Datenformate HTML und XML

Anna Fleischer <anna.fleischer@stud.tu-ilmenau.de>
Matrikelnummer: 35578
Manuel Löffelholz <manuel.loeffelholz@stud.tu-ilmenau.de>
Matrikelnummer: 35884

Erstelldatum 09.03.06

Seminar Digitale Kommunikation
Prof. Dr. Grimm, WS 2005/06
Uni Koblenz / TU Ilmenau

Zusammenfassung

Die Hyper Text Markup Language (HTML) ist als Auszeichnungssprache in der Welt des World Wide Web weitgehend verbreitet und bei der Programmierung von überwiegend statischen Seiten sehr gebräuchlich.

Die erweiterbare Auszeichnungssprache (XML) trennt die Darstellungsweise und den darzustellenden Inhalt voneinander und eignet sich somit für die Speicherung von dynamischen, sich schnell ändernden Inhalten wie beispielsweise „Wetterdaten“. Zudem stellt XML ein Datenformat dar, dass über die reine Präsentation im World Wide Web hinausgeht.

Diese Arbeit bietet einen grundlegenden Überblick über die Datenformate HTML und XML.

Nach einer historischen Herleitung der beiden Formate im Kapitel 2 und einem grundlegenden Überblick über Auszeichnungssprachen im Kapitel 3 werden in den Kapiteln 4 und 5 die beiden Datenformate HTML und XML behandelt.

1. Einleitung

Um die Auszeichnungssprachen HTML und XML näher zu untersuchen, sollte man sich vorerst mit deren Überbegriff „Datenformate im Internet“[1] eingehender auseinandersetzen.

Zur vollständigen Beschreibung digitaler Daten sind vier Aspekte notwendig. Während die *logische Struktur* die innere Aufteilung in Teildatensätzen und deren Verhältnis zueinander beschreibt, sollen die *Rohdaten* als digitale Symbole zur Beschreibung der einfachsten logischen Einheiten keine logischen Untereinheiten mehr besitzen. Die *Präsentation* von Daten hängt von dem darzustellenden Medium ab, sowie von den Subjekten, welche die Darstellung rezipieren sollen. Wie auch die *Präsentation* wird die *digitale Kodierung* strikt von der abstrakten Syntaxnotation getrennt gehalten. Der letzte Aspekt kommt erst zur Realisierung der Daten in den verschiedenen Medien zum Tragen.

Während die *logische Struktur* und die Zuordnung von *Rohdatentypen* zu den Elementen der lokalen Struktur gemeinsam in einer abstrakten Syntaxnotation beschrieben werden, sollten die *Präsentationsregeln*, ebenso wie die *Kodierung*, in einem Beschreibungsteil getrennt von Logik und Rohdaten sein.[2]

In dieser Trennung der verschiedenen notwendigen Aspekte liegt auch der grundlegendste Unterschied von HTML gegenüber XML, der im Weiteren erläutert wird. XML folgt der Trennung von Layout und Inhalt – beide Bestandteile werden in zwei verschiedenen Dateien gespeichert. HTML jedoch ist das Beispiel in der Kommunikationstechnik, bei dem die Präsentation im Gegensatz zu Logik und Rohdaten auf derselben Stufe stehen. Zwar ist es durch Cascading Style Sheets möglich, HTML - Tags Formatierungen zuzuweisen, jedoch differenziert erst XML sauber zwischen Inhalt und Layout.

2. Historische Entwicklung der Formate HTML und XML

Die Ursprünge der Auszeichnungssprache HTML (**H**yper**t**ext **M**arkup **L**anguage), die im Jahre 1990 von Tim Berners-Lee entwickelt wurde, gehen bereits zurück in die siebziger Jahre, als an der Universität Stanford das Programm TeX von Professor Donald Knuth geschrieben, und von Leslie Lamport zu LaTeX ergänzt wurde. Dieses Programm unterscheidet sich von „normaler“ Textverarbeitung, indem es Text und Stilvorlagen getrennt speichert und somit mehr Flexibilität verleiht. Zur optischen Veränderung eines Textes müssen also nur noch die Stile ausgewechselt werden. Aus LaTeX entwickelte sich die **S**tandard **G**eneralized **M**arkup **L**anguage (SGML)[3].

Berners-Lee’s Entwicklung von HTML erweiterte die Darstellungsmöglichkeiten, da diese Sprache auch Formeln, Zeichnungen und Grafiken in Texte integrieren und übermitteln kann. Zur Übertragung der HTML-Daten musste ein neues Protokoll eingeführt werden, das **H**yper**t**ext **T**ransfer **P**rotocol (http). Angezeigt werden diese Daten durch einen Browser. 1993 entstand als einer der ersten Webbrowser Mosaic, später gefolgt von Netscape und dem Internet Explorer, die gratis erhältlich sind.

---

[1] Grimm (2005) S.235

[2] Vgl. Grimm (2005) S.238

[3] Vgl. Harms/Koch/Kürten (2000) S. 19-20

Im Prinzip ist HTML nichts anderes als ein spezieller Dokumenttyp von SGML und wird mittlerweile als „die Programmiersprache des World Wide Webs“[4] bezeichnet. Dies ist jedoch nicht ganz exakt, da HTML im Grunde genommen „nur“ eine Seitenbeschreibungs- und Auszeichnungssprache ist, die definiert, wie das Erscheinungsbild eines Textdokumentes aussieht. Der Begriff der Programmiersprache scheint also etwas übertrieben zu sein.

Das World Wide Web entwickelte sich mit den Jahren immer weiter, so dass die Möglichkeiten, die HTML zur Gestaltung von Webseiten bietet, ausgereizt wurden. Es zeichnete sich zunehmend ab, dass HTML den Anforderungen professioneller Webseiten keineswegs mehr gerecht werden konnte. Viele Entwickler von Web-Programmen fühlten sich in ihren Möglichkeiten mehr als nur eingeschränkt. Ihrer Meinung nach fehlten dem Web regelrecht die Möglichkeiten von SGML[3]. Dies führte Ende der 90er Jahre dazu, dass neben Erweiterungen für HTML auch an der Entwicklung einer neuen vereinfachten Variante von SGML gearbeitet wurde. Das Ergebnis dieser Entwicklung ist XML. Während die grundlegendsten Spezifikationen von SGML erhalten blieben, wurden die weniger gebräuchlichen Spezifikationen nicht in XML übertragen. Unter dem Aspekt der Nutzung von XML zur Gestaltung von Webseiten sind insbesondere drei Fakten von besonderer Bedeutung. Zum einen sollte XML einfach für das Internet nutzbar sein und zum anderen sollte XML eine Vielzahl von Anwendungen unterstützen sowie kompatibel zu SGML sein.

3. Einführung in Auszeichnungssprachen

Auszeichnungssprachen (engl. *Markup Language*) dienen zur Beschreibung oder zur Darstellung von Daten.

Es existieren zwei Gruppen von Auszeichnungssprachen. Die *Procedural Markup Languages* weisen Ausgabegeräten (wie z.B.: den Bildschirm oder den Drucker) an, wie das Dokument dargestellt werden soll. Die *Descriptive Markup Languages* beschreiben den Inhalt oder geben Auskunft darüber wie die Formatierung der Daten zu erfolgen hat. Beispiele für *Procedural Markup Languages* sind PDF und PostScript. Beispiele für *Descriptive Markup Languages* sind XML und HTML[5]

3.1. Logische Auszeichnung

Eine logische Auszeichnung ist eine inhaltliche Beschreibung des Textes. Es kann so festgelegt werden, ob es sich bei dem gekennzeichneten Begriff um ein Zitat, einen Namen oder eine Überschrift handelt. In HTML sind einige logische Auszeichnungen spezifiziert. Beispielsweise <author>Manuel Loeffelholz</author>

Diese Auszeichnung hilft Suchmaschinen bei der Auswertung des Textes.

[4] Vgl. Kobert (1999)

[5] Wikipedia (2005)

3.2. Physische Auszeichnung

Physische Auszeichnungen verfolgen das Ziel, Möglichkeiten zur visuellen Textdarstellung zu geben. Eine gebräuchliche physische Auszeichnung in HTML ist zum Beispiel:

Schriftart XXX

3.3. Semantische Auszeichnung

Semantische Auszeichnungen „beschreiben weder Formatanweisungen noch die logische Struktur, sondern geben Rückschlüsse über den Inhalt des zwischen den Tags stehenden Textes."[6]

In XML werden fast ausschließlich semantische Tags eingesetzt.

Eine semantische Auszeichnung sieht beispielsweise so aus:

<Preis>54</Preis>

Solche semantischen Tags bieten die Möglichkeit, die entsprechenden Felder auszuwerten oder beispielsweise nach einem bestimmten Preis zu suchen.

Ganz besonders bei der Verknüpfung von XML-Dokumenten mit Datenbanken spielen diese semantischen Tags eine bedeutende Rolle.

3.4. Bedingungen an eine zukunftsorientierte Auszeichnungssprache

Flexible Datenstruktur

Die Sprache darf nicht auf einer festen Datenstruktur basieren, sondern muss ständig erweiterbar sein. Sie muss sich den wechselnden Gegebenheiten und Anforderungen der Nutzer anpassen. Sie muss sich leicht in möglichst unterschiedliche Formate konvertieren lassen. Beispielsweise sollte eine Mitarbeiterliste leicht in ein optisch ansprechendes, druckbares Format wie PDF konvertiert werden können. Die gleichen Daten müssen sich auch in eine Adressdatenbank integrieren lassen.

Plattformunabhängigkeit

Die Daten müssen flexibel zwischen den verschiedensten Plattformen wie beispielsweise Windows, Macintosh und Unix austauschbar sein.

Textorientierung

Schon die plattformübergreifende Struktur legt ein Textformat nahe, und lässt eine binäre Datenstruktur in den Hintergrund treten. Die Textstruktur bedingt auch, dass die Kennzeichnung der einzelnen Daten durch Textmarken erzeugt wird.[7]

[6] Vgl. Pott & Wielage (1999) S. 16

[7] Vgl. Pott & Wielage (1999) S. 13, 14

4. HTML

HTML wird gern als die „Lingua Franca“[8] des World Wide Web bezeichnet: jeder kennt, spricht und braucht sie und außerdem ist sie vergleichsweise schnell zu erlernen. HTML ist eine Sprache zur Strukturierung von Texten mit der Möglichkeit, Graphiken und multimediale Inhalte in Form einer Referenz einzubinden und in den Text zu integrieren.[9]

HTML wird als Klartextformat bezeichnet, weil es möglich ist, HTML-Dateien in einem Texteditor zu erstellen und zu bearbeiten. Zur simpleren Handhabung gibt es zahlreiche „WYSIWYG-Programme“[10] zum Schreiben von HTML-Dateien, jedoch sind diese nicht notwendig. Somit ist eine Bindung an ein kommerzielles Programm nicht gegeben.[11]

Die Definition von Verweisen („Hyperlinks“) ist eine der wichtigsten Eigenschaften von HTML. Mit Verweisen kann der Nutzer direkt zu einer anderen Stelle im Dokument, oder auch zu anderen Adressen im World Wide Web geführt werden. HTML dient also der Vernetzung von Inhalten - auch verschiedener Anbieter.[12]

4.1. Einführung in die HTML-Programmierung

Um eine Website mit der Auszeichnungssprache HTML zu erstellen, ist zu beachten, dass diese zum einen aus den Inhalten besteht, die später auf der Seite zu sehen sein sollen, zum anderen aus den HTML-Befehlen, die als Tags bezeichnet werden.[13] Das bedeutet also, dass Inhalt, Struktur und Layout gemeinsam in einem Dokument festgelegt werden. Sollten also später Veränderungen im Layout vollzogen werden, ist der Aufwand, diese durchzuführen, relativ hoch.

Um das Layout zu bestimmen, Graphiken einzubinden, Tabellen darzustellen usw., benötigt man die richtigen Befehle. Diese Tags werden in spitze Klammern (< >) eingefasst. Zum Eingrenzen der Textpassagen, die einem bestimmten Tag unterliegen, gibt es ein Anfangs- und ein Ende-Tag. So leitet z.B. das Tag < EM > eine Hervorhebung ein, und beendet diese mit dem Tag < / EM >. Man wiederholt also zum Abschluss der Textpassage dasselbe Tag mit einem vorangestellten Schrägstrich (/).

```
< EM >
Hier steht der hervorgehobene Text.
< / EM >
Ab hier wird wieder normal geschrieben.
```

Jedoch gibt es auch Befehle, die kein Ende-Tag besitzen. Der Zeilenumbruch bezieht sich beispielsweise nicht auf einen längeren Abschnitt, sondern eben nur genau auf eine bestimmte Stelle. Ein Beispiel für ein solches Tag ist < BR>.

---

[8] SELFHTML e.V. (2005a)

[9] Vgl. SELFHTML e.V. (2005a)

[10] WYSIWYG ist die Abkürzung für „What You Ses Is What You Get“ und steht für Tools, die HTML- Quellcode automatisch generieren und ihn gleichzeitig wie ein Browser interpretieren, sodass das Erstellen von HTML-Seiten auch für Personen ohne Kenntnisse über die von HTML verwendeten Tags möglich ist.

[11] Vgl. SELFHTML e.V. (2005b)

[12] Vgl. SELFHTML e.V. (2005c)

[13] Vgl. Harms/Koch/Kürten (2000) S. 20

HTML-Befehle lassen sich nicht nur hintereinander verwenden, sondern können auch ineinander verschachtelt werden. So kann man etwa in einer Überschrift (Header; eine bestimmte Überschrift: < H1 >) einige Worte kursiv (< I >) gestalten.

```
< H1 >
Diese Überschrift < I > wird an dieser Stelle< / I > kursiv geschrieben
< / H1 >
```

Zur übersichtlichen Darstellung des Quelltextes ist es vorteilhaft, die unterschiedlichen Befehle in einem Satz in eine neue Zeile einzurücken. Dies ändert die spätere Darstellung im Browser nicht, sondern erleichtert nur die ggf. nachträgliche Bearbeitung.

Des Weiteren ist es möglich, den Tags Attribute zuzuweisen. Im einleitenden Tag der Überschrift kann also auch noch angegeben werden, wie die Ausrichtung der Überschrift erfolgen soll (linksbündig, zentriert oder rechtsbündig).

```
< H1 ALIGN=LEFT >
Diese Überschrift erscheint linksbündig.
< / H1 >
```

Es ist im Übrigen dem Autor überlassen, ob er die Tags groß oder klein schreibt.

4.2. Aufbau von HTML-Seiten

Die Grundstruktur ist bei jeder HTML-Seite gleich.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
        <head>
        <title>Seite mit Textblock</title>
        </head>
        <body>
        Im Body steht also alles, was man später auf der Website als norma-
len Text sieht. Im Quelltext kann man jedoch einen Zeilenumbruch
        durchführen           und Leerzeichen einführen, die so nicht im
Browser dargestellt werden.
        </body>
</html>
```

Speichert man diesen Text im Texteditor unter beispiel.html ab, kann man die Datei mit dem Webbrowser aufrufen und erhält die entsprechende Browserinterpretation der HTML-Datei. (vgl. Abbildung 3 auf Seite 26.)

Im Folgenden werden die einzelnen Elemente der Grundstruktur erläutert.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

Dieser Teil (DOCTYPE HTML PUBLIC) besagt, dass der Text auf der öffentlich zugänglichen HTML-DocTypeDefinition (DTD) beruht, festgelegt vom W3-Consortium. Des Weiteren beschreibt diese Zeile die HTML-Version (HTML 4.01 Transitional) und die Sprache, in der die Tag-Befehle definiert wurden (EN = Englisch).[14]

```
<html>
```

Dieser Befehl zeichnet den gesamten Text als HTML-Seite aus, die aus einem Kopfteil (Informationen über die Seite, also Metadaten; <head>) und einem Inhaltsteil (dargestellte Informationen; <body>) bestehen.[15]

Der Titel (<title>) im Kopfteil sollte prägnant die Inhalte der Seite wiedergeben, da er zum einen in der Fensterzeile des Browsers zu sehen ist und zum anderen von Suchmaschinen dazu verwendet wird, Informationen zu finden.

Ein genereller Hinweis über das Arbeiten mit HTML in der deutschen Sprache ist noch zu erwähnen. Die Umlaute Ä, Ö und Ü sowie andere Sonderzeichen werden nicht in allen Sprachversionen unterstützt. Wird nicht die entsprechende HTML-Codierung verwendet, sieht die Bildschirmausgabe im Webbrowser oftmals seltsam aus. Die Umlaute sollten also durch bestimmte Schlüsselcodes ersetzt werden, z.B. „ä“ durch „ä“ (= a + Umlaut).

4.3. Funktionen von HTML

Mit Hilfe von Tags wird also der Text einer Webseite formatiert, sodass er wie jedes beliebige Dokument eines Textverarbeitungsprogramms aussieht. Zur Erläuterung der Funktionsweise von HTML genügen einige Beispiele, auf die wir uns in den folgenden Abschnitten beziehen. Wir werden hier auf die Integration von Graphiken und Tabellen eingehen.

4.3.1. Graphiken

In der Regel dienen Graphiken zur Erklärung und Auflockerung der Textinformation, jedoch sollte man sich bewusst sein, dass der Einsatz von Graphiken auch eine höhere Datenmenge bedeutet.[16] Eine größere Datenmenge äußert sich für den Leser in einer längeren Wartezeit bis das Bild geladen ist. Es ist also sinnvoll sich vor der Integration von Bilddaten über Dateiformate und Komprimierungsarten Gedanken zu machen, sodass die Bilddaten bei geringer Dateigröße noch eine annehmbare Qualität zeigen.

Für kleine Bilder, Bitmaps, Steuergraphiken und vektorbasiertes Bildmaterial ist das GIF-Format hilfreich, da es viel Platz spart und immer noch zufrieden stellende Qualität liefert. Für Bilder ab 200 mal 200 Pixel mit mehr als 256 Farben, bieten sich Bildformate an, die eine starke Kompression unterstützen wie z.B.: JPEG oder PNG, da sonst mit starken Qualitätseinbußen und langen Ladezeiten gerechnet werden muss.[17]

Um eine Graphik in die HTML-Seite einzubinden, wird ein Verweis (auch Referenzierung genannt) auf die Bilddatei geschrieben.

---

[14] Vgl. Harms/Koch/Kürten (2000) S.33

[15] Vgl. Tolksdorf (2000) S.15

[16] Harms/Koch/Kürten (2000) S.67

[17] Vgl. Harms/Koch/Kürten (2000) S.75

Als Beispiel:
```
<IMG SRC=“Photo.jpg“>
```

Das Tag IMG steht für Image (Bild), das Attribut SRC für Source (Quelle). „Photo.jpg“ bezeichnet die Bilddatei, die eingebunden werden soll. Der Dateiname muss immer in Anführungszeichen stehen. Bei diesem Beispiel muss sich die Bilddatei im selben Ordner befinden wie die HTML-Datei.[18]

Ist die Bilddatei an einem anderen Ort abgelegt, müssen z.B.: Server und/oder Verzeichnis zu der Bild-Datei angegeben werden:
```
<IMG SRC=“http://www.spi.tu-ilmenau.de/photos/aenneken/aenneken-
060110131935.jpg“>
```

Um eine größere Anzahl von Bildern darzustellen, bietet sich eine Bildergalerie an. Hier erstellt man Thumbnail-Bilder als Übersicht von den Originalen, sodass die Ladezeit akzeptabel ist. Die kleinen Bilder, auch Thumbnails (eng. Miniaturbild, Daumennagel) genannt, leiten den Nutzer über einen Link zum Originalbild weiter.
```
<a href="Today.jpg"><img src="Today_klein.jpg"></a>
```

Dies ist nur ein kurzer Einblick für die simpelste Art, Bilder in eine HTML-Seite einzubinden. Zur Intensivierung des Themas sollte die verwendete Literatur genutzt werden.

4.3.2. Tabellen

Tabellen werden oft verwendet, um Informationen gegliedert darzustellen. Der einleitende Befehl für Tabelle ist <table>. Um die Tabelle von einer Liste zu unterscheiden, werden Gitternetzlinien eingesetzt. Diese bezeichnet man als „border“ und müssen einem Wert größer null zugeordnet werden. Dann folgen die einzelnen Tabellenzeilen <tr> (tr = table row = Tabellenzeile) und die Spalten der Reihe. Wie gewöhnlich muss der Tag mit </tr> wieder geschlossen werden. Die Kopfzelle ist definiert mit <th> und die Datenzelle mit <td>.[19] Im Anhang ist ein gekürztes Beispiel. In Tabellenform ist das Rezept eines Rotweinkuchens auf Seite 27 zu finden. Hier ist auch direkt das Breitenverhältnis der Spalten „Menge“ und „Zutaten“ zueinander mit dem Befehl <colgroup> festgelegt. Die Spalte „Menge“ ist mit <col width=„1*“> nur halb so breit wie die Spalte „Zutaten“ mit dem Breitenverhältnis „2“.[20]

4.4. Dynamic HTML

Der Aufbau einer gewöhnlichen HTML-Seite ist statisch und verhält sich ähnlich einer gedruckten Zeitschrift. Man kann zwar die Seiten durchblättern, jedoch ändert sich nicht der Inhalt.

DHTML (Dynamic Hypertext Markup Language) ist keine klassische HTML-Erweiterung in Gestalt neuer HTML-Elemente, sondern vielmehr ein Sammelbegriff für verschiedene Lösungen, die es dem Autor einer Webseite ermöglichen, Elemente der Webseite während der Anzeige dy-

[18] Vgl. Harms/Koch/Kürten (2000) S.83

[19] SELFHTML e.V. (2005d)

[20] Vgl. Harms/Koch/Kürten (2000) S. 134

namisch zu ändern. Einzelne angezeigte Inhalte können ausgetauscht werden, ohne die Seite vollständig neu zu laden. Eine Webseite kann mit DHTML wie eine Anwendung arbeiten, die einmal im Arbeitsspeicher geladen wird und dann durch Interaktion mit dem Nutzer das Geschehen am Bildschirm lenkt.[21] DHTML eröffnet eine Vielzahl von Möglichkeiten. Als Voraussetzung für DHTML sollte man fortgeschrittene Programmierkenntnisse in JavaScript besitzen.

Großer Nachteil des Dynamischen HTML ist die Inkompatibilität der beiden bekanntesten Browser Netscape und Internet Explorer in der 4.0-Version. Die Realisierungen unterscheiden sich sehr und sollen beide Browser die Webseite lesen können, muss getrennt kodiert werden. Dank des W3-Consortiums zeichnet sich das „Document Object Model“ (DOM) als einheitlicher Standard ab.

4.5. XHTML

XHTML steht für e**X**tensible **H**yper**T**ext **M**arkup **L**anguage, eine notwendig gewordene Auszeichnungssprache um HTML mit Hilfe von XML zu definieren.

Wurde HTML bisher zum Darstellen und Gestalten von Webdokumenten eingesetzt, so ist jetzt eine Auszeichnungssprache gefragt, mit der Inhalte strukturiert und aufgenommen werden können. Die Darstellung der Inhalte soll in Zukunft ausschließlich mit anderen Sprachen erfolgen.[22] Um diesen Anforderungen gerecht zu werden, wurde im Jahre 2000 XHTML 1.0 als Empfehlung des W3-Consortiums verabschiedet.

XHTML verfügt über die gleichen Elemente, Attribute und Verschachtelungsregeln wie HTML, jedoch sollte man nach Herold (2002) folgende 14 Unterschiede beachten:

1. Tag- und Attributnamen werden in XHTML klein geschrieben.
2. Die Tags <head>, <title> und <body> müssen in XHTML angegeben sein.
3. Endtags sind in XHTML immer anzugeben.
4. Allen Attributen muss in XHTML ein Wert zugewiesen werden.
5. Alle Attribute müssen in XHTML mit “…” umgeben sein.
6. Weitere MIME-Typen für XHTML-Dokumente
7. Neue Dateiendungen für XHTML-Dokumente
8. XML-Deklaration am Anfang einer XHTML-Datei
9. Unterschiedliche Dokumenttyp-Angaben
10. Namensraum-Angabe in XHTML
11. Hyperlinks nur auf id=-Namen in XHTML
12. Attribut xml:lang= statt lang= in XHTML
13. Skript- und Style-Angaben müssen in XHTML im CDATA-Bereich stehen.
14. Neue Verschachtelungsregeln in XHTML.[23]

[21] Vgl. SELFHTML e.V. (2005e)

[22] Hess & Karl (2000) S.15

[23] Herold (2002) S. 486-487

5. XML

Dieses Kapitel bietet Ihnen einen grundlegenden Überblick über die „Extensible Markup Language“. Nach der Definition und der Identifizierung als Metasprache wird auf die Grammatik und das Regelwerk eingegangen. Anschließend wird die „Extensible Stylesheet-Language“ mit ihren zwei Hauptaufgaben vorgestellt und anhand von einigen Beispielen die Bedeutung von XML für das World Wide Web verdeutlicht.

5.1. Begriffsdefinition

Nach dem World Wide Web Consortium beschreibt die Extensible Markup Language (XML), „eine Klasse von Datenobjekten, genannt XML-Dokumente, und beschreibt teilweise das Verhalten von Computer-Programmen, die solche Dokumente verarbeiten. XML ist ein Anwendungsprofil (application profile) oder eine eingeschränkte Form von SGML, der Standard Generalized Markup Language [ISO 8879]. Durch ihre Konstruktion sind XML-Dokumente konforme SGML-Dokumente.“[24]

XML stellt eine Teilmenge von der *Structured Generalized Markup Language* SGML dar und wurde entworfen, um eine „einfache Implementierung und Zusammenarbeit sowohl mit SGML als auch mit HTML zu gewährleisten.“[25]

5.2. Unterschiede zu HTML

XML bietet zu HTML vielerlei Unterscheidungspunkte, auch was die Möglichkeiten bei der Überführung in Präsentationsformate angeht. Der grundlegendste Unterscheidungspunkt beruht dabei darauf, dass XML eine Metasprache ist und dadurch höhere Flexibilität bei der Lösung von Problemen bietet.

5.2.1. Metasprache bietet höhere Flexibilität

Das Problem bei HTML ist, dass dem Nutzer eine bestimmte Anzahl an vorgegebenen Tags zur Verfügung steht, egal, ob er diese nun benötigt oder nicht. So gibt es beispielsweise Tags zur Darstellung chemischer Formeln oder auch Musiknoten, die eigentlich nur für eine Minderheit aller Webseiten benötigt wird, aber so allgemein gehalten sind, dass sie von jedem verwendet werden können.[26]

Durch die Verwendung von XML ist die Kreativität nicht mehr an eine limitierte Anzahl von Tags gebunden. Jeder kann seine eigene Auszeichnungssprache entwickeln, die exakt auf die spezifische Problemstellung und Bedürfnisse abgestimmt ist.[27] Wenn ein Befehl fehlen sollte, dann

[24] W3C_1 (2005)

[25] W3C (2005)

[26] vgl. Dobnig, Mario (2002)

[27] vgl. Dobnig, Mario (2002)

entwirft der Nutzer sich ihn einfach selbst. Unbrauchbare Tags, die keine häufige Anwendung finden, fallen hingegen weg.

Des Weiteren ermöglicht XML die Definition von Tags jeder nur denkbaren Art. Während bei HTML Tags „nur" das Erscheinungsbild eines Textdokumentes definieren, erlaubt XML „Tags, die Daten oder das Verhältnis von Daten zueinander beschreiben".[28] Dies ist darauf zurückzuführen, dass XML-Tags weniger die Formatierung als vielmehr den Inhalt beschreiben und somit also der Strukturierung und Abgrenzung von Daten dienen. Sie sind daher auch als Metadaten charakterisierbar, was die Grundlage für den zweiten Unterscheidungspunkt zwischen XML und HTML liefert.

5.2.2. Metadaten erhöhen Informationsgehalt

Durch Metadaten wird der Informationsgehalt bei XML erhöht.

Bisher war die Suche nach Informationen im World Wide Web recht aufwendig. Im Prinzip wurde jede Seite nach jedem einzelnen Suchbegriff durchforstet. Die Suche nach Wörtern, die in einer bestimmten Beziehung zueinander stehen, war damit nicht möglich.

XML scheint in dieser Beziehung einen großen Schritt für die Zukunft zu machen. Da die Tags den Inhalt der Daten beschreiben, erklärt sich XML wie von selbst. Somit werden die Informationen nicht nur für Außenstehende leichter verständlich, sondern sind auch wesentlich gezielter von Suchmaschinen auswertbar. Vielleicht wird dadurch endlich erreicht, dass Suchmaschinen tatsächlich auch das Ergebnis liefern, wonach gesucht wurde.[29]

5.2.3. Trennung von Inhalt, Struktur und Layout

Der dritte Unterscheidungspunkt zwischen XML und HTML beruht auf der Trennung von Inhalt, Struktur und Layout (vgl. Abbildung 6, S. 27).

Bei HTML ist die Textformatierung mit den Daten vermischt. Dies stellt sich spätestens dann als problematisch heraus, wenn nun eine Seite geändert werden soll. Egal wie groß oder klein die Veränderung auch sein mag, müssen höchstwahrscheinlich alle Seiten durchgearbeitet werden, was natürlich mit viel Aufwand und Zeit verbunden ist.

XML vereinfacht dieses Verfahren durch die Trennung von Daten und Darstellung in unterschiedlichen Dokumenten. Während das XML-Dokument an sich vergleichbar mit Datenbankinhalten ist, wo „nur" die Daten strukturiert abgelegt sind, ist ein zusätzliches „Stylesheet" notwendig, um das Layout des Dokuments festzulegen. Somit ist einerseits gewährleistet, dass eine Datenbasis für unterschiedliche Anwendungen genutzt werden kann (beispielsweise können Wetterdaten sowohl als Grafik als auch als Tabelle ausgegeben werden)[30] und andererseits sind Änderungen im Dokument mit weniger Aufwand verbunden. Je nachdem, ob nun die Daten oder das Layout verändert werden soll, kann auf das XML-Dokument oder das Stylesheet direkt zugegriffen werden.

[28] vgl. Dobnig, Mario (2002)

[29] vgl. Dobnig, Mario (2002)

[30] vgl. Wikipedia (2005)

5.3. Allgemeine Regeln für XML-Dokumente

XML-Dokumente unterliegen in ihrem Aufbau einigen Regeln, die nun im Folgenden näher erläutert werden.

5.3.1. Aufbau eines XML-Dokuments

Ein XML-Dokument beginnt in der Regel mit dem Prolog. Obgleich er nicht zwingend vorgeschrieben ist, ist er aus Gründen der Identifikation als XML-Dokument unbedingt zu empfehlen.

In der ersten Zeile des Prologs findet man meist die XML- Deklaration. Mit ihr wird die Übereinstimmung des Dokuments mit der gültigen Spezifikation von XML deklariert.

Im folgenden Beispiel sind noch das Attribut der Zeichensatzcodierung und die Markupdeklaration angegeben. Attribute werden verwendet, um bestimmte Eigenschaften oder Besonderheiten eines Elements zu beschreiben.[31]

<?xml version=“1.0“ encoding=“UTF-16“ standalone=“no“?>

„encoding=“UTF-16““ gibt an, dass das Dokument die Zeichensatzkodierung UTF-16 verwendet. Mit "standalone=“no““ wird angegeben, dass externe Markupdeklarationen wie etwa eine externe DTD vorhanden sind.

Weiterhin werden im Prolog Verknüpfungen zu einem Stylesheet oder einer DTD angegeben wie beispielsweise:

<?xml-stylesheet type="text/xsl" href="choose.xsl"?>

<!DOCTYPE adressen SYSTEM “adresse.dtd”>

Nach dem Prolog werden die eigentlichen XML- Daten in Baumstruktur geschrieben. Die Struktur besteht aus Elementen und Attributen.

Jedes Element wird wie im nachfolgenden Beispiel beschrieben mit einem Start-Tag und einem End-Tag umschlossen. Dabei liefern die Tags Metainformationen über den jeweiligen Inhalt des Tags.

<name> Albert Einstein</name>

5.3.2. Document Type Definition (DTD)

Die DTD enthält die Grammatik eines XML-Dokuments. Sie steht im Prolog einer XML-Datei oder in einer externen Datei mit der Endung „.dtd“ und wird aus dem Prolog heraus referenziert. Die DTD gibt vor, welche Elemente im Dokument erscheinen dürfen und bestimmt die Anzahl der Elemente. Sie gibt weiterhin vor, welche Unterelemente ein Element haben darf. Eine DTD ist jedoch nicht zwingend erforderlich.[32] Es wurde mit XML-Schemata eine komplexere und XML-konforme Sprache entwickelt, die mehr Möglichkeiten als die Document Type Definition bietet.

[31] vgl. Vonhoegen, Helmut (2005)

[32] vgl. Vonhoegen, Helmut (2005)

5.3.3. XML-Schema

XML-Schemata sind Produkte der Weiterentwicklung der Document Type Definition. Seit 2001 sind sie als Empfehlung des W3C zum Definieren von XML-Dokumentstrukturen beschrieben. Die Struktur eines XML-Schema wird anders als bei den klassischen DTD's durch die Form eines XML-Dokuments bestimmt.

XML-Schema ist eine komplexe Sprache zur Beschreibung eines XML-Typsystems. Dieses XML-Typsystem umfasst die Spezifikation neuer XML-Elemente, deren Attribute, sowie deren Kindelemente und den Datentypen. XML-Schema Definitionen unterstützen viele vordefinierte Datentypen z.B.: string, integer, decimal, time, date etc. . Des Weiteren werden Schema Definitionen an Namensräume gebunden, was den Vorteil mit sich bringt, dass XML-Dokumente mehrere Schemata voneinander unterscheiden können. Im Gegensatz zu DTDs kann bei Verwendung von XML-Schema zwischen dem Namen des XML-Typs und dem in der Instanz verwendeten XML-Tagnamen unterschieden werden.[33]

5.3.4. Wohlgeformtheit eines XML Dokuments

Ein XML-Dokument ist wohlgeformt, wenn es eine Reihe von syntaktischen Regeln einhält. Im Folgenden sind einige Regeln für Tags und Attribute aufgeführt.

Regeln für Tags und Attribute

Tags müssen mit einem:
- Buchstaben
- Unterstrich oder
- Doppelpunkt

beginnen.

Die Zeichenfolge „xml" darf in keiner der möglichen Schreibweisen am Anfang eines Tags stehen. Die Groß- bzw. Kleinschreibung wird bei Tags unterschieden.

Somit ist das Element <Name> Albert Einstein</name> nicht zulässig.

Jedes öffnende Tag muss wieder geschlossen werden.[34] Es muss ein Wurzelelement existieren, das alle nachfolgenden Tags umschließt.

Attributwerte müssen in Anführungszeichen stehen.

Beispiel: <haus nr="12" farbe="gelb">…</haus>

5.3.5. Gültigkeit eines XML Dokuments

Ein XML-Dokument ist gültig, wenn es sämtliche Regeln der Wohlgeformtheit erfüllt und ein Format einhält, dass in einer Grammatik wie etwa einer DTD oder einem XML-Schema definiert

[33] Vgl. Wikipedia (2005)

[34] vgl. Vonhoegen, Helmut (2005)

ist. Somit ergibt sich, dass ein wohlgeformtes XML-Dokument nicht unbedingt gültig sein muss. Für ein gültiges Dokument ist die Wohlgeformtheit jedoch zwingend notwendig.

Die Überprüfung der Gültigkeit oder Wohlgeformtheit obliegt den XML-Prozessoren. Dies ist ein Programm oder Programmteil welches XML-Daten ausliest, interpretiert und gegebenenfalls die Wohlgeformtheits- oder Gültigkeitsprüfung vornimmt.[35]

5.4. Überführung von XML-Dokumenten in Präsentationsformate

Um XML- Daten in einer optisch ansprechenden Form zu präsentieren ist ein Transformationsprozess notwendig, der die XML-Daten in ein Ausgabeformat bringt. Um diesen Transformationsprozess durchzuführen, ist eine zusätzliche Beschreibungssprache notwendig, die beschreibt, wie die einzelnen XML-Tags dargestellt werden sollen. Im Folgenden wird die Überführung von XML-Dokumenten in ein Präsentationsformat anhand von zwei „Style Sheet"-Sprachen demonstriert.

5.4.1. Cascading Stylesheets CSS

Wie schon im Kapitel 3.3 angesprochen wurde, handelt es sich bei XML um eine semantische „Markup Language". Diese liefert keine Informationen über die Darstellung der einzelnen Elemente, sondern lediglich Angaben zu ihrer logischen Bedeutung und wird deshalb auch als „logische Auszeichnungssprache bezeichnet"

Um die im XML-Dokument enthaltenden Daten in einem Browser darstellen zu können, ist eine Transformation nötig. Wie die Abbildung 1 zeigt, ist solch eine Transformation durch Style Sheets zu generieren.

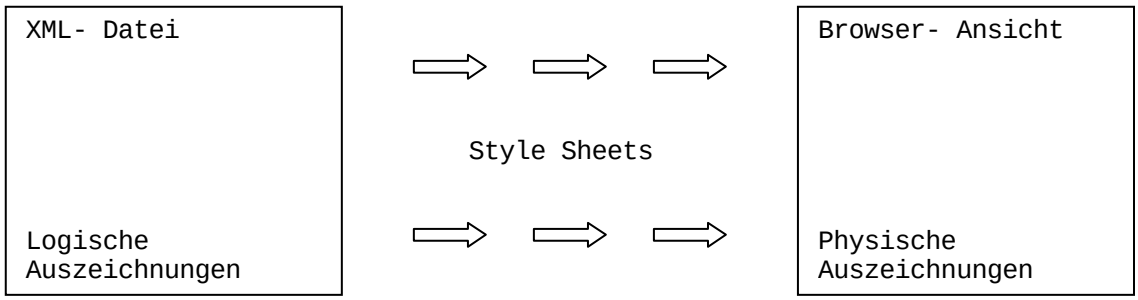


Abbildung 1

Um eine solche Browser-Ausgabe zu erreichen, hat man mehrere Möglichkeiten. Die erste Variante ist die Ausgabe mittels Cascading-Stylesheets. Cascading Stylesheets enthalten einen Konstruktionsplan für die Darstellung der Quelldatei. Sie erlauben es, den im XML-Dokument defi-

[35] vgl. W3C (2005)

nierten Tags Gestaltungsinformationen zuzuordnen. CSS dienen folglich als Vorlage zur Umwandlung der logischen Auszeichnungen in die physischen Auszeichnungen.

Die zweite, von uns betrachtete Möglichkeit ist die Ausgabe mittels XSL-Datei, mit der sich das Kapitel 5.4.2 beschäftigt. Eine andere Darstellungsmöglichkeit, auf die hier aber nicht weiter eingegangen wird, ist die Formatierung mittels XHTML-Tags. Hier werden direkt in der XML-Datei XHTML-Tags eingearbeitet, was einen logischen Bruch zu dem eigentlichen Ziel von XML, der Trennung von Struktur und Inhalt, darstellt.

Während die ursprünglichen Funktionen von CSS auf HTML beschränkt waren, wurden in der CSS-Version 2.0 einige Veränderungen im Bezug auf XML einbezogen. Mit CSS ist es jedoch nicht möglich, einzelne Elemente in eine komplexere Struktur von Objekten zu integrieren und eine Interaktivität bzw. dynamische Gestaltung von Webseiten zu erreichen. Beispielsweise können mit CSS keine Inhaltsverzeichnisse, Seiten- und Kapitelnummerierungen oder feste Kopf- und Fußzeilen automatisch erstellt werden.

Die Funktionsvielfalt von CSS beschränkt sich somit auf eingeschränkte Veränderungen an einzelnen Elementen.

Wenn aufwendigere Formatierungen zu realisieren sind, dann sollte man die extra für XML entwickelte „Extensible Stylesheet Language“ nutzen.

Stylesheets referenzieren

Die CSS-Stylesheet-Datei wird direkt im XML-Dokument referenziert. Nach der Referenz werden die Tags mit den jeweiligen Inhalten erstellt, die mittels Stylesheet-Datei formatiert werden.

<?xml:stylesheet <!--hierdurch wird dem Browser mitgeteilt, dass eine Stylesheet-Definition folgt-->

type="text/css" <!--dadurch wird dem Browser mitgeteilt, dass dieses Dokument lediglich Textelemente enthält-->

href="adresse.css"?> <!--hier wird auf die CSS-Datei verwiesen-->

Beispiel: XML- Datei mit CSS und Browserausgabe

Ein Beispiel für eine Browserausgabe einer XML-Datei mittels Cascading Stylesheets befindet sich im Anhang auf Seite 28.

5.4.2. XSL (Extensible Stylesheet Language)

Die erweiterbare Stylesheet-Sprache XSL erwächst aus dem Sprachraum der XML-Familie und stellt eine der kompliziertesten Spezifikationen dar. XSL teilt sich in zwei Bereiche auf: XSLT, dass für Transformationen benutzt wird und XSL-FO (Formatting Objekts), mit dem, wie der Name schon ausdrückt, Objekte formatiert werden.

Nachdem die Formatierung von XML-Dokumenten mittels CSS schon erläutert wurde, wird nun auf die Darstellung und Veränderung von XML-Dokumenten mit der Extensible Stylesheet Language (XSL) eingegangen. Sie ist eleganter und den Ansprüchen von XML besser gewachsen, da

sie „mehr Möglichkeiten bezüglich der Formatierung und des Funktionsumfanges zur Verfügung stellt.“[36]

XSL unterstützt die meisten Funktionen, die auch aus CSS bekannt sind und bietet außerdem noch weitere Möglichkeiten der Auszeichnung.

Über die reine Formatierung hinaus bietet XSL eine Sprache, mit deren Hilfe XML-Instanzen und Formatting Objekts transformiert werden können. XSL macht sich die Baumstruktur eines XML-Dokuments zunutze, indem es Formatierungsregeln auf einen Quellbaum anwendet.[37] Mit XSLT können Transformationsregeln auf eine Dokumentenquelle angewendet und durch Änderung der Baumstruktur ein neues Dokument erzeugt werden. Weiterhin macht es XSLT möglich, mehrere Dokumente in ein einziges Dokument zu überführen bzw. aus einer XML-Datei mehrere Dokumente zu erzeugen. Mittels XSL-Definition können somit XML-Dokumente in andere Formate konvertiert werden. Die häufigsten Formate sind hierbei: HTML, RTF oder PDF.

Der Inhalt der generierten HTML-, RTF- oder PDF-Dokumente entstammt der XML-Datei und die visuelle Umsetzung ist im zugehörigen XSL-Stylesheet festgelegt.

Die folgende Abbildung 2 soll die eben erwähnten Möglichkeiten, die XSL bietet, visualisieren.

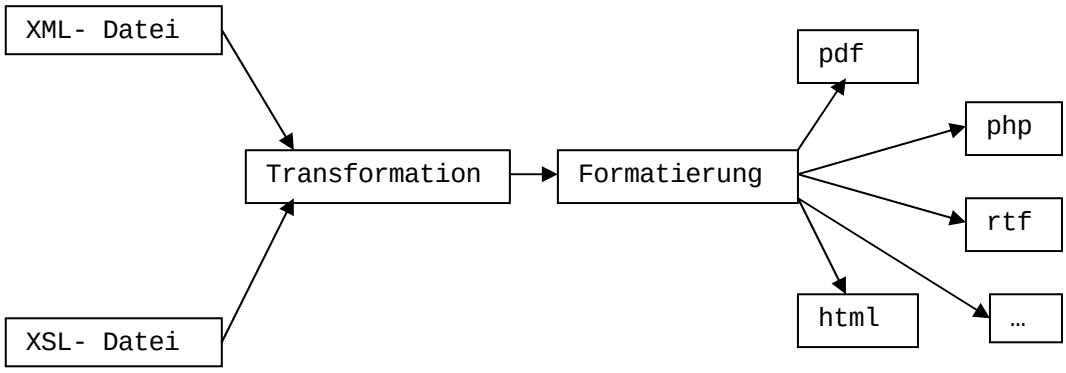


Abbildung 2

Formatierung von Elementen

Mit der Extensible Stylesheet Language XSL bestehen unter anderem folgende Formatierungsmöglichkeiten:

- Schriftarten / Schriftgrößen
- Farben
- Vertikale und horizontale Ausrichtungen
- Gliederung (Überschriften)
- Tabellen
- Behandlung von Leerräumen (Leerzeichen und -zeilen)

[36] Harms/Koch/Kürten (2000), S.547

[37] vgl. Eckstein R. & Casabianca M. (2002), S. 44

Die zwei Hauptaufgaben von XSL

Die XSL hat vor allem zwei Hauptaufgaben. Die erste liegt in der Bildung einer Sprache, mit der sich XML-Dokumente in andere Formate konvertieren lassen (Beispielsweise HTML oder PDF). Die zweite liegt darin, ein Vokabular zur Verfügung zu stellen, das den semantischen Tags bestimmte Formatierungen zuweist (ähnlich der bestehenden CSS).

Einführung in die Programmierung von XSL

Im Folgenden werden einige Grundregeln für die XSL-Programmierung dargestellt. Es wird das Grundgerüst erläutert und auf die Konstruktionsregeln eingegangen. Anschließend wird die Verwendung von Templates dargelegt und anhand eines Beispiels verdeutlicht.

Grundgerüst einer XSL-Datei:

<xsl>

<rule>

<!-- Konstruktionsregel 1 -->

</rule>

<rule>

<!-- Konstruktionsregel 2 -->

</rule>

</xsl>

XSL- Anweisungen werden in einem externen Dokument mit der Endung ».xsl« abgelegt. Diese Datei enthält die physischen Übersetzungen zu den im XML-Dokument definierten Befehlen. Sie muss außerdem in der XML- Datei korrekt referenziert werden, um die gewünschten physischen Auszeichnungen darstellen zu können.

Aufbau der Konstruktionsregel

Eine einzelne Stilanweisung oder auch Konstruktionsregel gliedert sich in zwei zusammenhängende Teile:

Das *Muster* (pattern) stellt den ersten Teil dar.

Durch dieses wird festgelegt, auf welchen (selbstdefinierten) XML-Befehl sich die dann folgende Stilanweisung (action) bezieht. Das Muster ist das Auswahlkriterium, wann die definierte Ausgabeform auf den Inhalt eines Markups umzusetzen ist.

Die *Aktion* (action) ist der zweite Teil der Konstruktionsregel.

Wurde das angegebene Muster im XML-Dokument erkannt, dann folgt die Umwandlung des betreffenden Elements in die angegebene Ausgabeform. Eine »action«-Anweisung kann neben passiven Elementen zur reinen Stilgestaltung auch dynamische Anweisungen, beispielsweise den Aufruf eines JavaScripts oder anderer Skriptsprachen, enthalten.

XSLT- Stylesheet- Struktur

Es gibt ein paar einfache Regeln beim Programmieren von XSL-Stylesheets, die auf die Einhaltung einer allgemeinen Reihenfolge hin zielen. Übergreifend ist vorab noch zu nennen, dass alle XSL-Stylesheets wohlgeformte XML-Dokumente sein müssen. Sie müssen somit den Regeln der „Wohlgeformtheit“ entsprechen.[38]

Erstens müssen alle XSL-Stylesheets mit dem Start-Tag des XSL-Wurzelelements beginnen.

<xsl:stylesheet version=“1.0“

Und am Ende des Dokuments mit dem entsprechenden End-Tag geschlossen werden.

<xsl:stylesheet/>

Zweitens muss jedes <XSL>-Element den Namensraum verwenden, der durch xmlns-Deklaration im <stylesheet>-Element(Start-Tag) spezifiziert ist.

<xsl:stylesheet version=“1.0“ xmlns:xsl=http://www.w3.org/1999/XSL/Transform>

Nach dem Wurzelelement können zusätzlich noch externe Stylesheets mit dem Element <xsl:import> importiert werden. Somit wird die Aufteilung größerer Projekte in einzelne Module möglich und die Wiederverwendung von fertigen XSL-Dokumenten wird dadurch erleichtert.

Templates

Durch die Anwendung von Templates wird ein XML-Dokument für einen gegebenen Knotentyp transformiert. Das Template- Element hat folgendes Aussehen:

<xsl:template match=“Knotentyp“>

…

</xsl:template>

Bei “Knotentyp“ wird der Knotentyp angegeben, der verarbeitet werden soll. In unserem Beispiel im Anhang auf Seite 30 wird mit der Zeile

<xsl:template match="/"> der Knoten „/“, der für das Wurzelelement steht, in der nachfolgenden Spezifikation in HTML transformiert.

Es wird mittels HTML-Tags eine Tabelle generiert, die eine Rahmenstärke von „4pt“ und eine Rahmenfarbe „gelb“ besitzt. Anschließend werden in einzelne Zellen mit vordefinierter Schriftgröße und Farbe die Wörter Vorname, Nachname, Straße, Hausnummer, Postleitzahl und Ort geschrieben. Hiernach werden durch das Tag <xsl:apply-templates/> rekursiv alle anderen Templates dieses Stylesheets auf das Wurzelelement „/“ angewendet. Das heißt, dass auch das später definierte Template „Dokument“ auf das Wurzelelement Anwendung findet.

Anschließend wird durch den Aufruf <xsl:template match="Dokument"> ein Template für das Tag „Dokument“ beschrieben. Durch das folgende Schleifen-Element <xsl:for-each select="Adresse"> werden alle <Adresse>-Elemente in nachfolgender weise bearbeitet. Es werden jeweils Tabellenzellen erstellt und durch die nachfolgenden Anweisungen <td><xsl:value-of select="Vorname"/> </td> <td><xsl:value-of select="Nachname"/></td>… werden die Inhalte des in Anführungsstrichen stehenden Tags in die Zellen geschrieben.

[38] vgl. Eckstein R. & Casabianca M. (2002), S. 18

Im Anhang auf Seite 31 kann man die Browserausgabe der Datei Adresse.xml durch die Stylesheet-Datei Adresse.xsl betrachten.

Im Anhang auf Seite 27 befindet sich eine Zusammenfassung aller verwendeten Tags.

Stylesheets referenzieren

Das Stylesheet-Dokument wird durch folgende Zeile direkt im XML-Dokument referenziert:

<?xml-stylesheet type="text/xsl" href="choose.xsl"?>

In dem Sonderfall, dass gleichzeitig ein XSL-und CSS-Stylesheet im XML-Dokument referenziert ist, wird automatisch das XSL-Stylesheet bevorzugt behandelt.

5.4.3. XSLT als Werkzeug von XSL

Transformationen mit XSLT

Der XSLT-Teil von XSL hat die Aufgabe, Elemente einer XML-basierten Sprache in eine andere XML-gerechte Sprache zu transformieren. Beispielsweise werden Elemente aus einer eigenen XML-Datei, wie vorname und zuname, in Auszeichnungs-Konstrukte einer anderen Sprache transformiert, um damit eine formatierte Ausgabe der Elemente zu erzeugen. Um XML-Daten in HTML zu transformieren, muss eine Verbindung zwischen Elementen und Attributen der XML-Daten und bestimmten HTML-Konstrukten hergestellt werden.

Zum Beispiel wird im XSLT-Stylesheet angeben, dass ein Element namens vorname in den HTML-Code <td>[...]</td> umgesetzt werden soll. Aus einer Notation, wie <vorname>Stefan</vorname> wird dann daraus beim Transformieren als Ergebnis das HTML-Konstrukt <td>Stefan</td> erzeugt. Bei der Transformation wird aus einem Quellbaum ein Ergebnisbaum erzeugt. Dies ist möglich, weil sich alle XML-basierten Daten als Baumstruktur darstellen lassen.

Es ist oft nötig, ganz gezielt auf einzelne Bestandteile (Knoten) der XML-Daten im „Baum“ zuzugreifen, um diese beim Erzeugen des Ergebnisbaums ebenfalls an ganz bestimmten Stellen unterzubringen. Dafür benutzt XSLT eine Syntax, die in einer eigenen Sprache definiert wird: in XPath. XPath beschreibt, wie man in XML-gerechten Datenstrukturen auf beliebige Strukturbestandteile zugreifen kann.

XSLT-Elemente

Es gibt sehr viele Element in XSLT: z.B. xsl:apply-import, xsl:for-each, xsl:apply-templates, xsl:if, xsl:attribute, xsl:import, xsl:call-template, xsl:key, xsl:choose, xsl:message, xsl:comment, xsl:number, xsl:copy, xsl:output, xsl:element, xsl:sort, xsl:fallback, xsl:stylesheet, xsl:strip-space, xsl:text, usw.

Im Nachfolgenden wird nun xsl:choose als Beispiel herausgenommen und näher erläutert.

Einen Ergebnisbaum generieren mit xsl:choose

xsl.choose ist der Rahmen für eine Reihe von Abfragen, die mit xsl:when und xsl:otherwise durchgeführt werden. Es können beliebig viele xsl:when-Abfragen und eine abschließende xsl:otherwise-Anweisung bestehen. Ausgewählt wird diejenige Abfrage, deren Bedingung als erste zutrifft.

xsl:choose besitzt keine Attribute und kann innerhalb von xsl:template vorkommen.

Beispieldatei *choose.xsl* siehe Anhang Seite 32.

Im Beispiel-Stylesheet wird mit <xsl:template match="postleitzahl"> das Template definiert, das die HTML-Ausgabe der Zahlen steuert. Mit <xsl:value-of select="." /> wird der Wert der Zahl ausgegeben. Anschließend wird eine Variable namens *wert* definiert, die sich den Wert, also den Inhalt zwischen <zahl> und </zahl>, merkt. Mit <xsl:choose> wird sodann eine Abfragereihe eingeleitet. Sie enthält eine xsl:when-Abfrage und eine xsl:otherwise-Anweisung. Bei der xsl:when-Abfrage wird abgefragt, ob der Wert der Variablen *wert* kleiner als 10000 ist. Wenn ja, wird hinter die bereits ausgegebene Zahl der Text (unvollständige Postleitzahl) ausgegeben, andernfalls (otherwise) kein Text. Für die Textausgabe wird xsl:text benutzt.

Zur bildlichen Darstellung wurde eine der Postleitzahlen (aus Adresse.xml) manipuliert und das ganze im Internet Explorer ausgegeben:

Die Postleitzahlen lauten:
98693
247 (unvollständige Postleitzahl)
37327

Also erscheint nun hinter einer Zahl der Text, während bei den anderen beiden kein Text angefügt wird.

Im Anhang auf Seite 32 befindet sich der Quellcode für die XML-Datei und die XSL-Datei sowie eine Grafik mit der Browserausgabe dieses Beispiels.

6. Fazit

In diesem einführenden Artikel haben wir versucht, einen grundlegenden Einblick in die beiden Datenformate HTML und XML zu geben und deren Funktionsumfang anzudeuten.

Zusammenfassend lässt sich sagen, dass die Vorzüge von XML unbestritten sind. Hierzu seien nochmals zwei Begriffe genannt, die XML und seine Vorteile, insbesondere auch für die Gestaltung von Webseiten, versinnbildlichen. Das sind zum einen der Begriff „Offenheit“ und zum anderen der Begriff „Standardisierung“. Mit „Offenheit“ ist dabei gemeint, dass mit Hilfe der XML weitere Auszeichnungssprachen entstehen, die leicht interpretierbar sind und daher auch in einem breiten Nutzerfeld Anklang finden. Da XML die Grundlage für weitere Auszeichnungssprachen ist, besteht keine Sorge über die Kompatibilität bzw. die Möglichkeiten für den Datenaustausch.

XML kann überall da zum Einsatz kommen, wo Bedarf für Datenaustausch besteht. Die Zahl möglicher Anwendungen scheint letztlich unbegrenzt zu sein, da jeder seine eigene, an seine Bedürfnisse angepasste, Auszeichnungssprache definieren kann. Daher wird der Verbreitungsgrad von XML sich an den der HTML angleichen.

Jedoch sei am Schluss nochmals hervorgehoben, dass XML kein Nachfolger von HTML ist und es auch nicht werden wird. Vielmehr bildet XML die Grundlage für XHTML, eine neue Form der ursprünglichen HTML. Dies macht deutlich, dass HTML durch XML erweiterbar ist und XML daher eher als Ergänzung zu HTML gesehen werden kann.

Abschließend lässt sich festhalten, dass XML zwar nicht immer die beste Lösung ist, es sich aber immer lohnt, XML in Erwägung zu ziehen oder wie es im Originalton des World Wide Web Consortium (W3C) heißt: „XML isn´t always the best solution, but it´s always worth considering.“

7. Literatur

[Dobnig, Mario(2002)] Einführung: XML verstehen. Online im Internet: URL: http://www.go4xml.com/xml/index.php [19.11.2005]

[Dour_02] Blake Dournaee: XML Security. RSA Press, McGraw-Hill, New York etc, 2002, 379 pages.

[Eckstein R. & Casabianca M. (2002)] Eckstein, Robert/ Casabianca, Michel (2003): XML kurz & gut. O'Reilly Verlag GmbH & Co. KG, Köln

[EZ_02] Esswein, Werner; Zumpe, Sabine: Realisierung des Datenaustauschs im elektronischen Handel. Informatik Spektrum 25/4, August 2002, Springer, Berlin, Heidelberg, 251-261.

[Grimm (2005)] Grimm, Rüdger (2005): Digitale Kommunikation. Oldenbourg, München

[Harms, Koch, Kürten (2000)] Harms, Florian/Koch, Daniel/Kürten, Oliver (2000): Das große Buch (X)HTML & XML. Data Becker GmbH & Co. KG, Düsseldorf

[Herold (2002)] Herold, Helmut (2002): Das HTML/XHTML-Buch: mit cascading style sheets und einer Einführung in XML. SuSE-Press, Nürnberg

[Hess & Karl (2000)] Hess, Uwe/Karl, Günther (2000): XHTML 1.0. Das bhv Taschenbuch. bhv Verlag GmbH, Kaarst

[Kobert (1999)] Kobert, Thomas (1999): XML- Das bhv Taschenbuch. bhv Verlag GmbH, Kaarst

[Pott & Wielage(1999)] Pott, Oliver; Wielage, Gunter 1999: XML- Praxis und Referenz

[SELFHTML e.V. (2005a)] SELFHTML e.V. (2005): HTML als „Lingua Franca" des Web. Online im Internet: URL: http://de.selfhtml.org/intro/technologien/html.htm [21.01.2006]

[SELFHTML e.V. (2005b)]
SELFHTML e.V. (2005): HTML als software-unabhängiges Klartextformat. Online im Internet: URL: http://de.selfhtml.org/intro/technologien/html.htm#klartextformat [21.01.2006]

[SELFHTML e.V. (2005c)]
SELFHTML e.V. (2005): HTML als Hypertext. Online im Internet: URL: http://de.selfhtml.org/intro/technologien/html.htm#hypertext [21.01.2006]

[SELFHTML e.V. (2005d)]
SELFHTML e.V. (2005): Aufbau einer Tabelle. Online im Internet: URL: http://de.selfhtml.org/html/tabellen/aufbau.htm [21.01.2006]

[SELFHTML e.V. (2005e)]
SELFHTML e.V. (2005): Allgemeines zu Dynamischem HTML. Online im Internet: URL: http://de.selfhtml.org/dhtml/intro.htm [21.01.2006]

[Tolksdorf (2000)] Tolksdorf, Robert 2000: Die Sprache des Web: HTML und XHTML. dpunkt-Verlag, 4. Auflage, Heidelberg, S. 15

[Vonhoegen, Helmut (2005)]
Einstieg in XML – 3., erweiterte Auflage Galileo Computing

[W3C (2005)] Bray, Tim; Paoli, Jean; Sperberg-McQueen, C. M.; Maler, Eve; - Zweite Auflage im Internet unter: http://edition-w3c.de/TR/2000/REC-xml-20001006/

[W3C_1 (2005)] Bray, Tim; Paoli, Jean; Sperberg-McQueen, C. M.; Maler, Eve; - Zweite Auflage im Internet unter: http://edition-w3c.de/TR/2000/REC-xml-20001006/#sec-intro

[W3C_2 (2005)] Bray, Tim; Paoli, Jean; Sperberg-McQueen, C. M.; Maler, Eve; - Zweite Auflage im Internet unter: http://edition-w3c.de/TR/2000/REC-xml-20001006/#sec-intro

[Wikipedia (2005)] Wikimedia Foundation Inc. (2005): Extensible Markup Language. Online im Internet: http://de.wikipedia.org/wiki/XML [19.11.2005]

8. Diskussionsprotokoll

Eingeleitet wurde die Diskussion mit einem Zitat des World Wide Web Consortium (W3C): „XML isn´t always the best solution, but it´s always worth considering."

Die Seminarteilnehmer sollten sich dazu äußern, bzw. berichten, welche Programmier- oder Auszeichnungssprache sie für die eigene Webseite nutzen und worin sie die Vor- und Nachteile der einzelnen Sprachen sehen. Ein Seminarteilnehmer brachte es auf den Punkt, indem er meinte, dass man, je nach dem welche Ziele man hat, eine dementsprechende Sprache wählen soll. Also eine, die den Anforderungen entspricht.

9. Anhang

Abbildung 3: beispiel.html

```
<h1>Rotweinkuchen</h1>
<table border="1">
 <colgroup>
  <col width="1*">
  <col width="2*">
 </colgroup>
  <tr>
    <th>Menge</th>
    <th>Zutaten</th>
  </tr>
  <tr>
    <td>250 g</td>
    <td>Zucker</td>
  </tr>
  <tr>
    <td>300 g</td>
    <td>Margarine</td>
  </tr>
</table>
```

Abbildung 4: Beispiel für eine Tabelle „Rotweinkuchen“

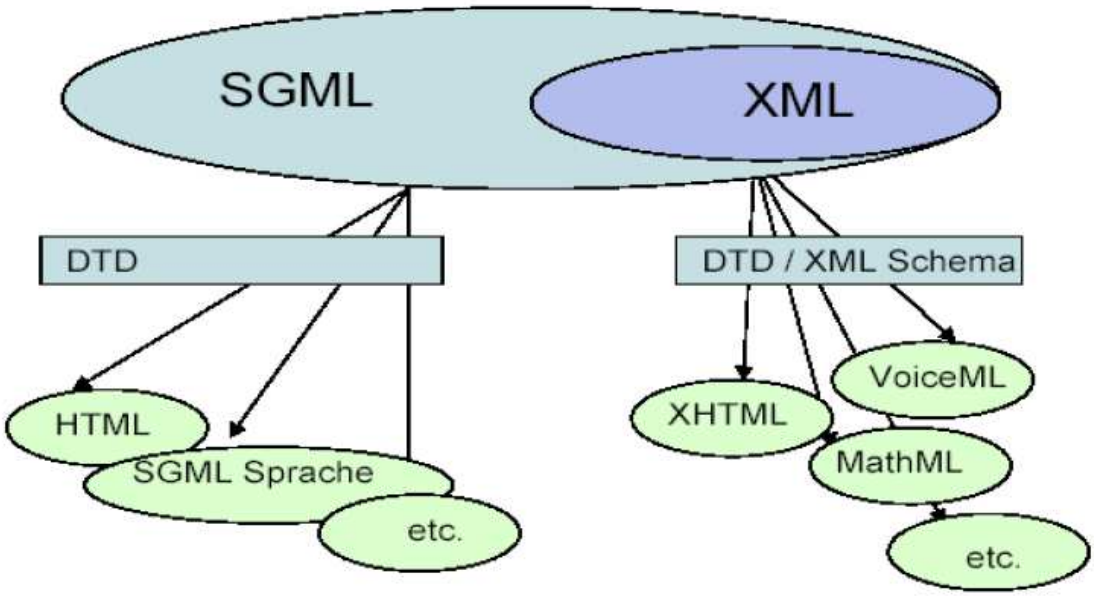


Abbildung 5: Einordnung der Sprachfamilie

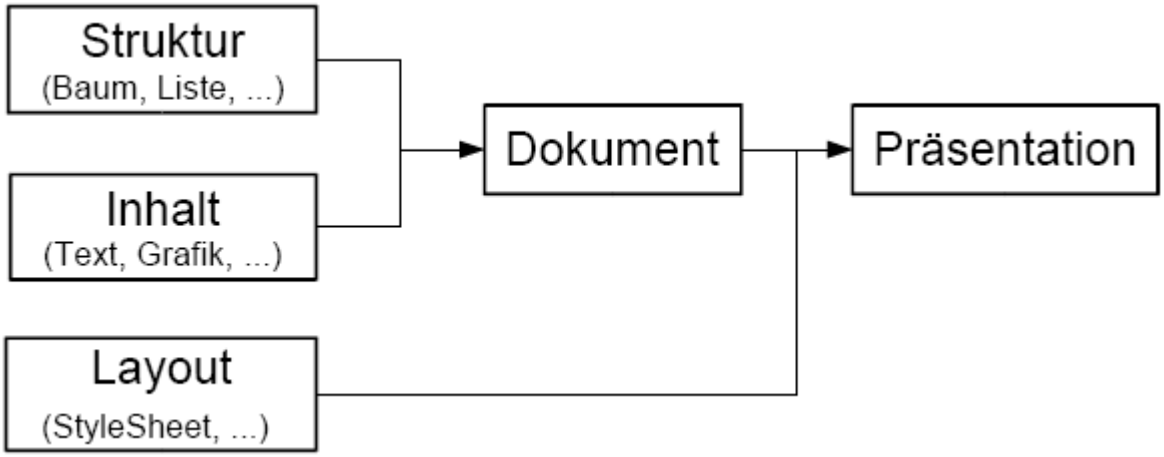


Abbildung 6: Trennung von Struktur, Inhalt und Layout

Zusammenfassung der verwendeten Tags:

<?xml:stylesheet
{hierdurch wird dem Browser mitgeteilt, dass eine Stylesheet- Definition folgt}

type="text/xsl"
{dadurch wird dem Browser mitgeteilt, dass dieses Dokument lediglich Textelemente enthält}

href="style.xsl"
{hier wird auf die XSL-Datei verwiesen}

xsl:apply-templates
{Templates werden angewendet}

xsl:for-each
{Tag erlaubt, alle Tags eines bestimmten Types im aktiven Template auszugeben und zu bearbeiten. Daher ist es z.B. für Inhaltsverzeichnisse und Stichwortlisten prädestiniert.}

<xsl:for-each select="//Adresse">
{findet alle (// entspricht an jeder Position des Dokuments) „Adressen“ des Dokuments und gibt sie mit Namen aus → interessanter Nebeneffekt = Element hinter “select” wird zum aktiven Knoten}

<xsl:value-of select="Vorname"/>
{Zwischen den Anführungszeichen steht der Wert der ausgegeben wird}

Beispiel: XML- Datei mit CSS und Browserausgabe

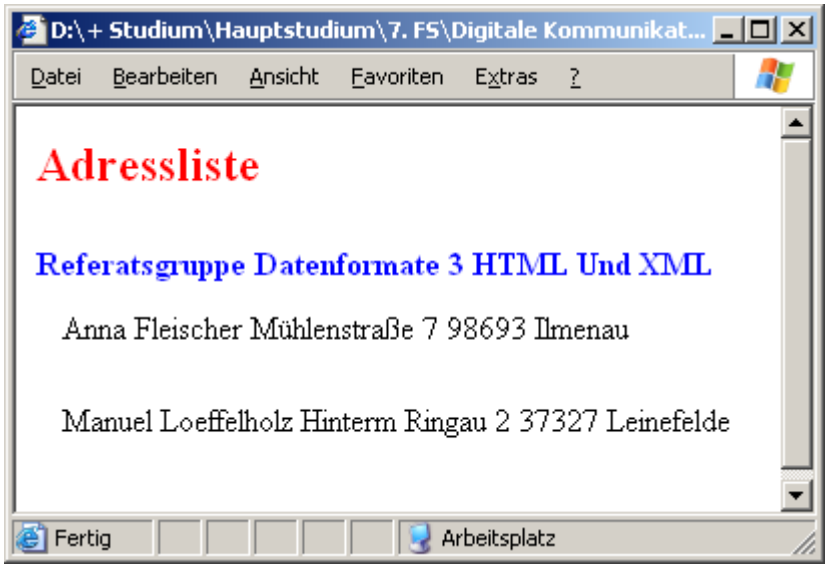
Inhalt der Datei adresse.xml:

```
<?xml version="1.0"?>
<!-- <!DOCTYPE Adresse SYSTEM "adresse.dtd"> -->
<?xml-stylesheet type="text/css" href="adresse.css"?>

<Dokument>
<titel>Adressliste</titel>
<autor>referatsgruppe datenformate 3 HTML und XML</autor>

	<Adresse>
		<Vorname>Anna</Vorname>
		<Nachname>Fleischer</Nachname>
		<Strasse>Mühlenstraße</Strasse>
		<Hausnummer>7</Hausnummer>
		<Postleitzahl>98693</Postleitzahl>
		<Ort>Ilmenau</Ort>
	</Adresse>
	<Adresse>
		<Vorname>Manuel</Vorname>
		<Nachname>Loeffelholz</Nachname>
		<Strasse>Hinterm Ringau</Strasse>
		<Hausnummer>2</Hausnummer>
		<Postleitzahl>37327</Postleitzahl>
		<Ort>Leinefelde</Ort>
	</Adresse>
</Dokument>
```

Inhalt der Datei adresse.css:

```
TITEL
      { display: block;
       font-size: 150%;
       font-weight: bolder;
       text-align: left;
       color: red;
       padding-top: 1pt;
       padding-bottom: 20pt
       }


AUTOR
      {      display: block;
       font-weight: bolder;
       //text-transform: capitalize;
       text-align: left;
       color: blue;
       padding-bottom: 10pt
       }

adresse
      {      display: block;
       text-align: left;
       text-indent: 10pt;
       line-height: 1.2;
       padding-bottom: 20pt
       }
```

Browserausgabe der XML- Datei:



Beispiel: XML- Datei mit XSL-Stylesheet und Browserausgabe

Inhalt der Datei „Adresse.xml“:

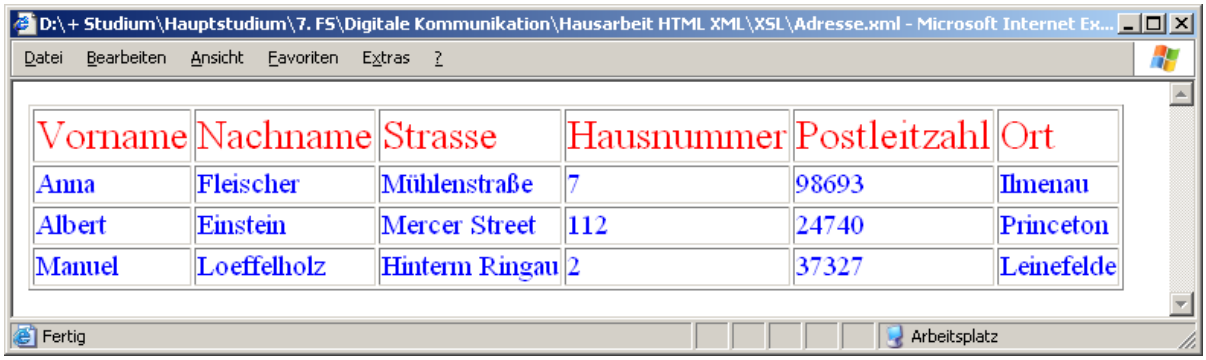
```
<?xml version="1.0"?>
<!-- <!DOCTYPE Adresse SYSTEM "adresse.dtd"> -->
<?xml-stylesheet type="text/xsl" href="Adresse.xsl"?>

<Dokument xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.schniederlihof.de adresse.xsd">
       <Adresse>
              <Vorname>Anna</Vorname>
              <Nachname>Fleischer</Nachname>
              <Strasse>Mühlenstraße</Strasse>
              <Hausnummer>7</Hausnummer>
              <Postleitzahl>98693</Postleitzahl>
              <Ort>Ilmenau</Ort>
       </Adresse>
       <Adresse>
              <Vorname>Albert</Vorname>
              <Nachname>Einstein</Nachname>
              <Strasse>Mercer Street</Strasse>
              <Hausnummer>112</Hausnummer>
              <Postleitzahl>24740</Postleitzahl>
              <Ort>Princeton</Ort>
       </Adresse>
       <Adresse>
              <Vorname>Manuel</Vorname>
              <Nachname>Loeffelholz</Nachname>
              <Strasse>Hinterm Ringau</Strasse>
              <Hausnummer>2</Hausnummer>
              <Postleitzahl>37327</Postleitzahl>
              <Ort>Leinefelde</Ort>
       </Adresse>
</Dokument>
```

Inhalt der Datei „Adresse.xsl“:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
       <xsl:template match="/">
       <html>
              <TABLE border="1pt">
                     <TR STYLE="font-size:20pt; color:red">
                            <TD>Vorname</TD>
                            <TD>Nachname</TD>
                            <TD>Strasse</TD>
```

```
                        <TD>Hausnummer</TD>
                        <TD>Postleitzahl</TD>
                        <TD>Ort</TD>
                    </TR>
                    <xsl:apply-templates/> <!-- wendet alle Templates dieses Styles-
heets rekursiv auf das gegenwärtige Knotenelement an(Knotenelement ist in unserem Fall
das Wurzelelement "/") -->
             </TABLE>
       </html>
       </xsl:template>

  <xsl:template  match="Dokument">

       <xsl:for-each select="Adresse">
             <TR STYLE="font-size:14pt; color:blue">
                    <TD>
                        <xsl:value-of select="Vorname"/>
                    </TD>
                    <TD>
                        <xsl:value-of select="Nachname"/>
                    </TD>
                    <TD>
                        <xsl:value-of select="Strasse"/>
                    </TD>
                    <TD>
                        <xsl:value-of select="Hausnummer"/>
                    </TD>
                    <TD>
                        <xsl:value-of select="Postleitzahl"/>
                    </TD>
                    <TD>
                        <xsl:value-of select="Ort"/>
                    </TD>
             </TR>
       </xsl:for-each>

  </xsl:template>

</xsl:stylesheet>
```

Browserausgabe der XML- Datei:



Vorname	Nachname	Strasse	Hausnummer	Postleitzahl	Ort
Anna	Fleischer	Mühlenstraße	7	98693	Ilmenau
Albert	Einstein	Mercer Street	112	24740	Princeton
Manuel	Loeffelholz	Hinterm Ringau	2	37327	Leinefelde

Beispiel bestimmten Ergebnisbaum erzeugen:

Inhalt der modifizierten Datei Adresse.xsl:

```
<?xml version="1.0"?>
<!-- <!DOCTYPE Adresse SYSTEM "adresse.dtd"> -->
<?xml-stylesheet type="text/xsl" href="choose.xsl"?>

<Dokument xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.schniederlihof.de adresse.xsd">
       <Adresse>
              <Vorname>Anna</Vorname>
              <Nachname>Fleischer</Nachname>
              <Strasse>Mühlenstraße</Strasse>
              <Hausnummer>7</Hausnummer>
              <Postleitzahl>98693</Postleitzahl>
              <Ort>Ilmenau</Ort>
       </Adresse>
       <Adresse>
              <Vorname>Albert</Vorname>
              <Nachname>Einstein</Nachname>
              <Strasse>Mercer Street</Strasse>
              <Hausnummer>112</Hausnummer>
              <Postleitzahl>247</Postleitzahl>
              <Ort>Princeton</Ort>
       </Adresse>
       <Adresse>
              <Vorname>Manuel</Vorname>
              <Nachname>Loeffelholz</Nachname>
              <Strasse>Hinterm Ringau</Strasse>
              <Hausnummer>2</Hausnummer>
              <Postleitzahl>37327</Postleitzahl>
              <Ort>Leinefelde</Ort>
       </Adresse>
</Dokument>
```

Inhalt der Datei choose.xsl:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"/>

<xsl:template match="/">
<html>
<body>
<h3>Die Postleitzahlen lauten:</h3>
<xsl:for-each select="Dokument/Adresse/Postleitzahl">
```

```
<div>
<xsl:value-of select="." />
<xsl:variable name="wert" select="." />
<xsl:choose>
<xsl:when test="$wert &lt; 10000">
  <xsl:text> (unvollständige Postleitzahl)  </xsl:text>
  </xsl:when>
  <xsl:otherwise>
  <xsl:text />
  </xsl:otherwise>
</xsl:choose>
</div>
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Browserausgabe der XML- Datei: