DHBW
Duale Hochschule
Baden-Württemberg

HARMAN

Web- und Multimediabasierte Informationssysteme

Vorlesunsmitschrieb

für die Prüfung zum

Bachelor of Engineering

des Studiengangs Informationstechnik

von

Jan Ulses

15. September 2014

Dozent:	Jürgen Röthig
E-Mail:	jr@roethig.de
Vorlesungszeitraum:	29.09.14 - 31.03.14
Klausurtermin:	19.12.2014
Autor:	Jan Ulses
Kurs:	TINF12B3
Ausbildungsfirma:	Harman/Becker Automotive Systems GmbH
Studiengangsleiter:	Jürgen Vollmer

Inhaltsverzeichnis

KAPITEL 1

XML

- eXtensible Markup Language
- Mittel um konkrete Auszeichnungssprachen zu definieren
- XML selbst ist eine Metasprache, keine eigene (konkrete) Sprache

Auszeichnungssprache: Sprache um reinen Text weitere Eigenschaften mitzugeben

Designorientiert: Textbestandteile bekommen Aussehen (z.B. Fettdruck, rote Farbe). Beispiel: klassisches Word 1990, Druckformate (PostScript, PCL)

Strukturorientiert: Struktur oder spezielle Funktion (z.B. Überschrift, Absatz, Liste, Tabelle). Beispiel: HTML, LaTeX, SGML, die meisten XML-basierten

1.1 Beispiele für XML-basierte Sprachen

- XHTML (auch HTML5 in XML-Variante)
- SVG (Grafikformat)
- XSD (Sprache zur Definition XML-basierter Sprachen)
- Viele Konfigurationsdateien vieler Software-Pakete (z.B. Apache)
- Dokumentformate (aktuellere) von Microsoft Office (.docx) oder Open Office
- Austauschformate für Inhalte von relationalen Datenbanken

1.2 DOCTYPE-Deklaration

```
<!DOCTYPE html(root Tag) | PUBLIC(bzw. SYSTEM, PRIVATE) "Public-
   Id"(bei PUBLIC) "Syst-Id"(nicht bei PRIVATE)>
```

Listing 1.1: Syntax einer DOCTYPE-Deklaration

Public-ID: ungefähr wie bei HTML "–//W3C/DTD/XHTML1.1/EN"
System-ID: URL, verweist auf konkrete Grammatik in Form einer DTD

1.3 Wesentliche Eigenschaften von XML-Dateien

Es gibt zwei wesentliche Eigenschaften, welche jedes XML-basierte Dokument erfüllen muss bzw. sollte.

- Wohlgeformtheit (z.B. XML-Deklaration)
  - Wurzel-Tag, welcher das komplette Dokument umschließt
  - Tags paarweise, also Start- und Endtag
  - Korrekte Schachtelung (letzter geöffneter Tag zuerst schließen)
- Gültigkeit (z.B. DOCTYPE-Deklaration, insbesondere Verweis auf DTD)
  - Entspricht einer konkreten Grammatik (Tagnamen, Attributnamen und Zugehörigkeit, Enthaltenseinsmodell (Inhalt eines Tags)

1.4 Document Type Definition

Document Type Definitionen (kurz: DTD)

- Definiert eine konkrete Grammatik (XML-basiert)
- Besteht aus Text
- Besteht nur aus Deklarationen (wegen fehlendem Wurzeltag nicht XML-basiert)

1.5 ELEMENT-Deklaration

```
<!ELEMENT tagname inhaltsmodell>
```

Listing 1.2: Syntax einer ELEMENT-Deklaration

tagname: Name des Elements/Tags bestehend aus Buchstaben (Klein- und Großschreibung wird unterschieden, ¡bla¿ ist nicht gleich ¡bLa¿) und Ziffern, 1. Zeichen muss Buchstabe oder Unterstrich sein, theoretisch beliebig lang, praktisch kleiner 256 Zeichen, keine Umlaute verwenden.

inhaltsmodell:

EMPTY	(Bsp. aus XHTML: <!ELEMENT br EMPTY> für leere Tags ohne Inhalt)
(#PCDATA)	für beliebige Zeichenfolgen (außer „<“, „>“, „&“ und „”“) insbesondere keine Tags z.B. <!ELEMENT title (#PCDATA)>
sequenz	z.B. (tagname1, tagname2) -¿ Abfolge von tagname1 und tagname2 z.B. ¡!ELEMENT html (head, body)
auswahl	z.B. (tagname1 — tagname2)
gemischt	((#PCDATA) — auswahl)*

Alle Inhaltsmodelle können mit nachgestellten Quantoren versehen werden:

- (inhaltsmodell)* beliebig oft (inkl. Keinmal)
- (inhaltsmodell)+ beliebig oft, aber mindestens einmal
- (inhaltsmodell)? einmal oder keinmal

Entitäten:

&lt	„<“	Less than
&gt	„>“	Greater than
&amp	„&“	Ampersand
&quot	„”“	Quotation mark
&auml	„ä“	
&Auml	„Ä“	

1.6 ATTLIST-Deklaration

```
<!ATTLIST tagname attrname attrtype voreinstellung (optional) >
```

Listing 1.3: Syntax einer ATTLIST-Deklaration

attrname: Name des Attributs, genauso aufgebaut wie Tagnamen

attrtype:

CDATA	beliebige Zeichenfolgen inklusive „<“ und „>“, Einschränkung bei einfachen/doppelte
ID	dokumentweit eindeutiger Wert, Einschränkung an Werteraum wie bei Tag- und Attri
IDREF	Referenz/Verweis auf einen ID-Wert, Einschränkung der Werte wie oben, aber keine E
IDREFS	beliebig viele ID-Werte, getrennt durch Leerzeichen
NMTOKEN(S)	„Name“, d.h. Zeichenfolge von beliebig vielen Buchstaben, Ziffern, manchen Sonderzei
	aufzaehlung: (nmtoken1—nmtoken2—nmtoken3—...) der Attributwert kann nur einer
ENTITY	Verweis auf Entitäten -> externe (auch binäre) Daten
ENTITIES	Verweis auf Entitäten -> externe (auch binäre) Daten
NOTATION	Daten mit spezieller Interpretation

voreinstellung:

#REQUIRED	Pflichtattribut
#IMPLIED	optionales Attribut
#FIXED	wert, Attribut mit festem Wert wert
wert	#IMPLIED-Attribut mit Default-Wert wert
[fehlt]	#IMPLIED-Attribut ohne Default-Wert

Die „Gültigkeit“ einer XML-Datei kann anhand der DOCTYPE-Deklaration und der darin referenzierten DTD überprüft werden -> mittels einem Validator z.B. für HTML-Dateien ”http://validator.w3.org/”.
Problem: Zugriff des Validators auf die DTD? Muss die DTD auf einem öffentlich zugänglichen WebSpace liegen? -> NEIN, Abhilfe: Inline-DTD, siehe das Beispiel aus Listing 3.1 auf Seite 12.

```
<?xml version="1.0" ?>
<!DOCTYPE bla [
  <!ELEMENT ...>
    .
    .
  <!ATTLIST ...>
    .
    .
]>
<bla>
    .
    .
</bla>
```

Listing 1.4: Inline-DTD Beispiel

1.6.1 Beispiel

Name	Vorname	Matrikelnummer	Kursbezeichnung	Wahlfach
Müller	Max	012345	TINF12B3	WuMBasis
Maier	Moritz	4711	TINF12B3	Shit
Schulze	Marta	0815	TINF12B5	Gaming

```
<studis>
  <studi matrikelnummer="012345">
    <name nach="Mueller" vor="Max" />
    <kurs> TINF12B3 </kurs>
    <wahlfach> WuMM </wahlfach>
  </studi>
  <!-- entsprechend fuer Maier und Schulze -->
</studis>
```

Listing 1.5: Inline-DTD Beispiel

```
<!ELEMENT studis (studi)*>
<!ELEMENT studi (name, kurs, wahlfach)>
<!ELEMENT name EMPTY>
<!ATTLIST name nach CDATA REQUIRED attrtype evtl. NMTOKEN vor CDATA
    #REQUIRED attrtype evtl. NMTOKENS>
```

Listing 1.6: Inline-DTD Beispiel

Anzeige abstrakter XML-Daten (nicht HTML oder SVG) im XML-fähigen WebBrowser?

- strukturierte Liste (mit Einschränkungen, Elemente aus- und einklappbar)
- nicht besonders anschaulich
- kann mit CSS deutlich „aufgehübscht“ werden
- bessere Variante: XSLT (XML Style Sheet Language Transformation)
  Achtung: Trotz des Namensbestandteils “Stylesheet macht eine XSLT viel mehr als nur Aussehen festzulegen!

KAPITEL 2

XSL

2.1 Bestandteile

Die XML Stylesheet Language besteht aus:

- XSLT: XSL Transformation, Sprache zur Transformation von „XML-Konstrukten“ in andere XML-Konstrukte (oder auch „Konstrukte“ in textbasierten Sprachen)
- XPath: XML Path Language, Sprache zur Auswahl von spezifischen „XML-Konstrukten“ aus der XML-Quelldatei
- XML-FO: XML-Formatting Objects, spezielle XML-basierte Sprache zur layoutgetreuen Ausgabe (nicht struktur- sondern designorientierte Sprache)
- im Folgenden für uns in der Vorlesung interessant: XSLT, XPath nicht jedoch XML-FO

Bsp. für Anwendung: XSLT zur Wandlung der abstrakten „Studis-Datei“ in eine HTML-Datei mit entsprechender Tabelle der Studis

Wer führt die Transformation durch?

- *standalone-Tool:* XSLprocessor (in gängigen Linux-Distributionen enthalten) Apache xalan saxon (von Michael Kay) (unterstützt auch XSLT Version 2)

- *serverseitig:* Integration der XSLT in einem WebServer, d.h. der WebServer liefert bei Anforderung der XML-Datei bereits die mittels XSLT transformierte Datei aus! Apache Project Cocoon Perl-Modul AxKit
- *clientseitig:* integriert in gängige WebBrowser -> Mozilla Firefox, MS Internet Explorer, Opera, Chrome, Safari können XSLT!

KAPITEL 3

XSLT

3.1 Transfomation

Werkzeug zur Transformation von XML-basierten Daten in (meist andere) XML-basierte Daten.

```
<?xml version="1.0" ?> <!--Hinweis: Attribut encoding="UTF-8" ist
    bei XML default-->
<!DOCTYPE studis SYSTEM "url/zur DTD"> //<?+<! sind Deklarationen
    wobei <! auf > endet.
[KEINE DOCTYPE-Deklaration!]
<xsl:stylesheet
  version="1.0" //version->Namensraumdeklaration fuer XSLT,
     Praefix->Postfix
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns="Namespace der Ausgabesprache, z.B. HTML" //
     Namensraumdeklaration fuer Ausgabesprache, Verwendung ohne
     Postfix und Praefix (Grund: Ersparung von Schreibarbeit)
>
  <xsl:output method="xml" encoding ="UTF-8" //method->auch html(
     bitte nicht angeben!) oder text
  doctype-public"..." //Public Doctypes (Doctype definiert den
     HTML-Standart)
  doctype-system"url/zur/DTD" //fuer system Doctype deklaration
  <!--Template fuer die Definition der Transformation-->
</xsl:stylesheet>
```

Listing 3.1: Definition einer XML-Datei zur Transformation

```
  <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE studis SYSTEM "url/zur/DTD">
```

Listing 3.2: Transformierte XML-Datei

Transfoframtionsvorschriften in Form von Templates (Schablonen)

- Templates werden nacheinander notiert, d.h. sie können nicht geschachtelt werden.
- Templates ersetzen irgendwelche Knoten (Tags und Attribute) aus der Quelldatei.

```
<xsl:template match="XPath-Ausdruck">
2 <!--(wohlgeformte) Ausgabe des des Templates, also Text, Tags(
    inklusive Attribute) und  weitere Verarbeitungsanweisungen-->
</xsl:template>
```

Listing 3.3: Syntax einer xsl:template-Deklaration

KAPITEL 4

XPath

Sprache zur Auswahl bestimmter Knoten eines XML-Dokuments. Meist relativ zur aktuellen Position im XML-Dokument

Abbildungsverzeichnis

Tabellenverzeichnis

Listings