

Vorlesung Werkzeuge der Informatik

Grundlagen und Werkzeuge des WWW (Teil 2)

Jörg P. Müller

Inhalt

- **Entwicklung von Internet und WWW**
- **WWW-Architektur und Protokolle**
 - Web Ressourcen (oder: Was ist eine URL)
 - Basisprotokoll des Internet: TCP/IP
 - WWW-Architektur (Client-Server)
 - Das HTTP-Protokoll
- **Darstellung von WWW-Inhalten**
 - HTML als Web Content Sprache
 - Die eXtensible Markup Language (XML)
 - XML Schema (Basics)

Das Dokumentenmodell des WWW

- Dokumente kann Referenzen (Hyperlinks) auf ein anderes Dokument enthalten
- Hyperlinks werden durch Browser explizit angezeigt → Auswahl durch Anklicken
- Auswahl eines Hyperlinks erzeugt Anforderung an den Server, auf dem das entsprechende Dokument abgelegt ist (über URL)
- Von dort wird es auf die Maschine des Nutzers übertragen und im Browser angezeigt
- Sprache für Webdokumente:
HTML: Hypertext Markup Language

Hypertext Markup Language

- Markup-Sprache zur Annotation und Strukturierung von Dokumenten
- HTML-Dokumente bestehen aus Kopfteil `<HEAD>` und Rumpf `<BODY>`
- HTML-Sprachelemente (=Tags) zum
 - Markieren von Überschriften, Listen, Tabellen, Formularen
 - Einfügen von Bildern und Animationen
 - Gestaltung der Darstellung im Browser (z.B. Schriften, Schriftgrößen, kursive oder fettgedruckte Anzeige, Ausrichtung, Farben, Textabsätze)
- Notation von HTML-Tags in spitzen Klammern, z.B.
 - `<H1>Hello World</H1>`
- HTML-Tags schließen in der Regel Text ein
- Web Browser verwendet HTML-Parser: Software, die HTML-Tags erkennt und in strukturierten Text umsetzt.
- Erweiterungen erlauben die Darstellung von Teilen eines Dokumentes in Form eines im Browser ausführbaren Skriptes (z.B. Javascript)

HTML – Ein erstes Beispieldokument



```
<HTML>
<HEAD>
  <TITLE>Meine Homepage</TITLE>
</HEAD>
<BODY>
  <H1>Homepage von J&ouml;rger M&uuml;ller</H1>
  Dies ist meine erste HTML-Seite.
  <p>
    Sie enth&auml;lt einen Paragraphen und einen <br>Zeilenumbruch.
  </p>
  Au&szlig;erdem kann man auch <b>fett</b> und <i>kursiv</i>
  schreiben.
</BODY>
</HTML>
```

Zeichendarstellung in HTML

- **HTML-Texte werden in Dateien in binärer Form (bytes) abgespeichert**
- **Web-Browser muss Bytes wieder in Zeichen (Buchstaben etc.) umwandeln**
 - Zeichenkodierung, erfolgt mithilfe einer Codetabelle, die jedem erlaubten Zeichen aus dem Zeichenvorrat eine fortlaufende Nummer zuweist
 - Verschiedene Zeichenkodierungsschemata
 - z.B. UNICODE: "a"→97, "b"→98, "c"→99, etc.
- **Browser muss bei der Anzeige die gleiche Kodierung angeben, die der Editor beim Abspeichern verwendet**
 - Spezifikation der Zeichenkodierung im Kopf des HTML-Dokumentes

Zeichendarstellung in HTML (2)

- **Beispiel:**

```
<head>
<meta http-equiv="content-type"
      content="text/html; charset=ISO-8859-1">
<!-- ... andere Angaben im Dateikopf ... -->
</head>
```

- **Darstellung von Sonderzeichen mittels benannter HTML-Elemente, z.B.**

- `ü` für "ü", `Ü` für "Ü"
- `&` für "&", `ß` für "ß"

- **Beispiel:**

```
J&ouml;rg M&uuml;ller trinkt im Hofbr&auml;uhaus
Bier aus Ma&szlig;kr&uuml;gen.
```

Elemente zur Textstrukturierung

- **Bereits kennengelernt: Überschriften, Textabsätze**
- **Darstellung von Listen und Aufzählungen**

```
<ul> <!-- dies ist eine unsortierte Liste -->
<li>Element 1</li>
<li>Element 2</li>
<li>Element 3</li>
</ul>
```

```
<ol> <!-- dies ist eine nummerierte Liste -->
<li>Element 1</li>
<li>Element 2</li>
<li>Element 3</li>
</ol>
```

- **Listen können geschachtelt sein**

Beispiel: Listen in HTML



```
<HTML>
<HEAD><TITLE>Meine Homepage</TITLE></HEAD>
<BODY>
<H1>Homepage von Jörg Müller</H1>
  Berufliche Laufbahn:
  <ul>
    <li>seit 2005: Professor an der TU Clausthal</li>
    <li>1999-2006: Principal Research Scientist bei Siemens
      Corporate Technology, München</li>
    <li>Davor: <ol><li>Drei Jahre Berufserfahrung</li>
      <li>Studium und Promotion</li>
      <li>Schule und Kindergarten</li>
    </ol>
  </li>
  </ul>
</BODY>
</HTML>
```

Wirtschaftsinformatik
© Prof. Dr. Jörg P. Müller, 2008

9

HTML Tabellen

- Tabellen werden häufig zur Strukturierung von HTML-Dokumenten eingesetzt
- Aufbau einer Tabelle in HTML
 - Tabelle <table>
 - Zeile <tr>
 - Tabellenkopfelement <th>
 - Tabellendatenelement <td>
 - Weitere Attribute für Form (z.B. border)

Wirtschaftsinformatik
© Prof. Dr. Jörg P. Müller, 2008

Quelle: de.selfhtml.org

10

Beispiel: Tabellen zur Strukturierung von Dokumenten



```
<HTML>
<HEAD><TITLE>Meine Homepage</TITLE></HEAD>
<BODY>
  <H1>Homepage von J&ouml;rger M&uuml;ller</H1>
  <table border="1">
    <tr> <td colspan="3">Berufliche Laufbahn</td></tr>
    <tr> <th>Jahr</th> <th>T&auuml;tigkeit</th>
      <th>Organisation / Ort</th></tr>
    <tr> <td>seit 2005</td> <td>Professor</td>
      <td>TU Clausthal</td> </tr>
    <tr> <td>1999-2006</td> <td>Principal Research Scientist</td>
      <td>Siemens Corporate Technology, M&uuml;nchen</td> </tr>
    <tr> <td colspan="3">und so weiter</td></tr>
  </table>
</BODY>
</HTML>
```

HTML Formulare

- **Zweck: Eingabe (über Eingabefelder o. Auswahllisten) und Senden (über Button) von Daten vom Client zum Server**
- **Erstellen eines Formulars in HTML, dabei wird spezifiziert, was mit den eingegebenen Daten passieren soll**
- **Anwendungen:**
 - Suchmaschinen
 - Benutzeranmeldung bei einer Web-Applikation
 - Erfassen von Nutzer- und Bestelldaten in E-Shops

Definition von Formularen: Das <form> Tag



```
<HEAD><TITLE>Meine Homepage</TITLE></HEAD>
<BODY>
  <H1>Homepage von J&ouml;rg M&uuml;ller</H1>
  <p> Mit diesem Formular können Sie mit mir Kontakt aufnehmen
  </p>
  <form action="http://de.selfhtml.org/cgi-bin/comments.pl">
    <!-- Hier wird das Formular definiert -->
  </form>
</body>
</html>
```

Der Formularinhalt (1)



```
<form action="http://de.selfhtml.org/cgi-bin/comments.pl">
  <table border="0" cellpadding="5" cellspacing="0"
    bgcolor="#E0E0E0">
    <tr>
      <td align="right">Vorname:</td>
      <td><input name="Vorname" type="text" size="30"
        maxlength="30"> </td>
    </tr>
    <tr>
      <td align="right">Zuname:</td>
      <td><input name="Zuname" type="text" size="30"
        maxlength="40"></td>
    </tr>
    <tr>
      <td align="right" valign="top">Kommentar:</td>
      <td><textarea name="Text" rows="5" cols="50">
        </textarea></td>
    </tr>
  </table>
</form>
```

So sieht's dann im Web Browser aus ...

Meine Homepage - Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

file:///C:/Users/jmue/TUClaustral/Lehre/WILV

Google

Homepage von Jörg Müller

Mit diesem Formular können Sie mit mir Kontakt aufnehmen

Vorname: Jörg

Zuname: Müller

Kommentar: Guten Morgen

Fertig

Was fehlt hier?

Der Button zum Absenden!



```
<form action="http://de.selfhtml.org/cgi-bin/comments.pl">
  <table border="0" cellpadding="5" cellspacing="0"
    bgcolor="#E0E0E0">
    <tr><!-- Hier Vorname --> </tr>
    <tr><!-- Hier Zuname --> </tr>
    <tr><!-- Hier Kommentar --> </tr>
    <tr>
      <td align="right">Formular:</td>
      <td>
        <input type="submit" value=" Absenden ">
        <input type="reset" value=" Abbrechen">
      </td>
    </tr>
  </table>
</form>
```


Das sieht dann so aus:

The screenshot shows a Mozilla Firefox browser window titled "Meine Homepage - Mozilla Firefox". The address bar contains the file path "file:///C:/Users/jmue/TUClausthal/Lehre/WIL/V". The page content includes a title "Homepage von Jörg Müller" and a sub-header "Mit diesem Formular können Sie mit mir Kontakt aufnehmen". Below this is a contact form with three input fields: "Vorname: Jörg", "Zuname: Müller", and "Kommentar: Guten Morgen". At the bottom of the form are two buttons: "Absenden" and "Abbrechen". The status bar at the bottom of the browser window shows "Fertig".

Und das wird übertragen:

<http://de.selfhtml.org/cgi-bin/comments.pl?Vorname=J%F6rg&Zuname=M%FCller&Text=Guten+Morgen>

Wirtschaftsinformatik
© Prof. Dr. Jörg P. Müller, 2008

17

Und hier nochmal der ganze HTML Code

```
<HEAD><TITLE>Meine Homepage</TITLE></HEAD>
<BODY>
  <H1>Homepage von J&ouml;rj M&uuml;ller</H1>
  <p><b>Mit diesem Formular können Sie mit mir Kontakt aufnehmen </b></p>
  <form action="http://de.selfhtml.org/cgi-bin/comments.pl">
    <table border="0" cellpadding="5" cellspacing="0" bgcolor="#E0E0E0">
      <tr> <td align="right">Vorname:</td>
        <td> <input name="Vorname" type="text" size="30" maxlength="30"> </td> </tr>
      <tr> <td align="right">Zuname:</td>
        <td><input name="Zuname" type="text" size="30" maxlength="40"></td></tr>
      <tr> <td align="right" valign="top">Kommentar:</td>
        <td> <textarea name="Text" rows="5" cols="50"> </textarea></td> </tr>
      <tr><td align="right">Formular:</td>
        <td> <input type="submit" value=" Absenden ">
          <input type="reset" value=" Abbrechen">
        </td></tr>
    </table>
  </form>
</body>
</html>
```

Wirtschaftsinformatik
© Prof. Dr. Jörg P. Müller, 2008

18

Attribute des <form> Tags

- **action:** Gibt an, was mit den Eingabedaten passieren soll, wenn das Formular abgesendet wird, z.B.
 - **Email senden:**
`<form action="mailto:mueller@tu-clausthal.de" ...>`
 - **Programm auf dem Server aufrufen, das die Daten weiterverarbeitet**
`<form action="http://www.tu-clausthal.de/cgi-bin/feedback.pl" ...>`
- **method:** Auswahl der http-Übertragungsmethode für die Formulardaten (s.o.)
 - **get:** Daten des ausgefüllten Formulars als Parameter an die Aufrufadresse angehängt
 - **post:** Daten des ausgefüllten Formulars werden vom Web-Server über den Standardeingabekanal (d.h. im Body des HTTP-Requests) zur Verfügung gestellt
 - **put:** Verwendet für das Upload einer Datei vom Client zum Server

HTML: Zusammenfassung

- **Hypertext Markup Language (HTML)**
 - einheitliches, fixes Markup-System
- **Vorteile:**
 - einfach
 - portabel
 - einheitliche Handhabung von Verweisen (Links)
- **HTML erlaubt erstmals den breiten Einsatz einheitlicher Schnittstellen zu Information und Software-Applikationen innerhalb einer Organisation**

Nachteile von HTML

- **HTML ist nicht erweiterbar:**
 - Markup (Elemente und Attribute)
 - Verweise
- **HTML unterstützt nicht die Modellierung strukturierter Informationen (Objektmodelle, relationale Datenbanken)**
- **Integration von Software-Applikationen**
 - JavaScript, Plugins
 - Proprietäre Formate und Sprachen
- **Handhabbarkeit: unzureichende Entkopplung von Inhalt und Präsentation / Layout**
- **Ständiger Wandel der HTML Sprache**

XML

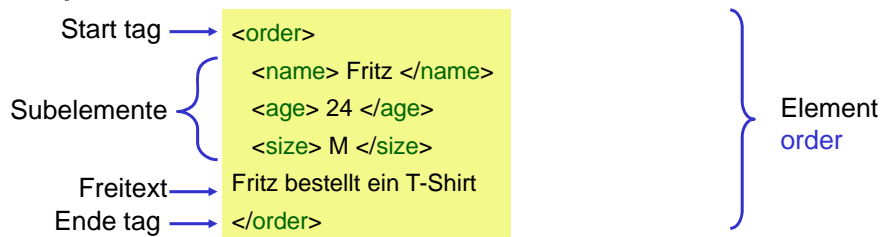
- **XML = eXtended Markup Language**
- **Ursprung: HTML4.0 \in XML \subset SGML (Standard Generalized Markup Language, ISO 8879)**
- **Web-Standard (W3C) für semi-strukturierte Dokumente**
- **Zwei Arten von Dokumenten:**
 - Document-Type-Definitions (DTD's) definieren über eine kontextfreie Grammatik, was ein legales XML Dokument ist
 - XML Dokumente: sind die Datendokumente, strukturiert bzgl. einer DTD
- **HTML & XML:**
 - HTML beschreibt die Präsentation eines Dokumentes durch feste Tags
 - XML beschreibt den Inhalt durch benutzerdefinierte Tags
- **Grundidee von XML: Trennung von Inhalt, Struktur und Präsentation**

Beispiel eines XML Dokumentes

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes">
<ORDER>
  <SOLD-TO>
    <PERSON><LASTNAME>Layman</LASTNAME>
      <FIRSTNAME;>Andrew</FIRSTNAME>
    </PERSON>
  </SOLD-TO>
  <SOLD-ON>19970317</SOLD-ON>
  <ITEM>
    <PRICE>5.95</PRICE>
    <BOOK>
      <TITLE>Number, the Language of Science</TITLE>
      <AUTHOR>Dantzig, Tobias</AUTHOR>
    </BOOK>
  </ITEM>
  <ITEM">
    <PRICE>12.95</PRICE>
    <BOOK>
      <TITLE>Introduction to Objectivist
        Epistemology</TITLE>
      <AUTHOR>Rand, Ayn</AUTHOR>
      <ISBN>0-452-01030-6</ISBN>
    </BOOK>
  </ITEM>
</ORDER>
```

XML Element


- Ein XML Element ist ein Dokumentfragment eingeschlossen in Name Tags `<my_element> </my_element >`
 - `<my_element>` Markiert den Elementanfang
 - `</my_element>` Markiert das Elementende
- Zwischen Anfangs- und End Tag können sein :
 - beliebiger Freitext
 - Andere Elemente (Verschachtelung)
 - nichts, leere Elemente können abgekürzt werden durch: `<my_element />`
- Beispiel:



XML Attribute

- XML Elemente können Attribute besitzen.
- Ein XML Attribut wird als Attribut-Wert-Paar dargestellt
- XML Attribute werden im Start-Tag kodiert
`<my_element my_attr1="value1" my_attr2="value2" >`

- **Beispiel:** 3 Attribute



```
<order name="Fritz" age="24" size="M">
Fritz bestellt ein T-Shirt
</order >
```

Document Type Definitions (!ELEMENT)

- **Beschreibung der zulässigen XML Dokumente**
 - Deklaration der Tags und Attribute
 - Deklaration der Verschachtelungsstruktur
- **Syntax: <!ELEMENT name (definition)>**
 - **name:** Name des Element Tags
 - **definition:** erlaubte Sub-Elemente und andere Inhalte
- **Syntax für definition:**
 - **EMPTY:** nur leeres Element erlaubt
 - **#PCDATA (Parsed Character DATA):** Freitext
 - **(a,b,c):** Liste der Sub-Elemente (a,b,c sind die Namen der Sub-Elemente die in dieser Reihenfolge vorkommen müssen)
 - **(a|b|c):** Liste alternativer Sub-Elemente (Auswahl)
 - **Jedes Sub-Element kann eine Kardinalitätsspezifikation besitzen:**
 - * Element kommt kein mal oder beliebig oft vor
 - + Element kommt mindestens einmal vor (oder öfter)
 - ? Element kommt kein mal oder einmal vor
 - Ohne Angabe: Element kommt genau einmal vor

Beispiel

- DTD Fragment

```
<!ELEMENT person (name, age?, size?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT size (#PCDATA)>
```

- XML Document

```
<person>
  <name> Fritz </name>
  <age> 24 </age>
  <size> M </size>
</person>
```

Sonstige XML Sprachkonstrukte

- Objekt-Referenzen, z.B.

```
&lt; &gt; &amp; &apos; &quot;&uuml;&tuclogo;
```

- Kommentare

```
<!-- dies ist ein Kommentar -->
```

- Anweisungen

- dienen zur Übergabe von Information an Applikationen, z.B.

```
<?XML VERSION=1.0 RMD="NONE" ENCODING="UTF-8"?>
```

- allgemeine Form:

```
<?name parameterliste?>
```

- CDATA Sektionen

- Zeichen in CDATA Sektionen werden vom XML-Prozessor ignoriert; z.B. Programm-Listings:

```
<! [CDATA[ *p = &q;
          b = ( I <= 3 ); ]]>
```

Eigenschaften von XML-Dokumenten: Wohlgeformtheit und Gültigkeit

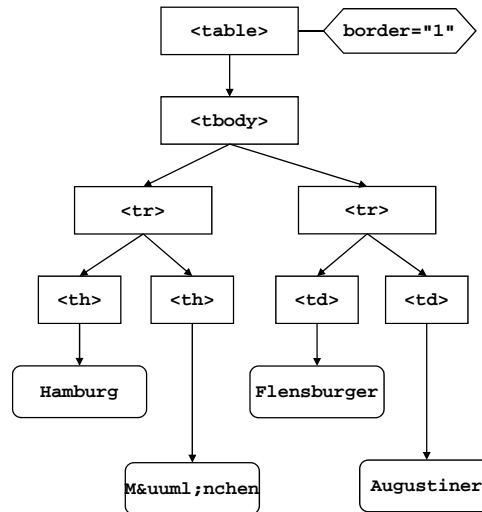
- **Strikte Sprachdefinition erlaubt Test auf wichtige Eigenschaften**
 - Wohlgeformtheit (bzgl. der Syntax von XML)
 - Gültigkeit (bezüglich einer DTD)
- **XML-Dokument heißt wohlgeformt, wenn:**
 - Es mit der Anweisung `<?xml ...>` beginnt
 - das gesamte Dokument in ein einzelnes Wurzelement eingeschlossen ist
 - Alle unbedingt erforderlichen Attribute angegeben sind
 - Alle Attribute den richtigen Wertebereich / Typ haben
 - Alle eingesetzten Elemente korrekt ineinander verschachtelt sind
- **XML-Dokument heißt gültig (bzgl. einer DTD), wenn:**
 - das Dokument ist wohlgeformt
 - eine zugehörige interne oder externe DTD existiert und ist verfügbar
 - das Dokument entspricht den in der DTD ausgestellten Regeln

Interne Darstellung eines XML-Dokuments

- **Darstellung als Baumstruktur**
- **Knoten sind Elemente**
- **Kanten: Beziehungen zwischen Elementen**
 - Nachfolger ("Kind")
 - Vorgänger ("Elternknoten")
- **Existenz eines "Wurzelknoten",**
- **Nachfolger ("Kinder") eines Knotens sind die in ihm enthaltenen Elemente**
- **Attribute sind dem ihr Element entsprechenden Knoten zugeordnet**
- **XML-Anwendungen (Parser, XSLT) arbeiten auf dieser Baumstruktur**
- **Standard-Modell: (www.w3.org/DOM)
DOM (Document Object Modell) mit Implementierungen z.B. für Java**

Beispiel: DOM Darstellung eines XML-Dokuments

```
<table border="1">
  <tbody>
    <tr>
      <th>Hamburg</th>
      <th>M&uuml;nchen</th> </tr>
    <tr>
      <td>Flensburger</td>
      <td>Augustiner</td>
    </tr>
  </tbody>
</table>
```



XML Schema

- Schwäche von XML: Es gibt keine Datentypen, es gibt nur STRINGS

```
<ITEM>
  <PRICE>5.95</PRICE>
</ITEM>
```

Keine Möglichkeit, auszudrücken, dass hier nur Zahlen erlaubt sind

- XML Schema:
 - Standard entwickelt vom W3C (siehe : www.w3.org/XML/Schema)
- Standardisierte Beschreibung komplexerer (auch zusammengesetzter) Datenformate
 - Vordefinierte Datentypen
 - Untertypenbildung
 - Namensräume
- XML Schema beschreibt ein XML Dokument
 - ähnliche Funktion wie die DTD, aber mehr Struktur
 - ist selbst ein XML Dokument (Meta-Beschreibung) und keine DTD

Einfache Datentypen

- **Standard-Datentypen sind vordefiniert z.B.:**
 - string, integer, positiveInteger, boolean,
 - time, date, duration,
 - ID und IDREF (aus XML)
- **Datentypen können eingeschränkt werden, z.B.:**
 - length, minLength, maxLength (z.B. Länge von Strings)
 - minInclusive, maxInclusive: Wertebereiche bei numerischen Typen
 - enumeration: Definition von Aufzählungstypen (basierend auf string)
- **Definiert durch Schema-Elemente:**
`<xsd:simpleType>` und `<xsd:restriction>`

Syntaxbeispiele

```
<xsd:simpleType name="sizeType">  
  <xsd:restriction base="xsd:string">  
    <xsd:enumeration value="S" />  
    <xsd:enumeration value="M" />  
    <xsd:enumeration value="L" />  
  </xsd:restriction>  
</xsd:simpleType>
```

Aufzählungstyp
{S, M, L}

```
<xsd:simpleType name="ageType">  
  <xsd:restriction base="xsd:integer">  
    <xsd:minInclusive value="0" />  
    <xsd:maxInclusive value="150" />  
  </xsd:restriction>  
</xsd:simpleType>
```

Range [0...150]

Komplexe Datentypen

- Definiert durch Schema-Elemente: `<xsd:complexType>`
- Definition von verschachtelten XML-Elementen:
 - `<xsd:sequence>` : Folge von Elementen
 - `<xsd:choice>` : Auswahl aus einer Menge von Elementen
 - `<xsd:all>` : Menge von Elementen, Reihenfolge egal
 - `<xsd:group>` : Definition und Referenz auf wiederverwendbare Gruppe von Elementen
- Innerhalb dieser Schemaelemente werden lokale XML-Elemente definiert durch:
 - `<xsd:element name="...Elementname..." type="...Typname..." minOccurs="...Zahl..." maxOccurs="...Zahl..." />`

Kardinalität: untere Schranke

Kardinalität: obere Schranke

Syntaxbeispiele

```
<xsd:complexType name="personType">  
  <xsd:sequence  
    <xsd:element name="name" type="xsd:string" />  
    <xsd:element name="age" type="ageType" minOccurs="0"/>  
    <xsd:element name="size" type="sizeType" minOccurs="0"/>  
  </xsd:sequence>  
</xsd:complexType>
```

Komplexer Typ

```
<xsd:element name="person" type="personType" />
```

Top-Level Element für Instanz

Instanz des Schemas:

```
<person>  
  <name> Fritz </name>  
  <age> 24 </age>  
  <size> M </size>  
</person>
```