



Universität Frankfurt am Main
Fachbereich Biologie und Informatik
Institut für Informatik
Lehrstuhl für Datenbanken und Informationssysteme

Seminar-Ausarbeitung

XSL -Dokumente mit Stil

Fabian Wleklinski

Mai 2001

Betreuer: Karsten Tolle



1 Inhaltsverzeichnis

1	INHALTSVERZEICHNIS.....	2
2	ABBILDUNGSVERZEICHNIS	5
3	VORWORT	6
4	EINLEITUNG	7
5	DOKUMENTEIMWANDEL DERZEIT	8
5.1	HTML	8
5.2	60erJahre:GML	8
5.3	80erJahre:SGML	8
5.4	90erJahre:DasWorldWideWebentsteht	8
5.5	1996:BedarffürXMLentsteht	9
5.6	1996:DSSSL.....	9
5.7	1996/97:XMLwirdveröffentlicht	9
5.8	1997:XMLüberall	10
5.9	1997:DieGeburtvonXSL	11
6	EXTENSIBLESTYLE SHEET LANGUAGE(XSL)	12
6.1	Einleitung	12
6.2	XSLProzessoren	14
6.3	XSLvs.DSSSL	15
6.4	StatusderSpezifikationen	16
7	XSLTRANSFORMATIONS (XSLT).....	17
7.1	Einleitung	17
7.2	DasXSLT -Wurzelement	17
7.3	XSLTTemplateRules	18
7.4	ElementeundAnweisungen	20



xsl:template.....	20
xsl:apply-templates.....	20
xsl:value-of.....	20
xsl:for-each.....	21
xsl:if.....	21
xsl:choose.....	22
xsl:element.....	22
xsl:attribute.....	22
xsl:copy.....	23
8 XMLPATHLANGUAGE(X PATH).....	24
8.1 Einleitung.....	24
8.2 Bedarf für XPath.....	24
Referenzierung in der HTML -Welt.....	24
Referenzierung in der Welt der Datenbanken.....	24
Referenzierung mittels XPath.....	25
8.3 Die XPath Syntax.....	25
Locationpaths.....	25
Locationsteps.....	27
XML Knotentypen.....	28
Blickrichtungen und der Kontextknoten.....	29
Prädikate.....	30
Beispiele für Locationpaths.....	31
abgekürzte Ausdrücke für Prädikate.....	32
9 EINPRAXISBEISPIEL FÜR XSL.....	33
9.1 Einleitung.....	33
9.2 telemallCOMMUNITY.....	33
9.3 Warum XML und XSL/XSLT?.....	34
9.4 Technischer XML -Background der telemallCOMMUNITY.....	35
9.5 XML/HTML Konvertierung mittels Sablotron.....	37
10 ANHANG.....	41
10.1 Häufige Fragen.....	41
Vorteile gegenüber CSS.....	41
Können XSL und CSS kombiniert werden?.....	41
Welche Bedeutung hat XML/XSL heute?.....	41
Welche Bedeutung wird XSL in Zukunft haben?.....	41
10.2 Ausblick.....	41
11 LITERATURVERZEICHNIS.....	43
11.1 Eine Anmerkung vorneweg.....	43



11.2	Literatur	43
12	GLOSSAR.....	51
13	INDEX	52



2 Abbildungsverzeichnis

ABBILDUNG 1 –XSL BEINHÄLTET MUSTER, ANWEISUNGEN UND STIL.....	13
ABBILDUNG 2 –XSL KOOPERIERT MIT XSLT, XPath UND XSLFO	13
ABBILDUNG 3 –XML, SGML, XSL UND DSSSL	14
ABBILDUNG 4 –XSL IST MEHR ALS NUR DSSSL.....	15
ABBILDUNG 5 –UNTERSCHIED ZWISCHEN <i>TEMPLATE</i> UND <i>TEMPLATE RULE</i>	19
ABBILDUNG 6 –VERZEICHNISSTRUKTUR FÜR DAS BEISPIEL.....	26
ABBILDUNG 7 –EINERLEGTER <i>LOCATIONPATH</i>	27
ABBILDUNG 8 –EINERLEGTER <i>LOCATIONPATH</i> –TEIL 2	28
ABBILDUNG 9 –TRANSFORMATION 1 DER COMMUNITY XML-DATEN	36
ABBILDUNG 10 –TRANSFORMATION 2 DER COMMUNITY XML-DATEN	36
ABBILDUNG 11 –TRANSFORMATION 3 DER COMMUNITY XML-DATEN	36
ABBILDUNG 12 –DER XML-STROM DER TELEMAIL COMMUNITY	37
ABBILDUNG 13 –TRANSFORMATION 1 EINER TELEMAIL COMMUNITYS STARTSEITE	38
ABBILDUNG 14 –TRANSFORMATION 2 EINER TELEMAIL COMMUNITYS STARTSEITE	39
ABBILDUNG 15 –TRANSFORMATION 3 EINER TELEMAIL COMMUNITYS STARTSEITE	40



3 Vorwort

Diese Ausarbeitung handelt über die XML Stilsprache „XSL“ und ist im Mai 2001 im Rahmen des Seminars „WWW und Datenbanken“ des Lehrstuhls für Datenbanken und Informationssysteme am Fachbereich Biologie und Informatik der Universität Frankfurt am Main entstanden.

Der aktuelle Stand dieser Ausarbeitung, sowie die parallel zu dieser Ausarbeitung entstandene Präsentation, ist unter der folgenden Internetadresse erhältlich:

<http://www.stormzone.de/uni/Hauptstudium/seminare/wwwdb/list.php3>

Aus Platzgründen habe ich in dieser Ausarbeitung auf einen Teilbereich verzichten müssen, der in der Präsentation seinen Platz gefunden hat: die XSL Formatting Objects. Interessierte Leser seien an dieser Stelle auf die zu dieser Ausarbeitung gehörige Präsentation verwiesen, oder auf eine der zahlreichen anderen Quellen zu diesem Thema, siehe 11.2.

Im Text vorkommende Abkürzungen und Akronyme wie *XHTML*, *XSL* etc. habe ich mich bemüht kursiv auszuzeichnen, und in einem Glossar zusammenzufassen, siehe 12.

Diese Ausarbeitung wurde mit Hilfe von MS Word angefertigt, die dazugehörige Präsentation mit Hilfe von MS PowerPoint.

Ich wünsche ein angenehmes Lesen,

Fabian Wleklinski

(Fabian@Wleklinski.de)



4 Einleitung

XML, XSL, XSLT, XLink, XPath, XPointer, XHTML...?

XML, XSL, XSLT, XLink, XPath, XPointer, XHTML –soodersoähnlichlesensichimJahre2001die TitelblättereinesGroßergängigenFachliteratur.Wasaberhatesdamitaufsich?Fürwensinddiese Technikennützlich?Werkannsieeinsetzen?Waserfordernsie?Undwokommensieeigentlichher?

*XHTML, XML, XSL, XSLT*sindnurwenige vonvielen Spezifikationen des W3C-Konsortiums (www.w3c.org).Das W3CisteineOrganisation,diesichderStandardisierungvonDokumenten im Zusammenhangmitdem Internetverschriebenhat.Dadurch,dasssiemehrsals400 Mitgliedsorganisationenzählt,kannsiemitstarkerUnterstützung undrelativschnellerUmsetzungder Standardsrechnen.

DieseAusarbeitungkonzentriertsichauf *XSL*,unddiedamitengverbundenenTechnologien *XSLT*, *XSLFormattingObjects (XSLFO)*und *XMLPathLanguage (XPath)*.Kenntnissein *HTML*sowiein *XML*werdenvorausgesetzt.BeieinzelnenFragenzu *HTML*kannmanz.B.inderReferenz „SelfHTML“(<http://www.netzwelt.com/selfhtml/>)nachsichsehen.

DieMedienlandschaftsichverändert.GabesfrüherDokumente wieBriefe,Bücherundtechnische DokumentationenausschließlichinPapierform,sindheuteDokumenteinverschiedenstenMedien gebräuchlich –unteranderemimelektronischenMediumInternet .UmDokumenteimInternetzu veröffentlichen,wardieEntwicklungvonSprachennötig,mitdenendieDokumenteabgebildet werdenkonnten.

FürdieKodierungvonInformationenexistierenunzähligeFormate.FormatefürDateienund Protokolle,teilweisestandardisiert,teilweiseproprietär.NureinekleineAuswahlgängigerFormate sindz.B. *.html, .ps, .pdf, .doc, .rm, .mp3, .wma, .jpg, .png, .ppt, .lyx, .bmp, .lwp, .vcf*,

ImAlltagistderInformationensaustauschdurchdieseVielzahlanverschiedenenundvor allem inkompatiblenFormatensehrschwer.DienötigeDatenkonvertierungistnurinwenigenFällen möglich,underfordertdannjenachAusgangs- undZielformateinindividuelles,teilweise mehrstufigesVorgehen.

NurinwenigenBereichenexistierenQuasi-Standardswiez.B.*HTML, MS Word*oder*PDF*für Textdokumente,*MS PowerPoint*fürPräsentationenoder*MS Excel*fürTabellenkalkulationsdaten .Die meistendieserStandardssindaberdurchdiemarktbeherrschendeStellung einzelnerFirmen entstanden,undsinddaherproprietärundnichtoffenfürMitbewerber.

*XML*istangetreten,vielendieserFormateeinstandardisierter,universeller,erweiterbar, dokumentierterundintuitiverNachfolgerzusein.



5 Dokumente im Wandel der Zeit

5.1 HTML

Die wohl bekannteste Sprache überhaupt ist derzeit, *HTML*, ein Akronym für, *HyperText Markup Language*. Im Wesentlichen bezeichnet alles das, was man gemeinhin als „Website“ bezeichnet, auf *HTML*. Diese Sprache weist aber verschiedene Nachteile auf; so ist sie zum Beispiel layoutabhängig, das heißt sie vereint Strukturierungs- und Formatierungselemente in ein und demselben Dokument.

Eine abstrakte Sprache, die Dokumentenstrukturen ohne jede Layoutinformation beschreiben kann, scheint die Lösung für dieses Problem. Ansätze für derartige generische Sprachen existieren bereits seit Jahrzehnten – woran hängt es also?

Um das zu erklären, muss etwas ausgeholt werden...

5.2 60er Jahre: GML

Bereits in den frühen 60er Jahren entwickelten Charles **Goldfarb**, Edward **Mosher** und Raymond **Lorie** bei der IBM die, *Generalized Markup Language* (*GML*). Sie schufen damit eine Sprache, die sowohl mächtig und generisch gewesen ist, dass alle Dokumentbeschreibungssprachen als Teilmenge dieser Sprache formuliert werden konnten. In dem Akronym der neu geschaffenen Sprache verewigten sie zugleich ihre Namen.

5.3 80er Jahre: SGML

In den 80er Jahren trat die, *Standard Generalized Markup Language* („bekannt als, *SGML*“, die Nachfolgerin dieser Sprache an. 1986 wurde *SGML* zu einem ISO-Standard (ISO 8879:1986). *SGML* hat sich nur in wenigen Bereichen durchsetzen können, da es als „unüberschaubar“ gilt, und die Entwicklung entsprechender Software mit hohem Aufwand verbunden ist. Aus heutiger Sicht hat sich nach der Schöpfung von *SGML* in einigen Jahren nicht viel verändert. *SGML* war zwar die am weitesten verbreitete Sprache dieser Art, aber dennoch nicht sonderlich populär.

Die bekannteste Anwendung von *SGML* ist *HTML*. *SGML* bildet also gewissermaßen einen Grundpfeiler des World Wide Web. Insofern, „lebt“ *SGML* weiter.

5.4 90er Jahre: Das World Wide Web entsteht

In den frühen 90er Jahren entstand unbemerkt das World Wide Web. Um nicht mit „Kanonen auf Spatzen zu schießen“, wurde mit *HTML* eine sehr unkomplizierte *SGML*-Sprache als Seitenbeschreibungssprache gewählt.

Im Juni 1997 lag *HTML* mittlerweile in Version 3.2 vor, und das World Wide Web war immer noch in seiner Entstehungsphase. Der marktbeherrschende Netscape Navigator lag in Version 3.x vor, nur vom herausfordernden MS Internet Explorer existierte schon eine 4.0 Beta-Version.

Für Firmen waren damals noch lang nicht selbstverständlich, eine Website einzurichten. Private Internet-Präsenzen waren allenfalls etwas für Computerfreaks. Websites bestanden zum größten Teil aus statischen Inhalten, serverseitige Skriptsprachen wie *PHP*, *ASP*, *JSP*, ... waren entweder noch nicht geboren, oder zumindest noch nicht populär. Die Informationen im World Wide Web, die man mit dem heutigen Informations-Overkill verglichen nur als „spärlich“ bezeichnen kann, wurden mit einfachen Texteditoren oder rudimentären *HTML*-Werkzeugen handschriftlich eingegeben.



5.5 1996:BedarffürXML entsteht

Mit der explosionsartigen Verbreitung des World Wide Web, die nach 1996 einsetzte, und die bis heute nicht abgeklungen ist, stieg auch die Komplexität der darin enthaltenen Seiten. Die Anzahl der Seiten, der Server, der Internetbenutzer, der weltweite Internetverkehr – all diese Kenngrößen stiegen in den folgenden Monaten und Jahren exponentiell an. Es wurden mehr und mehr Informationen auf Webseiten angeboten. Und vor allem wurden die Informationen eines: nämlich bunter. *HTML* begann, sich von einer Strukturauszeichnungssprache zu einer Seitenbeschreibungssprache zu verändern.

Zu allem Überflus setzte 1997 der bittere Konkurrenzkampf der Browser-Hersteller „Netscape“ (<http://www.netscape.com>) und „Microsoft“ (<http://www.microsoft.com>) ein, der als „Browser War“ in die Geschichtsbücher eingegangen ist. Im Kampf um jede zusätzliche Funktion und jedes Prozent Marktanteil begannen beide großen Browser-Hersteller, jeweilige eigene Interpretationen und Erweiterungen des *HTML*-Standards in ihren Produkte zu implementieren.

Der steigende Umfang und die steigende Komplexität der Websites, die zunehmende Überfrachtung der Websites mit visuellen Elementen, und nicht zuletzt die geschwundene Browserunabhängigkeit der *HTML*-Dokumente gestaltete die Entwicklung und die Pflege von Webseiten mehr und mehr zu einer Sisyphus-Arbeit.

Bereits gegen Ende 1996 ist *CSS* veröffentlicht worden, im Zusammenhang mit *HTML* einer der ersten Ansätze zur Trennung von Strukturinformationen und Daten einerseits, und Layoutinformationen andererseits. Obwohl *CSS* als kompakte und mächtige Layoutdefinitionssprache noch heute verwendet wird, erlaubt es weiterhin keine Trennung von Strukturinformationen und Daten.

CSS erlaubt es, logischen Textauszeichnungen wie Überschrift, Absatz, Fußzeile, usw. in einer separaten Stylesheet-Datei physische Textauszeichnungen wie z.B. bestimmte Schriftfarben, -arten und -größen, Absatzabstände, kursive und fette Schrift, etc. zuzuweisen. Dadurch mussten in den *HTML*-Dokumenten lediglich noch die logischen Textauszeichnungen vorhanden sein, was man gemeinhin als „Textstruktur“ bezeichnet.

5.6 1996:DSSSL

Im gleichen Jahr wird *DSSSL* (gesprochen: „Dissel“) (*Document Style Semantics and Specification Language*) zu einem ISO-Standard. *DSSSL* ist die Stilsprache für *SGML*, und ermöglicht unter anderem die Umwandlung von *SGML*-Dokumenten in *HTML*-Dokumente.

Die Grundidee von *DSSSL* war die Schaffung einer standardisierten Layout-Semantik für *SGML*-Dokumente, jedoch sollten die *SGML*-Dokumente ansich weiterhin nur logische Auszeichnungen enthalten. Im Laufe der Zeit wurde der *DSSSL*-Standard immer stärker erweitert, um allen Anforderungen gerecht werden zu können. Dies hatte zur Folge, dass die Formatierungsmöglichkeiten zwar sehr mächtig wurden, aber auch keine existierende Anwendung bis heute den kompletten *DSSSL*-Standard umsetzen konnte.

DSSSL ist von ähnlicher Kompliziertheit wie *SGML* selber, und hat genau wie *SGML* nie besondere Popularität erlangt.

5.7 1996/97:XML wird veröffentlicht

1996 entwickelt das W3C insgesamt 10 Design-Richtlinien für eine neue Sprache, die die Lücke zwischen dem „überforderten“ *HTML* und dem zu komplexen *SGML* füllen soll. Im Dezember 1997 veröffentlicht das W3C die „Recommendation“ für *XML 1.0* (*Extended Markup Language*). *XML* vereint Aspekte und Erfahrungen aus *SGML* und *HTML* und wurde konzipiert, um ausgezeichnete Daten zu veröffentlichen, zu transportieren und zu speichern.

Die Einsatzmöglichkeiten von *XML* sind unüberschaubar groß. Beinahe jede Applikation muss Daten speichern und/oder austauschen. Wo bisher Funktionalität zum Lesen, Schreiben, Bearbeiten, Komprimieren, Verschlüsseln, Übertragen, Validieren, Konvertieren, ... von Daten teurer entwickelt



werden musste, ist dank *XML* nun noch die Erstellung einer *DTD* nötig – für die Bereitstellung der eigentlichen Funktionalität kann hingegen preiswert eine Standardsoftware verwendet werden.

SGML wird jedoch in einigen Bereichen, vor allem an Hochschulen, nach wie vor eingesetzt. Ein *XML*-Dokument ist auch absolut *SGML*-konform, bereits bestehende *SGML*-Anwendungen können also auch mit *XML*-Daten umgehen.

5.8 1997: XML überall

Die Einfachheit von *XML*, man könnte beinahe sagen: die Intuitivität, hat für einen rasanten und weiten Verbreitung dieses Standards gesorgt. Zahlreiche Firmen haben spezielle Software für *XML* entwickelt, oder *XML*-Aspekte in ihre bestehende Software integriert.

Es ist kaum noch möglich, die Menge der Applikationen, Sprachen und Protokolle aufzuzählen, die *XML* verwenden oder darauf aufbauen, aber um einen kleinen Überblick zu bekommen, seien hier einige Beispiele genannt:

- Office-Pakete
 - MS Office
 - Star Office
 - ...
- Internet-Formate
 - Channel Description Format (CDF)
 - Customer Profile Exchange (CPEX)
 - Querying XML
 - Resource Description Framework (RDF)
 - Simple Markup Language (SML)
 - Voice ML
 - Wireless Markup Language (WML)
 - XML Hypertext Markup Language (*XHTML*)
 - XML Linking Language (*XLink*)
 - XML Path Language (*XPath*)
 - XML Pointer Language (*XPointer*)
 - XML Protocols
 - XML-Schemata
 - XML Stylesheet Language (*XSL*)
 - XSL Transformations (*XSLT*)
 - ...
- Naturwissenschaftliche oder technische Fachsprachen
 - Chemical Markup Language (CML)
 - Mathematical Markup Language (MathML)
 - ...



- Grafikformate
 - VectorMarkupLanguage(VML)
 - ScalableVectorGraphics(SVG)
 - ...

5.9 1997:DieGeburtvonXSL

Sopraktisch *XML* aus der Sicht eines Techniklers ist, sounschönistesausSichteinesTechniklaren: hierarchischverschachtelteDatenstrukturen ,Datentyp -Deklarationenund Verarbeitungsdirektiven statt schmuckenSchriftauszeichnungen, Tabellen undÜberschriften. EinUmwandlungswerkzeug mussteher.

In derAnfangszeitdientedasursprünglichfür *HTML* entwickelte *CSS* alsAuszeichnungssprache für *XML*. *CSS* warzudiesemZeitpunkt bereitsinvielenProduktenimplementiert, verbreitet, akzeptiert underprobt. AllerdingsreichtendieMöglichkeiten, die *CSS* bot, bei weitemnichtaus, jedem AnspruchaneineoptischeAufbereitungder *XML*-Datengerechtzuwerden.

UmdenGradzwischen derUnkompliziertheit von *XML/HTML/CSS* und derMächtigkeit von *SGML/DSSSL* zubewandern, reichtenimAugust1997 dieVertreter vonMicrosoft ,ArborText, Inso, sowieHenryThompsonundJamesClark(entwickeltedie *DSSSL*-Engine, „*Jade*“) einenVorschlagfür *XSL* beim *W3C* ein.

Aberwasist *XSL* dennuneigentlich...?



6 eXtensibleStylesheet Language(XSL)

6.1 Einleitung

Was ist eigentlich... XSL?

Die Abkürzung *XSL* steht für „*eXtensibleStylesheet Language*“. Dahinter verbirgt sich eine Sprache, die 1999 vom W3C entworfen worden ist, um *XML*-Dokumente in andere *XML*-Dokumente zu transformieren. Ein *XSL*-Dokument beschreibt die Präsentation einer Klasse von *XML*-Dokumenten, oder, anders ausgedrückt: *XSL* ist die Stilsprache von *XML*.

Ein *XSL*-Dokument besitzt im Allgemeinen die Dateiendung „.XSL“, man bezeichnet *XSL*-Dokumente oft auch als „*Stylesheets*“. Aber Achtung: *XSL*-Stylesheets sind von *CSS*-Stylesheets zu unterscheiden!

Die Anwendung von Stylesheets auf bestimmte *XML*-Dateien (oder –Ströme) übernimmt spezielle Anwendungen, die man „*XSLT*-Prozessoren“ nennt. Dazu gehört z. B. „Sablotron“. Für eine Auswahl siehe 11.2.

Eine *XSL*-Datei enthält „Baumkonstruktionsregeln“. Die Verarbeitung einer *XSL*-Datei durch einen *XSL*-Prozessor läuft so ab, dass ein *XSL*-Prozessor ein gegebenes *XML*-Dokument einliest, es in eine interne Baumstruktur umwandelt (den *XML*-Quellbaum), Element für Element die Vorgaben aus dem *XSL*-Stylesheet bearbeitet, und währenddessen einen *XML*-Zielbaum erzeugt.

Beider Entwicklung von *XSL* lag der Hauptaugenmerk auf den folgenden Zielen:

- einfache Nutzung über das Internet
- *XML*-Syntax
- deklarative Sprache für Textformatierungen wie bei *CSS* oder *DSSSL* vorhanden
- „als letzte Möglichkeit“ eine Skriptsprache, um komplexe Formatierungen und Erweiterbarkeit zu garantieren
- schnelle Entwicklung von *XSL*-Anwendungen
- intuitive Lesbarkeit von *XSL*-Dokumente
- der *XML*-Zielbaum soll eine völlig andere Struktur als der *XML*-Quellbaum besitzen können
- es sollen „neue“ *XML*-Strukturen hinzugefügt werden können, z. B. ein Inhaltsverzeichnis, „on the fly“ erstellt werden

Hier ein Beispiel für ein *XSL*-Dokument:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:template xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <html><body>
    <xsl:value-of select="mail/autor/name"/>
  </body></html>
</xsl:template>
```

XSL bezeichnet eine Sprache, die ihrerseits aus drei Teilsprachen besteht:

1. *XSL Transformations* (*XSLT*)
2. *XML Path Language* (*XPath*)

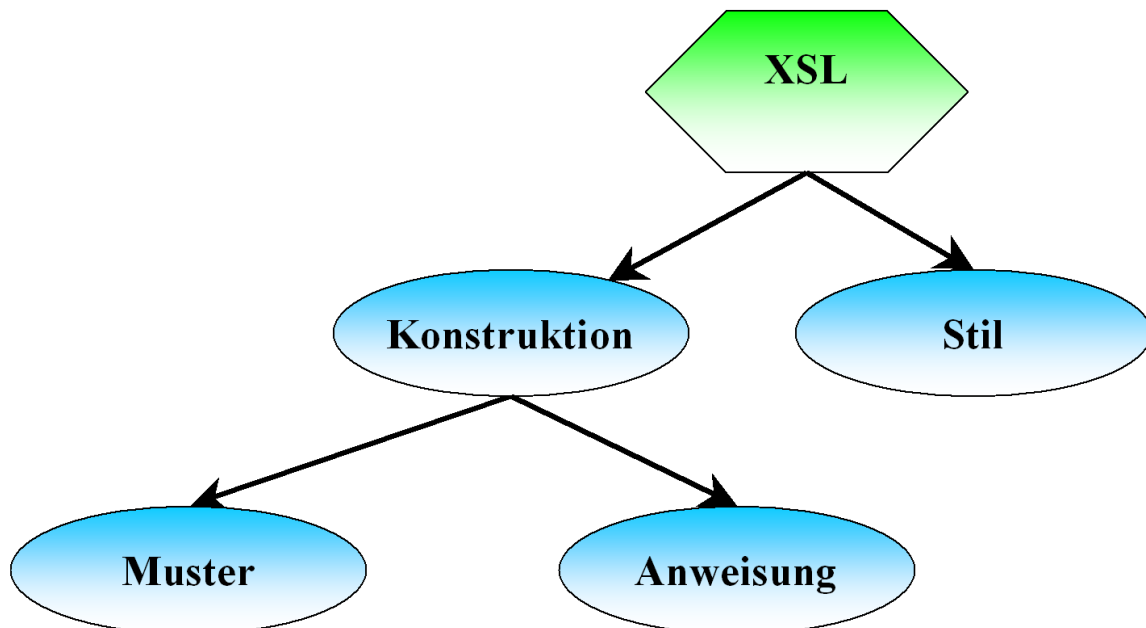
3. *XSLFormattingObjects* (XSLFO)

Abbildung 1 –XSLbeinhaltetMuster,AnweisungenundStil

Der grundlegend neue Teil der XSL-Sprache ist XSLT. Die *XSLFormattingObjects* entsprechen von ihrem Prinzip her (aber nicht formal!) den altbekannten CSS-Stylesheets, und die *XMLPathLanguage* tut auf den ersten Blick wie ein in der XSLT untergeordnete Technologie an.

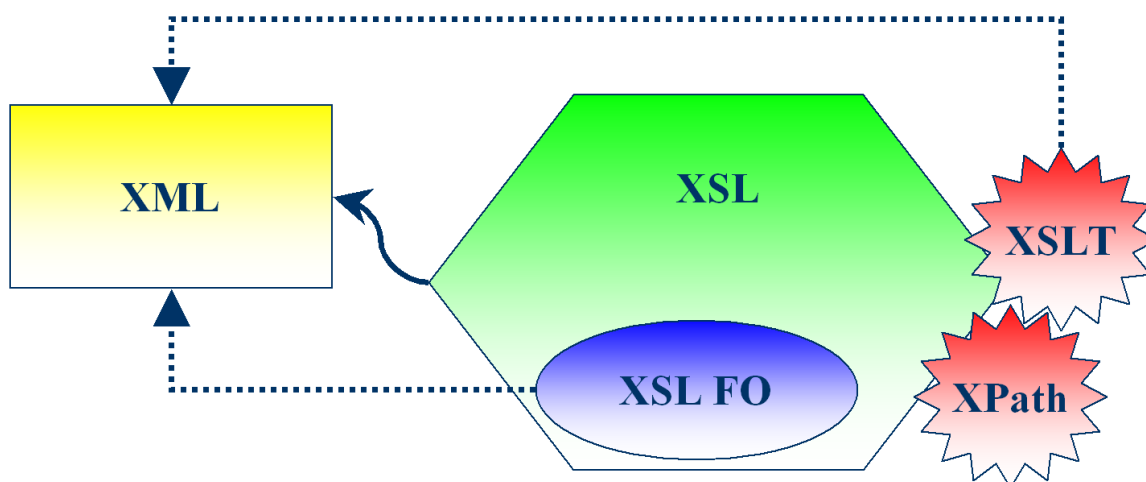


Abbildung 2 –XSLkooperiertmitXSLT, XPathundXSLFO

Die Begriffe XSL und XSLT werden oft synonym verwendet, aber nicht trotzdem ist XSLT nur eine von drei Komponenten von XSL. Die *XSLFormattingObjects* gehören zu XSL, nicht zu XSLT. Die *XMLPathLanguage* gehört weder zu XSL noch zu XSLT, sondern ist ein eigenständiger Standard. (In Zukunft wird XPath wohl auch für den neuen XLink-Standard verwendet werden.)



Indie Entwicklung von XSL sind zahlreiche Aspekte und Erfahrungen von DSSSL eingeflossen. Ähnlich wie XML auf SGML zurückgeht, fußt XSL somit auf DSSSL. Somit haben SGML und DSSSL ihre Daseinsberechtigung als Beispiele, was alles möglich ist, und als Wegbereiter für XML und XSL:

„XSL is based on DSSSL, which is based on Lisp. It is amazing what people will do to avoid Lisp.“

Abbildung 3 verdeutlicht den Zusammenhang zwischen den verschiedenen Sprachen:

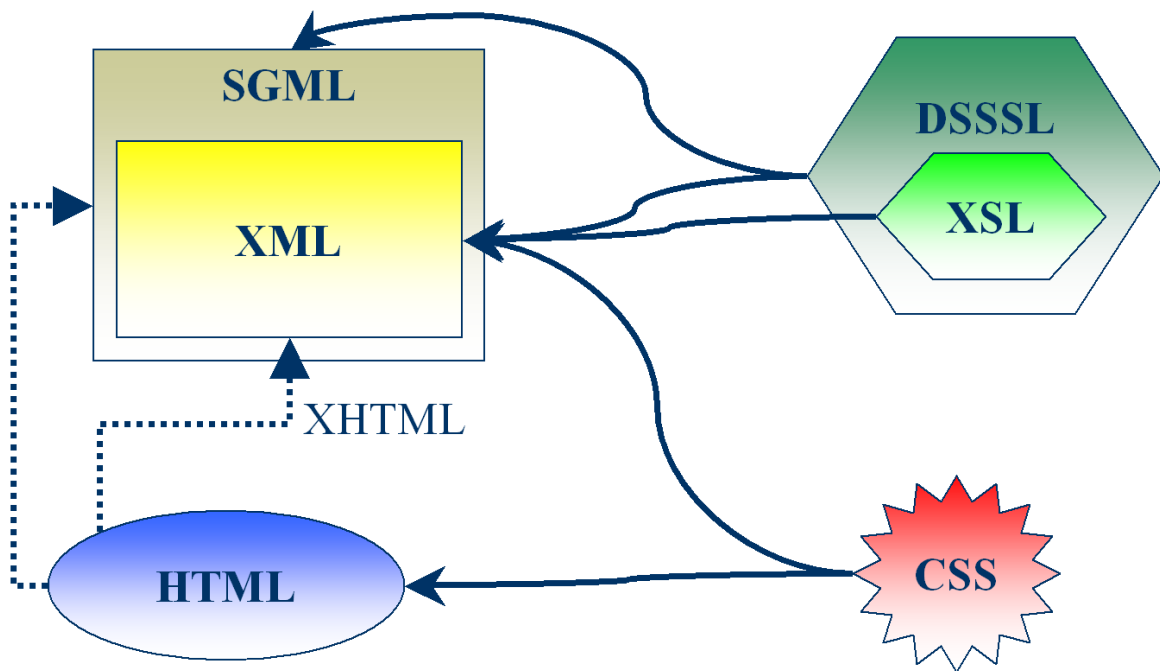


Abbildung 3 –XML, SGML, XSL und DSSSL

XSLFO wird jedoch von den meisten, derzeit erhältlichen XSL-Processoren nicht unterstützt, und ist auch noch nicht endgültig spezifiziert. Aus diesem Grunde, und auch, weil die Funktionalität von XSLFO sehr gut von der restlichen XSL-Funktionalität abgespalten werden kann, bleibt XSLFO in dieser Ausarbeitung außen vor.

Interessierte Leser seien an dieser Stelle auf die zu dieser Ausarbeitung zugehörige Präsentation verwiesen, oder auf eine der zahlreichen anderen Quellen zu diesem Thema, siehe 11.2.

6.2 XSL Prozessoren

Was ist ein XSL -Prozessor? Wasmacher? Wiemacher das?

Es sind inzwischen ein Handvoll XSLT-Processoren erhältlich, einige von ihnen sind kommerziell, andere sind freierhältlich. Den meisten Processoren ist es aber gemein, dass sie **keine XSL-Processoren** sind; denn streng genommen muss zwischen den beiden Begriffen, „XSL-Processor“ und „XSLT-Processor“ unterschieden werden.

Wie bereits in 6.1 erwähnt wurde, setzt sich XSL aus XSLT, XSLFO und XPath zusammen. Die meisten Processoren unterstützen XSLT mehr oder weniger vollständig. Das gleiche gilt für XPath, denn ohne XPath ist XSLT kaum möglich. Ganz anders steht es um XSLFO: beinahe alle gängigen



Prozessoren unterstützen *XSLFO* **nicht** (Stand: 05/2001). Daher spricht man im Allgemeinen von *XSLT-Prozessoren*.

Allen Prozessoren ist aber die Vorgehensweise gemeinsam, mit der sie ein Dokument bearbeiten:

1. Lesendes *XML*-Quelldokumentes, Abbildung der Dokumentstruktur auf einen Baumstruktur: den *XML*-Quellbaum
2. Lesendes *XSL*-Dokumentes und Anwendung der *XSL*-Befehle auf den *XML*-Quellbaum; dabei wird der *XML*-Zielbaum aufgebaut
3. Anwendung von *XSL Formatting Objects* auf dem *XML*-Zielbaum
4. Ausgabe des *XML*-Zielbaumes

Die *XSLT-Prozessoren* verzichten bei der Umwandlung auf den dritten Schritt.

6.3 XSL vs. DSSSL

Der oben erwähnte Vorschlag beinhaltete, dass *XSL* *DSSSL*-Eigenschaften und *CSS*-Möglichkeiten vereinen sollte. Damit wurde praktisch die Vorgabe von *W3C* erfüllt, in der *XML*-Stilsprache *DSSSL*-Funktionen zu implementieren. Aber bei der Arbeit stellte sich heraus, dass gewisse Abweichungen und Ergänzungen in Abgrenzung zu *DSSSL* erforderlich waren.

Zumein sind die *XSL*-Style-Sheets im Gegensatz zu *DSSSL*, das *Scheme*- bzw. *LISP*-Syntax verwendet, in *XML*-Notation gehalten, was bedeutet, dass (beinahe) alles in spitzen Klammern steht. *LISP*-Syntax benutzt für Elementnamen runde Klammern, während *XSL* diese ausschließlich in Skripten verwendet.

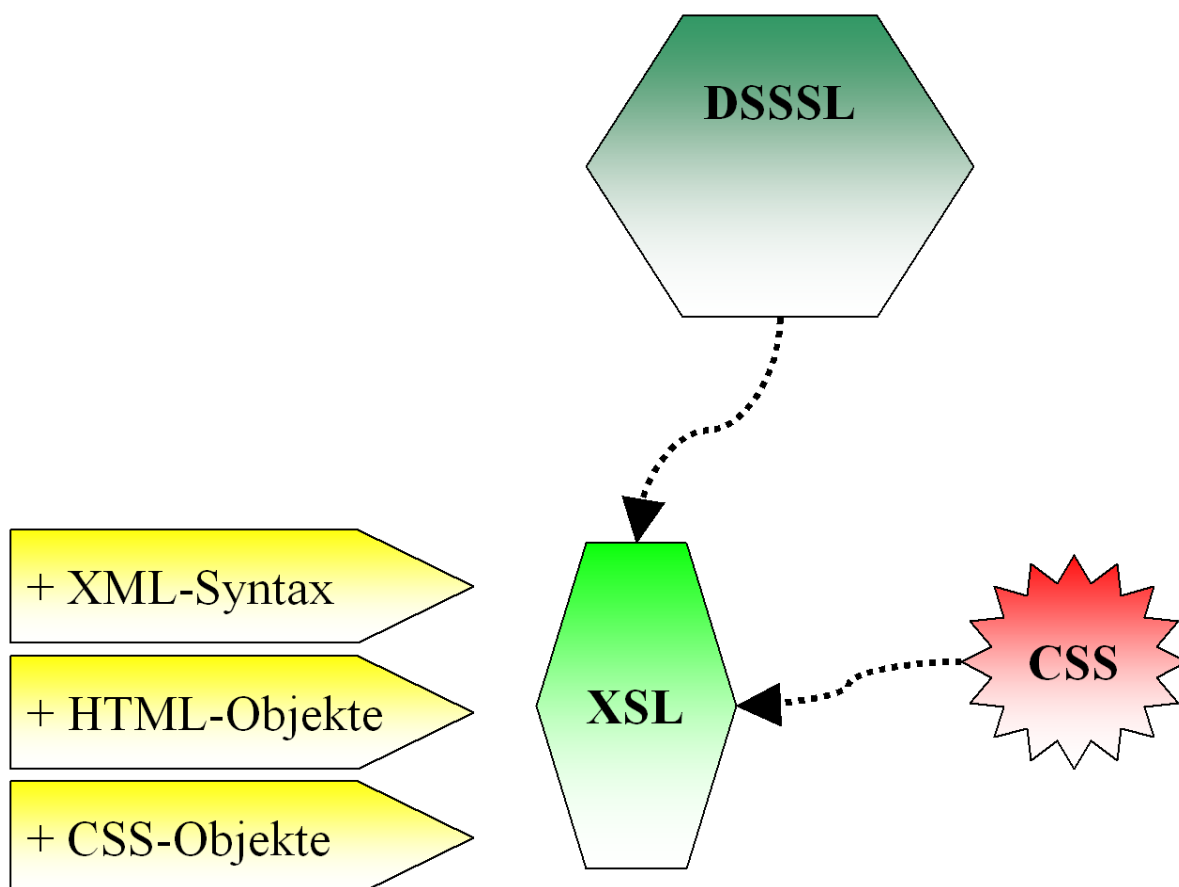


Abbildung 4 – XSL ist mehr als nur DSSSL



Der, *Flow ObjectTree* "istumElementeerweitert worden, welche die Einbeziehung von *HTML*-und *CSS*-Objektenerlauben. Diese Abweichungen sind auch für eine Ergänzung in *DSSSL* vorgeschlagen. Würde diese festgeschrieben, wäre *DSSSL* eine Obermenge von *XSL* (siehe Abbildung 4).

6.4 Status der Spezifikationen

Spezifikation und Versionsnummer	Status
<i>XSL</i> 1.0	Candidate Recommendation
<i>XSLT</i> 1.0	Recommendation
<i>XSLT</i> 1.1	Working Draft
<i>XSLT</i> 2.0 requirements	Working Draft
<i>XPath</i> 1.0	Recommendation
<i>XPath</i> 2.0 requirements	Working Draft



7 XSL Transformations (XSLT)

7.1 Einleitung

Was ist eigentlich...XSLT ?

Der Name sagt es: „Transformations“ bedeutet übersetzt „Transformationen“, und dahinter verbirgt sich das Umwandeln einer Menge von Informationen in einer bestimmten Darstellungsform durch die Anwendung von Regeln in eine andere Darstellungsform.

Will man nicht die gesamte Menge an Informationen transformieren, sondern nur eine Teilmenge der selben, dann benötigt man ein Selektionskriterium. Im Falle von XSLT übernimmt diese Aufgabe XPath, siehe 8.

Bei einer Transformation können Informationen verloren gehen, es können aber auch neue Informationen hinzukommen. Vorallem aber ist eine Transformationalsonichtinjedem Fall umkehrbar.

XSLT ist eine XML -Sprache. Alle Elemente, die zur XSLT-Anwendungsdomäne gehören, werden durch die folgende Namensraum -Deklaration erkannt:

```
http://www.w3.org/1999/XSL/Transform
```

Standardmäßig wird für diesen Namensraum der Präfix „xsl“ verwendet.

7.2 Das XSLT -Wurzelement

Alle XSLT-Anweisungen in einer XSL-Datei müssen unterhalb dieses Elementes stehen:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"/>
```

Dieses Element muss auf oberster Ebene, als das XML-Wurzelement sein. Daher darf es auch nur ein einziges Mal in XSL-Dokument stehen.

Das Element `<xsl:transform/>` gilt als Synonym, einen Unterschied gibt es lediglich in ihrer Bedeutung hinsichtlich Drittanwendungen. Handelt es sich bei der XSL-Datei um eine Transformationsvorschrift, ist `<xsl:transform/>` die richtige Wahl. Handelt es sich aber in erster Linie um Ausgabeformatierungen mittels XSLFO, sollte `<xsl:stylesheet/>` bevorzugt werden.

Attributesindz.B.:

- **version:**

Das Attribut „**version**“ ist obligatorisch. Prozessoren verfügen übereinensogenannten „Forward Compatible Processing“-Modus, den sie immer dann aktivieren, wenn sie eine XSL/XSLT-Version entdecken, die neuer ist, als die in ihnen implementierte Version. Prozessoren benötigen das Attribut „**version**“, um den Kompatibilitätsmodus ausschalten zu können.

Durch dieses Attribut ist sichergestellt, dass XSL-Dateien auch von älteren Prozessoren noch verarbeitet werden können.

- **default-space:**

Dieses optionale Attribut bestimmt, ob Leerzeichen im XML-Quellbaum erhalten bleiben sollen. Der Wert für dieses Attribut ist „**preserve**“.



- **extension-element:**

Mittels dieses Attributes ist es möglich, eine Menge von *XML*-Namensräumen zu definieren, die von *XSL/XSLT*-Prozessoren ignoriert werden sollen. Dazu wird als Wert dieses Attributes eine Liste von Namensräumen angegeben, die durch Leerzeichen getrennt sind.

- **id:**

Mittels des Attributes „**id**“ kann man das *XSL*-Dokument ggf. innerhalb einer anderen Ressource eindeutig identifizieren.

- **indent-result:**

Mittels dieses optionalen Attributes kann bestimmt werden, ob in der Ausgabe eventuelle Leerzeichen erhalten werden sollen. Der Vorgabewert für dieses Attribut ist „**yes**“.

- **language:**

Mittels dieses optionalen Attributes kann die gegebene Sprache definiert werden. Der Vorgabewert für dieses Attribut ist „**JScript**“.

- **result-ns:**

Mittels dieses Attributes wird der Namensraum der Ausgabe des Prozessors definiert.

7.3 XSLT Template Rules

Was steht unter dem *XSLT*-Wurzelement...?

XSLT-Anweisungen bestehen aus sogenannten „*Template Rules*“. Jede *Template Rule* wird durch ein bestimmtes *XML*-Element definiert. Eine *Template Rule* wird zur Formulierung eines Transformationsprozesses für eine Menge von Knotentypen im *XML*-Quellbaum verwendet.

Eine *Template Rule* besteht aus zwei Komponenten, und zwar

- einem *Muster*, und
- einem *Template*.



Die folgende Abbildung zeigt, wie *Template*, *TemplateRule* und *Muster* Hand-in-Hand greifen:

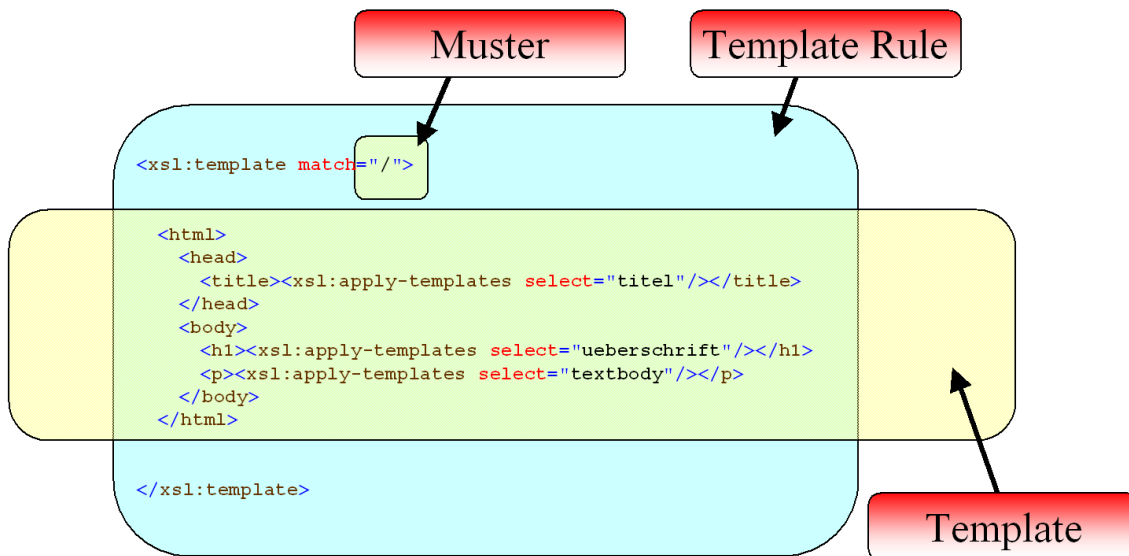


Abbildung 5 – Unterschied zwischen *Template* und *TemplateRule*

Das *Muster* hat die Aufgabe, einen Teil des zu konvertierenden *XML*-Dokumentes zu beschreiben, oder es ist formal ausgedrückt: eine Teilmenge aller *XML*-Elemente des *XML*-Quelldokumentes zu beschreiben.

Das *Muster* ist der Punkt, an dem *XPath* ins Spiel kommt – denn *XPath* ist die Syntax der Ausdrücke, die als *Muster* zulässig sind. Siehe 8.

Man kann sich ein *Muster* ähnlich einem regulären Ausdruck vorstellen, der auf eine bestimmte Menge angewendet wird, und eine Teilmenge derselben herausfiltert. Gäbe es keine *Muster*, so würde jede Transformationsanweisung **alle** Elemente des *XML*-Quelldokumentes betreffen!

Das *Template* ist der eigentliche Transformationsschritt. Es entspricht einer Verarbeitungsanweisung, die als Eingabe eine Menge von *XML*-Elementen benötigt. Ein *Template* verwendet immer genau die *XML*-Elemente als Eingabe, die das zugehörige *Muster* beschreibt.

Der Begriff „*Template*“ ist so zu erklären, dass hier eine Art von „Vorlage“ für die zu erzeugende *XML*-Struktur steht, welche bei dem Transformationsprozess noch mit konkreten Werten gefüllt wird. Bei dem *Template* handelt es sich gewissermaßen um „ein Stück Stylesheet“, das im *XML*-Zielbaum instanziiert wird.

Ein *Template* kann den Transformationsprozess rekursiv fortführen, d.h. es kann seinerseits wieder eine *TemplateRule* beinhalten, deren *Muster* die *XML*-Element-Teilmenge des *Template* als Eingabe gegeben wird.

Das *Muster* ist in diesem Fall der Ausdruck „/“, und das *Template* ist alles, was innerhalb des `<xsl:template/>` Elementes steht.



7.4 Elemente und Anweisungen

Wiesehendennundie XSLT-Anweisungenaus...?

xsl:template

`<xsl:template/>` weist den Prozessor an, eine *TemplateRule* zu definieren. Mittels des Attributes „`match`“ wird die Elementmenge definiert, auf die das *Template* angewendet werden kann soll.

- `<xsl:template match="titel">`
 `<h1><xsl:value-of/></h1>`
 `</xsl:template>`

Weist den Prozessor an, eine *TemplateRule* zu definieren, die auf alle *XML-Elemente* vom Typ „`titel`“ angewendet wird. Falls ein *XML-Element* vom Typ „`titel`“ verarbeitet wird, werden *HTML-Anweisungen* erzeugt, um den Wert des Elementes „`titel`“ als Überschrift erster Ordnung zu formatieren.

xsl:apply-templates

`<xsl:apply-templates/>` weist den Prozessor an, auf eine bestimmte Menge von Elementen *Templates* anzuwenden. Welche *Templates* das im Einzelnen sind, entscheidet der Prozessor. Diese Anweisung wird häufig benutzt, um die Nachfahren des aktuellen Knotens zu verarbeiten.

Mittels des Attributes „`select`“ kann die Menge der Elementespezifiziert werden, welche verarbeitet werden sollen. Falls dieses Attribut nicht verwendet wird, setzt der Prozessor den Vorgabewert „`node()`“, und verarbeitet allen nachfolgenden Elementen des aktuellen Elementes (allerdings keine Attribute).

- `<xsl:apply-templates/>`

Weist den Prozessor an, alle Nachfahren des aktuellen Elementes zu verarbeiten.

- `<xsl:apply-templates select="muster"/>`

Weist den Prozessor an, alle Nachfahren des aktuellen Elementes vom Typ „`muster`“ zu verarbeiten.

- `<xsl:apply-templates order-by="-krit1;+krit2"/>`

Weist den Prozessor an, alle Nachfahren des aktuellen Elementes vom Typ „`krit1`“ aufsteigend, und nach „`krit2`“ absteigend sortiert zu verarbeiten.

xsl:value-of

`<xsl:value-of/>` weist den Prozessor an, eine bestimmte Menge von Elementen *als Text* in den *XML-Zielbaum* einzufügen.

Mittels des Attributes „`select`“ kann die Menge der Elementespezifiziert werden, welche verarbeitet werden sollen. Falls dieses Attribut nicht verwendet wird, setzt der Prozessor den Vorgabewert „`.`“, und kopiert den Inhalt des aktuellen Knotens.

Achtung: Durch Anwendung dieses Befehls lassen sich lediglich *Texte* bzw. *Strings* in den *XML-Zielbaum* einzufügen, es lassen sich aber keine *XML-Elemente* oder *Attribute* einzufügen bzw. kopieren!

- `<xsl:value-of/>`

Weist den Prozessor an, den Wert des aktuellen Elementes als Text in den *XML-Zielbaum* einzufügen.



- `<xsl:value-of select="muster"/>`

Weist den Prozessoran, den Wert aller Nachfahren vom Typ, `muster` des aktuellen Elementes als Text in den XML-Zielbaum einzufügen.

xsl:for-each

`<xsl:for-each/>` weist den Prozessoran, durch eine bestimmte Menge von Elementen zu iterieren, und für jedes dieser Elemente bestimmte Anweisungen auszuführen.

Mittels des Attributes, `select` kann die Menge der Elemente spezifiziert werden, durch welche iteriert werden soll. Falls dieses Attribut nicht verwendet wird, setzt der Prozessor den Vorgabewert `node()`, und iteriert durch alle nachfolgenden Elemente des aktuellen Elementes (allerdings keine Attribute).

- `<xsl:for-each select="muster" order-by="elem"/>`

Weist den Prozessoran, sortiert nach den Wert des Feldes, `elem` durch die Menge aller Nachfahren vom Typ, `muster` des aktuellen Elementes zu iterieren.

- ```
<xsl:for-each select="images/image">
 <tr>
 <td>

 </td>
 </tr>
</xsl:for-each>
```

Weist den Prozessoran, durch die Menge aller Nachfahren vom Typ, `image` aller Nachfahren vom Typ, `images` des aktuellen Elementes zu iterieren, jeweils die dargestellten *HTML*-Anweisungen für eine *HTML*-Tabellenzeile mit einer eingebetteten Grafik auszugeben, und dabei an der Position des Attributes, `src` des Elementes, `img` den Wert des Attributes, `org` des XML-Quellbaumes einzufügen.

### xsl:if

`<xsl:if/>` weist den Prozessoran, eine bestimmte Menge von *XSLT*-Anweisungen nur unter bestimmten Umständen auszuführen, man spricht von einer „bedingten Ausführung“.

Die *XSLT*-Anweisungen, die bedingt ausgeführt werden sollen, sind innerhalb des Elementes. Die auszuwertende Bedingung wird durch das Attribut, `test` spezifiziert.

`<xsl:if/>` entspricht der klassischen „if“-Anweisung vieler Programmiersprachen.

- ```
<xsl:if test="geschlecht[.='female']">
  Frau
  <xsl:apply-templates/>
</xsl:if>
<xsl:if test="geschlecht[.='male']">
  Herr
  <xsl:apply-templates/>
</xsl:if>
```

Weist den Prozessoran, in einem ersten Schritt für jeden Nachfahren vom Typ, `geschlecht` des aktuellen Elementes, den Wert `female` hat, den Text, `Frau` auszugeben, und anschließend mittels `<xsl:apply-templates/>` die Verarbeitung mit den nachfolgenden Elementen fortzuführen, und in einem zweiten Schritt Entsprechendes für die Elemente mit dem Wert, `male` zutun.



xsl:choose

`<xsl:choose/>` weist den Prozessor an, in Abhängigkeit von mehreren Bedingungen jeweils andere Mengen von XSLT-Anweisungen auszuführen, um anspruchsvollere, bedingte Ausführung.

`<xsl:choose/>` entspricht der klassischen „select case“ - bzw. „switch case“ - bzw. „case of“ - Anweisung vieler Programmiersprachen.

- ```
<xsl:choose>
 <xsl:when test="geschlecht[.='female']">
 Frau
 <xsl:apply-templates/>
 </xsl:when>
 <xsl:when test="geschlecht[.='male']">
 Herr
 <xsl:apply-templates/>
 </xsl:when>
 <xsl:otherwise>
 Sehr geehrte Damen und Herren,
 </xsl:otherwise>
</xsl:choose>
```

Weist den Prozessor an, in einem ersten Schritt für jeden Nachfahren vom Typ „geschlecht“ des aktuellen Elementes, deren Wert „female“ hat, den Text „Frau“ auszugeben, und anschließend mittels `<xsl:apply-templates/>` die Verarbeitung mit den nachfolgenden Elementen fortzuführen, und in einem zweiten Schritt Entsprechendes für die Elemente mit dem Wert „male“ zu tun. In einem dritten Schritt schließlich wird für jeden Nachfahren vom Typ „geschlecht“ des aktuellen Elementes, deren Wert als „female“ oder „male“ hat, der Text „Sehr geehrte Damen und Herren,“ auszugeben.

### xsl:element

`<xsl:element/>` weist den Prozessor an, dem XML-Zielbaum ein XML-Element hinzuzufügen. Der Name des hinzuzufügenden Elementes wird in dem Attribut „name“ spezifiziert.

- ```
<xsl:element name="xsl.template"/>
```

Fügt dem aktuellen Element im XML-Zielbaum ein Element mit dem Namen „xsl.template“ hinzu.

xsl:attribute

`<xsl:attribute/>` weist den Prozessor an, dem aktuellen Element des XML-Zielbaums ein XML-Attribut hinzuzufügen. Der Name des hinzuzufügenden Attributes wird in dem Attribut „name“ spezifiziert. Der Wert des hinzuzufügenden Attributes ist innerhalb des Elementes.

`<xsl:attribute/>` wird sehr oft in Zusammenhang mit `<xsl:element/>` benutzt, um Attribute für hinzugefügte Elemente hinzuzufügen.

Achtung: Bevor Nachfolger eines Elementes hinzugefügt werden, müssen alle seine Attribute hinzugefügt worden sein! Denn nach dem Hinzufügen eines Nachfolgers ist dieser „Nachfolger“ das aktuelle Element!

- ```
<xsl:attribute name="autor">Manfred</xsl:attribute>
```

Weist den Prozessor an, dem aktuellen Element im XML-Zielbaum ein Attribut mit dem Namen „autor“ und dem Wert „Manfred“ hinzuzufügen.



- ```
<img>
  <xsl:attribute name="src">
    <xsl:value-of select="img/imgname"/>
  </xsl:attribute>
</img>
```

Weist den Prozessor an, dem aktuellen Element im *XML*-Zielbaum ein Element mit dem Namen „**img**“ hinzuzufügen, und diesem Element ein Attribut mit dem Namen „**src**“ hinzuzufügen, dessen Wert auf den Wert des Nachfolgers, „**imgname**“ des Nachfolgers, „**img**“ des aktuellen Elementes gesetzt wird.

xsl:copy

`<xsl:copy/>` weist den Prozessor an, das aktuelle Element in den *XML*-Zielbaum zu kopieren.

- ```
<xsl:template match="MeinElement">
 <xsl:copy>
 <xsl:attribute name="kopiert">
 true
 </xsl:attribute>
 </xsl:copy>
</xsl:template>
```

Weist den Prozessor an, gefundene Vorkommen von Elementen des Typs „**MeinElement**“ in den *XML*-Zielbaum zu kopieren, und ihnen ein Attribut mit dem Namen „**kopiert**“ und dem Wert „**true**“ hinzuzufügen.

s., „**MeinElement**“ in „**kopiert**“ und dem



## 8 XMLPathLanguage(XPath)

### 8.1 Einleitung

Was ist eigentlich... XPath?

Das W3C hat die Empfehlung für XPath 1999 veröffentlicht. XPath ist kein Bestandteil von XSL, noch viel weniger ein Bestandteil von XSLT, sondern ein eigenständiger Standard. Insbesondere wird XPath wohl auch dem zukünftigen XLink dienen, dem Äquivalent zu den aus der HTML-Welt bekannten Hyperlinks, und damit seine Rolle als eigenständige und unabhängige Spezifikation wahrnehmen.

Mittels XPath ist es möglich, einen beliebigen Teil eines XML-Dokumentes zu beschreiben, oder etwas formaler ausgedrückt: beliebige Teilmengen von Knoten eines XML-Baumes zu beschreiben. Daher kommt XPath im Zusammenhang mit XSLT eine große Bedeutung zu, denn es gibt keine XPath, sondern es gibt keine Muster, und dann würde jede XSLT-Transformationsanweisung alle Elemente des XML-Quelldokumentes betreffen!

### 8.2 Bedarf für XPath

Warum schon wieder eine neue Sprache?

Warum ist es nötig, mit XPath eine weitere Sprache zu definieren? Ist es nicht möglich, auf eine bestehende Sprache aus der HTML-Welt oder der Welt der Datenbanken zurückzugreifen?

#### Referenzierung in der HTML -Welt

Das bekanntere, aber wesentlich röhre Äquivalent zu XPath aus der HTML-Welt ist die für Hyperlinks verwendete „dateiname.html#anker“-Notation. Warum reicht diese einfache und intuitive Notation für XML-Dokumente nicht aus?

In der HTML-Welt ist das HTML-Dokument die zweitkleinste Einheit. Die kleinsten der HTML-Welt sind sogenannte „anchors“ (englisch: „Anker“), darunter versteht man im HTML-Dokument explizit definierte Positionen.

In der HTML-Welt ist es also entweder möglich, auf ein anderes HTML-Dokument zu referenzieren, oder auf eine definierte Position in einem anderen HTML-Dokument zu referenzieren. Diese Möglichkeiten reichen aber selbst für HTML nicht aus, z.B. sind die folgenden Referenzierungen damit nicht möglich:

- überlappende Bereiche („Klicken Sie hier für Kapitel 1 bis 3, hier für Kapitel 2 bis 4,...“)
- relative Bereiche („Klicken Sie hier, um zum nächsten Kapitel zu springen“)
- komplexe Bereichsdefinitionen (z.B. zum Erzeugen einer Zusammenfassung, indem die ersten Wörter jedes Absatzes dargestellt werden)
- ...

Die Möglichkeit der Referenzierung aus der HTML-Welt reicht daher für XML-Dokumente nicht aus.

#### Referenzierung in der Welt der Datenbanken

Das Äquivalent zu XPath aus der Welt der Datenbanken ist die SQL-Abfrage. Warum reicht diese etablierte Notation für XML-Dokumente nicht aus?



Im Gegensatz zur der zeilen- und spaltenorientierten Repräsentation der Daten in einem relationalen Datenbanksystemen, haben *XML*-Dokumente eine hierarchische Struktur. Der *SQL*-Standard in seiner derzeitigen Version nimmt jedoch keinerlei Rücksicht auf hierarchische Strukturen.

Zwar haben einige Hersteller von Datenbanksystemen proprietäre *SQL*-Erweiterungen implementiert, die das Traversieren hierarchischer Beziehungen erlauben, aber diese Erweiterungen wären weder ausreichend für *XML*-Dokumente, noch sind sie standardisiert, noch erweiterbar, noch intuitiv.

Sofern man auch Referenzen zwischen den verschiedenen Elementen eines *XML*-Dokumentes, und mehrerer *XML*-Dokumente untereinander zulässt, hat man es nicht mehr mit einer hierarchischen Struktur zutun, sondern mit einem „Graph“. Der Graph ist die allgemeinste aller Strukturen dieser Art.

Spätestens für Beziehungen zwischen *XML*-Elementen reicht die Möglichkeit der Referenzierung aus der Welt der Datenbanken reichend ahernicht aus.

### Referenzierung mittels XPath

Mittels *XPath* ist es möglich, beliebige Teile eines *XML*-Dokumentes anzusprechen, oder es etwas formaler ausgedrückt: beliebige Teilmengen von Knoten eines *XML*-Baumes zu beschreiben.

Strenggenommen handelt es sich nicht um Mengen und Teilmengen, sondern um Sequenzen und Teilsequenzen, da die Elemente geordnet sind. Aber der Einfachheit und Konsistenz mit der englischsprachigen Terminologie (vgl. B., „nodeset“) zuliebe, und nicht zuletzt, weil die Ordnung oft ohnehin keine Rolle spielt, wird in folgend die Begriffe „Menge“ und „Teilmenge“ verwendet.

Zu diesem Zweck stellt *XPath* verschiedene Auswahlkriterien zur Verfügung, wie z. B.:

- die Position eines Knoten innerhalb des Baumes,
- der Type eines Knotens,
- der Inhalt eines Knotens,
- ...

Mit diesen Auswahlkriterien ist es möglich, Dokumentbereiche zu beschreiben wie z. B.:

- den letzten Absatz des dritten Abschnittes
- das erste Kapitel des dritten Buches
- jeden Absatz, der das Wort „Warnung“ enthält
- alle Grafiken
- ...

Man kann sich *XPath* als eine Abfrage vorstellen, die ähnlich wie z. B. eine Datenbankabfrage auf eine Menge von Elementen angewendet wird, und eine Teilmenge dieser Elemente zurückgibt.

*XPath*-Ausdrücke haben also einen filternden Charakter, der dazu dient, eine Teilmenge einer gegebenen Menge von *XML*-Elementen zu beschreiben.

Jede *XPath*-Abfrage liefert demzufolge eine Menge von *XML*-Knoten zurück die man, „nodeset“ nennt.

## 8.3 Die XPath Syntax

### Locationpaths

Wider Erwarten besitzt *XPath* keine *XML*-Syntax. Die Syntax von *XPath* ist hervorgehoben in der Syntax von Verzeichnispfaden („paths“) eines Dateisystems („/usr/local/...“). Man nennt die *XPath*-Ausdrücke daher auch, „locationpaths“, die einzelnen Komponenten, „locationsteps“. Wie im



Dateisystem werden die *locationstep* durch den Schrägstrich („/“) als Pfadtrenner voneinander separiert, und von links nach rechts abgearbeitet.

Genau wie einzelnen Komponenten eines Dateipfades Verzeichnisnamen sein können, **können** die Komponenten eines *locationpath* XML-Elemente sein. Im Gegensatz zu Dateipfaden operiert *XPath* aber nicht auf Dateien und Verzeichnissen, sondern auf den Knoten eines XML-Dokumentes.

Die formale Syntax eines *locationpath* ist:

```
[/][location_step (/ location_step) *]
```

Das folgende Szenario soll die Verwendung und die Komposition von *Locationpaths* verdeutlichen:

Ausgangspunkt sei die in Abbildung 6 dargestellte Struktur eines Dateisystems, in der für jedes Jahr zwischen 1997 und 2001 ein Archivverzeichnis existiert. In diesem Verzeichnis mögen sich archivierte Dokumente des Benutzers „Mustermann“ befinden.

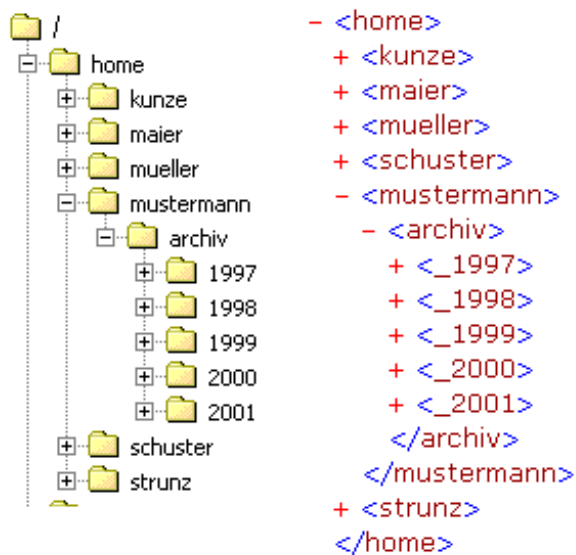


Abbildung 6 – Verzeichnisstruktur für das Beispiel

Mustermann möchte nun auf ein Dokument zugreifen, welches er 1999 erstellt hat, und an dessen exakten Namen ersich nicht mehr erinnert. Mustermann kann aber davon ausgehen, dass dessen Dateiname mit „brief“ beginnt.



Mustermann wird als „Dateipfad“, `/home/mustermann/archiv/1999/brief*` verwenden, der alle Dateien beschreibt, die sich im Verzeichnis, `/home/mustermann/archiv/1999` befinden, und deren Namen mit „`brief`“ beginnt. Dieser Dateipfad setzt sich wie folgt zusammen:

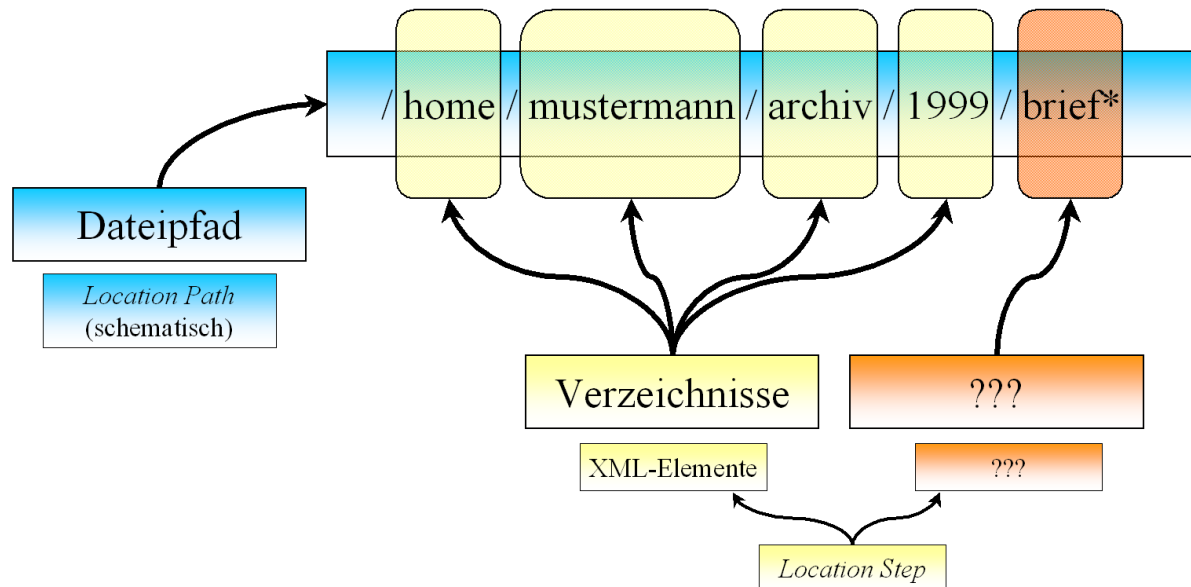


Abbildung 7 –Einzerlegter *locationpath*

Achtung: Der Dateipfad ist nur schematisch, nicht syntaktisch ein *Locationpath* !

Wie zu sehen ist, besteht der Dateipfad aus mehreren Komponenten, darunter einige Verzeichnisse, und eine unbenannte Komponente: ein Dateiname mit Platzhalter. Für die letztgenannte Komponente existiert in Bezug auf Dateisysteme keine formale Bezeichnung, es handelt sich aber nicht um den Namen einer Datei oder eines Verzeichnisses.

Auf *XPath* übertragen entspricht der Dateipfad einem *Locationpath*, aus den Verzeichnissen sind *XML-Elemente* geworden. Jede einzelne Komponente, sowohl die Verzeichnisnamen, die nun *XML-Elemente* beschreiben, als auch der Name mit Platzhalter, entsprechen nun *Locationsteps*.

### Locationsteps

Es wurde bereits erwähnt, dass *Locationsteps* dazu dienen, *Locationpaths* auszudrücken, und dass es sich bei *Locationsteps* um *XML-Elemente* handeln kann. Was aber sind *Locationsteps*, die keine *XML-Elemente* sind?

Als kleiner Appetitanreizer wird das bereits definierte Szenario mit Herr Mustermann nun erweitert: Es liegt das Jahr 1997 so weit zurück, dass sich Mustermann nicht mehr daran erinnern kann, in welchem Jahr er das gesuchte Dokument erzeugt hat.



Herr Mustermann verwendet zum Zugriff auf seinen Briefknoten, „Dateipfad“  
 „/home/mustermann/archiv/199?/brief\*“, der sich wie folgt zusammensetzt:

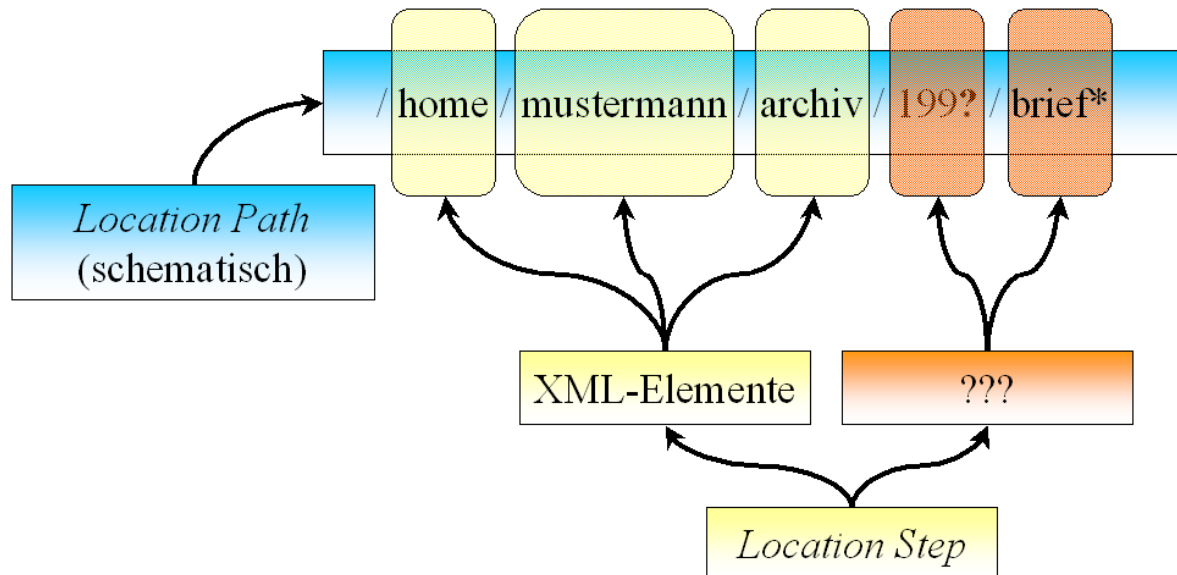


Abbildung 8 -Einzellegter *locationpath* –Teil2

Achtung: Der Dateipfad ist nur schematisch, nicht syntaktisch ein *Locationpath* !

Dieser Dateipfad stößt nun schon an die Grenzen der gängigen Befehlsinterpretation, bzw. überschreitet sie sogar – aber für *XPath* ist das erst der Anfang. In diesem Beispiel soll deutlich werden, dass sich *locationpaths* von Pfaden eines Dateisystems dadurch unterscheiden, dass sie wesentlich stärkeren Gebrauch von Platzhaltern machen, und dass diese Platzhalter auch wesentlich komplexer und mächtiger sein können als die bekannten Platzhalter „\*“ und „?“ der Dateisysteme.

Ein *locationstep* besteht, wie bereits erwähnt, im einfachsten Fall nur aus dem Namen eines *XML-Elementes*. *XPath* bietet wesentlich mächtigere Ausdrucksmöglichkeiten als jedes Dateisystem, denn ein *locationstep* kann aus

- einem Knotentyp und/oder
- einer Richtung und/oder
- einem Prädikat

bestehen. Die formale Syntax eines *locationstep* ist:

`richtung::typ([prädikat])*`

### XML Knotentypen

Die Knoten eines *XML-Dokumentes* wie z.B. Elemente, Attribute etc. lassen sich in verschiedene Knotentypen einteilen. Die Menge der definierten Knotentypen ist relativ kompakt, bekommt aber u.U. von Zeit zu Zeit Zuwachs durch einen neuen, vom W3C definierten Knotentyp. Gegenwärtig (05/2001) definierte Knotentypen sind:

- **Wurzelknoten** („root“)
- **Elemente**
- **Attribute**



- **Texte**
- Namespace-Attribute
- Processinginstructions
- Kommentare

Für einfache XSL/XSLT-Anwendungen sind vor allem die fett -markierten Knotentypen von Bedeutung.

Die Verfeinerung eines *locationstep* mittels der Angabe eines Knotentyps verkleinert die Menge der XML-Knoten, die der *locationstep* zurückgibt. Dazu sind in XPath verschiedene Bezeichner definiert, die die Auswahl eines *locationpath* auf bestimmte Typen von XML-Knoten weiter einschränken, z.B:

| Knotentyp                                 | Beschreibung                                                                                |
|-------------------------------------------|---------------------------------------------------------------------------------------------|
| <code>node()</code>                       | Element                                                                                     |
| <code>namespace</code>                    | alle Knoten im Namensraum <code>namespace</code>                                            |
| <code>namespace:typ</code>                | alle Knoten vom Typ <code>typ</code> im Namensraum <code>namespace</code>                   |
| <code>typ</code>                          | alle Knoten vom Typ <code>typ</code> und ohne besonderen Namensraum                         |
| <code>text()</code>                       | alle Textknoten                                                                             |
| <code>comment()</code>                    | alle Knoten vom Typ „comment“                                                               |
| <code>processing-instruction()</code>     | alle Knoten vom Typ „processing-instruction“                                                |
| <code>processing-instruction(name)</code> | alle Knoten vom Typ „processing-instruction“ mit <code>name</code> als <i>Expanded-Name</i> |

XPaths sieht einige Abkürzungen für Knotentypen vor, um kompakte *locationpaths* zu ermöglichen, z.B:

| abgekürzter Knotentyp | Knotentyp                                                                                           |
|-----------------------|-----------------------------------------------------------------------------------------------------|
| *                     | alle Knoten vom Typ der Achse. Achsentyp ist immer „element“ außer bei „attribute“ und „namespace“. |
| @                     | attribute::                                                                                         |

### Blickrichtungen und der Kontextknoten

XPath kennt sowohl absolute Pfade, als auch relative Pfade; auch hier wieder eine Gemeinsamkeit mit Dateisystemen. Der Begriff des relativen Pfades impliziert, dass es einen „Zustand“ geben muss, und diesen nennt man den „Kontextknoten“. Der Kontextknoten entspricht dem Arbeitsverzeichnis eines Dateisystems.

Genau wie bei einem Dateisystem beginnen auch bei XPath absolute Pfade mit dem Pfadtrenner („/“), alle anderen Pfade sind relative Pfade.



Verwendet man in einem bestimmten Arbeitsverzeichnis eines Dateisystems eine relative Pfadangabe, so navigiert man in eine von zwei „Richtungen“: nämlich entweder vorwärts („z.B., `cd texte`“), oder zurück („`cd ..`“). Mit *XPath* und dem Kontextknoten verhält sich genauso, mit dem Unterschied, dass es mehr als zwei Richtungen gibt.

Die Achsbezeichnung ist ausgehend vom Kontextknoten die „Blickrichtung“. Eine mögliche Blickrichtung wäre z.B. in Richtung der nachfolgenden *XML*-Elemente, eine andere wäre in Richtung des vorhergehenden *XML*-Elementes, und wieder eine andere wäre in Richtung der zugehörigen *XML*-Attribute.

Die Angabe einer Richtung ist nicht unbedingt nötig. Je nach Kontext gibt es für die Richtungen einen definierten Vorgabewert, ist der Kontextknoten z.B. ein *XML*-Element so ist der Vorgabewert für die Richtung „nachfolgende Elemente“ (d.h., „Kinderelemente“).

| Richtung           | Bedeutung                                     |
|--------------------|-----------------------------------------------|
| self               | aktueller Knoten                              |
| child              | direkt nachfolgende Elemente                  |
| descendant         | nachfolgende Elemente                         |
| descendant-or-self | nachfolgende Elemente oder aktuelles Element  |
| parent             | direkt vorhergehendes Element                 |
| ancestor           | vorhergehende Elemente                        |
| ancestor-or-self   | vorhergehende Elemente oder aktuelles Element |
| attribute          | nicht-Namespaces-Attribute                    |
| namespace          | Namespaces-Attribute                          |
| preceding-sibling  | direkter Vorgänger                            |
| following-sibling  | direkter Nachfolger                           |
| preceding          | alle Vorgänger                                |
| following          | alle Nachfolger                               |

Darüber hinaus gibt es noch einige Abkürzungen für Richtungen, die folgende Auswahl ist allerdings nur exemplarisch:

| abgekürzte Richtung | Richtung                   |
|---------------------|----------------------------|
| (leer)              | child::                    |
| .                   | self::node()               |
| ..                  | parent::node()             |
| //                  | descendant-or-self::node() |

## Prädikate

Mittels eines Prädikates lässt sich die Auswahl eines *locationstep* noch weiter und noch granularer einschränken, als dies durch die Angabe von Knotentyp und/oder Richtung möglich ist. Ein Prädikat



isteineBedingung,d.h.eineFunktion,diebeiderInterpretationzu,,wa hr“oder,,falsch“ausgewertet werdenkann.

DieSprache,indereinPrädikatspezifiziertwird,isteineTeilsprachevon XPath.IneinemPrädikat könnenVergleicheangestelltwerden,arithmetischeundboolescheOperatorenangewendetwerden, undsofarrelativkomplexeFunktionenbenutztwerden,einigedavonz.B.zurStringbearbeitung.

Einige,beispielhafteFunktionenin XPath sind:

- last()
- position()
- count()
- id()
- name()
- substring()
- concat()
- true()
- round()
- translate()

XPathkenntübrigensDatentypeninnerhalb esDokumentes,alsoistz.B.füretwaige ZahlenvergleicheeineKonvertierungmitder number()-Funktionvorzunehmen.

Auchfüreinige XPath-PrädikategibtesAbkürzungen,soz.B:

| abgekürztesPrädikat | Prädikat         |
|---------------------|------------------|
| [ 2 ]               | [ position()=2 ] |

### Beispielefürlocationpaths

HiernuneinigeBeispielefür locationpaths ,dierechtwillkürlichgewählt sind:

| locationpath                                  | Abkürzung       | Beschreibung                                  |
|-----------------------------------------------|-----------------|-----------------------------------------------|
| child:text()                                  | text()          | Textelemente derdirekt nachfolgenden Elemente |
| /child:HTML                                   | /HTML           | Elemente „HTML“ unterhalbvom Wurzelknoten     |
| /descendant-or-self::node()/child::table      | //table         | Elemente „table“überall imDokument            |
| /parent::node()/child::table/attribute::width | ../table/@width | Attribute „width“aller                        |



| locationpath                                                    | Abkürzung                          | Beschreibung                                                      |
|-----------------------------------------------------------------|------------------------------------|-------------------------------------------------------------------|
| <code>/child::HTML/child::body/child::HTML[2]</code>            | <code>/HTML/body/table[2]</code>   | Tabellen, die Kinder des direkt vorfahrenden Knotens sind         |
| <code>/descendant-or-self::table[attribute::width="50%"]</code> | <code>//table[@width="50%"]</code> | Die zweite Tabelle im HTML-Body                                   |
| <code>//td[position() != last()]</code>                         |                                    | alle „td“-Elemente im Dokument mit Ausnahme des letzten           |
| <code>//namespace:head</code>                                   |                                    | alle „head“-Elemente, die im Namensraum namespace deklariert sind |

### abgekürzte Ausdrücke für Prädikate

| abgekürztes Prädikat | Bedeutung                                    |
|----------------------|----------------------------------------------|
| <code>'Name'</code>  | Testet, ob Knoten diesen Namen hat           |
| <code>*</code>       | Knoten mit beliebigem Namen                  |
| <code>text()</code>  | Ein Textknoten                               |
| <code>..[..]</code>  | Zusätzliches Prädikat, das erfüllt sein muss |



## 9 Ein Praxisbeispiel für XSL

### 9.1 Einleitung

*Und die Praxis...?*

Dieses Kapitel ist ein Beispiel für die Anwendung von *XSLT* gewidmet, und nicht ein beliebiges Beispiel, sondern ein **Praxisbeispiel**.

Auf der Suche nach Literatur für diese Ausarbeitung und den zugehörigen Vortrag bin ich wieder und wieder auf sogenannte „Hello World!“-Beispiele gestoßen. Da wurden drei Adressdaten als *XML*-Datei gespeichert und via *XSL* auf dem Bildschirm ausgegeben. Da wurden vier oder fünf Buchbestellungen, Getränkevorräte und Kinofilme *XML*-kodiert und anschließend transformiert. Allesamt Beispiele, die die prinzipielle Funktionsweise von *XSL* demonstrieren, aber einen skeptischen Leser doch nicht wirklich von *XSL* überzeugen können.

Aus diesem Grund, und auch aus der Gunst der Stunde heraus möchte ich in dieser Ausarbeitung einen anderen Weg beschreiten: ich werde im Folgenden ein Beispiel aus der Praxis vorstellen, an dessen Realisierung ich selber mitgearbeitet habe:

die Software **telemallCOMMUNITY** der Frankfurter Firma telemall AG (<http://www.telemall.de>).



Weitere Informationen zur telemallCOMMUNITY sind unter [info@eWorks.de](mailto:info@eWorks.de) oder [info@telemall.de](mailto:info@telemall.de) erhältlich.

### 9.2 telemallCOMMUNITY

*telemallCOMMUNITY - Was ist das wie eine Software...?*

telemallCOMMUNITY ist eine *XML/XSLT*-basierende Standardsoftware der Firma telemall AG (<http://www.telemall.de>) für den Bereich der Internet-Communities, mit Anlehnungen an Content Management-(CMS-) und Portal-Features.

telemallCOMMUNITY wurde im letzten Quartal 2000 von Kai Klüber, Martin Klossek ([Klossek@eWorks.de](mailto:Klossek@eWorks.de)), und mir selber ([Wleklinski@eWorks.de](mailto:Wleklinski@eWorks.de)) konzipiert, und im ersten Quartal 2001 implementiert.

telemallCOMMUNITY fußt technisch auf

- dem Webserver *Apache*,
- der Skriptsprache *PHP*,
- dem Datenbankserver *MySQL* und
- dem *XSLT*-Prozessor *Sablotron*.

telemallCOMMUNITY erzeugt anders als konventionelle, serverseitige Skriptlösungen keine *HTML*-Ausgabe, sondern eine *XML*-Ausgabe. Die verwendete *XML*-Sprache wurde dafür eigens von den Entwicklern definiert. Die erzeugte *XML*-Ausgabe entspricht dabei den Informationen, die auf einer



Seite angezeigt werden sollen, und wird mittels eines *XSL*-Stylesheets, und eines *XSL*-Prozessors nach *XHTML* transformiert.

## 9.3 Warum XML und XSL/XSLT?

Welche Gründe sprechen beider Konzeption für die Verwendung eines XML/XSL -Framework...?

Es muss konkrete Gründe dafür geben, beider Implementierung einer Internetpräsenz eine Kombination von *XML* und *XSLT* zugeben, denn immerhin gibt es ja auch konkrete Gründe, die dagegensprechen, als das wären:

- *XML/XSL* ist langsamer als statisches *HTML*,
- *XML/XSL* ist nicht auf jedem System verfügbar,
- *XML/XSL* erfordert technische Weiterbildung bei Entwicklern und Webdesignern und
- *XML/XSL* erfordert Neugestaltung des Workflows in Bezug auf Webdesign.

Es gibt viele Gründe, die für die Verwendung von *XML/XSL* sprechen, die wichtigsten fünf davon sind die folgenden:

### 1. Trennung von Implementierung und Design

- Die Verwendung von *XML/XSL* erhöht die Wiederverwendbarkeit der Software. Es sind keine Layoutinformationen in dem Programmcode oder der erzeugten Ausgabe der Software enthalten.

Das Layout ist zwar keinesfalls alles, was zwei verschiedene Implementierungen auf Basis derselben Standardsoftware voneinander unterscheidet, aber Layout ist ein Aspekt, in dem sich zwei Implementierungen auf jeden Fall voneinander unterscheiden.

Durch die strikte Trennung von Implementierung und Design wird der layoutspezifische Teil des Anpassungsaufwandes, der bei einer Standardsoftware in diesem Bereich den größten Teil ausmacht, weg von den (rare) Programmierern, und hinzuden (preiswerteren) Webdesignern verlagert.

- Da es durch *XML/XSL* eine extrem flexible, und vor allem klar definierte Schnittstelle zwischen der Funktionalität und dem Layout gibt, ist der Verwender der Standardsoftware unabhängig von den Entwicklern derselben. Layoutänderungen erfordern keinen Zugriff auf den Programmcode mehr.
- *XML/XSL* kann den Abspracheaufwand zwischen Programmierern und Webdesignern verringern, wenn die verwendete *XML*-Sprache klar definiert und allen bekannt ist. Insbesondere können mehr Prozesse parallelisiert werden, z.B. kann die Erstellung des Layout zeitgleich mit der Implementierung einer Lösung beginnen:

der unternehmensinterne Workflow wird optimiert.

### 2. Personalisierung

- Unter „Personalisierung“ versteht man die Anpassbarkeit von Websites aus der Sicht des einzelnen Benutzers. *XML/XSL* erleichtert die Realisierung von personalisierbaren Websites, da alle Layoutänderungen durch *XSL*-Änderungen bewirkt werden können.

Das könnte im absurden Maße weit gehen, dass ein Benutzer ein eigenes *XSL*-Stylesheet besitzt, und dieses bearbeiten kann.



### 3. Erweiterbarkeit

- *XML/XSL* als ausgegebenes Datenformat garantiert die Erweiterbarkeit auf alternative und auch heute noch unbekannte Ausgabemedien und –formate wie *WAP/WML*, *Windows CE* - und *Palm* -Geräte, *Print* -Medien etc.

### 4. Strenge Modularisierung durch offene Protokolle

- Mehrere Komponenten des Systems sind als autonome Module realisiert, die über eine spezielle *XML*-Sprache und das *http* -Protokoll mit dem Rest des Systems kommunizieren, dazu gehören z.B. der Kommunikations -Server und die Benutzerverwaltung, „UserEngine“ (<http://www.userengine.de>)

Zwar sind die Verwendung von *XML/XSL* für die Datenausgabe und die Verwendung von *XML* für interne Protokolle grundsätzlich unabhängig voneinander, aber sie begünstigen sich zumindest gegenseitig –dagegen sind *XML*-Technologien wie sie bereits in die Software integriert ist.

### 5. Offene Schnittstellen

- Es existieren Schnittstellen für den Austausch von Daten mit externen Anbietern von Inhalten (Nachrichtenagenturen etc.) über *XML*-Sprachen wie *SOAP*, *NewsML*, *RDF* etc.

Auch hier gilt, dass die Verwendung von *XML/XSL* für die Datenausgabe und die Verwendung von *XML* für externe Schnittstellen grundsätzlich unabhängig voneinander sind, aber sie gegenseitig begünstigen.

## 9.4 Technischer XML -Background der telemail COMMUNITY

Wie sehen die erzeugten XML -Daten denn eigentlich aus...?

Die telemail COMMUNITY erzeugt einen umfangreichen Ausgabestrom von *XML*-Daten, von dem ich aus Platzgründen nur einen kleinen Teil vorstellen möchte.

Eine Internet -Community besteht im Allgemeinen aus vielen, vielen Datensätzen, die dazu prädestiniert sind, als Beispiel für die Transformation von *XML* mittels *XSL* herzuhalten. Solche Datensätze sind z.B .

- Nachrichtenmeldungen
- Diskussionsbeiträgen
- empfangene E -Mails
- Einträgen im Kleinanzeiger
- Veröffentlichungen an „schwarzen Brettern“
- Presseclippings
- ...

Die *XML*-Anweisungen, die von der telemail COMMUNITY für z.B. eine Nachrichtenmeldung erzeugt werden, können wie folgt aussehen:

```
<record ikey="1000" parentid="100">
 <v_1 ikey="1000">
 <date>2001-03-09 17:36:49</date>
 <authorid>12364</authorid>
 <authorname>
 Manfred Mustermann
 </authorname>
 <authoremail>
```



```
musterma@informatik.uni-frankfurt.de
</authoremail>
<authorinitials>
 MM
</authorinitials>
<nodeicon src="http:.."/>
<subject src="http:..">
 Datenschützer warnen vor gläsernem Internet-Nutzer
</subject>
<newlink src="http:..">neu</newlink>
<dellink src="http:..">loeschen</dellink>
</v_1>
</record>
```

TransformiertmandieseXML -Datenmiteinem XSL-Stylesheetundeinem XSL-Prozessornach HTML,soentstehenz.B.diefolgendenDarstellungen:

**Datenschützer warnen vor gläsernem Internet-Nutzer**  
vom 09.03.2001 - N D

Die Datenschutzbeauftragten des Bundes und der Länder sehen die Anonymität im Internet in Gefahr.

Abbildung 9 –Transformation1derCommunityXML -Daten

#### **Kino**

**09.03. Datenschützer warnen vor gläsernem Internet-Nutzer**

**Die Datenschutzbeauftragten des Bundes und der Länder sehen die Anonymität im Internet in Gefahr.**

Abbildung 10 -Transformation2derCommunityXML -Daten



Datenschützer warnen vor gläsernem Internet-Nutzer (neu) (löschen)

Abbildung 11 -Transformation3derCommunityXML -Daten

DieseDarstellungenindnurexemplarisc h,undkönnendurchModifikationendes XSL-Stylesheets beliebigabgeändertwerden.



In Wirklichkeit ist die Menge der erzeugten XML-Anweisungen wesentlich größer, als hier gezeigt werden kann. Ein Screenshot der Visualisierung der XML-Daten mit einem Webbrowser sieht etwa so aus:



Abbildung 12 –derXML -StromdertelemallCOMMUNITY

Auch dies ist nach wie vor ein Bruchteil der vom System erzeugten XML-Daten, wie sich an dem Scrollbalken im rechten Bereich des Bildes unschwer erkennen lässt.

### 9.5 XML/HTML Konvertierung mittels Sablotron

Das Erstellen der XSL-Stylesheets ist eine Aufgabe, die im Allgemeinen von den Entwicklern des Systems weder übernommen werden möchte, noch sollte – denn bei den Entwicklern handelt es sich um Entwickler, und nicht um Designer.

Demzufolge stand uns während der Entwicklung der telemallCOMMUNITY kein XSL-Stylesheet zur Verfügung, das wir hätten benutzen können – also schreiben wir uns selbster eines. Dieses ist zwar spartanisch und hässlich, erfüllt aber den Anspruch, dass es alle Informationen in HTML umwandelt.



Während der Entwicklungszeit wurde als weitestgehend **ohne Layout** gearbeitet. Die Darstellung sah in etwa wie folgt aus:

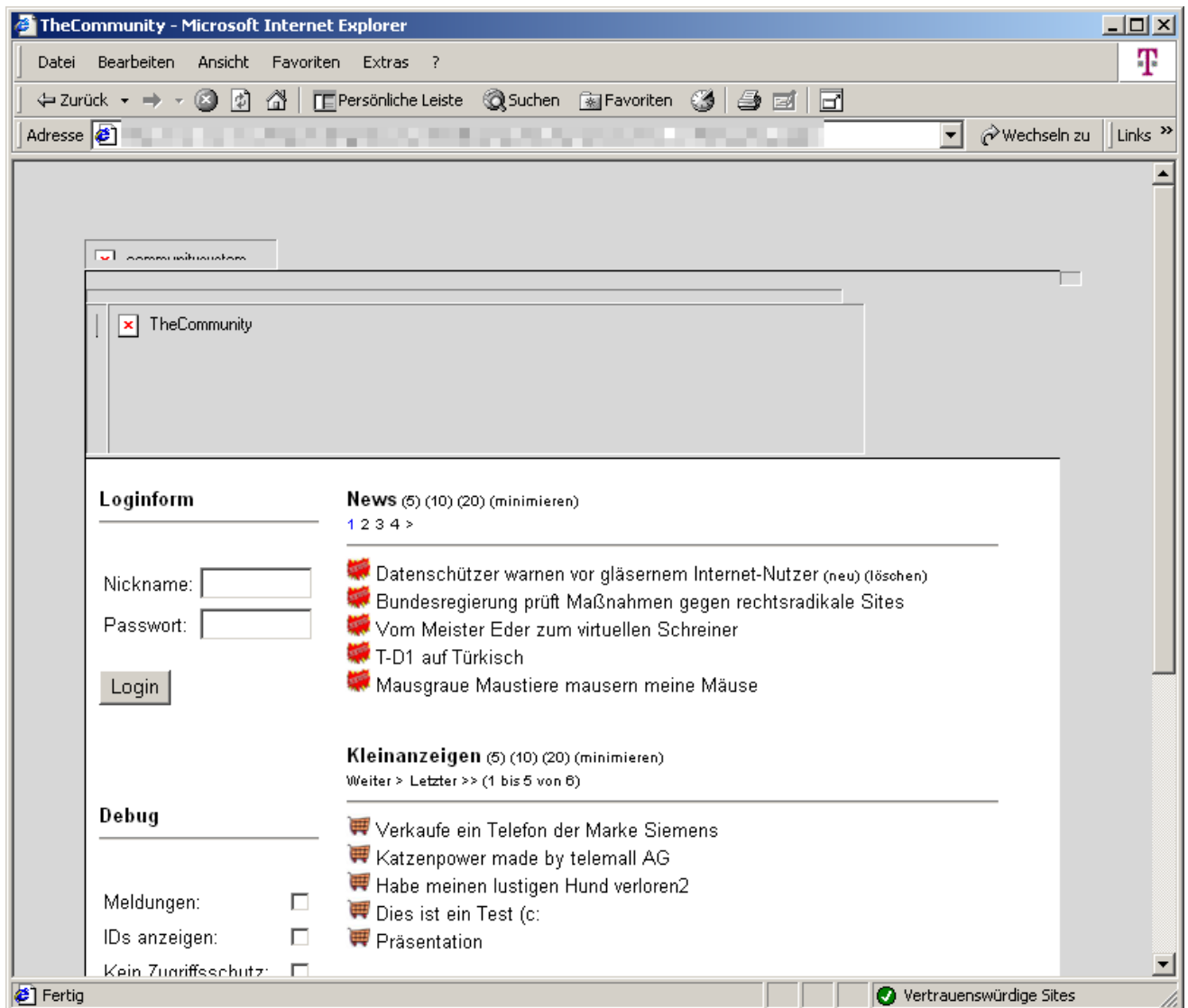


Abbildung 13 – Transformation 1 in eine telema IICOMMUNITY Startseite

Nachdem die Webdesign -Abteilung ihre Arbeit an den Stylesheet beenden hatte, in etwa zeitgleich mit dem Ende unserer Entwicklungszeit, konnten wir bereitgestellte Stylesheets in das System einbinden.

„Einbinden“ bedeutet in diesem Zusammenhang nicht das Schreiben oder Verändern von Programmcode, sondern lediglich das Kopieren der XSL-Datei in ein entsprechendes Verzeichnis!



DasLayoutsahnachderEinbindungdes XSL-Stylesheetsprunghaftbesserundvollkommenneuaus, undzwarso:

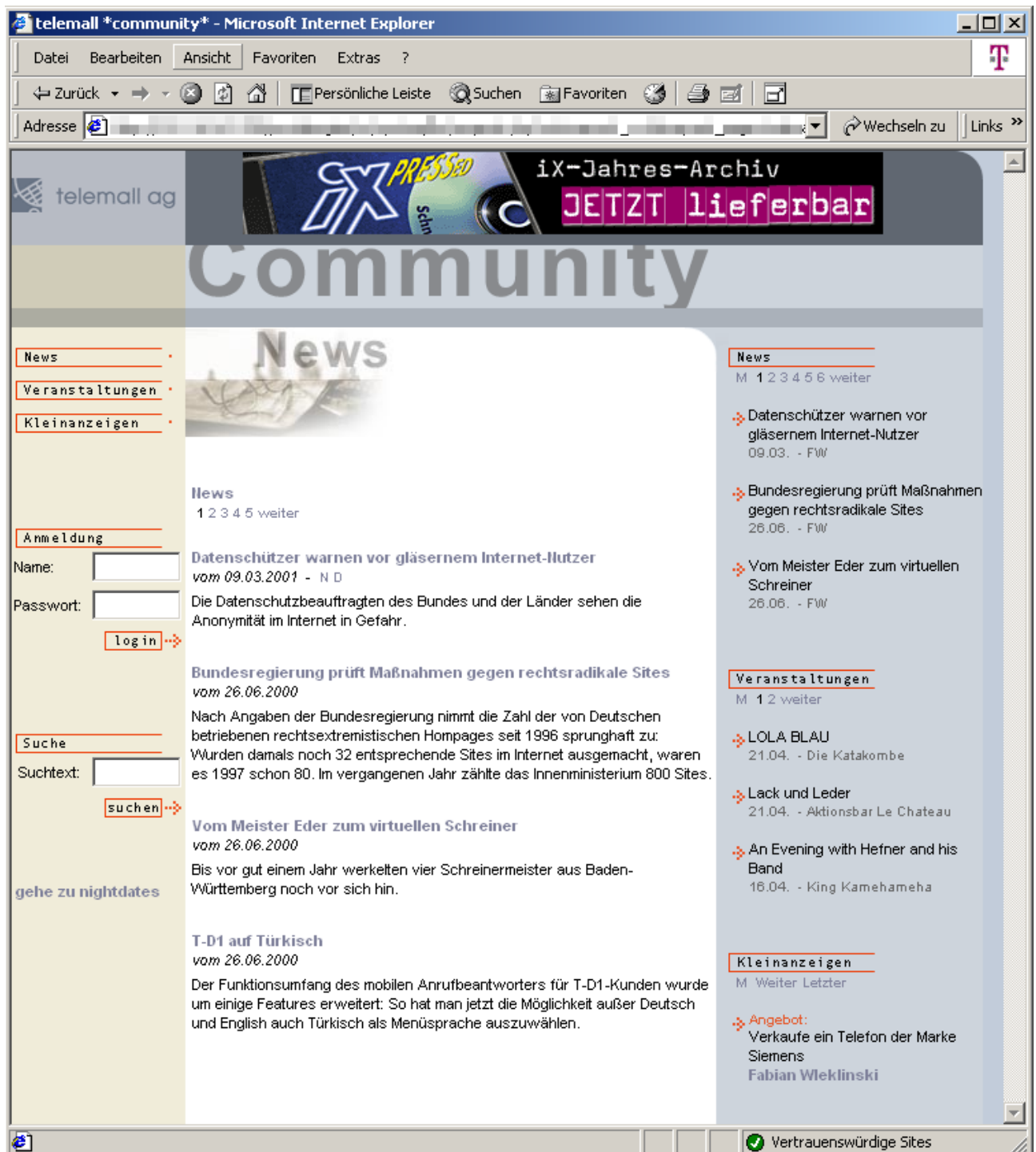


Abbildung 14 -Transformation2einertelemallCOMMUNiTYStartseite

UmdieneuartigenMöglichkeitenvonXML/ XSLbesondersherauszustellen,wurdefürdie PräsentationendesSystemsnichtnurein XSL-Stylesheet,sondernzwei XSL-Stylesheetsgeschrieben, zwischendeneninderWebbrowser -AnsichtübereinHyperlink(siehe Abbildung 14amlinkenRand) gewechseltwerdenkonnte -inEchtzeit,verstehtsich!



Durch ein Klick auf diesen Hyperlink wird das System angewiesen, für die Transformation nach HTML ein anderes XSL-Stylesheet zu verwenden, und schlagartig ändert sich die Ausgabe wie folgt:

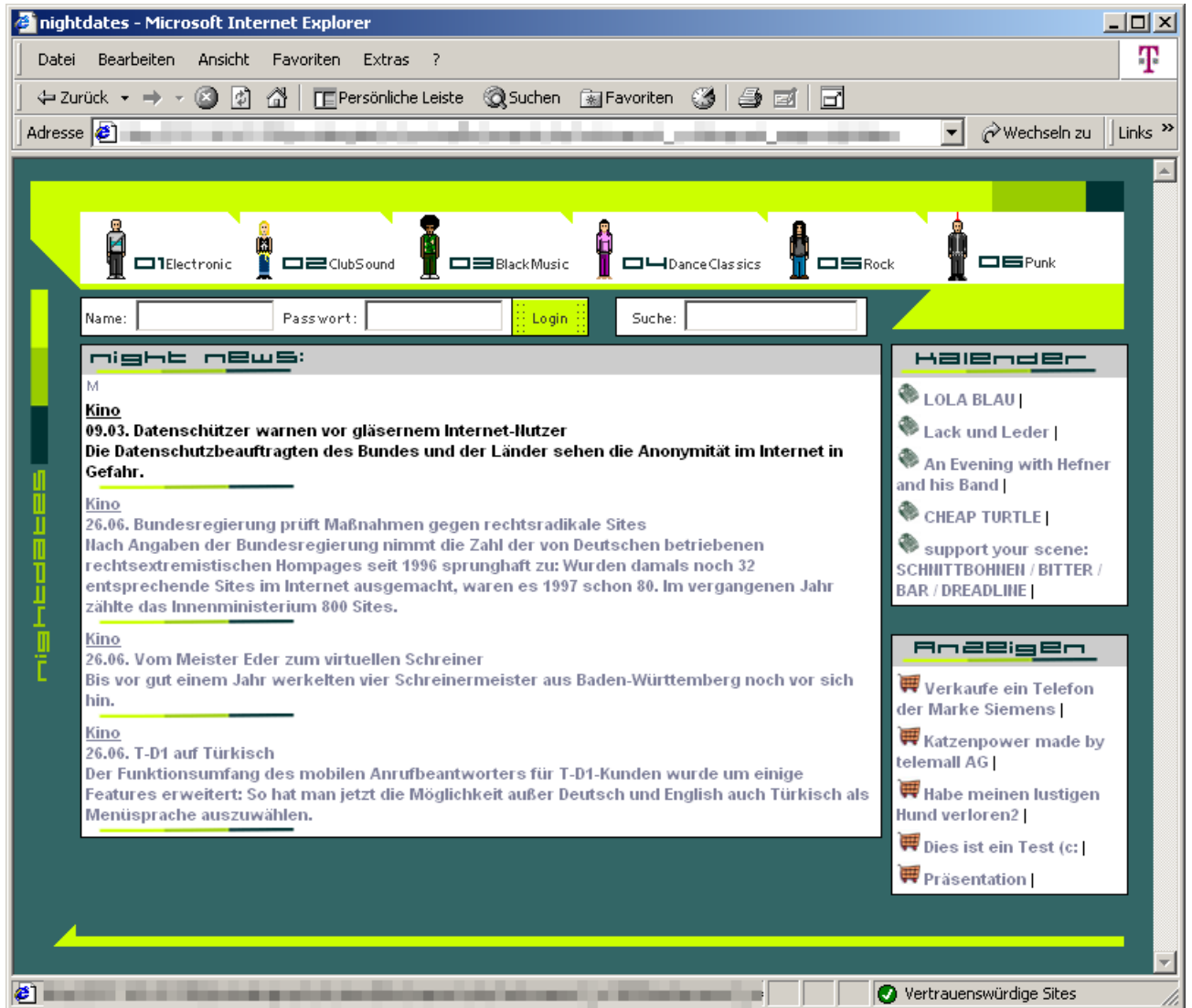


Abbildung 15 - Transformation einer telemallCOMMUNITY Startseite

Die Unterschiede zwischen den beiden Ausgabevarianten, die nur durch die Variation der Stylesheet hervorgerufen wird, sind sehr beeindruckend. Die hier erkennbaren Unterschiede sind nicht einmal das Ende der Fahnenstange, und doch sehen beide Darstellungen schon so grundverschieden aus, dass man beim Betrachten kaum glauben möchte, dass sie von derselben Software und demselben Datenbestand erzeugt werden!



## 10 Anhang

### 10.1 Häufige Fragen

Was ist so neu an XSL? Warum nicht CSS? Warum, warum, warum...?

#### Vorteile gegenüber CSS

- In etwa 90% der Formatierungseigenschaften aus XSLFO sind in derselben oder in ähnlicher Form CSS entlehnt. Wer also mit CSS Erfahrung hat, wird sich mit XSLFO nicht schwertun. Insbesondere, „fehlen“ in XSLFO keine Merkmale von CSS.
- XSL bietet durch XPath wesentlich weitreichendere und mächtigere Selektoren als CSS, dadurch sind granularere Beschreibungen und effektivere Baumkonstruktionen möglich.
- XSLFO bietet ein Seitenlayout -Modell, was CSS nicht leistet
- XSLFO bietet CSS gegenüber ein erweitertes Formatierungsmodell
- XSLFO bietet CSS gegenüber eine bessere Internationalisierung und mehr Ausgabemodi
- XSL ist eine XML-Sprache. Bestehende XML-Software wie XML-Editoren etc. können für XSL verwendet werden.
- Anders als CSS bietet XSL mit XSLT eine Möglichkeit zur Umstrukturierung des XML-Quellbaumes

#### Können XSL und CSS kombiniert werden?

- Ja, sinnvoll sind z.B. hybride Lösungen, d.h. eine XSL-basierende XML/HTML-Umwandlung auf dem Server, und eine CSS-Formatierung auf dem Client

#### Welche Bedeutung hat XML/XSL heute?

- Es gibt bereits eine Menge Websites, die Kombinationen von XML und XSL einsetzen, dazu gehört z.B. einer der weltweit am häufigsten besuchten Websites - nämlich [www.microsoft.com](http://www.microsoft.com).

#### Welche Bedeutung wird XSL in Zukunft haben?

- Die Bedeutung alternativer Plattformen steigt, erwähnt seien hier nur Handheld -PCs, Mobiltelefone, Kabel -TV, Spielekonsolen, und so weiter. Die Bereitstellung von Daten für diese Plattformen wird ebenfalls steigen. XML/XSL-Lösungen sind prädestiniert für diese Aufgabe.
- Sollten XSL-Prozessoren zukünftig in üblichen Client -Software wie z.B. Webbrowser Einzug halten, entstehen neue Möglichkeiten der Transformierung, z.B. können Websites dann sinnvoll in akustische Informationen, oder in Informationen für Blindenschrift -Lesegeräte transformiert werden

### 10.2 Ausblick

XML kann sehr breit verwendet werden, und wird dies auch bereits in der Gegenwart (05/2001). Es könnte in der Lage sein, die im Beginn dieser Ausarbeitung angesprochenen Probleme in Bezug auf den Datenaustausch sowohl in Informatik und Wirtschaft, sowie in vielen anderen Bereichen (Politik, Wissensbildung etc.) zu lösen.



Bereits heute wird XML von mehr Anwendungen eingesetzt, von mehr Firmen propagiert, und von mehr Menschen gekannt, beherrscht als dies bei der XML-Übermenge SGML jemals der Fall war. Zukünftig wird XML möglicherweise wirklich die Sprache des Internets – die Zeichen dafür stehen gut.

Durch seine Standardisierung seitens des W3C und seine große Akzeptanz sogar von Monopolisten und Marktherrschern, wird XML auch in der nahen Zukunft eine große Rolle spielen. Durch XML ist anwendungs- und plattformübergreifender Datenaustausch zu einem salonfähigen Marketinginstrument geworden.

Probleme werden zukünftig möglicherweise auftreten, wenn verschiedene Hersteller für die gleiche Art von Daten (z.B. Text/Textverarbeitung) jeweils eigene DTDs definieren. Allerdings werden diese Probleme im Gegensatz zu heute stark entschärft sein, denn die Anwendungen einer entsprechenden XSL-Transformation sollte dann zur Konvertierung ausreichen.



# 11 Literaturverzeichnis

## 11.1 Eine Anmerkung vorneweg

Warum ist das Literaturverzeichnis so umfangreich...?

Bei XML und XSL, sowie allem, was damit in irgendeiner Art und Weise zusammenhängt, handelt es sich um „hype“ Themen – daher gibt es Unmengen an offline und online verfügbarer Literatur darüber. Die Schwierigkeit der Recherche besteht nicht darin, Literaturquellen auszumachen, sondern eher darin, einen großen Teil der Literaturquellen auszuschließen.

Soberuht diese Ausarbeitung auch nicht auf der Essenz zweier oder dreier „großer“ Werke, sondern ist eher ein Konglomerat Dutzender von Spezifikationen, Beispielen, Kommentaren, Reportagen, Tutorien, Seminaren und Ausarbeitungen aller Art, aus den Federn anderer Studenten, Professoren, gestandener Autoren, Institutionen und Firmen.

Jede Literatur, die ich zur Vorbereitung auf diese Ausarbeitung und die zugehörige Präsentation verwendet habe, ist online verfügbar.

Von der ersten Website, die ich zwecks Informationsbeschaffung für diese Ausarbeitung besucht habe, bis zur allerletzten Website sind alle Informationsquellen in dem folgenden Literaturverzeichnis enthalten. Das Literaturverzeichnis ist deswegen sehr umfangreich. Ich habe darauf verzichtet, für jede einzelne Quelle Details wie Veröffentlichungsdatum oder Verlag bzw. Redakteure etc. anzugeben, nennend dafür aber die Adressen der entsprechenden Website.

Falls der Ein oder Andere dieses Literaturverzeichnis als Ausgangspunkt für eine eigene Recherche verwenden möchte, dann lege ich ihm besonders einen Besuch der aufgeführten XML-Portalseiten nahe.

## 11.2 Literatur

### w3c

- Extensible Markup Language (XML)  
<http://www.w3.org/TR/REC-xml>
- XSL Transformations (XSLT)  
<http://www.w3.org/TR/XSLT>
- Extensible Stylesheet Language (XSL)  
<http://www.w3.org/Style/XSL/>
- XML Path Language (XPath)  
<http://www.w3.org/TR/XPath>

### Anwendungen

- XJax: XML Schema -> Java Mapping  
<http://xjay.sourceforge.net/>
- XSL Barcode Generator  
<http://www.renderx.com/barcodes.html>



- XSLT Stylesheets - Useful things and other jokes  
<http://www.informatik.hu-berlin.de/~obecker/XSLT/>
- Merging XML Documents  
<http://www.devguy.com/fp/XML/XSL/Merge/index.html>

### Software

- XML Software Guide: Specialized XML Software  
<http://wdvl.com/Software/XML/special.html>
- XML Software Guide: Additional XML Software Resource Lists  
<http://wdvl.com/Software/XML/resources.html>

### XSLT Prozessoren

- SAXON homepage  
<http://users.iclway.co.uk/mhkay/saxon/>
- Sablotron: XSL Transformations Processor  
<http://www.gingerall.com/charlie-bin/get/webGA/act/sablotron.act>
- Unicorn XML Toolkit  
[http://www.unicorn-enterprises.com/products\\_xmlkit.html](http://www.unicorn-enterprises.com/products_xmlkit.html)
- AntennaHouse XSLFO formatter  
<http://www.antennahouse.com/XSLformatter.html>
- FOP (Apache)  
<http://xml.apache.org/fop/>
- Übersicht von Free XML Tools  
[http://www.garshol.priv.no/download/xmltools/name\\_ix.html](http://www.garshol.priv.no/download/xmltools/name_ix.html)

### Zeitschrift „ix“:

- 3/1998: „Klammern gehört zum Handwerk - DSSSL: XML - Dokument für Webformatieren“  
<http://www.heise.de/ix/artikel/1998/03/156/>
- 5/1998: „Daten verpflichten - XSL: die Stil - Sprache für XML“  
<http://www.heise.de/ix/artikel/1998/05/138/>
- 1/2001: „XSLT-Tutorial“  
<http://www.heise.de/ix/artikel/2001/01/167/>
- 11/1999: „Stil - Leben: Transformation von XML - Dokumenten“  
<http://www.heise.de/ix/artikel/1999/11/181/>
- 11/2000: „Dehnbare Formate: SVG und VML: XML - Alternativen für dynamische Webgrafiken“  
<http://www.heise.de/ix/artikel/2000/11/148/>
- 7/2000: „Erste Beta von KDE 2.0 - XML überall“  
<http://www.heise.de/ix/artikel/2000/07/110/>
- 9/2000: „BMEcat - XML mit Perl generiert - Kataloge für den Mittelstand“



- <http://www.heise.de/ix/artikel/2000/09/166/>
- 4/2000: „Wenn Markup lockt - XML - Daten auf's Handy bringen“  
<http://www.heise.de/ix/artikel/2000/04/178/>
- 5/1999: „Ein Titel ist ein Titel - Name namespaces in der Extensible Markup Language“  
<http://www.heise.de/ix/artikel/1999/05/142/>
- 11/1998: „Faulheit siegt - HTML-Seiten aus Datenbeständen über XML generieren“  
<http://www.heise.de/ix/artikel/1998/11/186/>
- 6/1998: „Vorsicht, Markup - Mozilla - Release mit XML und CSS“  
<http://www.heise.de/ix/artikel/1998/06/142/>
- 1/1998: „Mauerwerkendetail - Fortschritte im XML - Umfeld: Metadaten“  
<http://www.heise.de/ix/artikel/1998/01/116/>
- 4/2001: „Im Anschnitt - XML - Daten mit PHP verarbeiten“  
<http://www.heise.de/ix/artikel/2001/04/201/>
- 6/1997: „Revolution der Experten - XML: Professionelle Alternative zu HTML“  
<http://www.heise.de/ix/artikel/1997/06/106/>

## Portale

- XML/XSL Portal  
<http://www.bayes.co.uk/xml/index.xml>
- Jeni's XSLT Pages  
<http://www.jenitennison.com/XSLT/index.html>
- XMLSOFTWARE  
<http://www.xmlsoftware.com/>
- Web Developer's Virtual Library: Encyclopedia of Web Design Tutorials, Articles and Discussions  
<http://wdvl.com/>
- XML developer news from XMLhack: by and for the XML community  
<http://xmlhack.com>
- ActiveState -- Programming for the People  
<http://www.activestate.com/>
- ZVON.org  
<http://www.zvon.org>

## Benchmarks

- XSLBench - A XSLT Processor Benchmark  
<http://www.tfi-technology.com/xml/XSLbench.html>
- XSLTMark - the first XSLT processor performance benchmarking application  
<http://www.datapower.com/XSLTMark/>



- *XSLTProcessorBenchmarks*  
<http://www.xml.com/pub/a/2001/03/28/XSLTmark/>

### Mailinglisten

- *XSL-List --OpenForumon XSL*  
<http://www.mulberrytech.com/XSL/XSL-list/>
- *www-XSL-fo@w3.orgMailArchives*  
<http://lists.w3.org/Archives/Public/www-XSL-fo/>

### Kommentare

- *lucapassani:meand XSL*  
<http://groups.yahoo.com/group/XHTML-L/message/1015>
- *DoyouGrok XSLT?*  
<http://xmleverywhere.com/newsletters/20000329.htm>

### Praxisbeispiele

- *OnlineCarpentry:CraftingaNewMSDNTableofContents*  
<http://msdn.microsoft.com/voices/msdntoc.asp>

### Paper

- *DSSSL*  
<http://www-rn.informatik.uni-bremen.de/lehre/sgml/www/referate/dsssl/HTML/dsssl.html>
- *RobertWruck: XLink, XPath& XPointer*  
<http://www.fh-wedel.de/~si/seminare/ws00/Ausarbeitung/3.XLink/XLink0.htm>
- *JensWilke:AbfragesprachenfürXML -Dokumente*  
<http://www3.informatik.tu-muenchen.de/public/lehre/lehre.WS00/HS00-Ausarbeitungen/HS00-Ausarbeitung8.pdf>
- *HeikoFaasch: XSLT-Die XSLTransformationssprache*  
<http://www.fh-wedel.de/~si/seminare/ws00/Ausarbeitung/5.XSLT/XSLT0.htm>
- *Vorlesung"Bibliothekstechnik",SchwerpunktElektronischesPublizierenmitXML*  
<http://amor.rz.hu-berlin.de/~h0077dfz/bibliothekstechnik4.html>
- *MikeBecker: XSL -ExtensibelStilesheetLanguage*  
[http://i44www.info.uni-karlsruhe.de/~i44www/lehre/XML-WS-99-00/vortraege/XSL/XSL\\_frame.htm](http://i44www.info.uni-karlsruhe.de/~i44www/lehre/XML-WS-99-00/vortraege/XSL/XSL_frame.htm)
- *EugenDück : ProseminarDatenmodellierung - VortragüberXML*  
<http://www3.informatik.tu-muenchen.de/public/lehre/lehre.WS00/PS00-Ausarbeitungen/dueck-xmlproseminar.html>
- *AndreasKempf,MarioGanter,JörgKnaust,MarkusKurczek:ExtensibleStyleLanguage*  
<http://stio1.sari.fh-wuerzburg.de/student/i199/XSL/>



## XMLund XSLTEditoren

- Whitehill < XSL > Composer  
<http://www.whitehill.com/products/prod4.html>
- XSLTDocApplication  
<http://www.jenitennison.com/XSLT/utilities/>
- XMLCooktop  
<http://xmleverywhere.com/cooktop/>
- XMLSOFTWAREXMLEditors  
<http://xmlsoftware.com/editors/>
- XMLSOFTWAREXMLBrowsers  
<http://www.xmlsoftware.com/browsers/>
- XMLSoftwareGuide:XMLand XSLEditors  
<http://wdvl.com/Software/XML/editors.html>
- XML.comEditors  
<http://www.xml.com/pub/pt/3>

## Tutorien

- XSLTTips  
<http://xmleverywhere.com/tips/XSLT.htm>
- Using XSLFormatting Objects  
<http://www.xml.com/pub/a/2001/01/17/XSL-fo/index.html>
- HowtoCreate XSLTDocuments  
[http://www.devguy.com/fp/XML/XSL/XSLT\\_cookbook.htm](http://www.devguy.com/fp/XML/XSL/XSLT_cookbook.htm)
- XSLTheExtensibleStyleLanguage -StylingXMLDocuments  
<http://www.webtechniques.com/archives/1999/01/walsh/>
- ExtensibleStylesheetLanguage( XSL)  
<http://www.oasis-open.org/cover/XSL.html>
- XMLandQueryLanguages  
<http://www.oasis-open.org/cover/xmlQuery.html#software>
- XSLT, XPathand XSLFormatting Objects  
<http://wdvl.com/Authoring/Languages/XSL/>
- XSLTand XPath: XSLTransformationsandXMLPathLanguage  
<http://wdvl.com/Authoring/Languages/XSL/XSLT-XPath.html>
- XMLArticlesandTutorials  
<http://wdvl.com/Authoring/Languages/XML/Tutorials/>
- Zvon XSLTutorial  
<http://www.zvon.org/xxl/XSLTutorial/Books/Book1/index.html>



## Sonstiges

- ZvenoSwishXMLEditor  
<http://www.zveno.com/zm.cgi/in-products/in-swish/>
- XSLdoc  
<http://www.XSLdoc.org/>
- WordHTMLoutput2 FormattingObjects  
<http://www-uk.hpl.hp.com/people/fabgia/wh2fo/wh2fo.html>
- XMLdevelopernewsfromXMLhack:byandfortheXMLcommunity  
<http://xmlhack.com/list.php?cat=2>
- XSLTexpertspetitionagainstscriptelement  
<http://xmlhack.com/read.php?item=1092>
- jd.XSLT  
<http://www.aztecrider.com/XSLT/>
- eXcelon'sStylusStudio  
<http://www.exceloncorp.com/beta/studio.html>
- DocZillaTechnology  
<http://www.doczilla.com/>
- EXSLT  
<http://www.eXSLT.org/>
- EdinburghLanguageTechnologyGroup(LTG)  
<http://www.ltg.ed.ac.uk/>
- DavidMcKelvie'sXMLsoftwarepage  
<http://www.cogsci.ed.ac.uk/~dmck/XML.html>
- ProducingHTMLtableswithXSLT  
<http://www.cogsci.ed.ac.uk/~dmck/XSLT-tutorial.html>
- HenryS.Thompson'sHomePage  
<http://www.cogsci.ed.ac.uk/~ht/>
- ReutersNewsMLShowcase  
<http://newsshowcase.rtrlondon.co.uk/>
- XMLNews.org:XMLandthenewsindustry  
<http://www.xmlnews.org/>
- xmlTree  
<http://www.xmltree.com>
- XMLand XSLSamplesandDemos  
<http://msdn.microsoft.com/xml/demos/default.asp>
- XMLDeveloper'sGuide



- <http://msdn.microsoft.com/xml/xmlguide/default.asp>
- XSLTDeveloper'sGuide  
<http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/xmlsdk/XSLp8tlx.htm>
- XMLTutorial  
<http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/xmlsdk/xmlt7k18.htm>
- WebWorkshop:XML(ExtensibleMarkupLanguage)  
<http://msdn.microsoft.com/workshop/c-frame.htm?/workshop/xml/index.asp>
- XMLandwww.microsoft.com  
<http://msdn.microsoft.com/xml/articles/xml09182000.asp>
- StreamliningYourWebSiteUsingXML  
<http://msdn.microsoft.com/xml/articles/xml01172000.asp>
- "DXML"Redux:BuildingDynamic HTMLMenusfromXML  
<http://msdn.microsoft.com/workshop/author/dHTML/corner052499.asp>
- XSLTransformations:XSLTAlleviatesXMLSchemaIncompatibilityHeadaches  
<http://msdn.microsoft.com/msdnmag/issues/0800/XSLT/XSLT.asp>
- TheXMLFiles: XPath, XSLT,andotherXMLSpecifications  
<http://msdn.microsoft.com/msdnmag/issues/0500/xml/xml0500.asp>
- DuwamishOnlineSQLServerXMLCatalogBrowsing  
<http://msdn.microsoft.com/library/default.asp?URL=/library/techart/d51ctlgbrowse.htm>
- TheXMLCoverPages  
<http://www.oasis-open.org/cover/sgml-xml.html>
- Chapter14oftheXMLBible:XSLTransformations  
<http://www.ibiblio.org/xml/books/bible/updates/14.html>
- XSLINFO  
<http://www.XSLinfo.com/>
- XSLT.com  
<http://www.XSLT.com/>
- HenryS.Thompson'sHomePage  
<http://www.cogsci.ed.ac.uk/~ht/>
- xml.org  
<http://xml.org/>
- DSSSL:DocumentStyleSemanticsandSpecificationLanguage  
<http://www.heise.de/ix/raven/Web/dsssl.html>
- XMLPage  
<http://www.heise.de/ix/raven/Web/xml/>
- XML@iX:Inhaltsverzeichnis



- <http://www.heise.de/ix/raven/Web/xml/cebit99/>
- Xalan-Javaversion2.0.1  
<http://xml.apache.org/xalan-j/index.html>
- Fop  
<http://xml.apache.org/fop/index.html>
- SUN:FREEXSLTCOMPILER  
<http://xml.apache.org/fop/index.html>
- SunXML|DeveloperConnection  
<http://www.sun.com/software/xml/developers/XSLTc/>
- XML.com:WhatisXSLT?  
<http://www.xml.com/pub/a/2000/08/holman/index.html>
- XMLinderPraxis -vonHenningBehmeundStefanMintert  
<http://www.mintert.com/xml/buch/>
- EditingXML  
<http://www.mintert.com/xml/editing/>
- XT  
<http://www.jclark.com/xml/xt.html>
- 4xt.org  
<http://www.4xt.org/>
- XSLTinPerspective  
<http://www.jclark.com/xml/XSLT-talk.htm>
- AnIllustrationoftheXSLTKeyConstruct  
<http://www.cranesoftwrights.com/resources/XSLkeys/index.htm>
- OracleTechnologyNetwork -XMLDemo  
<http://technet.oracle.com/tech/xml/demo/demo1.htm>
- NataliyaAmirova:XML -Einführung  
[http://www.dbis.informatik.uni-frankfurt.de/TEACHING/DB-Seminar/1999\\_WS/seminar\\_ws9900\\_g.html](http://www.dbis.informatik.uni-frankfurt.de/TEACHING/DB-Seminar/1999_WS/seminar_ws9900_g.html)
- MarioJeckle:e XtensibleMarkupLanguage(XML)  
<http://www.jeckle.de/xml>
- KristianKöhntopp:XMLNormensalat  
<http://www.koehntopp.de/kris/artikel/xml-grundlagen/>
- Announcingdsc -2.0:AnonlineDSSSLsyntaxchecker  
<http://www.cogsci.ed.ac.uk/~ht/dsc-blurb.html>
- PublicSGML/XMLSoftware  
<http://www.oasis-open.org/cover/publicSW.html>



## 12 Glossar

- ASP  
<http://www.msdn.microsoft.com/workshop/c-frame.htm#/workshop/server/Default.asp>
- CSS  
<http://www.w3.org/Style/CSS/>
- DOM  
<http://www.w3.org/DOM/>
- HTML  
<http://www.w3.org/MarkUp/>
- MathML  
<http://www.w3.org/Math/>
- PDF  
<http://www.adobe.com/products/acrobat/adobepdf.html>
- PHP  
[www.php.net](http://www.php.net)
- PNG  
<http://www.w3.org/Graphics/PNG/>
- RDF  
<http://www.w3.org/RDF/>
- XHTML  
<http://www.w3.org/MarkUp/>
- XLink  
<http://www.w3.org/XML/Linking>
- XML  
<http://www.w3.org/XML/>
- XPath  
<http://www.w3.org/TR/XPath>
- XPointer  
<http://www.w3.org/XML/Linking>
- XSL  
<http://www.w3.org/Style/XSL/>  
*XSLT*
- <http://www.w3.org/Style/XSL/>



## 13 Index

- A**  
Anwendung8,9,12,15,17,20
- C**  
CSS9,11,12,13,15,16
- D**  
Datei12,17  
Datenstruktur11  
Datentyp11  
Deklaration17  
Design9  
Dokument1,7,8,9,10,12,15,17,18,19  
DSSSL9,11,12,14,15,16  
DTD10
- E**  
Excel7
- F**  
Format10  
Funktion9,15  
Fußzeile9
- G**  
GML8
- H**  
HTML7,8,9,11,12,16,20,21  
HTML-Dokument9
- I**  
Internet7,8,10,1 2  
Internet Explorer8  
ISO8,9
- K**  
Klasse12  
Komponente13,18
- L**  
Liste18
- M**  
Menge10,17,18,19,20,21,22
- Microsoft7,8,9,10,11  
Muster18,19
- N**  
Navigator8  
Netscape8,9
- O**  
Objekt16
- P**  
PowerPoint7  
Programmiersprache21,22  
Protokoll7,1 0
- R**  
Ressource18
- S**  
Server9  
SGML8,9,10,11,14  
Software8,10  
Status16  
Struktur12,19  
Stylesheet10,12,13,19
- T**  
Tabelle11,21  
Template18,19,20  
Templates20
- V**  
visuell9
- W**  
W3C7,9,11,12,15  
Web8,9  
Wert18,20,21,22
- X**  
XHTML7, 10  
XLink7,10,13  
XML7,9,10,11,12,13,14,15,17,18,19,  
20,21,22  
XMLSchema10  
XPath7,10,12,13,14,16,17,19



XSL1,7,10,11,12,13,14,15,16,17,18,  
19,20,21,22  
XSLFO7,13,14,15,17

XSLT7,10,12,13,14,15,16,17,  
22

18,20,21,