artikels

Der Dateiname *muß* die Form ‘name-`JJJJ`.xhtml ’ haben. Dabei ist *name* ein gut beschreibender, knapper Name für den Artikel (dieser ist i.A. *nicht* gleich der Überschrift, sondern kürzer), welcher nur aus den 26 Kleinbuchstaben und dem Bindestrich bestehen darf, und *JJJJ* ist das Jahr, in dem der Artikel verfaßt wurde.

[1] Theoretisch kann man auch andere Kodierungen verwenden, davon rate ich jedoch ab.

Für Spezialartikel (z.B. Impressum, Veranstaltungen, Kinoprogramm, KurzNotiert), die in vielen Ausgaben unter gleichem Titel vorkommen ist die Namensgeben ein wenig anders. Hier wird statt *JJJJ* das komplette Datum im *JJJJ-MM-TT*-Format angehängt, und es ist auch nicht das Verfassungs-Datum, sondern das Datum der Ausgabe, in der der Artikel erscheint.

Ein weiterer Spezialfall sind Artikel, die für die Druckausgabe z.B. aus Platzgründen in mehrere Teile gesplittet werden müssen. Solche Artikel müssen doppelt vorhanden sein: Einmal als Komplettdatei, die für's Web benutzt wird, und natürlich als Dateisammlung mit den einzelnen Teilen. Für letztere werden '-a', '-b' etc. angehängt. Beispiel: Der Artikel 'karlspreis-2005' muß in zwei Teile gesplittet werden, weil er zu lang ist. Dann existieren folgende Dateien:

```
karlspreis-2005.xhtml
karlspreis-2005-a.xhtml
karlspreis-2005-b.xhtml
```

Das funktioniert auch, wenn die Teilartikel in verschiedenen Ausgaben erscheinen. Im Web werden sie dann allein der ersten Ausgabe zugeschlagen. Sollen sie auch im Web getrennt werden, muß man die Dateien

```
karlspreis-2005.xhtml
karlspreis-a-2005.xhtml
karlspreis-b-2005.xhtml
```

nennen. All das genannte gilt auch für Spezialartikel. Insbesondere der Partykalender muß sehr oft gesplittet werden.

Aber mittlerweile befinden sich im Verzeichnis 'artikel/' ja für all diese Fälle genügend Beispiele.

4.2 Die Meta-Tabelle

Grundsätzlich beginnt *jeder* Artikel zur Zeitung, gleichgültig welcher Art er ist, mit folgendem Kopf:

1. Einer Tabelle mit zwei Spalten, die die sogenannten Meta-Informationen zu dem Artikel enthält. Deshalb heiß diese Tabelle die *Meta-Tabelle*.
2. Eventuell irgendwelchen Kommentaren, die nicht mitgedruckt werden sollen. Es wird von den Programmen also vollkommen ignoriert.
3. Der Titel des Artikels, in einer `<h1>`-Überschrift.

Jeder Artikel beginnt mit der *Meta-Tabelle*. Das ist zunächst einmal eine ganz normale HTML-Tabelle mit zwei Spalten. Sie könnte z.B. so aussehen:

Autor	Bugs Bunny <bugs.bunny@rwth-aachen.de>
Autor	Duffy Duck <duffy.duck@rwth-aachen.de>
Datum	2004-06-14
Schlagworte	RWTH; Studentenzeitung; Gründung

oder, in XHTML:

```
<table>
  <tbody>
    <tr>
      <td>Autor</td>
      <td>Bugs Bunny &lt;bugs.bunny@rwth-aachen.de&gt;</td>
    </tr>
    <tr>
      <td>Autor</td>
      <td>Duffy Duck &lt;duffy.duck@rwth-aachen.de&gt;</td>
    </tr>
    <tr>
      <td>Datum</td>
      <td>2004-06-14</td>
    </tr>
    <tr>
      <td>Schlagworte</td>
      <td>RWTH; Studentenzeitung; Gründung</td>
    </tr>
  </tbody>
</table>
```

In der ersten Spalte steht immer ein Schlüsselwort; das ist beispielsweise „Autor“, „Datum“ oder „Schlagworte“. In der zweiten Spalte steht dann der Wert. Es gibt die folgenden Schlüsselworte:

‘Autor’ Man kann mehrere Autoren angeben, aber mindestens einen, und immer nur einen pro Zeile. Jede Zeile muß in der ersten Spalte „Autor“ stehen haben. Man kann einen Autor im Web verbergen, indem man der Zeile das class -Attribut ‘versteckt’ gibt:

```
<tr class="versteckt">
  <td>Autor</td>
  <td> &lt;duffy.duck@rwth-aachen.de&gt;</td>
</tr>
```

Wichtig: Die Autoren *müssen* ihre Email-Adresse mitangeben, und zwar hinter dem Namen in spitzen Klammern, ansonsten funktioniert ZML nur fehlerhaft. Da die Email-Adresse nirgends veröffentlicht wird, darf das kein Problem sein.

‘Datum’ Das Datum muß angegeben werden, und zwar in dem Format JJJJ-MM-TT . Der 4. März 2003 wird also „2003-03-04 “ geschrieben. Andere Schreibweisen sind nicht erlaubt.

‘Schlagworte ’
Bitte vergebt zahlreiche aussagekräftige Schlagworte. Schlagworte sollten mit Semikolons voneinander getrennt werden.

Schlagworte müssen nicht unbedingt angegeben werden. Bei manchen Artikelsorten (z.B. Impressum) machen sie wenig Sinn. Aber wenn sie auch nur ein bißchen nützlich sein würden, soll man sie auch benutzen.

‘Stil ’ Ist der Artikel kein normaler Artikel, muß hier der Stil aufgeführt werden. Das wird beim Zusammenbauen der Webseiten benötigt. Für die Druckausgabe muß der Stil mit dem stil -Attribut in der ZML-Datei angegeben werden.

Bislang gibt es die Stile „Impressum “, „Kinoprogramm “[2], „Veranstaltungen “, „Kommentar “ und „Kurz notiert “, wobei „Kurz notiert “ noch nicht richtig unterstützt wird. Die Stile müssen *genau so* eingegeben werden, man beachte insbesondere die Groß-/Keinschreibung.

Die Reihenfolge der Zeilen in der Meta-Tabelle ist letztlich egal, aber man sollte sich der Lesbarkeit zuliebe an die obige Konvention halten. Es ist wichtig, auf die richtige Großschreibung bei den Schlüsselworten zu achten.

4.3 Überschrift bzw. Titel des Artikels

Direkt im Anschluß an die Meta-Tabelle muß die Überschrift für den Artikel kommen, und zwar als HTML-<h1>-Überschrift, also eine Überschrift der höchsten Ebene.

Jeglicher Text zwischen Meta-Tabelle und Überschrift wird ignoriert.

4.4 Medien-abhängige Verarbeitung

Manchmal möchte man Teile eines Artikel nur im Internet veröffentlichen, meist weil der Artikel in der Druckausgabe sonst zu lang würde. Alle HTML-Elemente, deren class -Attribut mit ‘web-only ’ beginnt,[3] werden nicht gedruckt, erscheinen aber im Web. Das kann man für einzelne Absätze benutzen:

```
<p class="web-only">  Weiterhin gibt es auch Dinge, die wir im
Kulturbereich machen, z. B. haben wir einen kompletten Becherverleih
...  </p>
```

[2] Es gibt für’s Web keine „Kompakt“-Version.

[3] Es reicht „beginnt“, um immer noch CSS benutzen zu können, da dann ja auch Dinge wie ‘web-only-Impressum ’ oder ‘web-only1 ’ nur online erscheinen.

Man kann aber auch mit ganze Bereiche auf einmal ausblenden:

```
<div class="web-only">
<p><b>Daniel: </b> Wir sind jetzt ja noch in der Planungsphase, da wir einen
harten Wechsel gemacht haben, und mußten uns erstmal einarbeiten. </p>

<p>... </p>
</div>
```

Möchte man, daß diejenigen Passagen, die nur im Web erschienen sind, als solche im Web markiert werden (damit der Leser besser erkennen kann, was er schon kennt), so muß man in das <head>-Element in der XHTML-Datei des Artikels folgendes einfügen:

```
<style type="text/css">
  .web-only { background: silver }
</style>
```

Achtung! Das '.web-only { background: silver } ' muß *genau so* eingegeben werden, inklusive Leerzeichen und Groß-/Kleinschreibung. Davor und danach können zwar noch andere CSS-Direktiven stehen, aber die Web-Only-Direktive wird nur *so* von ZML erkannt.

Auch der umgekehrte Fall, also daß Dinge nur in der gedruckten Zeitung erscheinen, ist möglich, mit dem Wert 'print-only ':

```
<p class="print-only"><em> Das gesamte Interview ist im Internet unter <a
href="http://www.karman-aachen.de/"> http://www.karman-aachen.de/ </a>
nachzulesen. </em></p>
```

Bei 'print-only ' muß allerdings das class-Attribut genau 'print-only ' sein, es reicht nicht, daß es nur damit beginnt.

4.5 Links ins Internet

Im Prinzip werden Internet-Links so angegeben, wie sie in HTML-Dateien auch normalerweise angegeben werden:

```
Weitere Informationen finden sich auf der <a
href="http://www.rwth-aachen.de"> Homepage der RWTH</a> .
```

ZML behandelt allerdings zwei Sonderfälle: Zum einen werden Email-Links (die mit 'mailto: ' beginnen müssen) geschützt, damit Spambots nicht die Email-Adressen abgreifen können und an sie Spam versenden können. Zum anderen kommt es häufig in Artikeln vor, daß man die Adresse selber verlinken möchte. In diesem Fall ist darauf zu achten, daß der Inhalt des <a>-Elements und des href-Attributs *exakt* übereinstimmen. Beispiel:

```
<p>Weitere Infos gibt es unter <a
href="http://www.laix.de/">http://www.laix.de/</a> .</p>
```

(Es ist auch möglich, für diesen Fall das <a>-Element einfach leer zu lassen. Das haben wir früher so gemacht. Ich empfehle aber, das in neuen Artikeln zu lassen, weil es die originale XHTML-Datei unlesbar macht.)

Wichtiger Hinweis: URLs, die auf Verzeichnisse und nicht eine einzelne Datei verweisen, müssen *immer* mit einem Schrägstrich '/' abgeschlossen sein. Also nicht

```
<a href="http://www.rwth-aachen.de">http://www.rwth-aachen.de</a>
```

sondern stets

```
<a href="http://www.rwth-aachen.de/">http://www.rwth-aachen.de/</a>
```

4.6 Grafiken

Grafiken werden entweder als JPEGs, PNGs oder PDFs eingebunden. Andere Grafikformate sind nicht erlaubt.

Sollen sie in der Zeitung gedruckt werden, müssen sie in einem eigenen Absatz stehen. Alles, was an Text unmittelbar hinter der Grafik in demselben Absatz steht, wird als Bildunterschrift interpretiert. Beispiel:

```
<p><img alt="AStA" src="../bilder/asta-vorsitzende-2005.jpg"/>Die
AStA-Vorsitzenden: Daniel George (rechts) und Christoph Rasim</p>
```

Das Pfad-Präfix '../bilder/' ist nötig, damit man die XHTML-Quellcodes der Artikel im Browser mit den Grafiken zusammen sehen kann, was zugegebenermaßen nicht allzu wichtig ist. Das ist aber nicht nur historischer Ballast; der XSLT-Prozessor erkennt daran, woher er das Bild holen soll:

'../bilder/'
: aus dem allgemeinen Bildarchiv

'../webpool/'
: aus dem Webpool, der online 'pool/' heißt

In allen anderen Fällen wird das src-Attribut einfach übernommen.

4.7 Elemente unverändert auf die Webseite bringen

In seltenen Fällen stört es, daß HTML-Elemente von ZML in veränderter Form auf die Webseiten gebracht werden. Beispielsweise möchte man auf der Begrüßungsseite eine Bitmap unterbringen, aber ZML interpretiert das als Bild, macht einen Rahmen darum und setzt eine Bildunterschrift drunter. Das ist in dieser Situation natürlich falsch.

Aus diesem Grund reicht ZML alle Elemente unverändert zur Webseite durch, wenn das class-Attribut mit 'plain' beginnt. Beispiel:

```
<p class="plain">
  <img src="pool/feed-icon32x32.png" alt="Feed-Icon"
  class="plain" height="28" width="28" style="float:left"/>                    Abonniere
  unseren Newsfeed, um über neue Ausgaben und Artikel auf dem Laufenden
  gehalten zu werden.  Es stehen zwei Formate zur Verfügung:                   </p>
```

ZML hätte ohne die beiden ‘plain ’ hier eindeutig eine Grafik erkannt, die besonders behandelt werden muß.

Anderes Beispiel: Wenn man nicht will, daß eine Email-Adresse geschützt wird, kann man das so realisieren:

```
<p>Kommt es dennoch zu irgendwelchen Problemen, kontaktiert den
Webmaster; das ist zur Zeit       <a class="plain"
href="mailto:bronger@physik.rwth-aachen.de">              Torsten  </a> . </p>
```

Man beachte, daß das class -Attribut nur mit ‘plain ’ *beginnen* muß, d.h. ‘plain-Bild ’ oder ‘plainGross ’ würden genauso funktionieren. Dadurch kann man weiterhin CSS-Direktiven nutzen.

4.8 Artikel-Stile

Bislang habe ich nur ganz allgemein etwas zu Artikeln gesagt. Es gibt jedoch verschiedene *Stile* von Artikeln, die jeweils ihre Besonderheiten haben. Der Standard-Stil ist „normaler Artikel“, was für die allermeisten Artikel zutrifft.

Es kann aber auch die Liste mit den Parties für die Woche sein, oder ein persönlicher Kommentar. Wenn es ein solcher Spezialartikel ist, muß dieser Spezialstil ausdrücklich angegeben werden, und zwar an *zwei* Stellen: im Artikel selber und in der ZML-Datei.

Im Artikel selber ist das das „Stil“-Feld in der Meta-Tabelle, see Abschnitt 4.2 [Meta-Tabelle] auf Seite 18. In der ZML-Datei ist es das stil -Attribut, see Abschnitt 6.4 [Das Element beitrag] auf Seite 36. Die folgende Tabelle zeigt, welche Werte jeweils korrespondieren:

stil -Wert	Meta-Tabellen-Wert
veranstaltungen	Veranstaltungen
kinoprogramm	Kinoprogramm
kinoprogramm-kompakt	Kinoprogramm
kurz notiert	Kurz notiert
kommentar	Kommentar
anzeige	—
anzeige-gerahmt	—
impressum	Impressum
comic	Comic

Es sei nochmal darauf hingewiesen, daß man sich für normale Artikel um den Stil nicht scheren muß.

Im folgenden werde ich die Besonderheiten der verfügbaren Stile genauer beschreiben.

4.8.1 Normale Artikel

Stilname: *nicht nötig*

Normale Artikel sind eben normale Artikel. Die meisten Beiträge einer Zeitung sind normale Artikel.

Unterüberschrift

Soll der Artikel neben der Hauptüberschrift noch eine Unterüberschrift bekommen, muß sie als <h2>-Überschrift direkt der <h1>-Überschrift folgen. Das könnte z.B. so aussehen:

```
<h1>RWTH will Kármán-Auditorium pink anstreichen </h1>
<h2>Tickt der Rektor nun völlig aus? </h2>

...
```

Danach kann eine Grafik folgen, die über den *kompletten* normalen Artikel gedruckt wird, see Abschnitt 4.6 [Grafiken] auf Seite 22.

Schließlich kommt der normale Artikel selber, alle Absätze, Bilder etc.

Einleitungsabsatz

Wenn der erste Text-Absatz *komplett* in fett gedruckt wird (dabei muß das HTML-Element und nicht etwa <b> zum Einsatz kommen), wird dieser Absatz als Kurz-Einleitung interpretiert und im Layout besonders behandelt.

Zwischenüberschriften

Jede HTML-<h3>-Überschrift im normalen Artikel wird als Zwischenüberschrift interpretiert und gedruckt.

Artikel-Rubrik

In die Meta-Tabelle kann ein zusätzlicher Eintrag namens „Rubrik“ erfolgen. Was man dort einträgt, wird im Web oben rechts neben den Artikel gedruckt. Beispiel:

```
    ...
<tr>
    <td> Rubrik </td>
    <td> Leser|Brief </td>
```

```
    </tr>
  </tbody>
</table>
```

druckt ein „LeserBrief“ über den Artikel, wobei „Leser“ schwarz und „Brief“ grau gedruckt wird. Beides wird also durch ein Pipe-Zeichen ‘| ’ voneinander getrennt.

Achtung! Das gilt nur für’s Web. Damit das auch im Druck passiert, muß das <beitrag> -Element ein rubrik -Attribut mit demselben Inhalt haben, see Abschnitt 6.4 [Das Element beitrag] auf Seite 36.[4]

4.8.2 Veranstaltungen

Stilname in der ZML-Datei: veranstaltungen
Stilname in der Meta-Tabelle: Veranstaltungen

Ein solcher Artikel besteht nur aus einer Tabelle mit fünf Spalten:

1. Name der Veranstaltung
2. Datum im Format JJJJ-MM-TT
3. Uhrzeit im Format HH:MM
4. Ort der Veranstaltung
5. Beschreibung

In jeder Tabellenzeile muß genau eine Veranstaltung stehen. Hier ein Beispiel:

```
<h1>PartiesVeranstaltungen</h1>

<table  border="1">
  <tbody>
    <tr>
      <th>Was?</th>
      <th>Tag</th>
      <th>Uhrzeit</th>
      <th>Ort</th>
      <th>Beschreibung</th>
      <th>URL</th>
    </tr>
    <tr>
```

[4] Das klingt nach einer überflüssigen Verdoppelung. Zum Teil hat das historische Gründe, zum Teil Performance-Gründe, vor allem aber könnte es sein, daß irgendwann einmal von der Möglichkeit Gebrauch gemacht wird, daß mehrere Dateien in einem <beitrag> -Element eingebunden werden. Dann ist sowas praktisch. Vermutlich ist es jedoch besser, die Rubrik gänzlich in die XHTML-Datei zu verschieben.

```
    <td>AHoi-Party - Die Aachener Hochschulinitiativen</td>
    <td>2005-12-20</td>
    <td>21:00</td>
    <td>Westbahnhof, Republikplatz</td>
    <td>VVK u. a. im Filmstudio und im AStA für 2,50 €, Abendkasse
      3,50 €</td>
    <td>http://www.ccac.rwth-aachen.de/ahoi/</td>
  </tr>
  <tr>
    ...
```

Die erste Tabellenzeile enthält die Spaltenüberschriften. Sie ist nicht nötig, dient aber der besseren Übersichtlichkeit und wird sowohl im Druck als auch in den Webseiten ignoriert.

Man beachte, daß Uhrzeiten, Daten und URLs genau so eingegeben werden müssen, weil diese Felder von ZML interpretiert und verarbeitet werden.

Manchmal möchte man über die Veranstaltungen noch etwas schreiben, z.B. wenn die Veranstaltungen im Druck gesplittet wurden. Das ist kein Problem:

```
<h1>PartiesVeranstaltungen</h1>

<p><em>(Fortsetzung)</em></p>

<table border="1">
  ...
```

Ein weiterer Sonderfall sind Partys, die herausgestellt werden sollen, z.B. weil Kármán Karten dafür verlost. Das geht mit dem `class`-Attributswert ‘`hervorgehoben`’:

```
  <tr class="hervorgehoben" style="&#x2744;">
    ..
```

Das `style`-Attribut enthält ein einzelnes Zeichen, daß für die Markierung oben und unten der Party genutzt wird. Das im Beispiel genutzte Zeichen ‘`&#x2744;`’ ist ein Unicode-Sternchen. In solchen Fällen kann man von der Möglichkeit Gebrauch machen, bei der Partybeschreibung mehrere Absätze mit `<p>` einzufügen. Das geht aber auch ohne ‘`hervorgehoben`’.

Es ist Konvention, der Datei, die die Parties für eine bestimmte Ausgabe enthält, den Namen ‘`parties-2004-07-27.xhtml`’ zu geben. Dabei ist ‘`2004-07-27`’ das Datum der Ausgabe.

Die `<h1>`-Überschrift eines Artikels mit Veranstaltungen muß „PartiesVeranstaltungen“ sein, auch wenn sie bislang ignoriert wird.

4.8.3 Kinoprogramm

Stilname in der ZML-Datei: `kinoprogramm`, oder `kinoprogramm-kompakt`, um Platz zu sparen.

Stilname in der Meta-Tabelle: Kinoprogramm

Dieser Stil ist geeignet, Filme des Filmstudios oder des AStA-Kinos aufzulisten. Dabei besteht der Artikel aus einer Tabelle mit sechs Spalten:

1. Datum im JJJJ-MM-TT -Format
2. Uhrzeit im HH:MM-Format, wobei hier auch freier Text zugelassen ist (nötig für die Feuerzangenbowle)
3. Ort der Vorführung
4. Filmtitel
5. spezielle Information wie z.B. „Original mit Untertitel“, Eintrittspreis etc.
6. Kurzbeschreibung des Films

Dabei ist es gängige Praxis geworden, über die Tabelle drüber zu schreiben

```
<h3>Filmstudio     </h3>

<p><a  href="http://www.filmstudio-aachen.de"></a></p>

<p>Wenn  nicht  anders  angegeben,  Beginn  jeweils  um  19:45 Uhr  in  der
Aula  im  Hauptgebäude,  Eintritt  € 2,50.                </p>
```

wenn es sich um das Programm des Filmstudios handelt, und dann eben die Tabellen-Felder leer zu lassen, die von diesen Angaben nicht abweichen. Dieser Beispiel-Schnippsel zeigt auch, wie man das Kino nennt: Es wird einfach als <h3>-Überschrift drübergeschrieben.

Das AStA-Kino-Programm kann und sollte in dieselbe Datei geschrieben werden. Man beginnt einfach eine zweite Tabelle und schreibt darüber:

```
<h3>AStA-Kino   </h3>

<p><a  href="http://www.asta.rwth-aachen.de"></a></p>
```

Die <h3>-Überschriften müssen genau den hier angegebenen Inhalt haben, weil ZML wissen muß, um welches Kino es sich handelt – sonst kann es nicht das entsprechende Logo danebendrucken.

Im HTML-Code könnte die Kino-Tabelle so aussehen:

```
<table  border="1">
   <tbody>
      <tr>
         <td>2005-11-02</td>
```

```
            <td>20:00</td>
            <td></td>
            <td>Star Wars: Episode III</td>
            <td><ul>
                  <li>OF</li>
                </ul></td>
            <td>Science-Fiction, bei der am Ende jeder sein Fett
              wegbekommt</td>
          </tr>
          ...
          <tr>
            <td>2005-11-18</td>
            <td>diverse Uhrzeiten</td>
            <td>diverse Hörsäle</td>
            <td>Die Feuerzangenbowle</td>
            <td><ul><li>OmU</li><li>Eintritt: 3 €</li>
              <li>Bericht zur Veranstaltung: Traditionen feiern an der RWTH
                Aachen</li></ul></td>
            <td>Kult-Komödie</td>
          </tr>
          </tr>
        </tbody>
      </table>
```

Am Feuerzangenbowlen-Eintrag erkennt man, daß man auch freien Text in die Uhrzeit schreiben kann.
Was noch fehlt ist, was in die fünfte Tabellenspalte geschrieben wird. Es handelt sich dabei um eine Liste mit `<ul>` , die `<li>` -Elemente enthält. Deren Inhalt wiederum kann normaler Text sein wie „Eintritt: 3 €“, oder es ist eine Abkürzung. Folgende Abkürzungen stehen zur Verfügung:

F	Eintritt frei
OF	Originalfassung
OmU	Original mit Untertiteln
OmeU	Original mit englischen Untertiteln
OmdK	Original mit deutschem Kommentar
RWTH	nur für Studenten und RWTH-Angehörige

Der Zweck der Abkürzungen ist übrigens nicht nur, ein wenig Tipparbeit zu sparen, sondern auch, in zukünftigen ZML-Versionen Symbole für diese Dinge zu ermöglichen.

Es ist Konvention, der Datei, die das Kinoprogramm für eine bestimmte Ausgabe enthält, den Namen ‘kinoprogramm-2004-07-27.xhtml ’ zu geben. Dabei ist ‘2004-07-27 ’ das Datum der Ausgabe.

Die <h1>-Überschrift eines Artikels mit Veranstaltungen muß „Kinoprogramm“ sein, auch wenn sie bislang ignoriert wird.

4.8.4 Kurz notiert

Stilname in der ZML-Datei: kurz notiert
Stilname in der Meta-Tabelle: Kurz notiert

Ein Kurz-Notiert-Artikel besteht aus kleinen Einzelartikeln, die jeweils aus einer <h2>-Überschrift und einem oder sehr wenigen Absätzen bestehen. Es ist, weil es sich hierbei um einen Sammel-Artikel handelt, besonders wichtig, gute Schlagworte zu vergeben.

Wenn die Einzelartikel von verschiedenen Personen stammen, ist es möglich, wenn auch nicht nötig, die Autoren vor den Artikeln manuell zu nennen:

```
<h3>Karlspreis an Juncker </h3>

<p><em>(von Florian Eßer) </em> Der luxemburgische Premierminister
Jean-Claude Juncker ist der Träger des Internationalen Karlspreises im
Jahre 2006. Dies ...
```

Es ist Konvention, der Datei, die die KurzNotiert-Artikel für eine bestimmte Ausgabe enthält, den Namen ‘kurz-notiert-2004-07-27.xhtml ’ zu geben. Dabei ist ‘2004-07-27 ’ das Datum der Ausgabe.

Die <h1>-Überschrift eines Kurz-Notiert-Artikels muß „KurzNotiert“ sein, auch wenn sie bislang aber ignoriert wird.

4.8.5 Kommentare

Stilname in der ZML-Datei: kommentar
Stilname in der Meta-Tabelle: Kommentar

Kommentare sind normale Artikel, die im Druck und im Web mit dem Zusatz „Meine-Meinung“ markiert werden. Man hätte das auch über eine Rubrik realisieren können, see Abschnitt 4.8.1 [Normale Artikel] auf Seite 24, allerdings gab es „Rubrik“ beim ersten Kommentar noch nicht. Dieser Stil ist also ein historisches Überbleibsel, das aber genutzt werden muß, wenn es sich um einen Kommentar handelt.

4.8.6 Impressum

Stilname in der ZML-Datei: impressum

Stilname in der Meta-Tabelle: Impressum

Das Impressum hat ebenfalls einen sehr einfachen Aufbau: Die <h1>-Überschrift muß „Impressum“ lauten, auch wenn sie bislang nicht genutzt wird. Danach folgen <h2>-Überschriften mit den einzelnen Impressum-Einträgen. So kann da beispielsweise stehen

```
<h2>Chefredakteur</h2>

<p>Bugs Bunny und Duffy Duck</p>
```

Es ist Konvention, der Datei, die das Impressum für eine bestimmte Ausgabe enthält, den Namen ‘impressum-2004-07-27.xhtml’ zu geben. Dabei ist ‘2004-07-27’ das Datum der Ausgabe. Allerdings ist es beim Impressum üblich, da es sich so selten ändert, die Impressum-Datei einer älteren Ausgabe wiederzuverwenden.

4.8.7 Anzeigen

Stilname in der ZML-Datei: anzeige, bzw. anzeige-gerahmt, wenn man noch ein Rähmchen drumherum haben möchte.

Anzeigen werden *nicht* über externe XHTML-Dateien eingebunden. Stattdessen wird die Grafikdatei, die die Anzeige enthält, direkt im Attribut datei im Element <beitrag> angegeben. Die einzigen erlaubten Bildformate sind PNG, PDF und JPEG. *Achtung:* Die Dateiendung darf nicht mitangegeben werden!

4.8.8 Comics

Stilname in der ZML-Datei: comic
Stilname in der Meta-Tabelle: Comic

Comics enthalten meist nur eine Grafik. Ein Comic-Artikel könnte nach der Meta-Tabelle so aussehen:

```
<h1>Hervé &amp;amp; Thea 4</h1>

<p><img alt="Hervé-&amp;amp;-Thea-Comic"
   src="../bilder/hervethea-4-2005.pdf"/></p>

<p class="web-only">Bitte auf das Bild klicken, um die größere Ansicht
zu erhalten.</p>
```

5 Typographische Richtlinien

Die Zeitung soll einen möglichst professionellen Eindruck machen. Das gilt sowohl für den Web-Auftritt als auch für die Printausgabe, wobei letztere typographisch erheblich kritischer ist. Aus diesem Grund gebe ich in diesem Kapitel einige Vorgaben und Tips, damit das Layout und die Feinheiten einigermaßen stimmen.

Vorab eine Anmerkung zur Wahl des Editors. Es ist wünschenswert, daß die Endkorrektur der Artikel mit einem Editor erfolgt, der die direkte Eingabe von Unicodes zuläßt. Man kann zwar in XHTML-Dateien auch sogenannte numerische Entitäten wie '`&#x2011;`' einfügen, aber die Verwendung der Zeichen selber macht die Datei erheblich les- und korrigierbarer. Ich werde in dieser Anleitung dennoch die Entitäts-Schreibweise (mit hexadezimalen Zahlen) verwenden, weil sich die Art und Weise, wie solche Sonderzeichen eingegeben werden, von Editor zu Editor stark unterscheidet.

Man kann übrigens auch richtige HTML-Editoren wie Amaya (http://www.w3.org/Amaya/) verwenden. Es ist allerdings sehr wünschenswert, wenn die XHTML-Dateien, die der HTML-Editor erzeugt, auch mit Texteditoren noch lesbar sind. Insbesondere sollten keine Mammut-Zeilen und unübersichtliche Tag-Suppe produziert werden.

Achtung: Im vorliegenden Text könnte die Darstellung der Sonderzeichen ein bißchen gelitten haben, abhängig davon, mit welchem Programm diese Anleitung erstellt wurde. Davon darf man sich nicht verwirren lassen.

5.1 Allgemeine Richtlinien

1. Es wird ausschließlich die Neue Rechtschreibung verwendet.

2. Sollen Worte hervorgehoben werden, geschieht das nicht durch Fettdruck, sondern durch *Kursivdruck*.

3. In großen Zahlen werden die Dreiergruppen durch Spatien abgetrennt, see Abschnitt 5.4 [Leerräume] auf Seite 33.

4. Euros werden entweder „Euro“ oder „€“ abgekürzt, niemals „EUR“.

5. Absätze sollten nie länger sein als zwei Zeitungs-Spalten breit sind. Ausnahmen sind möglich.

6. Daten werden „1. Januar 2006“ gedruckt, nie 01.01.06 o.ä. Uhrzeiten werden „20 Uhr“ oder „20:30 Uhr“ gedruckt, niemals mit führender Null. Im Veranstaltungska-

lender und im Kinoprogramm gelten u.U. leicht andere Regeln, die aber von den Skripten durchgesetzt werden.

7. Sind Studenten beiderlei Geschlechts gemeint, hat man sie „Studierende“ zu nennen. Ansonsten ist für den Plural die allgemein übliche Form zu benutzen, also „Sänger“, „Professoren“ etc.

8. Harte Zeilenumbrüche durch <br/> sind nur in Ausnahmefällen sinnvoll. Meist sind sie ein Indiz für fehlerhaftes Markup. Häufig wären beispielsweise mehrere Absätze durch <p> sinnvoller.

Grundsätzlich sollte man im mittlerweile bereits üppigen Artikel-Archiv von Kármán stöbern, um Positiv-Beispiele zu sehen.

5.2 Bindestriche & Co

Für den typographischen Anfänger mag es nur Bindestriche geben, in Wahrheit gibt es jedoch einige Unterarten davon:

- (Bindestrich)
: ist der eigentliche Bindestrich, der zwischen Teilworten benutzt wird, wie in „Physik-Lehrbuch“.

weicher Bindestrich ( )
: Wenn ein Wort falsch getrennt wird, kann man mit weichen Bindestrichen nachhelfen. Die werden nur sichtbar, wenn eben an ihrer Stelle ein Zeilenumbruch erfolgt. Beispielsweise findet sich im Artikel über das RWTH GYM:

```
Ausdauer-Ab=tei=lung
```

Die ‘=’ sind in Wahrheit diese weichen Bindestriche.

Wird dasselbe Wort immer wieder falsch getrennt, sollte man es der \hyphenation -Liste in der Datei ‘xslt/zml2latex/zml2latex.xsl ’ hinzufügen. Dort befinden sich schon einige Worte, an denen man erkennen kann, wie man es macht.

geschützter Bindestrich (‑)
: Dies ist ein normaler Bindestrich mit der Besonderheit, daß an ihm nie ein Zeilenumbruch erfolgt. Beispielsweise wäre es beim Ausdruck „Groß-/Kleinschreibung“ ja blöd, wenn er

Groß-
/Kleinschreibung

gedruckt würde.

− (Minuszeichen, −)
: wird in Formeln und als Vorzeichen von Zahlen benutzt: −3,4 Volt.

– (Gedankenstrich, –)
: wird benutzt, um einzelne Satzteile – so wie hier – vom Rest abzusetzen. In diesem Fall muß er von Leerzeichen umgeben sein.

  Es ist auch der Von-Bis-Strich, wie in „der Zug Hamburg–Berlin verspätet sich“, oder „das Spiel Hamburg–München endete unentschieden“. Hier wird er ohne Leerzeichen gesetzt.

— (langer Gedankenstrich, —)
: wird im Deutschen so gut wie nie eingesetzt.

5.3 Anführungszeichen

Anführungszeichen sollten sparsam eingesetzt werden. Insbesondere sind sie fehl am Platz, um Worte zu kennzeichnen, die nicht ganz passen. Dann muß man sich eben ein besseres Wort ausdenken.

Ist man sich sicher, Anführungsstriche zu brauchen, muß man sich die richtigen benutzen. Das Zitat beginnt mit „ („) und endet mit “ (“). Als hübsche Alternative stehen die Guillemets » und « (» und «) zur Verfügung.

Ein komplett englisches Zitat muß mit den englischen Anführungszeichen “ (“) und ” (”) umschlossen werden.

5.4 Leerräume

Im großen und ganzen gibt es drei Zeichen, die einen Leerraum darstellen:

Leerzeichen
: ist hoffentlich klar.

geschütztes Leerzeichen ( )
: ist ein Leerzeichen, an dem nie ein Zeilen- oder Seitenumbruch stattfindet. So etwas benötigt man oft, z.B. in „22 Uhr“. Es wäre ja häßlich, wenn daraus im Druck

  > Die Wohnheim-Bars beginnen um 21 Uhr in der
  > Vorlesungszeit, während die meisten Discos um 22
  > Uhr öffnen. …

  würde. Es gibt aber noch zahlreiche andere Anwendungsfälle, wie z.B. „Dr. Müller“ oder „3. Tag“.

Spatium ( )

Vorab erstmal ein Achtung: Im diesem Dokument kann man Spatien nicht sehen.

Das Spatium ist ein halbes Leerzeichen, an dem nie ein Zeilen- oder Seitenumbruch erfolgt. Es verhält sich also wie das geschützte Leerzeichen, ist allerdings nur halb so breit. Es ist besonders geeignet vor Einheiten wie „3€“ oder „4m“. Zwischen längeren Ausdrücken ist allerdings das volle geschützte Leerzeichen zu bevorzugen: „3 Euro“ oder „4 Meter“.

Außerdem wird das Spatium zwischen »d.h.«, »z.B.« etc. eingesetzt, und zum Abtrennen der Dreiergruppen in großen Zahlen benutzt: 100000.

Umbruchpunkt (‚)

Der Umbruchpunkt ist ein Leerzeichen ohne Breite, allerdings kann an ihm ein Zeilenumbruch erfolgen. Beispiel: Bachelor-/Masterstudenten muß man

```
Bachelor&#x2011;/&#x82;Masterstudenten
```

eingeben. (Je nach Editor, siehe oben.)

6 ZML – das Zeitungsgerüst

In der ZML-Datei wird eine Ausgabe der Zeitung zusammengestellt. Normalerweise enthält sie *nicht* den Inhalt selber, sondern sie verweist auf externe XHTML-Dateien, die die Artikel enthalten. ZML ist vergleichsweise einfach. Es gibt nur wenige Elemente, nämlich Seiten, Kolumnen, Rechtecke und Beiträge.

Eine einfache ZML-Datei könnte so aussehen:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE zeitung [<!ENTITY datum "2004-07-27">]>

<zeitung datum="&datum;" ausgabe="0" impressum="impressum-&datum;">
  <seite>
    <kolumne breite="5.26cm">
      <beitrag stil="veranstaltungen" datei="parties-&datum;"/>
      <beitrag stil="kurz notiert" datei="kurz-notiert-&datum;"/>
      <beitrag stil="hochschulsport" datei="hochschulsport-&datum;"/>
    </kolumne>
    <kolumne>
      <beitrag datei="partylandschaft-aachen" spalten="3" dehnen="5"/>
    </kolumne>
    <kolumne breite="5.26cm">
      <beitrag datei="ueber-karman"/>
    </kolumne>
  </seite>
</zeitung>
```

6.1 Das Element „zeitung“

Das Element <zeitung> umfaßt die ganze Zeitung. Es enthält mehrere <seite> n, die wiederum aus einer oder mehr <kolumne> n bestehen.

Es muß drei Attribute enthalten, nämlich datum , ausgabe und impressum . datum enthält das Erscheinungsdatum im Format ‘JJJJ-MM-TT ’. Andere Formate sind nicht erlaubt. ausgabe enthält die Ausgabennummer, z.B. "3" . Theoretisch kann das aber auch etwas wie "6a" sein (ist schon vorgekommen).

impressum schließlich ist der Dateiname (ohne Endung) des Artikels, der das Impressum enthält. Dessen Datum muß übrigens nicht gleich dem Datum der aktuellen Ausgabe sein. Da sich das Impressum nur selten ändert und es dämlich wäre, für jede Ausgabe

einen neuen gleichen Impressum-Artikel anzulegen, nimmt man einfach das Impressum einer früheren Ausgabe.

Als optionales Attribut kann man reine-webausgabe auf "ja" setzen. Dann wird diese Ausgabe nicht gedruckt (es gibt kein PDF), aber sie erscheint im Web.

6.2 Das Element „seite“

Eine Seite enthält eine oder mehrere Kolumnen. Das Element <seite> kann das Attribut rubrik enthalten, das dann die Rubrik der jeweiligen Seite enthält. Die Rubrik wird in die Kopfzeile der Zeitungsseite gedruckt.

Weiterhin kann das Attribut spalten die Zahl der Spalten der Seite angeben. Der einzige Sinn und Zweck dieser Angabe ist, daß daraufhin bei der Breite von Kolumnen neben den üblichen Einheiten cm usw. auch die Einheit „\Spaltenbreite “ möglich ist. Für die Berechnung dieser Einheit wird dann die hier angegebene Zahl der Spalten, die Textbreite und der Spaltenabstand angenommen. Es ist also nur eine unverbindliche Bequemlichkeit.

Wenn man unregelmäßige Spaltenbreiten haben möchte, oder schlicht bei der Angabe der Spaltenzahl lügt, nützt dieses Attribut nichts.

6.3 Das Element „kolumne“

Eine Kolumne nimmt immer die komplette Höhe ein, die frei ist. Das heißt insbesondere, daß Kolumnen, die direkt in <seite> stehen, die ganze Seitenhöhe haben.

Eine Kolumne kann außerdem ein Attribut haben, nämlich die *Breite* der Kolumne. Dafür schreibt man beispielsweise

```
<kolumne breite="5cm">
  ...
</kolumne>
```

Damit bekommt die Kolumne eine Breite von 5 cm. Wenn man keine Breite angibt, wird die Breite genommen, die noch frei ist. Hat man beim Mutter-<seite> -Element das Attribut spalten angegeben, kann man hier auch die Einheit „\Spaltenbreite “ benutzen. See Abschnitt 6.2 [Das Element seite] auf dieser Seite, um mehr dazu zu erfahren.

Bei Kolumnen gibt es zwei mögliche Elemente, die man da hineinschreiben kann: <artikel> und <rechteck> e.

6.4 Das Element „beitrag“

Dieses Element sollte vielleicht besser <artikel> heißen, aber es heißt nunmal <beitrag> . Wie auch immer, es bindet einen Artikel an der aktuellen Position in die Zeitung ein. Man

muß im Attribut datei den Dateinamen (ohne Endung) der XHTML-Datei, die den Artikel enthält, angeben. Beispiel:

```
<kolumne breite="5cm">
  <beitrag datei="herr-der-ringe-2005"/>
</kolumne>
```

Der Artikel wird dann in der Datei ‘herr-der-ringe-2005.xhtml’ erwartet. Man kann auch die Zahl der Spalten angeben, die der Artikel haben soll:

```
<kolumne breite="5cm">
  <beitrag datei="herr-der-ringe" spalten="3"/>
</kolumne>
```

Normalerweise gibt es nur eine Spalte, die so breit ist wie die umgebene Kolumne. Die Dateinamen für Artikel dürfen keine Leerzeichen enthalten. Man muß Bindestriche stattdessen verwenden.

Das Element <beitrag> ist immer leer.

Stile

Den Stil oder die Art des Artikels kann man im Attribut stil dem Element <beitrag> übergeben. Der vorgegebene Stil, also der, der aktiv ist, wenn man nichts ausdrücklich setzt, ist „normaler Artikel“. Das ist für Autoren selbstverständlich der wichtigste Stil. Aber es gibt noch andere, see Abschnitt 4.8 [Artikel-Stile] auf Seite 23.

Wenn der Artikel zu lang oder zu kurz ist . . .

. . . kann man das dehnen -Attribut benutzen. Es kann Werte von −9 (Artikel stark stauchen) bis 9 (Artikel stark dehnen) annehmen. Wenn man den Artikel etwas dehnen möchte, schreibt man also beispielsweise

```
<beitrag datei="herr-der-ringe-2005" spalten="3" dehnen="3"/>
```

Rubriken

Mit dem Attribut rubrik kann man einem einzelnen Artikel eine Rubrik zuordnen. Das darf man nicht mit der Rubrik für eine Seite verwechseln. Hier geht es um Dinge wie Leserbriefe oder eventuell Gegendarstellung oder was auch immer. Die Rubrik wird mit schwarz-grauer Schrift links oben über die Überschrift gedruckt. Beispiele:

```
<beitrag datei="muntu-afrika-2005" rubrik="Studentische|Initiativen"
  spalten="3" dehnen="-9"/>

<beitrag datei="leserbrief-1-2005" rubrik="Leser|Brief"
  autor-nennen="nein" spalten="2"/>
```

Der Teil vor dem Pipe-Zeichen '|' wird schwarz gedruckt, der dahinter grau. Zwischen beiden wird kein Leerraum gedruckt, ähnlich wie bei den festen Rubriken „KurzNotiert", „PartiesVeranstaltungen" etc. Achtung: In den meisten Fällen ist es sinnvoll, daß der Artikel die entsprechende Rubrik auch im Internet hat. Das muß dann in der Meta-Tabelle des Artikels gesondert eingestellt werden, see Abschnitt 4.8.1 [Normale Artikel] auf Seite 24.

Autorennamen unterdrücken

Wollen die Autoren eines Artikels nicht genannt werden, setzt man einfach das Attribut autor-nennen auf '"nein"':

```
<beitrag datei="leserbrief-1-2005" rubrik="Leser|Brief"
  autor-nennen="nein" spalten="2"/>
```

Reine Web-Artikel

Es ist nicht nur möglich, ganze Ausgabe als „nur Web" zu markieren, man kann dies auch mit einzelnen Artikel machen:

```
<beitrag datei="datenschutz-kommentar-2006" nur-web="ja"/>
```

6.5 Das Element „rechteck"

Rechtecke können neben Beiträgen in Kolumnen stehen. Rechtecke sind nichts anderes als „kleine Seiten". Sie enthalten wieder Kolumnen. Der einzige Sinn und Zweck ist, daß man dadurch sehr flexibel das Layout der Seite gestalten kann. Sie wurden allerdings bislang selten benötigt.

Zusätzlich kann man mit dem Attribut hoehe die Höhe das Rechtecks festlegen, wobei das eigentlich schon durch die Länge der Beiträge bestimmt wird.

6.6 Das Element „abstand"

Dieses Element ist für den letzten Schliff einer Ausgabe gedacht. Wenn trotz aller Maßnahmen immer noch zuviel Platz in der Spalte ist, kann man mit

```
<abstand hoehe="1cm"/>
```

beispielsweise einen Zentimeter einfügen. Das Element <abstand> ist immer leer.

7 Die Webseiten

7.1 Überblick über die Webseiten

Eine Webseite besteht aus der waagrechten Menüzeile oben, der linken Spalte, welche eine feste Breite besitzt, und der rechten Spalte, in der der eigentliche Text untergebracht wird.

Die größte Navigationstiefe ist zwei: In der Menüzeile sucht man sich den Hauptpunkt aus und kann dann eventuell in der linken Spalte noch einen Untermenüpunkt anwählen.

Meist wird die linke Spalte allerdings für Illustrationen oder das Inhaltsverzeichnis einer Ausgabe benutzt.

Folgende Eigenschaften müssen die Webseiten erfüllen:

1. Alle Webseiten müssen gültiges XHTML 1.1 sein. Das bedeutet, daß sie vom W3C-Validator (http://validator.w3.org) als solches validiert werden müssen.

2. Jeglicher CSS-Code muß gültiger CSS-2.1-Code sein. Er muß als solcher vom CSS-Validator (http://jigsaw.w3.org/css-validator/) validiert werden. Warnungen sind dabei egal.

3. Wenn Feeds angeboten werden, dann nur absolut streng nach Spezifikation. Zur Zeit werden Atom 1.0 und RSS 1.0 angeboten, weil es die einzigen Feed-Formate sind, die auf einer sauberen XML-Anwendung aufbauen. RSS 2.0 wäre auch akzeptabel.

4. Die einzigen erlauben Grafikformate sind PDF, JPEG und PNG. Insbesondere das völlig veraltete GIF-Format hat im Web nichts mehr verloren.

5. Schaltet man CSS ab, muß die Seite lesbar bleiben. Desweiteren sind die Richtlinien der WAI (http://www.w3c.de/Trans/WAI/webinhalt.html) weitgehend umzusetzen.

6. Die Webseiten sind mit üppigen Metainformationen auszustatten, z.B. nach dem Dublin Core (http://dublincore.org/).

Mit dem aktuellen Stand von ZML wird das alles auch erreicht. ZML erzeugt einen RSS-1.0- und Atom-1.0-Feed und enthält Metainformation nach dem Dublin Core.

Die Webseiten sind auch ohne CSS übersichtlich strukturiert (insbesondere verwandelt sich dann die Menüleiste in eine Liste). Die Benutzung des `accesskey` - und `tabindex` -Attributs, des `alt` -Attributs, der generell einfache Aufbau der Seiten sowie die Deklaration von Querverbindungen mit `<link rel="..." .../>` machen die Seite behindertengerecht und Textbrowser-geeignet.

7.2 Verzeichnisstruktur der Webseiten

Die Verzeichnisstruktur der Webseiten ist erheblich einfacher als die des Zeitungs-Verzeichnisses. Ich beschreibe hier alles relativ zur Wurzel-URL, die z.B. http://www.karman-aachen.de/ oder – wie es für Kármán aktuell der Fall ist – http://www-users.rwth-aachen.de/karman/ sein kann.

Obwohl es sich um XHTML-Dateien handelt, haben alle XHTML-Dateien der Webseite die Endung '.html '.

Alle Artikel sind – natürlich in umgewandelter Form – im Verzeichnis 'Artikel/ ' abgelegt. Die auf 300 px Breite skalierten Bilder befinden sich in 'Bilder/ ' und die Originale in 'Bilder/original/ '.

Alle bisher gedruckten Ausgaben sind als PDFs im Verzeichnis 'Ausgaben/ '.

Das Zeitungs-Verzeichnis 'webpool/ ' ist komplett in 'pool/ ' zu finden. (Das Zeitungs-Verzeichnis 'pool/ ' steht auf der Website nicht zur Verfügung.)

Im Zeitungs-Verzeichnis sind alle Newsfeeds und natürlich die Willkommenseite 'index.html '. Diese Seite sieht also der Besucher, wenn er einfach die Wurzel-URL in seinen Browser eingibt.

Die Webseiten haben ein Hauptmenü. Zu (fast) jedem Menüpunkt gibt es ein Unterverzeichnis, das gleich oder ähnlich heißt. Zum Menüpunkt „Impressum" existiert beispielsweise 'Impressum/ ', und darin wiederum 'Impressum/index.html '.

Einige Hauptmenüpunkt haben noch Untermenüpunkte, deren Dateien in demselben Verzeichnis stehen. Der Untermenüpunkt „Hilfe" des „Impressums" würde in der Datei 'Impressum/Hilfe.html ' stehen.

7.3 Die Datei 'webseite.xml '

In der Datei 'webseite.xml ' wird die Struktur der Webseiten definiert, insbesondere die Menü-Struktur. Die aktuelle 'webseite.xml ' von Kármán sieht wie folgt aus:

```
<!DOCTYPE webseite [
<!ENTITY ausgaben SYSTEM "ausgaben.lst">
]>
<webseite xmlns="http://www.karman-aachen.de/webseite"
          wurzel-url="http://www-users.rwth-aachen.de/karman/">
  <menue name="Aktuell" startseite="ja" accesskey="S"/>
  <menue name="Archiv" archiv="ja" accesskey="A"/>
  <menue name="Mitmachen" accesskey="M"/>
  <menue name="Veranstaltungen"
         url="http://www-users.rwth-aachen.de/karman/Veranstaltungen/"
         accesskey="V"/>
  <menue name="Ansprechpartner" accesskey="T"/>
  <menue name="Sport/Kármán-Ligen"
```

```
        url="http://karmanliga.karman-aachen.de"
        accesskey="L"/>
    <menue name="Impressum" accesskey="I">
      <submenue name="Hilfe" titel="Infos zu diesen Webseiten"
                accesskey="H"/>
    </menue>
    &ausgaben;
  </webseite>
```

Im Prinzip ist der Aufbau dieser XML-Datei sehr einfach. Es gibt nur drei Elemente:

<webseite>

ist das Wurzel-Element und kann auch nur als solches vorkommen. Das notwendige Attribut wurzel-url gibt die URL zur Startseite an. (Man beachte, daß alle URLs in ZML mit einem Slash ‘/ ’ enden müssen, wenn sie nicht auf Dateien verweisen.) Das Element <webseite> kann nur <menue>-Elemente als Kinder haben.

<menue>

Dieses Element definiert einen Hauptmenüpunkt auf den Webseiten. Das notwendige Attribut name gibt dabei den Namen der Snippet-Datei an, auf die der Menüpunkt später verweisen wird. Solche Namen müssen eindeutig sein und dürfen keine Punkte enthalten. Am besten beschränkt man sich auf die 2x26 Groß- und Kleinbuchstaben.

<menue>-Elemente können <submenue> -Elemente enthalten.

Ist ein url -Attribut vorhanden, verweist dieser Menüpunkt stattdessen auf diese URL. Auch in diesem Fall muß ein eindeutiger name angegeben werden, aber es gibt dann kein Snippet mit diesem Namen.

Dann gibt es noch das titel -Attribut, welches einen alternativen Namen für den Menüpunkt angibt. Das ist nur nötig, wenn die auf der Webseite angegebene Bezeichnung für den Menüpunkt vom Dateinamen der Snippet-Datei abweicht.

Das accesskey -Attribut bestimmt den HTML-Access-Key für diesen Menüeintrag. Man muß selber darauf achten, daß das nicht mit anderen Access-Keys auf der Seite kollidiert. Zur Zeit benutzen die Kármán-Seiten beispielsweise F , N und P für die Navigation innerhalb einer Ausgabe. Die sind in den XSLT-Dateien verdrahtet.

Schließlich ist es möglich, zwei Sonderseiten anzugeben (es darf immer nur jeweils *eine* geben):

Setzt man startseite auf ‘ja ’, wird die entsprechende Seite auf den Position der Wurzel-URL gelegt. Normalerweise würde ja das Snippet zu ‘Aktuell ’ auf die Position ‘Aktuell/index.html ’ gelegt. Das bedeutet, daß *keine* Webseite auf

der Wurzel-URL zu finden ist. Dort müßte man also eine reine Weiterleitungsseite zu ‘Aktuell/index.html ’ installieren (so wie es z.B. Campuslife macht). Das ist unschön, daher gibt es dieses Attribut.

Setzt man archiv auf ‘ja ’, geht ZML davon aus, daß das dazugehörige Snippet ein Gesamtverzeichnis der Ausgaben und Artikel enthält. Dann wird dieses Snippet *mehrfach* generiert, einmal pro Kalender-Jahrgang der Zeitung. Die dabei entstehenden Dateinamen sind (relativ zur Wurzel-URL)

```
Archiv/index.html
Archiv/Archiv-2006.html
Archiv/Archiv-2005.html
Archiv/Archiv-2004.html
```

Dabei gehe ich von name="Archiv" aus (wie im obigen Beispiel). Man beachte außerdem, daß der Inhalt von ‘index.html ’ und ‘Archiv-2006.html ’ identisch wären, d.h. per Default sieht man immer zuerst das neueste Jahr im Archiv.

<submenue>
: definiert einen Untermenü-Punkt der Webseite. Die Attribute name, url , titel und accesskey haben dieselbe Bedeutung wie bei <menue>. Andere Attribute gibt es nicht. <submenue> -Elemente sind immer leer.

7.4 Snippets

Mit *Snippets* sind die einzelnen XHTML-Dateien im Zeitungs-Verzeichnis ‘snippets/ ’ gemeint. Im Prinzip können die alles mögliche sein, aber zur Zeit sind sie beschränkt auf die Zielseiten der Menü- und Untermenü-Punkte der Webseite, also z.B. Begrüßungsseite, Impressum, Archiv etc.

Ihr Aufbau entspricht im großen und ganzen dem eines normalen Artikels, see Abschnitt 4.8.1 [Normale Artikel] auf Seite 24. Es fehlt allerdings meist die Meta-Tabelle.

7.4.1 Die Linke Spalte definieren

Genau genommen gibt es nur eine sinnvolle Anwendung der Meta-Tabelle in Snippets, nämlich die Definition einer linken Spalte. Die linke Spalte ist ja entweder leer oder enthält das Untermenü, was aber häufig ebenfalls leer ist.

Gibt es im Snippet aber einen Meta-Tabellen-Eintrag namens ‘Linke Spalte ’, so wird dessen Inhalt für die linke Spalte verwendet. Vorsicht! Es wird dann auch kein Untermenü automatisch generiert. Wäre also eines vorhanden gewesen, muß man das dann in der Meta-Tabelle nachbilden, oder es fehlt eben.

Die Willkommensseite der Kármán-Seiten enthält beispielsweise folgende Meta-Tabelle:

```
<table>
<tbody>
<tr>
   <td>Linke  Spalte</td>
   <td>
      <div  style="width:  225px">
         <img  src="../webpool/hauptgebaeude.jpg"
               alt="Das  Hauptgebäude  der  RWTH  Aachen"
               style="display:  block"/>
         <p  style="text-align:  right;  font-size:  xx-small;  margin-top:  0;
                     font-style:  italic">
            Foto:  Hendrik  Brixius
         </p>
      </div>
   </td>
</tr>
</tbody>
</table>
```

Damit wird ein Foto des Hauptgebäudes links auf die Willkommensseite gesetzt.

style -Attribut für die Linke Spalte

Alternativ kann man sich auch darauf beschränken, ein style -Attribut für die Linke Spalte zu definieren, indem man das <td> -Element nur mit einem solchen ausstattet und ansonsten leer läßt:

```
<table>
<tbody>
<tr>
   <td>Linke  Spalte</td>
   <td  style="background-color:  blue"/>
</tr>
</tbody>
</table>
```

In diesem Beispiel wird der Hintergrund der Linken Spalte auf Blau gesetzt.

Man beachte, daß in diesem Fall der Inhalt der Linken Spalte durch nichts ersetzt wird (es wäre ja auch nichts da, mit dem man ihn ersetzen könnte), d.h. Untermenü oder Ausgaben-Inhaltsverzeichnis oder was auch immer würden erhalten bleiben.

7.4.2 Spezielle Einfügungen

An nahezu beliebigen Stellen in Snippets kann man spezielle s einfügen, die dann von ZML durch anderes, dynamisches Material ersetzt werden.

: fügt an dieser Stelle das Inhaltsverzeichnis der aktuellen Ausgabe ein.

: fügt an dieser Stelle das Inhaltsverzeichnis der Ausgabe 15 ein. Funktioniert natürlich auch mit allen anderen Ausgaben.

: fügt das aktuellste Impressum der Druck-Ausgaben ein.

: fügt eine komplette Liste aller bisheriger Ausgaben und Artikel ein. Wird dieses in demjenigen Snippet benutzt, für das in 'webseite.xml' das Attribut archiv="ja" gesetzt ist (typischerweise der Fall), wird dieses Gesamtverzeichnis nach Jahren gesplittet, see Abschnitt 7.3 [webseite.xml] auf Seite 40.

7.5 Das make-Skript

Mit dem Aufruf

```
../zml/tools/make-karman-website.py
```

aus dem Zeitungs-Verzeichnis heraus wird die Webseite in 'webpages/' auf den aktuellen Stand gebracht. Bei 'make-karman-website.py' handelt es sich um ein Python-Skript.

Die Schritte, die das Skript durchführt, sind im einzelnen:

1. Die Bilder zu den Artikeln werden von 'bilder/' nach 'webpages/Bilder/original/' kopiert. Das Verzeichnis 'webpool/' wird nach 'webpages/pool/' kopiert.

2. Mittels 'convert' von ImageMagick (PDF) und der Python Image Library (JPEG und PNG) werden Versionen mit definierter Breite von allen Bildern in 'bilder/' erzeugt. Dabei werden PDFs auf 500 px bzw. 100 dpi Breite und alle anderen Bitmaps auf 300 px Breite getrimmt.

3. Ab diesem Zeitpunkt werden keine Bitmaps mehr zur Webseite hinzugefügt. Daher werden jetzt für *alle* Bitmaps in 'webpages/' und seinen Unterverzeichnissen die Höhe- und Breiteninformationen ermittelt und diese Daten in der temporären Datei 'webpages/bilder.xml' für die XSLT-Prozesse zur Verfügung stellt. Sinn der Übung ist, daß dann in den HTML-Dateien der Webseite alle <img> -Elemente mit width - und height -Attributen ausgestattet werden können.

4. Nun wird die temporäre Datei ‘ausgaben.lst ’ angelegt, die in ‘webseite.xml ’ eingebunden wird, see Abschnitt 7.3 [webseite.xml] auf Seite 40. ‘ausgaben.lst ’ enthält bloß eine Auflistung aller bisherigen Ausgaben, ob nur im Web erschienen oder nicht.

   Es folgt der erste XSLT-Prozeß: Das Skript ‘xslt/ausgaben-sammeln.xsl ’ macht sich über die Liste der Ausgaben-Dateien in ‘webseite.xml ’ her.[1] Ziel ist es, die temporäre Datei ‘ausgaben.xml ’ zu bauen, welche nicht nur die Ausgaben, sondern auch alle in ihnen erschienen Artikel enthält. Dabei wird auch schon vorsortiert (nur die Artikel innerhalb der Ausgaben, nicht die Ausgaben selber) und gesplittete Artikel durch ihre Gesamtfassung ersetzt.

   Übrigens werden Ausgaben, die noch zu neu für’s Web sind, nicht berücksichtigt. Somit wird vermieden, daß eine noch nicht erschienene Ausgabe versehentlich auf der Webseite landet. Die Regel ist, daß der Tag des Skriptaufrufs frühestens der Erscheinungstag sein muß, damit die Ausgabe online geht.

   Somit steht den XSLT-Prozessen, die noch folgen, in ‘ausgaben.xml ’ eine Liste aller Ausgaben und Artikel zur Verfügung. Das ist beispielsweise wichtig, um in der linken Spalte der Webseiten ein Inhaltsverzeichnis anbieten zu können.

5. Jetzt kommen die Snippets an die Reihe. Anhand von ‘webseite.xml ’ werden die Startseite, die Menüpunkte und das (u.U. mehrseitige) Archiv angelegt.

6. Nun werden *alle* Artikel in ‘artikel/ ’ mittels XSLT in ihre endgültige Form gebracht und in ‘webpages/Artikel/ ’ abgelegt (see Abschnitt 7.5.1 [Saxon-Aufrufe] auf der nächsten Seite).

   Dabei wird durch einen Aufruf von HTML Tidy sichergestellt, daß es sich auch um gültiges XHTML handelt, das da produziert wird, und daß die Ergebnis-HTMLs maximal Browser-kompatibel und schön formatiert sind. (HTML Tidy kommt auch bei den Snippets zum Einsatz.)

7. Schließlich werden die Newsfeeds erzeugt. Dafür ist das Skript ‘xslt/destilate-feed.xsl ’ zuständig, welches einen Atom-1.0-Feed in ‘webpages/karman.atom ’ generiert. Atom eignet sich als sehr reichhaltiges, streng XML-basiertes und gut spezifiziertes Feed-Format gut, um als Ausgangspunkt für die anderen Feed-Formate zu dienen. Bislang gibt es allerdings nur deren eines: Mittels ‘xslt/atom2rss10.xsl ’ wird aus dem Atom-Feed ein RSS-1.0-Feed erzeugt. RSS 2.0 wäre auch recht leicht möglich, ist aber noch nicht geschehen. Und auch nicht dringend.

8. Als allerletztes werden die temporären Dateien weggeräumt.

[1] Es ist sinnvoll wenn auch nicht nötig, diese Liste in eine eigene XML-Datei zu verschieben, weil die aktuelle Lösung mit einer eingebetteten DTD in ‘webseite.xml ’ Probleme mit einigen XML-Parsern macht.

7.5.1 Zu den Saxon-Aufrufen

An dieser Stelle möchte ich ein paar Sätze zu den Herzstücken der Webseiten-Generierung verlieren, nämlich den XSLT-Verarbeitungen. Dafür kommt ja der Saxon zum Einsatz. Interessanter ist allerdings das, was sonst noch an dem Aufruf hängt.

Wird beispielsweise der Artikel ‘artikel/karlspreis-2005.xhtml ’ in die Webseite ‘webpages/Artikel/karlspreis-2005.html ’ umgewandelt, so findet folgender Aufruf statt:

```
saxon artikel/karlspreis-2005.xhtml xslt/zml2html/zml2html.xsl \
    name=karlspreis-2005.xhtml
```

Es wird mittels des Haupt-XSLT-Skriptes ‘xslt/zml2html/zml2html.xsl ’ die Quelldatei ‘artikel/karlspreis-2005.xhtml ’ verarbeitet. Das Ergebnis wird innerhalb des Python-Skriptes allerdings noch etwas nachbearbeitet, u.a. von HTML Tidy, See Abschnitt 2.1.6 [HTML Tidy] auf Seite 10.

Wichtig ist die Direktive ‘name=karlspreis-2005.xhtml ’. Das ist ein sogenannter XSLT-Parameter. Damit steht dem XSLT-Skript der Name der Datei zur Verfügung, die gerade verarbeitet wird. Anhand derer kann das Skript beispielsweise erkennen, ob es sich um einen Artikel oder ein Snippet handelt – Snippets haben hier nämlich keine Dateiendung, da nur das name-Attribut übergeben wird, see Abschnitt 7.3 [webseite.xml] auf Seite 40.

Außerdem erkennt das XSLT-Skript beim prozessieren des Archivs (zu erkennen an ‘archiv="ja" ’ in ‘webseite.xml ’, see Abschnitt 7.3 [webseite.xml] auf Seite 40), anhand dieses name-Parameters, welches Jahr gerade prozessiert wird. Das wird nämlich angehängt: ‘Archiv-2005 ’, ‘Archiv-2006 ’, etc.

Technisch gesehen ist der name-Parameter identisch mit der Kármán-URL, see Abschnitt 7.7 [Kármán-URLs] auf Seite 48.

Offline-Webseiten

Manchmal kann es praktisch sein, die Seiten ohne Webserver offline im Browser überprüfen zu können. Dabei ist es ein Problem, daß beim Verweis auf ein Verzeichnis vom Browser nicht automatisch die Datei ‘index.html ’ in diesem Verzeichnis geöffnet wird. (Der Webserver würde diese Datei automatisch liefern.)

Um Abhilfe zu schaffen, gibt es den Parameter ‘--offline ’ für das make-Skript. Damit werden Webseiten gebaut, die sich offline in einem Browser lesen lassen. Eventuell ist es nötig, vorher ‘webpages/ ’ zu löschen, damit wirklich alles neu erstellt wird. Diese Webseiten sollte man nicht hochladen, auch wenn sie funktionieren würden.

Will man wieder auf Online-Webseiten umschalten, sollte man ebenfalls vorher ‘webpages/ ’ löschen.

7.6 Newsfeeds

Zur Zeit bietet ZML Newsfeeds in den Formaten Atom 1.0 (http://www.atomenabled.org/) und RSS 1.0 (http://web.resource.org/rss/1.0/) an.

RSS ist mit großen Abstand das beliebteste Feed-Format. Leider gibt es neun Untervarianten, die alle mehr oder weniger untereinander inkompatibel sind (teilweise zu sich selber). Das am besten spezifizierte RSS ist die Version 1.0, die vom W3C unterstützt wird. Leider ist das nicht die beliebteste, das ist die Version 2.0, die am W3C vorbei-spezifiziert wurde und daher vielen – auch mir – ein Dorn im Auge ist. Dennoch ist gegen einen guten RSS-2.0-Feed nichts einzuwenden, wenn ihn jemand macht.

Atom ist ein junges, hervorragendes Feed-Format, das sowohl vom W3C, als auch von der IETF unterstützt wird. Es wird von ZML aber nicht bloß aus idealistischen Gründen angeboten: Zum einen wird es auch in großem Maßstab eingesetzt, es wird von vielen Feed-Clients verstanden, Google nutzt es für seine Blogs, und es eignet sich als Ausgangsformat, um mit minimalem Aufwand in andere Feed-Formate umgewandelt zu werden. (Das XSLT-Skript zur Umwandlung Atom => RSS ist gerade mal 60 Zeilen lang.)

7.6.1 Inhalt des Feeds

Grundsätzlich werden alle Artikel der letzten Zeitungs-Ausgabe in den Feed aufgenommen, ein Feed-Eintrag pro Artikel. Artikel früherer Ausgaben sind nicht Teil des aktuellen Feeds.

Zusätzlich wird ein Übersichtseintrag erstellt, der auf die neue Ausgabe als Ganzes verweist.

Hinzu kommen Extra-Einträge der Datei ‘extra-feed.xml ’, die nicht älter sind als der Stichtag. Der Stichtag wiederum ist das Datum der aktuellen Ausgabe minus 14 Tage.

Alle Einträge, die zur aktuellen Ausgabe gehören (inklusive des Übersichtseintrags) sind auf das Erscheinungsdatum dieser Ausgabe datiert. Die Extra-Einträge tragen das Datum, das in ‘extra-feed.xml ’ angegeben ist.

Das Datum des ganzen Feeds wird bestimmt durch den Eintrag mit dem neuesten Datum.

7.6.2 Extra-Einträge im Feed

Es kann u.U. sinnvoll sein, die Leser des Feeds auf bestimmte Dinge aufmerksam zu machen, die nicht in der aktuellen Ausgabe sein. Das können z.B. Meldungen in eigener Sache sein. Aus diesem Grund existiert die Datei ‘extra-feed.xml ’ im ZML-Wurzelverzeichnis. Im (fast) einfachsten Fall sieht diese Datei so aus:

```
<feed>
  <entry  id="neujahrsgruss-2006"  date="2005-12-27"/>
</feed>
```

Man muß also für einen Extra-Eintrag erst einmal einen Artikel anlegen, in diesem Fall ‘artikel/neujahrsgruss-2005.xhtml ’, und dann trägt man den hier ein. Das Datum im Attribut date muß nicht mit dem Datum im Artikel übereinstimmen, vielmehr sollte es das Datum sein, an dem der Artikel dem Feed hinzugefügt wurde.

Danach erzeugt man mit ‘make-karman-website.py ’ neue Webseiten. Es ist nicht nötig, daß damit auch gleichzeitig eine neue Zeitungs-Ausgabe hochgeladen wird. Typischerweise wird der Feed-Eintrag sogar das einzige sein, das neu hinzugekommen ist.

7.7 Kármán-URLs

In Artikeln und Snippets ist es bei Querverweisen möglich, sogenannte Kármán-URLs anzugeben. Diese beginnen statt mit ‘http:// ’ mit ‘karman:// ’. Was dann folgt, ist kein Pfad, sondern ein einzelner Name. Es gibt folgende Möglichkeiten, die am besten anhand von Beispielen erklärt werden:

karlspreis-2005.xhtml
: Endet die Kármán-URL auf ‘.xhtml ’, verweist sie auf einen Artikel.

gym-einweihung-a-2005.jpg
: Handelt es sich um den Dateinamen eines Bildes, weist die Kármán-URL auf dieses Bild in ‘bilder/ ’. Achtung: Auf Bilder im Webpool kann so nicht verwiesen werden.

Impressum
: Enthält die Kármán-URL keinen Punkt, wird angenommen, daß es sich um das name-Attribut eines Menüpunktes handelt. In diesem Fall verweist die Kármán-URL auf die Seite, auf die der Menüpunkt führt.

Archiv-2004
: Hat die Kármán-URL die Form Archiv- Jahr , verweist sie auf eine Jahresseite im Archiv. Die URL karman://Archiv verweist hingegen auf das Archiv als solches.

  Das gilt allerdings nur, wenn das Archiv auch wirklich ‘Archiv ’ heißt. Das wird bestimmt in der Datei ‘webseite.xml ’, see Abschnitt 7.3 [webseite.xml] auf Seite 40.

Kármán-URLs sollten in Artikeln und Snippets benutzt werden, um innerhalb der Website zu verlinken, allerdings nur im href -Attribut des <a>-Elements. Insbesondere können sie (noch) nicht in <img>-Elementen benutzt werden. Ihr Nutzen in Beiträgen ist daher vergleichsweise gering.

Intern werden sie im ZML-System allerdings intensiv eingesetzt, um die relativen Pfade zwischen den einzelnen Webseiten zu berechnen. Beim bauen der Webseiten wird die Kármán-URL (ohne das ‘karman:// ’) bei jedem XSLT-Prozeß als XSLT-Parameter „name“ übergeben, see Abschnitt 7.5.1 [Saxon-Aufrufe] auf Seite 46.

7.8 Customisation-Layers

Customisation-Layers sind die Methode der Wahl, die originalen XSLT-Stylesheets, die ja für die Kármán Hochschulzeitung optimiert sind, für eine andere Zeitung anzupassen. Sie befinden sich im Zeitungs-Verzeichnis im Unterverzeichnis ‘pool/ ’.

‘html-local.xsl ’ dient zur Anpassung der Webseiten.

‘latex-local.xsl ’ dient zur Anpassung der mit LaTeX erzeugten Print-Ausgaben.

‘feed-local.xsl ’ dient zur Anpassung der Feeds. ‘rss10-local.xsl ’ ist für die spezielle Anpassung des RSS-1.0-Feeds, allerdings sollte diese Datei nicht nötig sein; man kann alles wichtige im RSS-Feed auch über ‘feed-local.xsl ’ ändern.

Ich kann hier nicht die genaue Vorgehendweise erklären, dafür möge man sich eine XSLT-Anleitung besorgen. Wichtig ist nur, daß alles, was in diesen ‘...-local.xsl ’-Dateien angegeben wird, alles andere in den originalen Stylesheets überschreibt. Auf diese Art und Weise kann man die Anpassung vornehmen. Der Vorteil gegenüber einer direkten Modifizierung der Stylesheets ist, daß neuere Versionen von ZML direkt verwendet werden können. Mit anderen Worten, man profitiert von Bugfixes und anderen Verbesserungen, muß aber nicht jedesmal wieder das System an die eigene Zeitung anpassen.

Um diesen Prozeß noch weiter zu vereinfachen, sind die ‘...-local.xsl ’-Dateien bereits mit allen wichtigen Templates gefüllt, die mir eingefallen sind. Sie sind allerdings alle auskommentiert. Man kann sie nun Schritt für Schritt einkommentieren und an die eigenen Bedürfnisse anpassen. So müßte die Anpassung einigermaßen schmerzfrei und vor allem rasch zu bewerkstelligen sein.

8 Ein typischer Arbeitszyklus

Steht die Erstellung einer neuen Ausgabe der Zeitung an, so sind typischerweise folgende Schritte durchzuführen.

8.1 Schritt 1: Erzeugen der XHTML-Dateien

1. Die Artikel, die in die aktuelle Ausgabe kommen sollen, sind nach dem Redaktionsschluß auf dem Kármán-Wiki (http://www.karman-aachen.de/wiki) einzufrieren, d.h. der Besitzer (Owner) ist auf root zu setzen und die Artikel sind als nur-lesbar zu markieren.

2. Die Artikel sind aus dem Wiki auf die Festplatte in das Verzeichnis ‘artikel/ ’ zu kopieren. Für die Namensgebung see Abschnitt 4.1 [Dateiname eines Artikels] auf Seite 17, wobei sie vorerst die Dateiendung ‘.pwiki ’ bekommen.

3. Nun sind die phpWiki-Dateien in das XHTML-Format umzuwandeln. Dabei kann das Python-Skript ‘tools/phpwiki2html.py ’ gute Dienste leisten, wenn auch nicht perfekte. Befindet man sich im Verzeichnis ‘artikel/ ’ und möchte die Datei ‘karlspreis-2005.pwiki ’ umwandelt, so gibt man ein

   ```
   ../tools/phpwiki2html.py karlspreis-2005.pwiki
   ```

   und es wird die Datei ‘karlspreis-2005.xhtml ’ angelegt.

   Diese Datei muß man dann nachbearbeiten, d.h. eventuell von zu langen Zeilen befreien und einigen Whitespace stutzen. Man beachte dabei insbesondere die Beschreibung des Dateiformats, see Kapitel 4 [Artikel] auf Seite 17, sowie Kapitel 5 [Typographische Richtlinien] auf Seite 31.

4. Schließlich wird die Datei in das CVS eingecheckt.

Die XHTML-Fassung eines Artikels ist die *eigentliche* Fassung. Die Fassung im Wiki ist ab dem Zeitpunkt des Einfrierens obsolet und könnte theoretisch sogar weggeschmissen werden.

8.2 Schritt 2: das Print-Layout

Dies ist der schwierigste Schritt, weil man darin erst einmal etwas Übung bekommen muß. Es geht darum, eine Datei namens ‘ausgabe- #.xml ’ in ZML-Wurzelverzeichnis zu erstellen, wobei »#« natürlich durch die Ausgabennummer ersetzt werden muß. Für das genaue Dateiformat see Kapitel 6 [ZML] auf Seite 35.

Es ist am einfachsten, die Datei der letzten Ausgabe als Ausgangspunkt zu nutzen. Dann fügt man alle Artikel hinzu und verteilt sie in einem ersten Versuch auf beide Seiten, selbst wenn das überhaupt nicht paßt.

So gewinnt man einen Überblick, wie die Ausgabe hinhauen könnte. Am besten arbeitet man in dieser Phase eng mit dem Chefredakteur zusammen, da u.U. Artikel auf die nächste Ausgabe geschoben werden müssen oder umgekehrt. (Instant Messaging ist hier das Kommunikationsmittel der Wahl.) Zwischendurch kann man mit einem Aufruf wie

```
./zmltopdf 10 && cp ausgabe-10.pdf ~/public_html/ \
                && sitecopy -u www-users
```

den aktuellen Stand für den Chefredakteur verfügbar machen.

Wenn das Endlayout zumindest absehbar wird, sollte man den Veranstaltungskalender splitten (es sei denn natürlich, man kommt mit einer Spalte hin, was eher selten der Fall ist) und in die endgültigen Spalten gießen. Schließlich werden die Artikel an ihre endgültige Position gerückt.

Das wichtigste in dieser Phase sind die Möglichkeiten, Artikel ein wenig zu stutzen oder zu dehnen. Folgende Varianten stehen zur Verfügung und können u.U. kombiniert werden. Dabei sind sie sortiert von „sanft“ nach „vermeiden, wenn’s geht“:

- Artikel dehnen oder stauchen mit dem dehnen -Attribut, see Abschnitt 6.4 [Das Element beitrag] auf Seite 36.
- Die kleine Grafik mit der Selbstwerbung für Kármán einfügen. Das geht mit

  ```
  <beitrag datei="karman-recruit" stil="anzeige"/>
  ```

- Den jeweils anderen Stil zwischen den Kinoprogramm-Stilen ‘kinoprogramm ’ und ‘kinoprogramm-kompakt ’ wählen.
- Zu einigen oder allen Filmen noch eine mehr oder weniger ausführliche Beschreibung beifügen.
- Einzelne Veranstaltungen streichen, oder nicht ganz so wichtige noch hinzunehmen.
- Bilder hinzunehmen oder weglassen.
- Abstände explizit einfügen, see Abschnitt 6.6 [Das Element abstand] auf Seite 38.

- Den Artikel kürzen mit web-only , see Abschnitt 4.4 [Medien-abhängige Verarbeitung] auf Seite 20. Das sollte natürlich nur in Rücksprache mit dem Autor und dem Chefredakteur erfolgen.
- Artikel in die nächste Ausgabe verschieben, bzw. Artikel, die eigentlich für die nächste Ausgabe geplant waren und schon fertig sind, in die aktuelle nehmen. Auch das nur mit Rücksprache mit dem Chefredakteur.

Wenn alles erledigt ist, wird die 'ausgabe- #.xml '-Datei eingecheckt (ebenso wie alle Artikel und Bilder, mit denen das noch nicht geschehen ist) und das PDF per Email an die Druckerei geschickt. Außerdem wird das PDF nach 'ausgaben/ ' kopiert und dort eingecheckt.

8.3 Schritt 3: die Webseiten

Als Belohnung für die Mühen des vorigen Schritts ist dieser nun recht simpel. Einfach

```
tools/make-karman-website.py
```

aufrufen, warten und glücklich sein. Für die Details see Abschnitt 7.5 [make-Skript] auf Seite 44.

9 Index