XML-Verarbeitung (VER)

Lernziele
• Sie kennen die Standardschnittstellen zum Zugriff auf XML-Dokumente. • Sie wissen, wie ein XML-Parser arbeitet. • Sie können einschätzen, welcher Parser der geeignete für Ihr Projekt ist. • Sie kennen die Möglichkeiten, um XML im Browser darzustellen. • Sie kennen verschiedene Varianten zur Formatierung von XML-Dokumenten. • Sie kennen die Paradigmen zur Transformation von XML-Dokumenten.

Warum Weiterverarbeitung?

> *Just because everyone is using XML does not mean the need for data conversion will disappear.*
>
> *Michael Kay*

- Weitergabe an unterschiedliche Medien
- Weitergabe an unterschiedliche Zielgruppen
- Hinzufügen, Entfernen und Umstrukturierung von Informationen
- Dynamischen Inhalt aus Datenbanken erzeugen
- Berichterstellung aus XML-Daten

Egal ob XML-Daten von Menschen gelesen werden sollen oder von anderen Softwareanwendungen: Selten werden die XML-Daten in der Form genutzt, in der sie ankommen. In der Regel müssen sie weiter verarbeitet werden.

Weiterverarbeitung ist also sowohl zum Datenaustausch notwendig (Datenkonversion) als auch zum Publishing (Datenpräsentation).

Prozess der XML-Verarbeitung

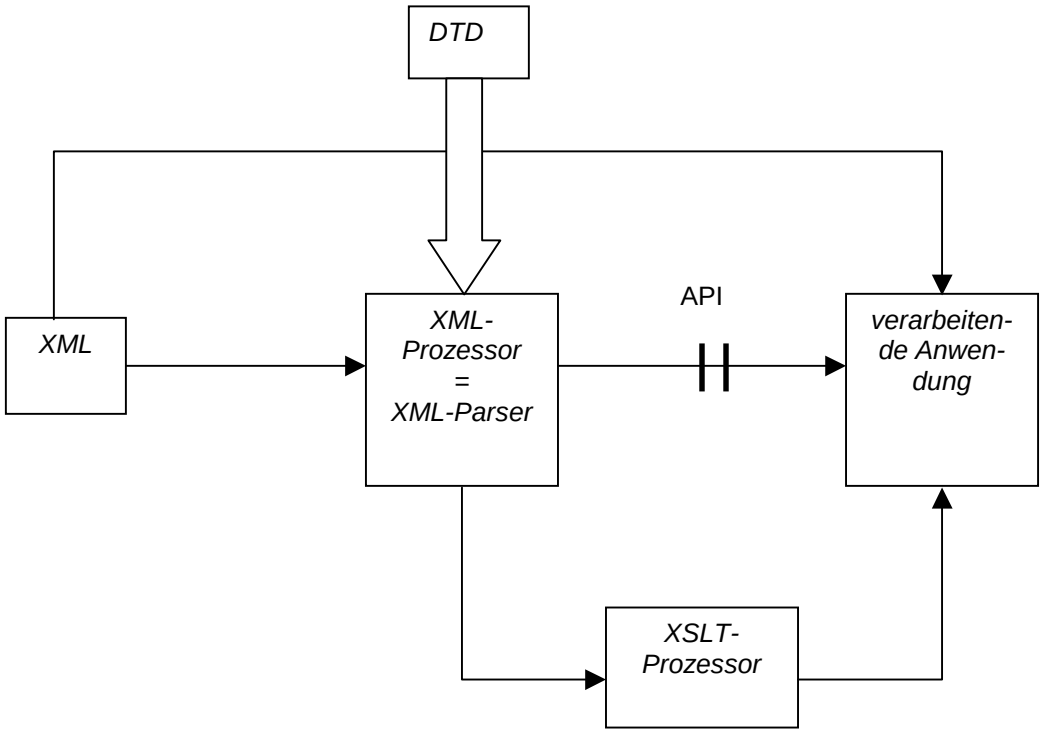

Abbildung VER-1: Schema Verarbeitungsprozess

Sichten auf ein XML-Dokument

XML-Dateien können betrachtet werden als:

- eine spezielle Art von Textdatei
- eine Abfolge von Ereignissen
- eine Hierarchie oder ein Baum

Beispiel VER-1: XML-Dokument als Textdatei

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<rechnung>
  <kunde>
    <name>Urs Wolf</name>
    <anschrift>
      <strasse>Kleestr. 2</strasse>
      <plz>56432</plz>
      <stadt>Köln</stadt>
      <land>DE</land>
    </anschrift>
  </kunde>
  <posten>
    <ware>Spanplatten</ware>
    <einheiten>4</einheiten>
  </posten>
  <posten>
    <ware>Nägel</ware>
    <einheiten>5000</einheiten>
  </posten>
  <posten>
    <ware>Dübel</ware>
    <einheiten>4000</einheiten>
  </posten>
  <posten>
    <ware>Schrauben</ware>
    <einheiten>4000</einheiten>
  </posten>
</rechnung>
```

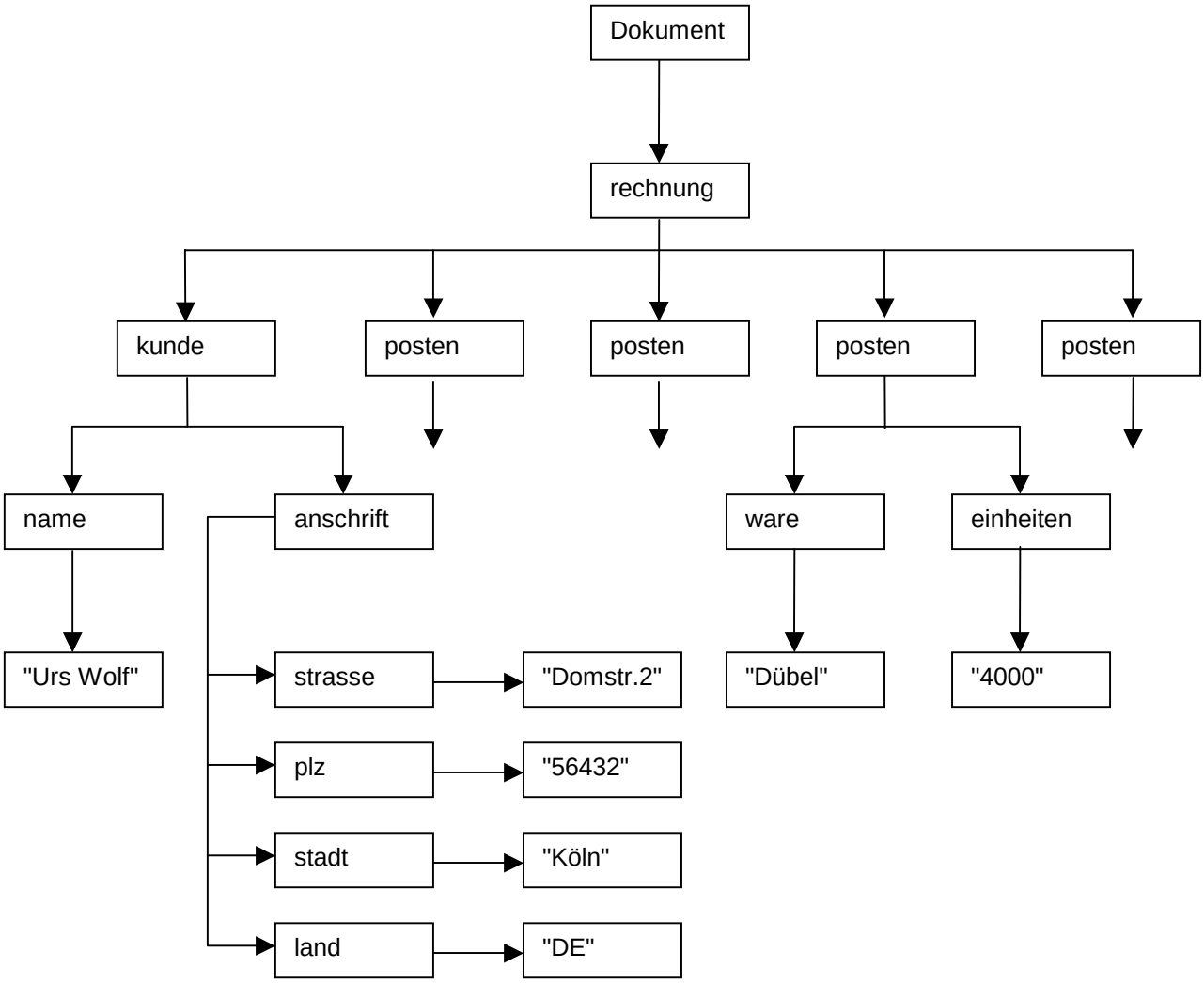

Abbildung VER-2: XML-Dokument als Baum

Elemente tauchen als **Knoten** im Baum auf. Alle Kinder eines Elements bilden eine **Knotenmenge (node set)**. Die Darstellung von Elementinhalten und Attributen sowie anderen Knotentypen unterscheidet sich je nach Baummodell. Ebenso ist die Frage, wie und ob Namensräume, CDATA-Abschnitte, Verarbeitungsanweisungen usw. über den Baum der Anwendung zur Verfügung gestellt werden. In der XML-Welt gibt es da unterschiedliche Ansätze.

Beispiel-VER-2: XML-Dokument als Folge von Ereignissen

```
1:  start document
2:  start element: rechnung
3:  start element: kunde
4:  start element: name
5:  text: Urs Wolf
6:  end element: name
7:  start element: anschrift
8:  start element: strasse
9:  text: Domstr. 2
10: end element: strasse
11: start element: plz
12: text: 56432
13: end element: plz
14: start element: stadt
15: text: Köln
16: end element: stadt
17: start element: land
18: text: DE
19: end element: land
20: end element: anschrift
21: end element: kunde
21: start element: posten
22: start element: ware
23: text: Spanplatten
24: end element: ware
25: start element: einheiten
26: text: 4
27: end element: einheiten
28: end element: posten
29: ......
30: end element rechnung
31: end document
```

Die Reihenfolge der Abarbeitung erfolgt natürlicherweise in Leserichtung, in XSLT auch **Dokumentordnung (Document Order)** genannt.

Arten von XML-Parsern

Ein XML-Parser ist das, was die XML-Empfehlung XML-Prozessor nennt. Ein XML-Parser filtert den Inhalt aus dem Markup eines XML-Dokuments und stellt ihn einer Anwendung über eine definierte Schnittstelle zur Verfügung.

Validierend - nicht validierend

Nach der XML-Spezifikation kann ein Parser

- validierend
- nicht validierend

sein.

Nicht-validierende Parser prüfen lediglich, ob ein Dokument wohlgeformt ist.

Ein **validierender Parser** verwendet auch eine DTD, um zu prüfen, ob ein Dokument in Form und Inhalt gültig ist. Viele Parser unterstützen beide Arten der Bearbeitung, die sich je nach Bedarf einstellen lässt.

Sie können nun hingehen und als echter Hacker ihren eigenen Parser schreiben. Einfacher ist es jedoch, Sie verwenden einen der mittlerweile zahlreichen verfügbaren Parser. Diese lassen sich im wesentlichen in zwei Gruppen einteilen.

baumorientiert - ereignisorientiert

Es gibt zwei Parsermodelle, die jeweils eine unterschiedliche Sicht auf den Inhalt von XML-Dokumenten zur Verfügung stellen.

- baumorientierte Parser
- ereignisorientierte Parser

Ein **baumorientierter Parser** liest ein XML-Dokument ein und baut aus der hierarchischen Struktur der Elemente einen Baum aus Objekten im Speicher auf. Man kann dann direkt auf den Elementen des Baums operieren.

Die Weiterverarbeitung mittels baumorientierten Parsern beinhaltet also zwei Durchläufe

1. Parsen und Aufbauen des Baums
2. eigentliche Datenverarbeitung

Das Prinzip der **ereignisorientierten Parser** ist Ihnen vielleicht von der Programmierung grafischer Benutzeroberflächen her bekannt. Ein solcher Parser ruft für jede Art von Komponente eines XML-Dokuments eine Methode oder Funktion auf.

Das bedeutet, dass Ihrer Anwendung nicht die Kontrolle obliegt. Wenn der Vorgang beginnt, dann ruft Ihr Programm nicht den Parser auf, vielmehr ruft der Parser Ihr Programm auf.

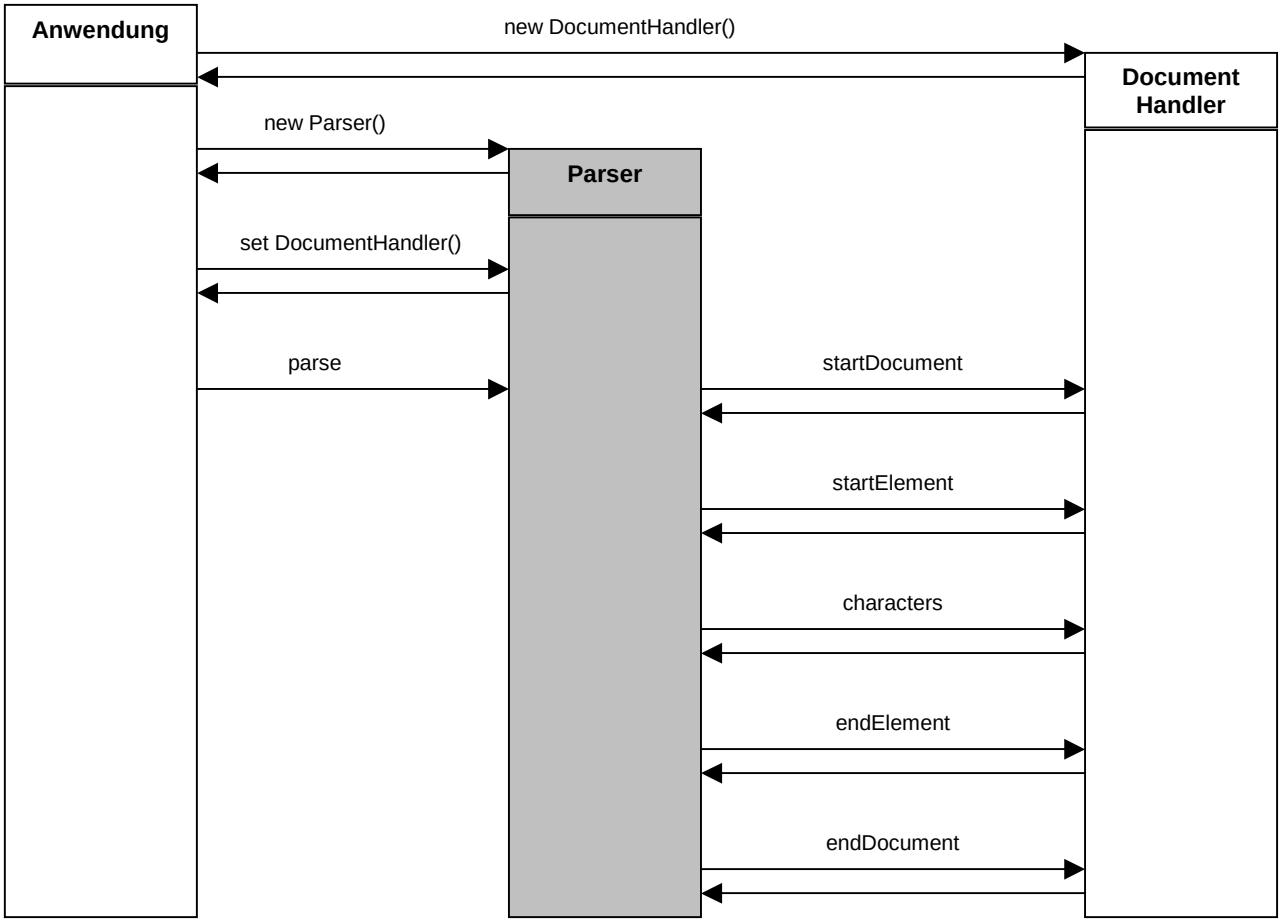

Abbildung VER-3: Arbeitsweise ereignisorientierter Parser

Standard-APIs für XML-Verarbeitung

DOM

DOM = Document Object Model

DOM ist eine vom W3C standardisierte, sprachneutrale Schnittstelle für den Zugriff auf XML-Dokumente. In der Regel implementieren baumorientierte Parser das DOM und stellen eine entsprechende API (*Application Programming Interface*) in einer Programmiersprache zur Verfügung. Diese API definiert Interfaces und Methoden zur Suche im DOM-Baum und Manipulation des DOM-Baums.

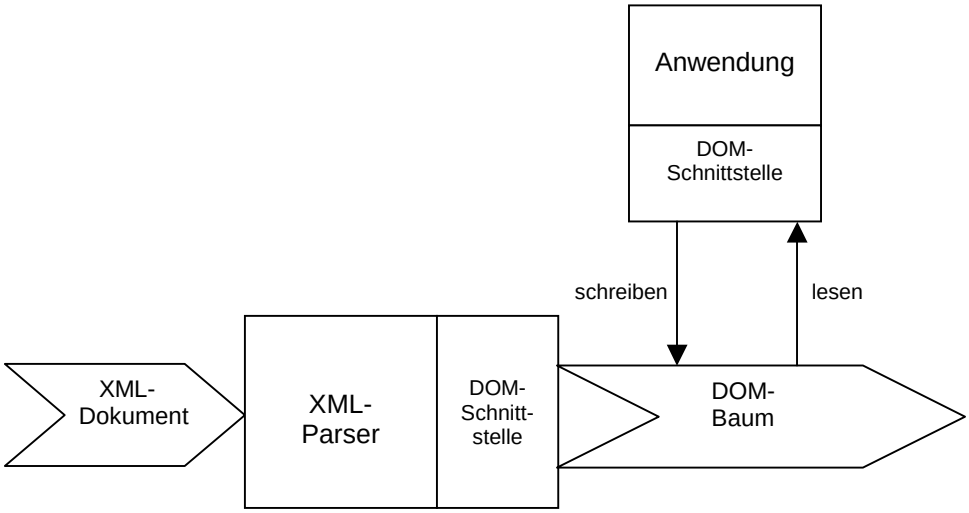

Abbildung VER-4: DOM-basierte Architektur von XML-Anwendungssystemen

Die DOM-Interfaces

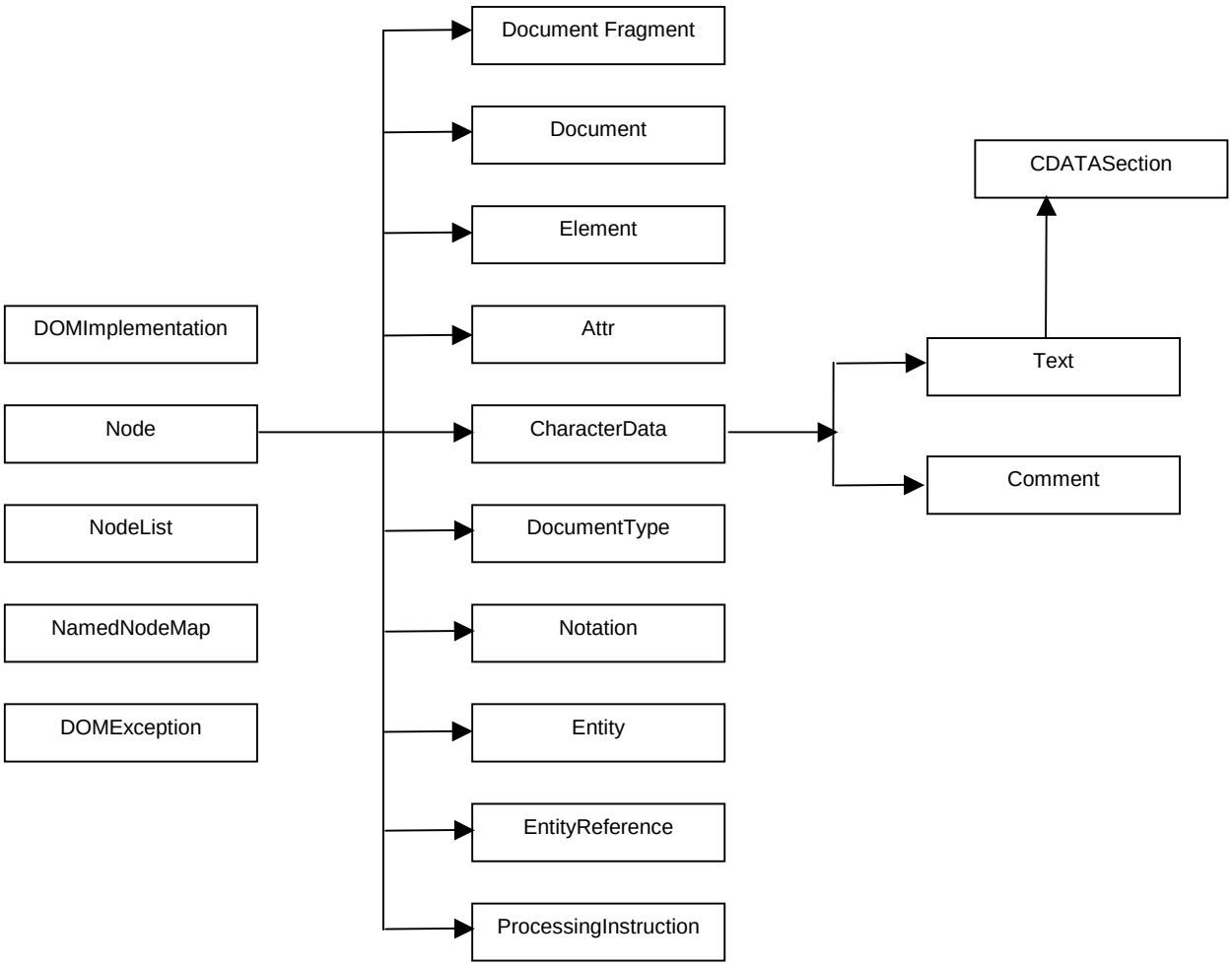

Abbildung VER-5: DOM-Interfaces (Level 1)

Das zentrale Interface von DOM heißt *Node*. In ihm werden die wichtigsten Methoden für den Zugriff auf Knoten im Strukturmodell, dem DOM-Baum, definiert. Weitere Interfaces wie *Element*, *Attr*, *Text* oder *Document* sind Spezialisierungen von *Node*, die alle Methoden von *Node* erben und weitere, spezielle Methoden hinzufügen.

DOM-Methoden (Auswahl)

Interface	Methoden

Node	`String getNodeName()`
	`String getNodeValue()`
	`Node getParentNode()`
	`NodeList getChildNodes()`
	`Node getFirstChild()`
	`Node getLastChild()`
	`Node getPreviousSibling()`
	`Node getNextSibling()`
	`NamedNodeMap getAttributes()`
	`Node insertBefore(Node newChild,`
	`Node refChild)`
	`Node appendChild(Node newChild)`
	`Node replaceChild(Node newChild,`
	`Node oldChild)`
	`Node removeChild(Node oldChild)`
	`boolean hasChildNodes()`
	`Node cloneNode(boolean deep)`

NodeList	`Node item(int index)` `int getLength()`

NamedNodeMap	`Node getNamedItem(String name)`
	`Node setNamedItem(Node arg)`
	`Node item(int index)`
	`int getLength()`

Attr	`String getName()`
	`String getValue()`
	`void setValue(String value)`

Element	`String getTagName()`
	`String getAttribute(String name)`
	`void setAttribute(String name, String value)`
	`void removeAttribute(String name)`

Document	`Element getDocumentElement()`
	`Element createElement(String tagName)`
	`Text createTextNode(String data)`
	`Attr createAttribute(String name)`
	`NodeList`
	`getElementsByTagName(String tagName)`

Ausblick

Das DOM wird ständig weiterentwickelt. Zur Zeit gibt es zwei Empfehlungen (*Recommendations*) des W3C zum Thema DOM:

- DOM Level 1 (18.10.1998)
- DOM Level 2 (13.11.2000)

DOM Level 1 beschränkt sich im Wesentlichen auf die Beschreibung von Schnittstellen für die verschiedenen Knotentypen und die Navigation zwischen diesen Knoten.

DOM Level 2 weitet die API auf eine ganze Reihe neuer Bereiche aus.

Bereiche des DOM Level 2

Namensräume	Die Interfaces des DOM Level 1 werden um die Namensraumunterstützung erweitert
Stylesheets	Die Interfaces dieses Pakets dienen zur Repräsentation von Cascading Style Sheets (CSS)
Views	Die beiden Interfaces dieses Pakets dienen als Grundlage für die Definition von Sichten auf das Dokument, die nur bestimmte Teilmengen des Dokuments beinhalten
Range	Klassen, die Bereiche von Knoten im DOM-Baum darstellen
Traversal	Klassen und Interfaces, die Traversierungen im DOM-Baum erleichtern
Events	Klassen und Interfaces eines neuen Event-Models

DOM Level 3 befindet sich zur Zeit noch im Status eines Entwurfs (*Working Draft*).

SAX

SAX = Simple API for XML

SAX ist eine Schnittstelle für die ereignisorientierte Verarbeitung von XML-Dokumenten. Die maßgebliche Spezifikation liegt in der Gestalt von Java-Interfaces vor, mittlerweile gibt es aber auch Implementierungen für Perl, Python u.a.

Ein SAX-Parser implementiert die SAX-Interfaces und stellt eine API für die Verarbeitung von Ereignissen zur Verfügung.

SAX wurde von Teilnehmern der Mailing-Liste XML-DEV unter Leitung von David Megginson entwickelt und hat sich zu einer Art defacto-Standard entwickelt.

SAX 1 wurde bereits am 11. Mai 1998 released.

SAX 2 wurde am 05.05.2000 freigegeben.

Die aktuellste Version von SAX ist **SAX 2.0.1** (29.01.2002). Sie ist vollständig abwärtskompatibel zu SAX 1, unterstützt darüber hinaus nun aber Namensräume und stellt Filter zur Verfügung.

Interfaces von SAX 2

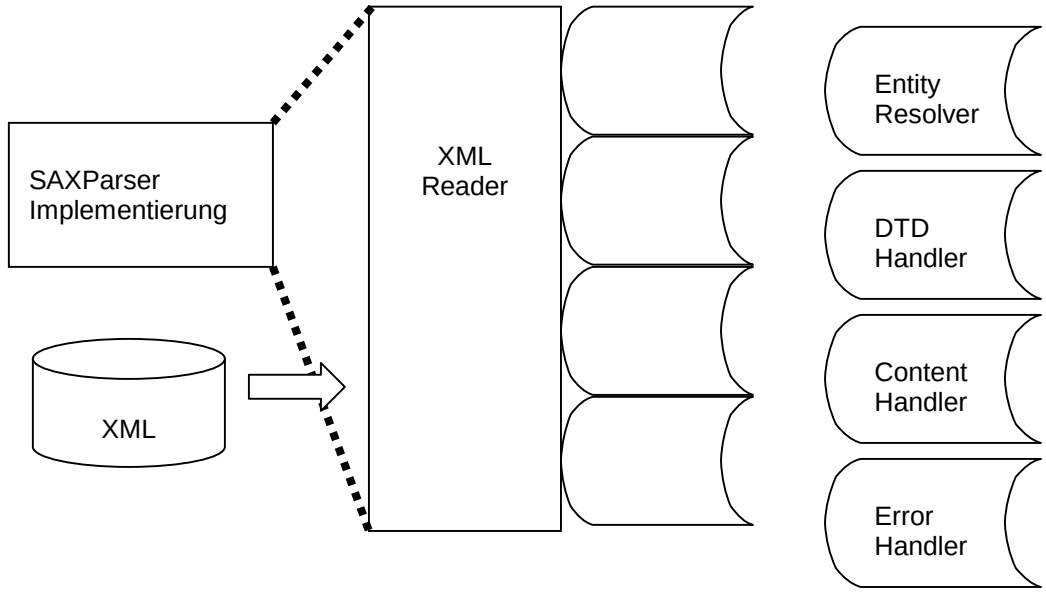

Abbildung VER-6: : Interfaces von SAX 2

Methoden von der SAX-2-Interfaces (Auswahl)

Interface	Methoden

ContentHandler	`void startDocument()`
	`void endDocument()`
	`void startElement(String namespaceURI,`
	`String localName,`
	`String qName,`
	`Attributes atts)`
	`void endElement(String namespaceURI,`
	`String localName,`
	`String qName)`
	`void characters(char[] ch,`
	`int start,`
	`int length)`

ErrorHandler	`void error(SAXParseException exception)`
	`void fatalError(SAXParseException exception)`
	`void warning(SAXParseException exception)`

DTDHandler	`void notationDecl(String name,`
	`String publicId,`
	`String systemId)`
	`void unparseEntityDecl(String name,`
	`String publicId,`
	`String systemId,`
	`String notationName)`

EntityResolver	`InputSource resolveEntity(String publicId,` `String systemId)`

DOM versus SAX

Vorteile SAX

- schnell
- geringer Speicherverbrauch
- kann Dateien beliebiger Größe parsen
- hilfreich zum Aufbau eigener Datenstrukturen

Nachteile SAX

- kein freier Zugriff auf beliebige Teile eines Dokuments
- beschränkt auf lesende Zugriffe
- komplexere Suchoperationen können nur schwer implementiert werden
- SAX wird nicht von Browsern unterstützt

Vorteile DOM

- Zugriff auf gesamtes Dokument
- schreibender Zugriff auf Dokument
- Browserunterstützung
- ideal für interaktive Aktionen auf dem XML-Dokument

Nachteile DOM

- hoher Speicherverbrauch
- nicht geeignet für sehr große Dateien
- langsamer wegen 2 Durchläufen bei der Verarbeitung

Übersicht über gängige XML-Parser

Name	Lizenz	Entwickler	Sprachen	API	Validierend
expat	frei	James Clark => Clark Cooper	ANSI-C, C++, Perl, Python	stream Wrapper für SAX u. DOM	nein
URL: http://expat.sourceforge.net					
xp	frei	James Clark	Java	SAX	nein
URL: http://www.jclark.com/xml/xp/index.html					
AElfred	frei	David Megginson => Microstar	Java (Applets)	SAX	nein
URL: http://www.opentext.com/services/content_management_services/aelfred.zip					
Xerces-J *Xerces-C++*	open source	Apache XML Project	Java, C++	DOM 2, SAX 2	ja
URL: http://xml.apache.org/xerces2-j/index.html (Xerces-J) http://xml.apache.org/xerces2-j/index.html (Xerces-C++)					
XML4J *XML4C*	frei	IBM alphaworks	Java, C++	DOM 2, SAX 2	ja
URL: http://www.alphaworks.ibm.com/tech/xml4j (XML4J) http://www.alphaworks.ibm.com/tech/xml4c					
MSXML	frei	Microsoft	DLL	DOM 2 SAX 2	ja
URL: http://www.msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/001/766/msdncompositedoc.xml					

XML im Browser

Die beiden großen Browser Netscape 6 und 7 und Microsoft Internet Explorer 5 und 6 "verstehen" XML. Das gilt auch für Mozilla 1.0. Dies heißt nichts anderes, als dass sie einen XML-Prozessor integriert haben, der XML-Dokumente parst und sie der Anwendung (dem Browser) zur Verfügung stellt.

Netscape 6, 7 und Mozilla

Rufen Sie ein XML-Dokument im Netscape 6 oder 7 auf, so gibt dieser den Inhalt der Elemente gänzlich ohne Formatierung wieder.

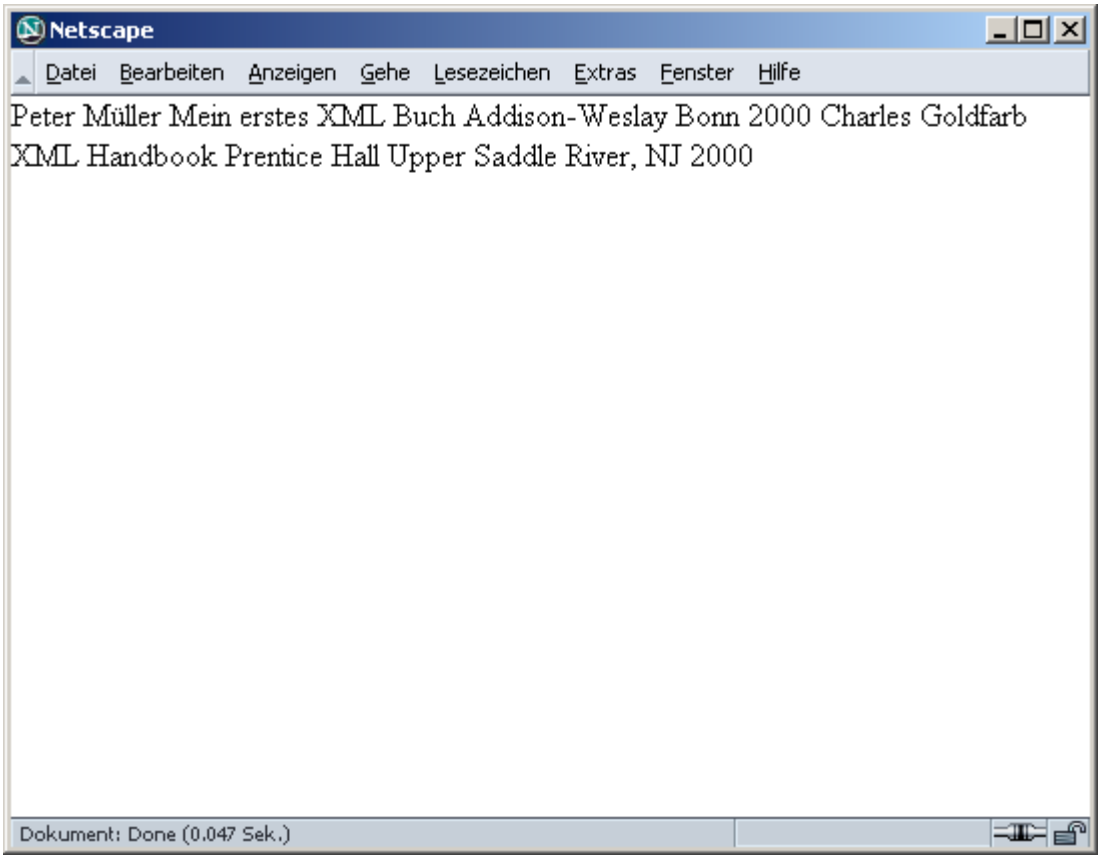

Abbildung VER-7: XML im Netscape-Browser 6

Netscape validiert die XML-Dokumente nicht, auch wenn eine DTD referenziert wird.

Opera 6 und Mozilla 1 verhalten sich ebenso wie NS 6!

Seit der Version 7.1 (Netscape) bzw. 1.2 (Mozilla) stellen die beiden Gecko-basierten Browser auch eine Baumansicht zur Verfügung:

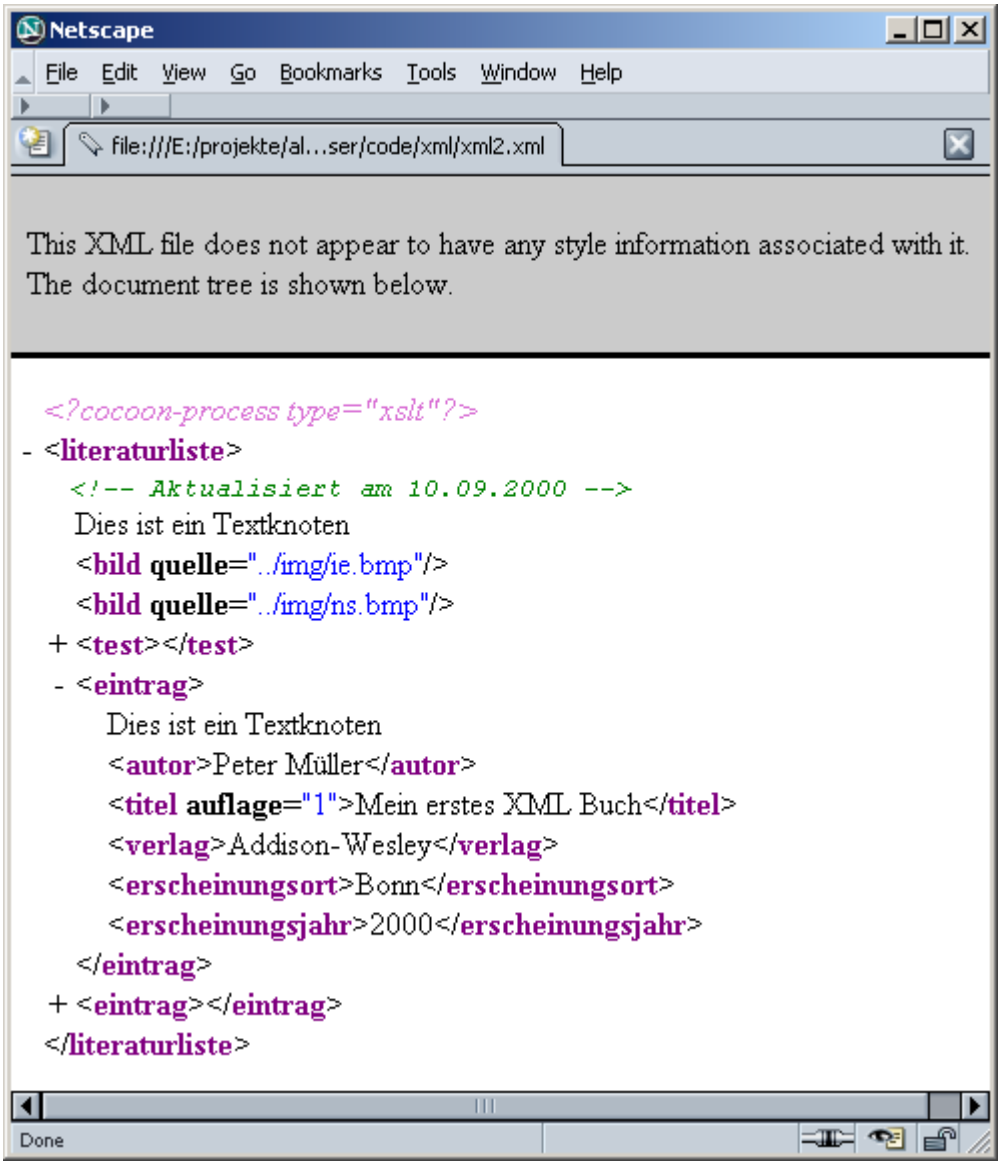

Abbildung VER-8: Baumansicht eines XML-Dokuments im Netscape 7.1

Internet Explorer 5 und 6

Der Internet Explorer 5 oder 6 verwendet ein internes Stylesheet und gibt eine strukturierte Baumdarstellung des XML-Dokuments aus:

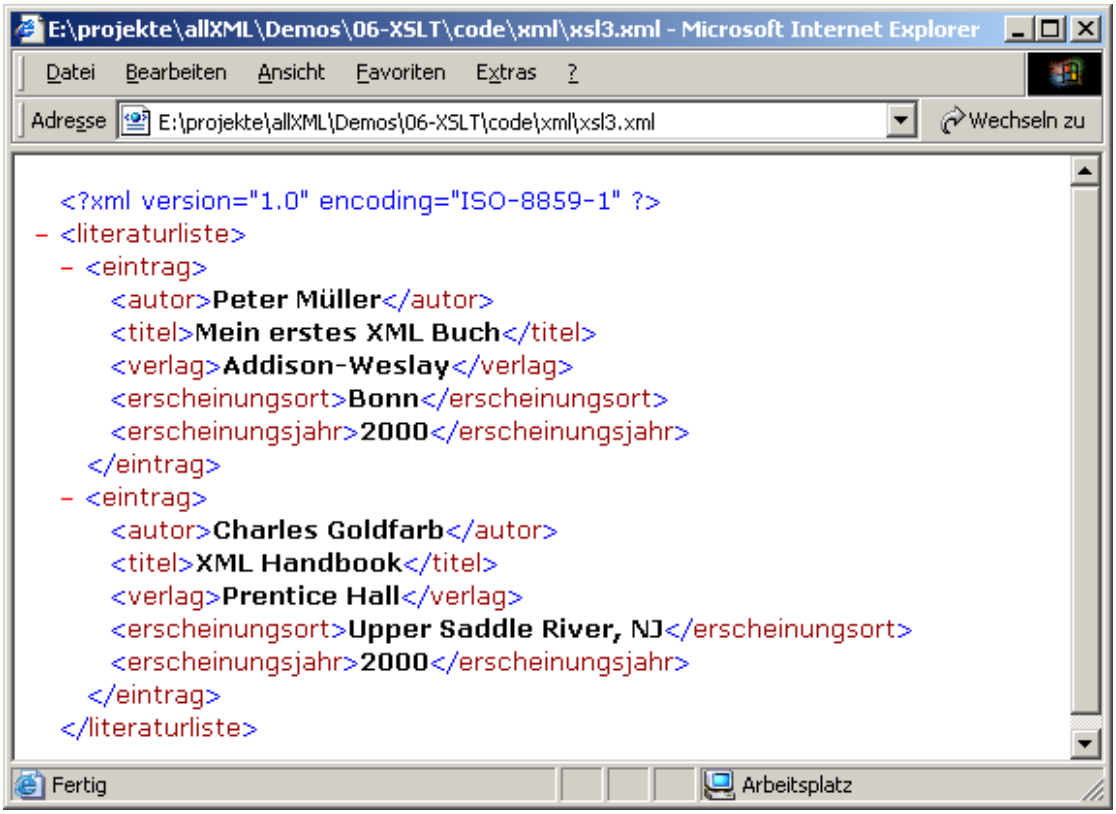

Abbildung VER-9: XML im Internet Explorer

Auch der Internet Explorer validiert die angezeigten XML-Dokumente trotz mitgelieferter DTD allerdings nicht.

TOOL-TIPP:

Sie können den IE durch zwei kleine Tools um zwei äußerst praktische Features erweitern. Laden Sie sich von der Microsoft-Webseite die ***Internet Explorer Tools for Validating XML and Viewing XSLT Output*** *herunter. Nach der Installation enthält Ihr Kontextmenü zwei zusätzliche Menüpunkte, die es Ihnen ermöglichen, Ihre XML-Dateien zu validieren und sich die Quelltext-Ausgabe einer XSLT-Transformation anzusehen, ohne großen Zauber mit der Kommandozeile anstellen zu müssen.*

URL:
`http://msdn.microsoft.com/downloads/default.asp?URL=/code/sample.asp?url=/msdn-files/027/000/543/msdncompositedoc.xml`

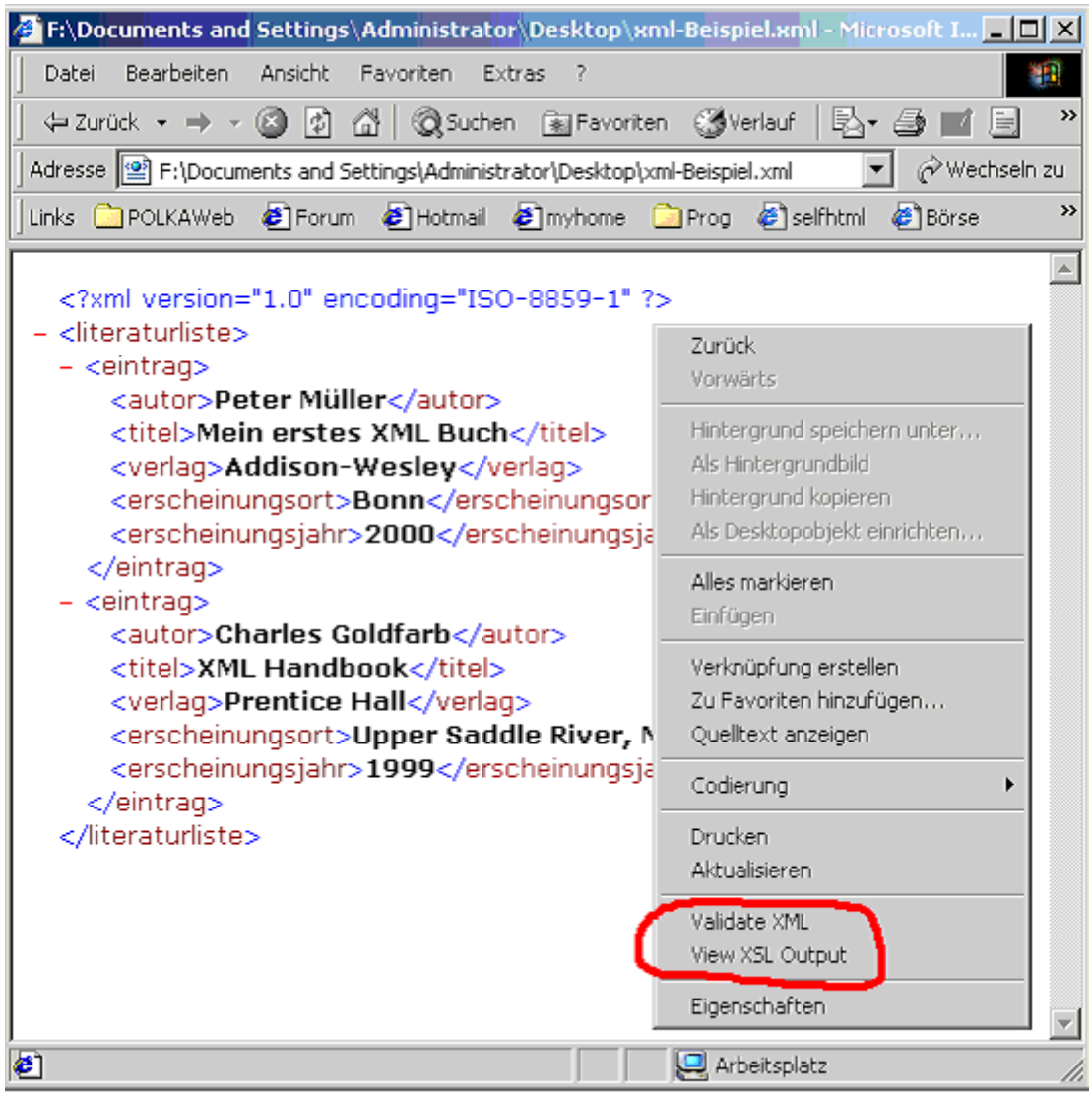

Abbildung VER-10: Internet Explorer Tools for Validating XML and Viewing XSLT Output

Formatierung von XML-Dokumenten

- Cascading Style Sheets (CSS)
- Extensible Stylesheet Language (XSL)
- Extensible Stylesheet Language Transformations (XSLT)

CSS

CSS dienen eigentlich der Formatierung von HTML-Dokumenten, sind aber auch auf XML-Dokumente anwendbar und das ideale Mittel zur clientseitigen Formatierung von XML-Dokumenten.

Beispiel VER-1: CSS und XML

XML-Dokument

```
<?xml version="1.0"?>
  <?xml-stylesheet href="../css/css.css"
                   type="text/css"?>
  <homepage>
    <ueberschrift>XML mit CSS ausgeben</ueberschrift>
    <text>
      Der IE und der NS 6 stellen dies alle beide ohne
Probleme dar
    </text>
</homepage>
```

CSS-Datei

```
ueberschrift {font-family:Arial;font-
size:35pt;color:red;}
text {font-family:Times;font-size:20pt;color:blue;}
```

Nachteile von CSS

- keine Umstrukturierung möglich
- es kann kein Baum gebaut werden
- keine inkrementelle Anzeige

Vorteile von CSS

- weniger Speicherverbrauch als XSLT
- für HTML-Autoren gewohntere Syntax

Sie können CSS auch zusammen mit XSLT einsetzen, wie wir später noch sehen werden!

XSL(-FO)

XSL ist die vom W3C für XML-Dokumente geschaffene XML-basierte Formatierungssprache.

XSL hat bereits eine längere Geschichte, die bis 1997 zurückreicht. Noch vor der Verabschiedung der offiziellen XML-Empfehlung (Februar 1998) begann die Arbeit an einer Formatierungssprache für XML. Wie XML sich von SGML ableitete, so sollte sich XSL von *DSSSL (Document Style Semantics and Specification Language)* ableiten. DSSSL ist die Formatierungssprache für SGML. XSL war allerdings von Anfang an als XML-Anwendung konzipiert, um die für XML entwickelten Parser für XSL wiederverwenden zu können.

Mit XSL definieren Sie die Formatierung und das Layout für die Darstellung von XML-Dokumenten auf Papier oder dem Bildschirm. XSL ist letztlich nichts anderes als ein XML-Vokabular, das die Bereiche einer darzustellenden Seite definiert und ihre Eigenschaften festlegt. Dies geschieht mittels so genannter **Formatierungsobjekte (*Formatting Objects*)**. Aus diesem Grund wird XSL in der Regel auch als **XSL Formatting Objects (XSL-FO)** bezeichnet, wenn man lediglich von der Formatierungssprache XSL ohne XSLT reden möchte.

XSL-FO ist ein sehr mächtiges und komplexes Werkzeug, mit dem sich die vielfältigsten Publikationsaufgaben in fast allen Sprachen der Erde bewältigen lassen. So unterstützt XSL beispielsweise beide Schreibrichtungen: von links nach rechts (wie in diesen Seminarunterlagen) und von rechts nach links (wie in arabischen Sprachen üblich). Kein Wunder also, dass die Empfehlung über 400 Seiten lang ist, ein Vielfaches länger als XSLT und XPath zusammen.

TOOL-TIPP

*Falls Sie sich für XSL interessieren, schauen Sie sich einmal das Formatting-Object-Project (**FOP**) des Apache-XML-Projekts an. FOP (eine Java-Applikation) erzeugt mittels Formatting Objects PDF-Dokumente.*

URL:
`http://xml.apache.org/fop`

CSS oder XSL-FO?

- Wenn es um die Formatierung im Web geht, dann ist aufgrund fehlender Browserunterstützung für XSL-FO CSS die richtige Wahl.
- XSL-FO ist eher für komplexe Layoutaufgaben im DTP-Bereich gedacht und die Browserhersteller werden es wohl in absehbarer Zeit auch nicht implementieren, zumal sich CSS schnell weiterentwickeln.

XSLT

Während des Entwicklungsstadiums von XSL wurde deutlich, dass dem Formatierungsprozess meistens eine strukturelle Transformation vorausgeht. Aus diesem Grund wurde XSL in zwei Teile aufgeteilt

- XSL Transformation (XSLT)
- XSL-FO

XSLT ist seit November 1999 in der Version 1.0 eine Empfehlung des W3C. Es gibt heute zahlreiche XSLT-Prozessoren, die diese Spezifikation implementieren.

Das W3C arbeitet bereits an einer Version 2. Sie hat z.Zt. allerdings erst den Status eines Entwurfs. Der zwischenzeitlich veröffentlichte Entwurf einer Version 1.1 wird nicht mehr weiter entwickelt.

XSLT dient zur Transformation von Dokumenten einer XML-Sprache in eine andere. Genauer gesagt: XSLT ist eine Sprache zur Transformation der Struktur eines XML-Dokuments. Somit lässt sich z.B. ein XML-Dokument mittels XSLT in (X)HTML transformieren, welches der Browser dann darstellen kann.

Transformation von XML-Dokumenten

Was ist Transformation?

Transformation bedeutet das Überführen einer Menge von Informationen durch Anwenden von Regeln in eine andere Darstellungsform. Dabei kann zum einen die gesamte Menge an Informationen transformiert werden oder nur ein Teil. Das Auswählen eines Teils wird **Selektion** genannt.

Zur Transformation von XML-Dokumenten können Sie die Low-Level-Standard-APIs oder auch XSLT im Zusammenspiel mit XPath verwenden.

Um Dokumententeile anzusprechen und auszuwählen verwendet XSLT **XPath**-Ausdrücke. Eine Transformation, die in XSLT formuliert ist, nennt man **Stylesheet**.

Beachten Sie:
Transformationen von XML-Dokumenten müssen nicht ausschließlich zu Präsentationszwecken durchgeführt werden. Beim Datenaustausch zwischen verschiedenen Applikationen ist es oftmals nötig, Dokumente der einen Anwendungsdomäne in das Schema der anderen Anwendungsdomäne zu transformieren. Somit ist der Name Stylesheet eigentlich etwas irreführend.

Transformation hat eine sehr weitreichende Bedeutung im Kontext von XSLT. So können Sie mit XSLT Dokumente sortieren, Elemente hinzufügen, nummerieren, kopieren, umgruppieren, Berechnungen durchführen usw.

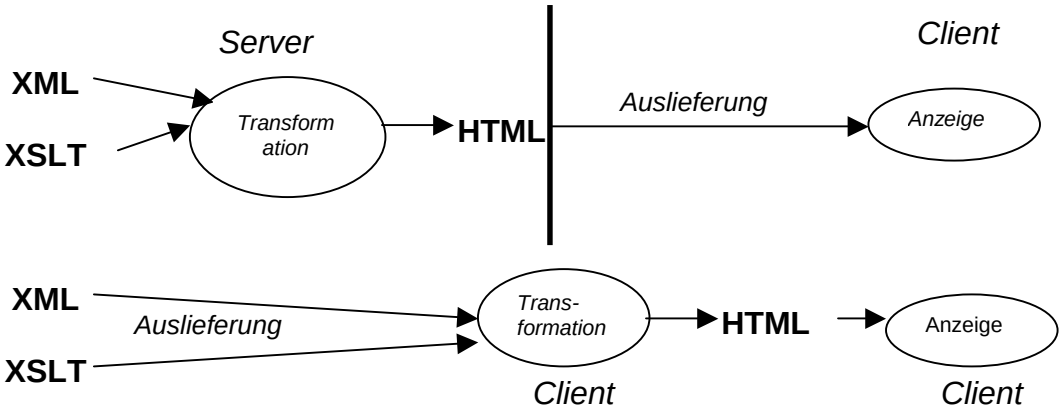

Abbildung-VER-1: Publikationsstrategien für XML

Transformation auf dem Server

Z.Zt. werden Transformationen noch überwiegend auf dem Server durchgeführt, da die wenigsten Clients XSLT verstehen. Dabei lassen sich zwei Verfahren unterscheiden:

- dynamische Transformation ("on the fly")
- statische Transformation ("Batch")

Dynamische Transformation

Eine Anwendung auf dem Server wandelt das angeforderte XML-Dokument z.B. in HTML und liefert es an den Browser. Eventuell liegen auf dem Server verschiedene Stylesheets für verschiedene Browser(versionen) bereit.

TOOL-TIPP:

Cocoon *ist ein Framework für das Web-Publishing mit XML auf dem Server. Cocoon nimmt ihnen die Arbeit ab, selbst eine Anwendung zu schreiben, die ihren XML-Parser aufruft, das Ergebnis an einen XSLT-Prozessor (oder andere Prozessoren) weiterreicht und schließlich HTML ausliefert. Was mit ihrer XML-Datei geschehen soll, geben Sie in Cocoon 1 mittels Verarbeitungsanweisungen) in ihrer XML-Datei und in Cocoon 2 über eine zentrale XML-Konfigurationsdatei (Sitemap) an. Cocoon parst und verarbeitet die Datei dann entsprechend ihren Anweisungen.*

URL:
`http://xml.apache.org/cocoon`

Statische Transformation

Die Daten werden in XML gehalten und ein XSLT-Prozessor wandelt dann die XML-Dateien in HTML um. Die HTML-Dateien werden auf den Webserver gelegt.

Transformation auf dem Client

Der Internet Explorer und Mozilla sowie Netscape ab Version 6.2 verstehen allerdings bereits XSLT, sodass man dort die XSLT-Verarbeitung auch clientseitig durchführen kann.

TOOL-TIPP:

Falls Sie vorhaben, mit XSLT im IE 5 zu arbeiten, sollten Sie in jedem Fall die ***aktuelle Version des msxml-Parsers*** *von der Microsoft-Webseite herunterladen. Die ausgelieferten Versionen des IE haben noch eine alte Version integriert, die nicht den neuen Namensraum der XSLT-Spezifikation unterstützt. Deshalb kommt es immer wieder zu Verwirrungen. Zudem wird die XSLT-Spezifikation nur unzureichend unterstützt.*

URL:
`http://msdn.microsoft.com/downloads/default.asp?URL=/code/sample.asp?url=/msdn-files/027/000/541/msdncompositedoc.xml`