Workshop: XML-basierte Webseiten mit Website-DTD

Seiten-Aufbau

XML und XSL-Stylesheets sind die Mittel der Wahl, um auf Webseiten Inhalt und Layout sauber getrennt zu halten. Das Framework Website-DTD hilft dabei. Oliver Fischer

HTML-Editoren nehmen einem Web-Programmierer mitunter mehr aus der Hand, als sie ihm nützen. Warum es nicht mal mit XML und XSL versuchen? Die Trennung von Layout und Inhalt klingt viel versprechend, doch die Vorstellung, ein eigenes Schema und die zugehörigen XML-Stylesheets zu entwickeln, verhindert so manchen Umstieg. Alternativ zur Eigenentwicklung bietet sich Website-DTD als Komplettlösung an, für die federführend Norman Walsh verantwortlich ist, bekannt durch seine Arbeit an der Docbook-DTD und den Docbook-Stylesheets.

Es verwundert also nicht, dass Website-DTD und Docbook eng verwandt sind. So ist die Document Type Declaration (DTD) von Website eine Spezialisierung der Docbook-DTD durch Elemente, die zum Aufbau einer Webseite notwendig sind. Durch diese Beziehung stehen Hunderte von Auszeichnungselementen von »a« wie Acronym bis »y« wie Year zur Verfügung. Elemente mit »z« sind derzeit nicht vorhanden.

Nach dem gleichen Muster verfahren die mit der DTD zusammen bereitgestellten Stylesheets: Sie definieren die Webseiten-spezifischen Templates, für die restlichen Elemente werden die Docbook-Stylesheets eingebunden. Um einen XML-basierten Netzauftritt zu realisieren, sind einige Vorbereitungen nötig.

XSLT-Prozessor wählen

Für die spätere Umformung der XML-Dokumente in HTML-Dateien ist ein aktueller XSLT-Prozessor wie Xalan oder der in der XSLT-C-Bibliothek des Gnome-Projekts Libxslt enthaltene »xsltproc« notwendig. Letzterer ist schnell und wahrscheinlich wegen seiner Verknüpfung mit Gnome auf den meisten Systemen standardmäßig installiert. Daher findet er auch für alle Beispiele in diesem Artikel Verwendung. Sollten Sie keine Binärpakete für Ihre spezielle Distribution finden, verwenden Sie die Quellen der Bibliothek unter [1] für den Eigenbau des Prozessors.

Zusätzlich empfiehlt es sich, die Libxml2 zu installieren, weil dann das Tool »xmllint« zur Verfügung steht. Verwenden Sie dabei nicht die Version 2.6.1, da diese einen Bug bei der Auflösung von Namensräumen aufweist. Bei »xmllint« handelt es sich um ein Programm zur Validierung von XML-Dokumenten. Natürlich können Sie aber jeden anderen XSLT-Prozessor zusammen mit Website-DTD einsetzen.

XML-Theorie

XML-Dokumente verweisen mit Hilfe der Doctype-Deklaration über eine URL auf ihre DTD, die beispielsweise auf einem Webserver zum Abruf bereitliegt. Ebenso können Sie über eine URL XSL-Stylesheets einbinden, die sich irgendwo im Internet befinden. XSLT-Prozessoren wie die erwähnten Xalan und »xsltproc« laden XSLs und DTDs aus dem Internet, wenn diese über eine gültige URL referenziert sind.

Somit ist es eigentlich nicht notwendig, sie lokal zu installieren. Dennoch sollten Sie genau dies tun: Komplexe XSL-Stylesheets bestehen oft aus vielen einzelnen Dateien und der XSLT-Prozessor lädt sie alle im schlechtesten Fall bei jedem Lauf. Sie können sich den gewaltigen Umfang der damit verbundenen Downloads leicht verdeutlichen, wenn Sie bedenken, dass Website-DTD als Spezialisierung der Docbook-DTD deren XSL-Stylesheets mit über 300 einzelnen Dateien einbindet.

Laden Sie zunächst über den Download-Service von Sourceforge [2] die aktuellen Versionen von Website-DTD und der Docbook-Stylesheets sowie von der Docbook-Webseite [3] die aktuelle Docbook-XML-DTD herunter. Sobald alle

Pakete lokal auf dem Rechner liegen, entpacken Sie sie in ein gemeinsames Verzeichnis. Bei der Docbook-DTD ist zu beachten: Das Zip-Archiv entpackt sich, wenn Sie es nicht anders angeben, in das aktuelle Verzeichnis.

Die Webseiten

Im Anschluss teilen Sie den beteiligten Programmen noch mit, dass Sie die lokale Installation statt der eigentlich referenzierten verwenden wollen. Das geschieht über einen so genannten XML-Katalog. Dieser enthält die Regeln für die Umsetzung von Uniform Ressource Identifiern (URI) (eine URL ist ebenfalls ein URI) und von öffentlichen Bezeichnern der DTDs. Der für diesen Workshop erstellte Beispielkatalog ist in Listing 1 zu sehen.

Abschließend setzen Sie die Umgebungsvariable »XML_CATALOG_FILES« so, dass sie auf die Katalogdatei zeigt. Die Programme »xsltproc« und »xmllint« binden die Informationen dann ein.

Wenn Sie mehr über XML-Kataloge wissen möchten, finden Sie unter [4] weitere Informationen.

Jedes XML-Dokument repräsentiert eine separate Webseite, die sich aus zwei logischen Bereichen zusammensetzt: dem Kopf – eingeschlossen in ein Head-Element – mit den Metadaten der Seite sowie dem Inhalt, aus dem später der Inhalt des Body-Elements der HTML-Datei erzeugt wird. Listing 2 zeigt so ein minimales XML-Dokument.

Der Eintrag »encoding = "iso-8859-15"« in der XML-Deklaration weist XSLT-Prozessoren an, Umlaute als Bestandteil des Zeichensatzes zu erlauben. Sonst müssten Sie diese Zeichen als Entitäten (»ä«) eingeben. Die sich anschließende Doctype-Deklaration legt die DTD für das Dokument fest, wobei in diesem Fall die Standard-DTD von Website zum Einsatz kommt.

Diese DTD stellt neben den für Website spezifischen Elementen auch eine Teilmenge der Docbook-DTD bereit. Möchten Sie Zugriff auf die gesamte Docbook-DTD haben, ändern Sie die Doctype-Deklaration wie folgt ab:

```
<!DOCTYPE webpage PUBLIC "-//Norman
Walsh//DTD Website Full V2.5.0//EN"
"http://docbook.sourceforge.net/release/webs
ite/2.5.0/schema/dtd/website-full.dtd">
```

Neben seiner Funktion als Wurzelelement erfüllt »<webpage>« über seine beiden Attribute »id« und »lang« zwei weitere Aufgaben: So stellt zum Beispiel »id = "kosmos"« einen für die Menügenerierung später notwendigen eindeutigen Bezeichner bereit. Zweitens aktiviert das Attribute »lang = "de"« länderspezifische Anpassungen, zum Beispiel den Gebrauch deutscher Bezeichnungen durch die Stylesheets (Tabelle statt Table) und den richtigen Umgang mit Anführungszeichen.

Die weiteren Elemente innerhalb des Elements »<head>« enthalten alle Metadaten für die zu erzeugende Seite. So wird der Inhalt von »<title>« genutzt, um den Titel des HTML-Dokuments und eine Seitenüberschrift zu er-

Listing 1: XML-Katalog

```
01 <?xml version='1.0'?>
02 <!DOCTYPE catalog PUBLIC "-//OASIS/DTD Entity Resolution
   XML Catalog V1.0//EN"
03  "http://www.oasis-open.org/committees/entity/release/1.0/catalog.dtd">
04 <catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
05   <rewriteSystem
06     systemIdStartString="http://docbook.sourceforge.net/release/xsl/
   1.64.1"
07     rewritePrefix="docbook-xsl-1.64.1" />
08
09   <rewriteSystem
10     systemIdStartString="http://docbook.sourceforge.net/release/xsl/
   current"
11     rewritePrefix="docbook-xsl-1.64.1" />
12   <rewriteSystem
13     systemIdStartString="http://www.oasis-open.org/docbook/xml/4.2"
14     rewritePrefix="docbook-xml-4.2" />
15   <rewriteSystem
16     systemIdStartString="http://docbook.sourceforge.net/release/
   website/2.5.0"
17     rewritePrefix="website-2.5.0" />
18 </catalog>
```

Listing 2: Einfaches Website-Dokument

```
01 <?xml version="1.0" encoding="iso-8859-15"?>
02 <!DOCTYPE webpage PUBLIC "-//Norman Walsh//DTD Website Full V2.5.0//EN"
03  "http://docbook.sourceforge.net/release/website/2.5.0/schema/dtd/web-
   site-full.dtd">
04
05 <webpage id="kosmos" lang="de">
06   <head>
07     <title>Sonne und Kosmos</title>
08     <titleabbrv>Sonne</titleabbrv>
09     <keywords>Sonne, Stern, ...</keywords>
10   </head>
11
12   <para id="sonne.eins.text" xreflabel="Die Sonne">Die Sonne ist für
13     die astronomische und astrophysikalische Forschung nicht nur das
14     auf der Erde Leben ermöglichende Zentralgestirn des Sonnensystems,
15     sondern auch der einzige Stern, auf dessen Oberfläche Details
16     erkennbar sind. Daraus erklärt sich auch die Sonderstellung, die
17     die Sonnenforschung innerhalb der Erforschung des Weltalls
18     einnimmt.</para>
19
20   <para id="sonne.eins.quelle">Der Text stammt aus dem <ulink
21       url="http://www.google.de">Netz</ulink>.</para>
22
23   <mediaobject>
24     <imageobject>
25       <imagedata fileref="../bilder/sonne.gif" format="GIF"
26         contentdepth="174" contentwidth="200" />
27     </imageobject>
28     <textobject>
29       <phrase>Die Sonne.</phrase>
30     </textobject>
31     <caption>
32       <para>Die Sonne ist der Stern in unserem System.</para>
33     </caption>
34   </mediaobject>
35 </webpage>
```

zeugen, und aus dem Element »<keyword>« wird »<meta name="keyword"...>«.

An das Element »<head>« schließt sich der eigentliche Inhalt an, den Sie durch die verfügbaren Docbook-Elemente auszeichnen können. So umfasst das Element »<para>« beispielsweise einen Absatz und ist somit das Äquivalent zum »<p>« in HTML. Über »<section>« definieren Sie Abschnitte mit einer Überschrift. Mit Hilfe von »<orderedlist>« und »<itemizedlist>« zeichnen Sie Listen aus und »<programlisting>« ist für genau das gedacht, wonach es klingt.

Navigation erleichtern

Der Umfang von Docbook – es besteht aus immerhin knapp 400 Elementen – sollte für fast jeden Anwendungsfall ausreichend sein. Die beste Referenz für alle Docbook-Elemente ist Norman Walshs „Docbook: The Definitive Guide“, das unter [5] online steht.

Zentrales Element für jeden Internetauftritt ist das Menü, es bestimmt die Navigation. Website entnimmt alle Informationen aus einem separaten XML-Dokument, das Sie als »layout.xml« im Basisverzeichnis der Webseite anlegen. Neben dem Menü legen Sie hier fest, aus welchem XML-Dokument Sie welche HTML-Datei in welchem Verzeichnis erzeugen. Zusätzliche Konfigurationsinformationen nehmen Sie über das »<config>«-Element) auf. Innerhalb von »layout.xml« leiten Sie mittels »<toc>« die Beschreibung des Menüs ein, das sich in den später erzeugten Seiten genauso wiederfindet.

Die Elemente unterhalb von »<toc>« mit dem Namen »<tocentry>« repräsentieren jeweils einen einzelnen Menüpunkt und lassen sich verschachteln. Das Element »<toc>« selbst nimmt die Angaben zur Startseite auf. Beide Elemente kennen die Attribute »dir«, »page« und »filename«, die der Prozessor für die Seitenerzeugung nutzt.

Das Attribut »page« nennt das Quelldokument, »filename« die zu erzeugende HTML-Datei und »dir« erlaubt es, das Zielverzeichnis für die HTML-Datei zu bestimmen. Dieses Verzeichnis muss allerdings vor der Erzeugung der Datei existieren. Ist das Attribut »dir« zwar nicht für das aktuelle Element »<tocentry>« definiert, aber für ein übergeordnetes, übernimmt der Prozessor dessen Wert für alle Tocentry-Elemente. Die jetzt noch fehlenden Menütexte für die Webseite übernimmt das Stylesheet aus dem Element »<title>« der Quelldokumente. Ist dieser Text zu lang, geben Sie alternativ per »<titleabbrev>« einen kürzeren Titel an.

Abweichend davon erscheint der Eintrag für den Link auf die Startseite am Anfang des Menüs nicht als Text, sondern als Grafik. Die beiden in Listing 3 gesetzten Parameter »banner-tabular« und »homebanner-tabular« bestimmen, welche Grafiken dafür verwendet werden. Die über »homebanner-tabular« angegebene Grafik erscheint auf der Startseite selbst, die über »banner-tabular« festgelegte auf allen anderen.

Wichtig: Für die richtige Darstellung des Menüs benötigen Sie Grafiken, die im Verzeichnis »example/graphics« der Website-Distribution liegen. Es empfiehlt sich, das gesamte Verzeichnis in das Basisverzeichnis der eigenen Seiten zu kopieren.

Automatisch generieren

Für die Erzeugung der einzelnen HTML-Dokumente kommt ein Makefile wie in Listing 4 zum Einsatz. Es automatisiert die einzelnen Schritte vollständig. Im ersten Schritt generieren Sie damit aus »layout.xml« die temporäre Datei »autolayout.xml«, die alle wichtigen Informationen aus den XML-Dokumenten und »layout.xml« zusammenführt. Die anschließend erzeugte Datei »depends.tabular« enthält automatisch generierte Abhängigkeitsbeschreibungen. Den eigentlichen Transformationsprozess beschreibt die Regel »%.html«. Mit Hilfe von »make« bauen Sie jetzt alle Seiten in einem Durchgang.

Alternativ gibt es noch die Möglichkeit, alle beschriebenen Schritte über einen XSLT-Prozessor ausführen zu lassen.

Listing 3: Layout des Webauftritts

```
01 <?xml version="1.0"?>
02 <!DOCTYPE layout PUBLIC "-//Norman Walsh//DTD Website Layout
   V2.5.0//EN"
03  "http://docbook.sourceforge.net/release/website/2.5.0/schema/dtd/
   layout.dtd">
04 <layout>
05   <config param="banner-tabular" value="graphics/nachhause.png"
06      altval="Zur Startseite"/>
07   <config param="homebanner-tabular" value="graphics/zuhause.png"
08      altval="Die Startseite"/>
09
10   <toc page="index.xml" filename="index.html">
11     <tocentry page="sonne/start.xml" filename="index.html" dir="sonne">
12       <tocentry page="sonne/eins.xml" filename="eins.html"/>
13       <tocentry page="sonne/zwei.xml" filename="zwei.html"/>
14       <tocentry page="sonne/drei.xml" filename="drei.html"/>
15     </tocentry>
16       ...
17     <tocentry page="kosmos.xml" filename="kosmos.html"/>
18   </toc>
19 </layout>
```

Abbildung 1: Eine Webseite, die mit den Standard-Stylesheets erzeugt wurde, sieht trotz ausreichender Funktionalität nicht sehr ansprechend aus.

Norman Walsh hat hierzu für Xalan und Saxon Erweiterungen geschrieben, deren Einsatz in dem zur Distribution gehörenden Beispiel beschrieben sind. Makefiles bieten allerdings den Vorteil, dass Sie weitere Schritte wie das Hochladen der Seiten auf einen Webserver ebenfalls automatisieren können.

Links auszeichnen

Ohne Links, egal ob interne oder externe, kommt keine Seite aus. Daher bietet auch Website die Möglichkeit an, zwischen einzelnen Seiten zu verlinken und Querverweise auf andere Seitenabschnitte anzulegen. Voraussetzung dafür ist, dass das Verweisziel und der Verweis selbst in den Quelldokumenten eindeutig identifizierbar sind.

Wollen Sie auf eine andere Stelle innerhalb desselben Dokuments verweisen, müssen Sie diese zuerst eindeutig markieren. Zu diesem Zweck setzen Sie für jedes XML-Element das bereits erwähnte Attribut »id«. Dessen Wert muss innerhalb eines Dokuments eindeutig sein. Falls Sie später zwischen separaten Seiten verweisen möchte, sollten die IDs so aufgebaut sein, dass sie in der gesamten Webseite eindeutig sind.

Querverweise innerhalb einer Seite lassen sich über die Elemente »xref« und »link« anlegen. Beide Elemente kennen das Attribut »linkend«, dem Sie den Wert des »id«-Attributs des Ziels zuweisen. Die Elemente »xref« und »link« unterscheiden sich nur in der Art, wie der Text erzeugt wird, auf dem der Verweis liegt. Während »link« dafür den Inhalt zwischen seinem Anfangs- und End-Tag nutzt, gewinnt »xref« den Text über die Stylesheets.

Verweisen Sie beispielsweise auf ein Section-Element, übernimmt das Stylesheet den Inhalt des Elements »title« aus diesem Abschnitt. Da Sie diesen Mechanismus nicht auf alle Elemente anwenden können, darf jedes Element zusätzlich das Attribut »xreflabel« aufnehmen, dessen Inhalt das Stylesheet alternativ als Linktext nutzt.

Dokumente verbinden

Beide Elemente – also »xref« und »link« – können Sie aber nur für Verweise innerhalb derselben Seite (desselben XML-Dokuments) nutzen. Verweise zwischen einzelnen Seiten eines Interntauftritts erfordern den Einsatz von so genannten Olinks. Bei diesen Links kommt im Gegensatz zu »xref« und »link« kein XML-Sprachmittel direkt zum Einsatz, sondern eine Datenbank, in der Sie alle Elemente mit gesetztem ID-Attribut ablegen. Im Beispiel-Makefile bewerkstelligt dies das Ziel »olink.db.xml«.

Innerhalb des verweisenden Dokuments fügen Sie anstelle von »xref« oder»link« ein Element »<olink>« mit den Attributen »targetdoc« und »targetptr« ein. Dabei nennt »targetdoc« das Zieldokument, das über den Wert des ID-Attributs eines Webpage-Elementes benannt wird. Mit »targetptr« weisen Sie die ID des Zielelements im referenzierten Dokument zu. Wenn Sie die HTML-Seiten durch Stylesheets erzeugen, müssen Sie nicht nur das Quelldokument, sondern zusätzlich noch die Olink-Datenbank laden, um auf alle notwendigen Informationen zugreifen zu können.

Olinks prüft leider nicht, ob das Ziel überhaupt existiert, und erzeugt auch blinde Links (Links ohne Zielangabe). Das geschieht besonders dann, wenn sich die Verweise und ID-Werte in den Dokumenten ändern, Sie das Dokument »olink.db.xml« aber nicht neu erzeugen. Verweise auf fremde Internetadressen erfordern den Einsatz des Elements »<ulink>«, dessen Attribut »url« die Ziel-URL aufnimmt.

Über das Element »mediaobject« nehmen Sie Bilder in eine Seite auf. Da »mediaobject« als Universalcontainer für unterschiedliche Medienarten (und Ausgabemedien) ausgelegt ist, benötigen Sie – wie aus den Beispielquellen ersichtlich

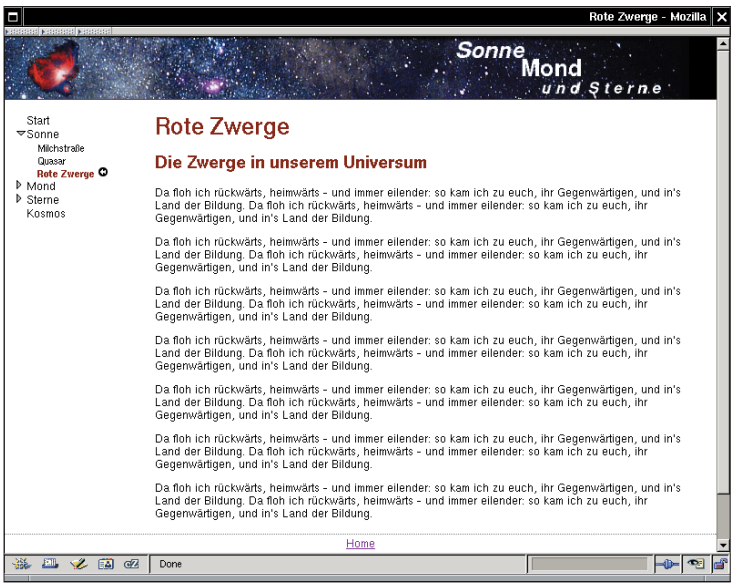

Abbildung 2: Mit etwas CSS und ein paar kleinen Änderungen an den XSL-Templates geben Sie den Seiten ein zeitgemäßes Design und profitieren trotzdem von den Stärken von Website.

Listing 4: Makefile

```
01 XSL_BASIS=http://docbook.sourceforge.net/release/website/2.5.0/xsl
02 PWD := $(shell pwd)
03
04 .SUFFIXES :
05 .PHONY    : clean
06
07 alles: depends.tabular olink.db.xml
08         @$(MAKE) website
09
10 -include depends.tabular
11
12 autolayout.xml: layout.xml
13         xsltproc --nonet -o $@ \
14         $(XSL_BASIS)/autolayout.xsl $<
15
16 olink.db.xml: autolayout.xml
17         xsltproc --nonet -o $@ $(XSL_BASIS)/website-targets.xsl $<
18
19 %.html: autolayout.xml
20         xsltproc --nonet -o $@ \
21         --stringparam autolayout-file "$(PWD)/autolayout.xml" \
22         --stringparam website.database.document "$(PWD)/olink.db.xml" \
23         treiber.xsl \
24         $(filter-out depends.tabular,$(filter-out autolayout.xml,$^))
25
26 depends.tabular: autolayout.xml
27         xsltproc -o $@ $(XSL_BASIS)/makefile-dep.xsl $<
```

ist – weitere Elemente, um alle erforderlichen Informationen unterzubringen. Das unter »imageobject« befindliche Element »imagedata« referenziert über »fileref« die Grafikdatei. In »contentdepth« und »contentwidth« legen Sie die Bildhöhe und -breite fest und den Text für das Alt-Attribut in HTML tragen Sie in das Phrase-Element ein.

Details anpassen

Die von den Standard-Stylesheets erzeugten HTML-Dokumente beschränken sich auf eine minimales Layout, das nicht besonders ansprechend ist (Abbildung 1). Änderungen des Layouts können Sie über CSS-Stylesheets, eigene beziehungsweise geänderte XSL-Templates sowie durch Setzen von Stylesheet-Parametern erreichen. Ein CSS-Stylesheet lässt sich am leichtesten über die Zeile »<style src="sms.css" type="text/css"/>« in »layout.xml« für persönliche Anpassungen nutzen.

Mehr Möglichkeiten eröffnet ein eigener Treiber für die XSL-Stylesheets, über den Sie die Parameter und die Templates der Website-Stylesheets verändern. Listing 5 zeigt einen solchen XSL-Treiber, der XSL-Parameter wie beispielsweise »navtocwith« für die Breite des Menüs und »navbgcolor« für dessen Hintergrundfarbe neu setzt. Eine komplette Referenz aller Parameter bietet die Datei »param.xsl« im Verzeichnis »xsl« der Website-Distribution.

Notwendig ist die Anpassung für die Vorlagen in »home.navhead.cell« und »home.navhead.upperright.cel«, die auf der Startseite die Texte Navhead und Upper-right erzeugen. Möchten Sie hingegen ein bestehendes Template verändern, suchen Sie zuerst das Original-Template. (Tipp: Nutzen Sie dazu das Programm »fgrep«) Kopieren Sie es in den Treiber und passen es dort an. Obwohl das Template zweimal existiert, nutzt der Prozessor die angepasste Variante. Die Priorisierung von XSL räumt den hinzugekommenen Templates einen höheren Vorrang ein.

Zum Schluss weisen Sie nur noch im Makefile der Variablen »XSL_BASIS = ...« den Pfad zum eigenen Treiber zu – schon erscheint die Seite im neuen Design (Abbildung 2). Sinnvoll ist es, bei der Nutzung von Olinks das generierte Teil-Makefile »depends.tabular« so zu erzeugen, dass es die eigene Olink-Datenbank bei einem »make distclean« auch gleich wegräumt (Listing 6).

Verwenden Sie dieses Stylesheet in Zeile 27 des Makefile für die Erzeugung von »depends.tabular« und übergeben »xsltproc« den Parameter »website.database.document«, dann enthält die erzeugte Datei die entsprechende Zeile zum Löschen der Datenbank.

Fazit

Website-DTD spielt seine Stärken besonders dann aus, wenn es um die korrekte Inhaltsauszeichnung und um konsistentes Layout geht. Die fast vollständige Auslagerung des Layouts in Stylesheets befreit den Programmierer außerdem vom übermäßigen HTML-Kodieren und die automatische Menügenerierung hilft beim Aufbau einer übersichtlichen Navigation. *(agr)* ■

Listing 5: XSL-Treiber für Anpassungen

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
03                 version="1.0">
04
05   <xsl:import href="http://docbook.sourceforge.net/release/
   website/2.5.0/xsl/tabular.xsl" />
06
07   <xsl:param name="navtocwidth">165</xsl:param>
08   <xsl:param name="navbgcolor">#ffffff</xsl:param>
09   <xsl:param name="header.hr">0</xsl:param>
10   <xsl:param name="footer.hr">0</xsl:param>
11   ...
```

Listing 6: Angepasstes Makefile

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
03                 version="1.0">
04
05   <xsl:import href="http://docbook.sourceforge.net/release/website/
   2.5.0/xsl/makefile-dep.xsl" />
06   <xsl:param name="website.database.document" select="''"/>
07
08   <xsl:template match="autolayout">
09     <xsl:text>website: </xsl:text>
10     <xsl:apply-templates select="toc" mode="all"/>
11     <xsl:apply-templates select="notoc" mode="all"/>
12     <xsl:text>&#10;&#10;</xsl:text>
13     <xsl:apply-templates select="toc"/>
14     <xsl:apply-templates select="notoc"/>
15     <xsl:text>&#10;</xsl:text>
16     <xsl:text>distclean: clean
17 &#9;rm -f </xsl:text>
18     <xsl:text>autolayout.xml depends.tabular</xsl:text>
19     <xsl:if test="$website.database.document != ''">
20       <xsl:text>&#10;&#9;rm -f </xsl:text>
21       <xsl:value-of select="$website.database.document"/>
22     </xsl:if>
23     <xsl:text>&#10;&#10;</xsl:text>
24     <xsl:text>clean:
25 &#9;rm -f </xsl:text>
26     <xsl:apply-templates select="toc" mode="all"/>
27     <xsl:apply-templates select="notoc" mode="all"/>
28     <xsl:text>&#10;&#10;</xsl:text>
29   </xsl:template>
30 </xsl:stylesheet>
```

Infos

[1] Libxslt- und Libxml-Quellen: [http://xmlsoft.org/downloads.html]
[2] Website-/Docbook-DTD sowie Stylesheets: [http://sourceforge.net/projects/docbook/]
[3] Docbook-XML-DTD: [http://www.docbook.org/xml/4.2/]
[4] Informationen zu XML-Katalogen: [http://www.oasis-open.org/committees/entity/spec.html]
[5] „Docbook: The definitive Guide" (Online-Version): [http://docbook.org/tdg/en/html/]

Der Autor

Oliver Fischer lebt, studiert und arbeitet in Berlin. In seiner Freizeit beschäftigt er sich mit allen Fragen rund um freie Betriebssysteme.