9. Web-Dokumente

Medieninformatik-Buch:
Kapitel 10

Weiterführende Literatur:

M. Knobloch, M. Kopp: Web-Design mit XML, dpunkt-Verlag 2001
H. Vonhoegen: Einstieg in XML, Galileo Computing 2009

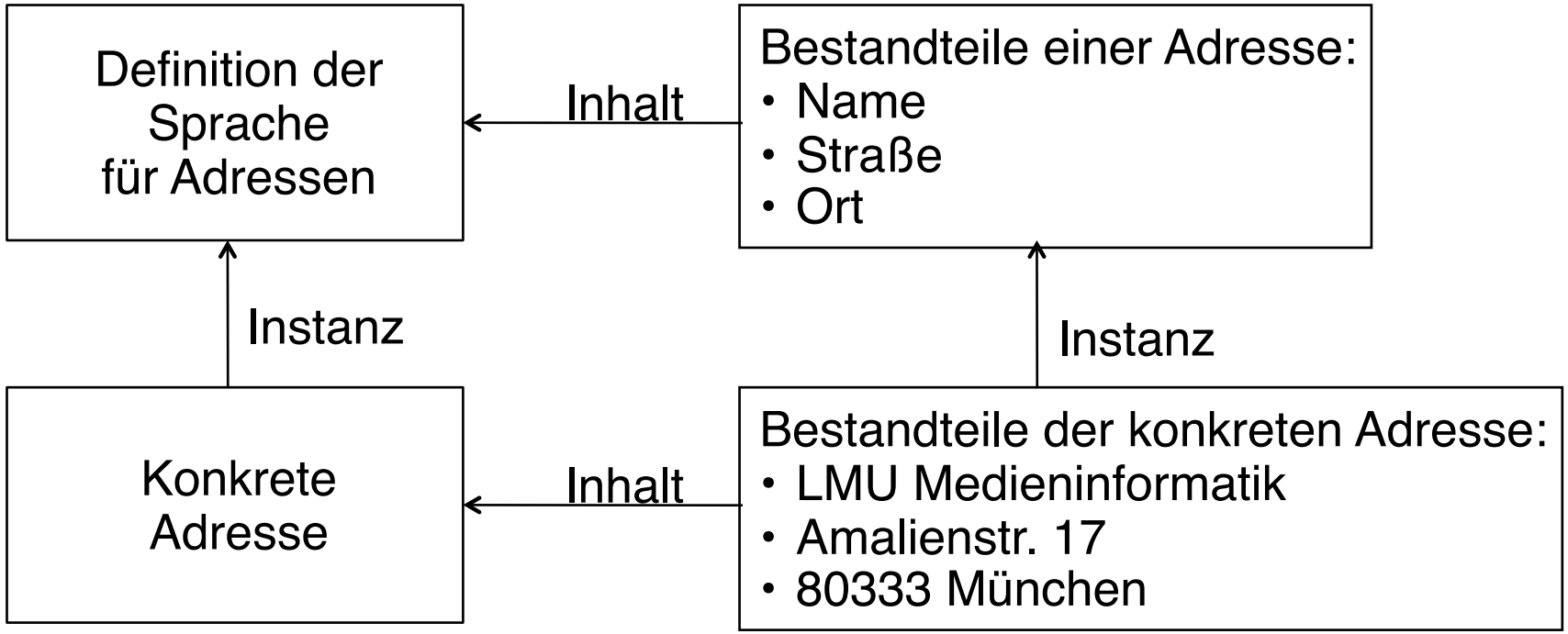
http://de.selfhtml.org/xml/

Generische Auszeichnungssprache: Idee

- Auszeichnungssprache *(markup language)*:
  - Text und eingebettete textuelle Zusatzinformation (insbesondere zur Darstellung)
  - Tag-/Attribut-Syntax weithin bekannt durch HTML
- Idee: „Familie“ von Auszeichnungssprachen gleicher Basissyntax für verschiedenste Anwendungsgebiete
  - Web-Seiten-Formatierung (HTML)
  - Strukturierte Daten, z.B. Adressen, Briefe, Texttypen
  - Austauschformate für Textverarbeitung und andere Software
  - Standardformate für Grafik, Multimedia-Präsentationen, ...
- Vorteile:
  - ***Trennung von Inhalt, Struktur und Präsentation***
  - Lesbarkeit durch Mensch und Maschine
  - Automatische syntaktische Überprüfungen
  - Erweiterbarkeit durch Definition neuer Tags/Attribute
- Nachteil: Lange Texte

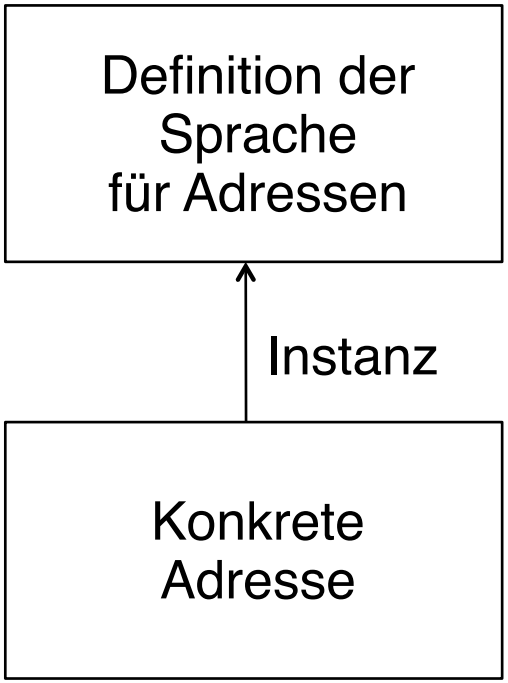
Sprachenbeschreibung und Sprachenverwendung

Ebene der Sprachenbeschreibung (= Metaebene)



Ebene der Sprachenverwendung

Mataebenen

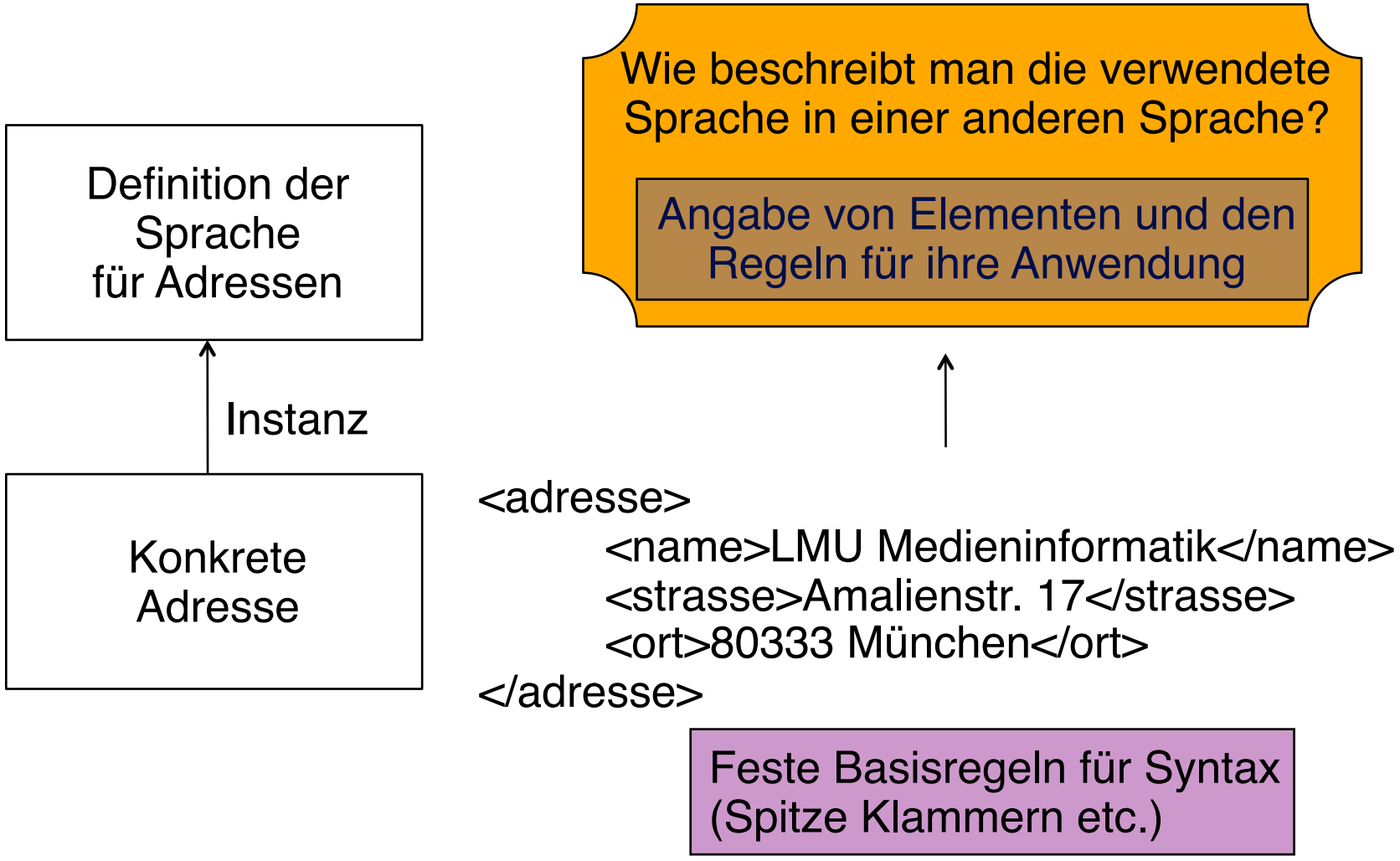


Wie beschreibt man die verwendete Sprache in einer anderen Sprache?

```
<adresse>
        <name>LMU Medieninformatik</name>
        <strasse>Amalienstr. 17</strasse>
        <ort>80333 München</ort>
</adresse>
```

Beispiel für eine relativ allgemeine Metasprache: Backus-Naur-Form (BNF)

Vereinfachung durch Sprachfamilie



Geschichte: SGML, HTML, XML, XHTML

- 1967: GenCode-Komitee
  - Norman Scharpf, Trennung Inhalt-Layout
- 1969: Goldfarb, Mosher, Lorrie (IBM):
  - Generalized Markup Language GML
- 1978: ISO-Standard 8879
  - Standard Generalized Markup Language SGML
  - Erlaubt Definition beliebiger *Dokumenttypen*
  - Sehr komplex, Verbreitung vorwiegend im akademischen Bereich und in der Definition weiterer Standards
- 1989: Berners-Lee, Cailleau
  - Hypertext Markup Language HTML
  - HTML ist ein spezieller Dokumenttyp von SGML
- 1998: WWW Consortium (W3C)
  - eXtensible Markup Language (XML)
  - Teilsprache von SGML
- 1999: Reformulierung von HTML als XML-Dokumenttyp
  - eXtensible Hypertext Markup Language XHTML
  - Etwas strengere Syntax als übliches HTML

Beispiele von XML-Anwendungen

- *XML-Anwendung* = Definition eines XML-Dokumenttyps für einen bestimmten Zweck
- Grafik-Sprachen:
  - SVG (Scalable Vector Graphics): 2D-Vektorgrafik
  - X3D: 3D-Vektorgrafik, Fortführung von VRML
- Ablage- und Austauschformat für Bürosoftware:
  - Open Document Format (odt, ods, odp, odb, odg, odf)
  - Office Open XML (Microsoft/ECMA), in MS Office seit Office 2007
- VoiceXML
  - Dialogbeschreibung für natürlichsprachige Dialoge
- MusicXML
  - Austauschformat für Musiknoten (westliche Musiknotation)
- Diverse Gebiete der Naturwissenschaften:
  - MathML (für mathematische Formeln)
  - CML (für chemische Formeln)
  - BSML (Bioinformatic Sequence Markup Language)
- ...

XML als Basistechnologie

- Netzdienste:
  - News-Feeds mit RSS (Really Simple Syndication) basieren auf XML
    - » siehe später
  - iTunes-Podcasts sind RSS-Feeds (also XML)
  - Messaging-Protokoll XMPP basiert auf XML
  - ...
- Software:
  - "Preference"-files in MacOS X
  - diverse proprietäre Dateiformate

XML-Dokumente

- Prolog:
  ```
  <?xml version="1.0"?>
  ```
- Dokumenttyp:
  ```
  <!DOCTYPE Typname SYSTEM "Dateiname.dtd">
  ```
- Tag- und Attribut-Syntax wie in HTML
  - Jedes geöffnete Tag muss explizit geschlossen werden.
    ```
    <xy> ... </xy>
    ```
  - Leere Tags müssen mit „/>“ enden
    ```
    <br/>
    ```
  - Jedes XML-Dokument hat genau ein Wurzel-Element *(root)*
  - Strenge hierarchische Schachtelung von Tags
  - Attributwerte immer in doppelten Anführungszeichen
    ```
    <xy a="..."> ... </xy>
    ```
  - Keine doppelten Attributwerte
  - Groß- und Kleinschreibung wird unterschieden

Document Type Definition (DTD)

- Festlegung der zulässigen Werte für Tags, Attribute etc. in den zugehörigen XML-Dokumentdateien
- Meist in separater Datei, kann aber auch Bestandteil eines XML-Dokuments sein
- Zwei verschiedene Syntax-Alternativen:
  - klassische DTD-Syntax (hier beschrieben)
  - "XML Schema" (siehe später)
- Wichtigste Deklarationen in DTDs:
  - ELEMENT: Element (Dokument-Tag)
  - ATTLIST: Attributliste für ein Element
  - ENTITY: Abkürzung für komplexes Element
  - NOTATION: Datentyp-Deklaration
- Eingebaute Datentypen in XML:
  - PCDATA (Parsed Character Data) - vom Parser analysierte Zeichenreihe
  - weitere Datentypen zum Einsatz z.B. in Attributen

ELEMENT-Deklaration in DTD

- Syntax:

  `<!ELEMENT` ***`Elementname`*** **`(`** ***`Inhaltsbeschreibung`*** **`)>`**

- *Elementname:*
  - Name muss mit Buchstaben oder Unterstrich beginnen
  - Groß- und Kleinschreibung wird unterschieden
- *Inhaltsbeschreibung:*
  - Struktur des Inhalts zwischen Start- und End-Tag
  - Bezieht sich auf weitere Elemente oder eingebaute Datentypen
  - Reguläre Ausdrücke:

Komma-Liste:	Sequentielle Abfolge
A | *B* :	Alternativen
A ?	Optional
A +	Mindestens einmal
A *	Beliebig oft

Beispiel: Dokumenttyp „Folien“ V.1

- Eine (sehr einfache) Folie einer Präsentation hat folgende Bestandteile
  - Titel: Zeichenreihe
  - Liste von Themen (d.h. Aussagen auf der Folie)
    - » Thema: Zeichenreihe

- Als DTD formal notiert:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT folie (titel, themenliste)>
<!ELEMENT titel (#PCDATA)>
<!ELEMENT themenliste (thema*)>
<!ELEMENT thema (#PCDATA)>
```

Beispiel in Anlehnung an Knobloch/Popp

Beispiel: Dokument des Typs „Folien“ V.1

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE folie SYSTEM "folien1.dtd">
<folie>
    <titel>XML-Übersicht</titel>
    <themenliste>
        <thema>Was sind DTDs?</thema>
        <thema>Struktur einer XML-Datei</thema>
    </themenliste>
</folie>
```

folien1.xpr

Wohlgeformtheit und Gültigkeit

- Ein XML-Dokument ist *wohlgeformt (well-formed)*, wenn es den allgemeinen Regeln der XML-Syntax genügt.
  - Beispiel (*well-formed*, aber nicht *valid*):

```
<folie>
    <ttel/>
    <tlist>
        <thm>Was sind DTDs?</thm>
        <tha>Struktur einer XML-Datei</tha>
    </tlist>
</folie>
```

- Ein XML-Dokument ist *gültig (valid)*, wenn es der angegebenen Document Type Definition entspricht.
- *Erweiterbarkeit:*
  - Anwendungen erzwingen Gültigkeit oft nicht
  - z.B. zusätzliche herstellerspezifische Tags – werden im Zweifelsfall ignoriert

folien1.xpr

Spezielle Inhaltsmodelle für ELEMENT

- ANY:
  - Erlaubt beliebige Zeichenreihen
  - Einschließlich Markup, d.h. Verwendung von Tags in der Zeichenreihe
  - Beispiel:

```
<!ELEMENT titel ANY>
<titel><thema>XML-Übersicht</thema></titel>
```

- EMPTY:
  - Verlangt leeren Inhalt
- Gemischte Daten (Literal und Element):
  - PCDATA-Angabe muss immer vorne stehen, z.B. in

```
<!ELEMENT titel (#PCDATA | thema)*>
```

ATTLIST-Deklaration in DTD

- Syntax:
  <!ATTLIST *Elementname Attributdefinition*+ >
- *Attributdefinition*:
  ***Attributname Attributtyp Standardwert* [*Festwert*]**
- *Attributtyp:*
  - Angabe eines Datentyps
    - » CDATA: Character Data, d.h. Zeichenreihe (nicht analysiert)
    - » ID: Eindeutiger Bezeichner für Verweise im Dokument
    - » IDREF: Verweis auf einen Bezeichner (vom Typ ID)
  - Explizite Werteliste (ohne Anführungszeichen!)
    - » (*Wert1* | *Wert2* | ....)
- *Standardwert:*
  - Angabe eines konkreten Werts: Default-Wert, Attribut ist optional
  - #IMPLIED: Attribut ist optional ohne Defaultwert
  - #REQUIRED: Attribut muss angegeben werden
  - #FIXED: Attribut muss immer mit dem selben Wert angegeben werden (der als *Festwert* angegeben ist)

Beispiel: Dokumenttyp „Folien“ V.2

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT folien (folie*)>
<!ATTLIST folien sprache (de | en)  "de">
<!ELEMENT folie (titel, themenliste)>
<!ATTLIST folie erstellt CDATA #REQUIRED>
<!ATTLIST folie autor CDATA #IMPLIED>
<!ATTLIST folie ident ID #REQUIRED>
<!ATTLIST folie sieheAuch IDREF #IMPLIED>
<!ELEMENT titel ANY>
<!ELEMENT themenliste (thema*)>
<!ELEMENT thema (#PCDATA)>
```

folien2.xpr

Beispiel: Dokument des Typs „Folien“ V.2

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE folien SYSTEM "folien2.dtd">
<folien sprache="en">
    <folie erstellt="04.06.2003" ident="f1">
        <titel>Attribute in XML</titel>
        <themenliste>
            <thema>Deklaration in DTD</thema>
            <thema>Verwendung in XML-Dokument</thema>
        </themenliste>
    </folie>
    <folie erstellt="03.06.2003" ident="f2">
        <titel>Identifikatoren</titel>
        <themenliste>
            <thema>Eindeutigkeit</thema>
        </themenliste>
    </folie>
</folien>
```

folien2.xpr

Elemente vs. Attribute

- Zusatzinformation zu Elementen als Unterelemente oder Attribute?

```
<buch>
   <isbn>3-932588-96-7</isbn>
   <autor>Fritz Müller</autor>
   ...
</buch>
```

oder

```
<buch
   isbn="3-932588-96-7"
   autor="Fritz Müller">
   ...
</buch>
```

- Unterelemente
  - sind leichter zu lesen
  - ermöglichen Wiederholungen
  - können weiter in Elemente untergliedert werden
  - können auch in anderem Kontext genutzt werden
- Attribute
  - sind leicht zu überprüfen
  - erlauben keine Wiederholungen
  - verringern die Hierarchietiefe
  - binden die Zusatzinformation eng an das zugehörige Element

Abkürzungen mit (internen) Entities

- (Interne) Entity
  - Abkürzungsmechanismus für XML-Abschnitte
  - Paar Name – Inhalt

- Allgemeine Entities - im Dokument benutzte Abkürzungen
  - Syntax (Definition):

    ```
    <!ENTITY Entityname Entityinhalt >
    ```

  - Syntax (Verwendung):

    ```
    &Entityname;
    ```

- Parameter-Entities - in der DTD benutzte Abkürzungen
  - Syntax (Definition):

    ```
    <!ENTITY % Entityname Entityinhalt >
    ```

  - Syntax (Verwendung):

    ```
    %Entityname;
    ```

Beispiele für Entities

- Allgemeine Entities (Abkürzungen im Dokument):

– DTD:

```
<!ENTITY einleitung "<thema>Einleitung</thema>">
<!ENTITY schluss "<thema>Zusammenfassung und
   Ausblick</thema>">
```

– XML-Dokument:

```
<themenliste>
     &einleitung;
  <thema>Kapitel 1</thema> ...
</themenliste>
```

- Parameter-Entities (Abkürzungen innerhalb der DTD):

– DTD:

```
<!ENTITY % _autor "Heinrich Hussmann">
<!ATTLIST folie autor CDATA "%_autor;">
<!ENTITY stdFusszeile "%_autor;">
```

folien3.xpr

Namensräume *(Namespaces)* (1)

- Mischen von XML-Information, die mehreren verschiedenen DTDs entspricht?
  - Mehrdeutige Tags?
- Namespace-Deklaration
  - Syntax:

    ```
    <Tagname xmlns:Namensraumname="URI" ...>
    ```

  - Definiert frei gewählten *Namensraumnamen*
    - » *URI* definiert den Urheber des Namensraums.
    - » Namensraum verwendbar in untergeordneten Dokumentteilen
    - » Deshalb meist bei Wurzel-Tags angegeben
  - *Namensraumname* wird als *Präfix* verwendet:

    ```
    Namensraumname : Tag
    ```

    - » Unterscheidung von evtl. gleichnamigen anderen Tags

Namensräume *(Namespaces)* (2)

- Default-Namensraum
  - gilt für Tags ohne Präfix
  - Deklariert durch

```
<Tagname xmlns="URI" ...>
```

  - Beispiel:

```
<html xmlns="http://www.w3.org/1999/xhtml" ...>
```

- Leider unbefriedigene Integration von Namespaces und DTDs!
  - Attribute und Elemente werden immer nur gegen eine DTD validiert!
  - Netzzugriff auf definierende DTD (gemäß URI) findet nicht statt

Namespaces: Beispiel (aus SVG)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE svg ... >
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">

  ... SVG-Inhalte ...
  <a xlink:href="http://www.medien.ifi.lmu.de">
     <circle cx="50" cy="50" fill="blue" r="20"/>
  </a>
  ...
</svg>
```

folien5.xpr

XML Schema: Idee

- Ersatz von XML DTDs durch ein spezielles XML-Dokumentformat
  - Ermöglicht Formen der reflexiven Definition: Schemasprache in sich selbst definierbar
  - Erlaubt homogene Werkzeuge für Schemata und Dokumente
- Datentypkonzept
  - Vielzahl eingebauter primitiver Datentypen (z.B. Zahlen)
  - Strukturierte Datentypen *(complex datatype)*
- Strukturierter Aufbau
  - Vererbung auf Schema-Ebene
- Verbesserte Dokumentationsunterstützung

- Leider: Sehr komplex
  - Mehrere hundert Seiten Spezifikation!
  - Deshalb Fortschritt im praktischen Einsatz nur sehr langsam

Beispiel: XML Schema Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>
  <xs:element name="folie" type="folie_type"/>
  <xs:element name="titel" type="xs:string"/>
  <xs:element name="thema" type="xs:string"/>
  <xs:complexType name="folie_type">
     <xs:sequence>
          <xs:element ref="titel"/>
          <xs:element ref="thema" maxOccurs="unbounded"/>
     </xs:sequence>
  </xs:complexType>
</xs:schema>
```

folienSchema.xpr

Beispiel: XML-Dokument basierend auf Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<folie xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance"
 xsi:noNamespaceSchemaLocation="folie1.xsd">
 <titel>XML-Übersicht</titel>
 <thema>Was sind DTDs?</thema>
 <thema>Struktur einer XML-Datei</thema>
</folie>
```

9. Web-Dokumente

9.1 Generische Auszeichnungssprachen: XML
9.2 XML und Style Sheets
9.3 XML für Multimedia: SMIL
9.4 XML für Web-Informationsdienste: RSS
9.5 Ausblick: XML Transformationen (XSLT)

Medieninformatik-Buch:
Kapitel 10

Weiterführende Literatur:

M. Knobloch, M. Kopp: Web-Design mit XML, dpunkt-Verlag 2001
H. Vonhoegen: Einstieg in XML, Galileo Computing 2009

http://de.selfhtml.org/xml/

CSS und XML

- XML-Dateien enthalten „reinen Inhalt“ (gemäß gegebener Struktur)
  - Zur Anzeige z.B. im Browser Zusatzangaben nötig
- Alternative Wege von einer XML-Datei zu einer Browseranzeige:
  - Cascading Style Sheets
  - Transformation in HTML-Text (z.B. mit Transformationssprache XSLT)
  - Kombination beider Ansätze

- Cascading Style Sheets
  - Separate Datei(en) mit Formatierungsangaben
  - Anbindung an XML-Dokument durch eine sogenannte *processing instruction* (PI):

```
<?xml-stylesheet type="text/css" href="folienstyle.css"?>
```

Beispiel: CSS-Datei für Dokumenttyp „Folien“ V.2

```
folie {
    font-family:sans-serif
}

titel {
    display:block; padding-top:10pt;
    font-size:200%; font-weight:bold; color:blue
}

thema {
    display:block; padding-left:30pt; padding-top:10pt;
    font-size:150%
}
```

folienstyle.xpr

Attribute in XML

- Deklaration in DTD
- Verwendung in XML-Dokument

Identifikatoren

Fortgeschrittene Konzepte in CSS

- Pseudo-Formate:
  - z.B. **`display:none`** zum Ausblenden (nicht darstellen)
- Pseudo-Elemente:
  - z.B. **`:first-letter`**, **`:first-line`** zur speziellen Formatierung von Textteilen
  - z.B. **`:before`**, **`:after`** zum Modifizieren von Texten bei der Anzeige
- Pseudo-Klassen
  - z.B. **`:hover`**, **`:focus`**, **`:active`** zur Darstellung abhängig von Benutzeraktionen
- Kontextabhängige Formatierung
  - z.B. für Elemente abhängig von bestimmten Attributwerten
  - z.B. für Unterelemente abhängig von den im Dokument vorhandenen Oberelementen
- Strukturierung von Formatierungsinformation
  - Vererbung und verschiedene Formen zur Einbindung von Stylesheets

folien2.xpr/folie5style.xml

9. Web-Dokumente

Weiterführende Literatur:

Dick Bulterman, Lloyd Rutledge: SMIL 3.0 – Interactive Multimedia for the Web, Mobile Devices and Daisy Talking Books, Springer 2008

SMIL - Idee und Geschichte

- Synchronized Multimedia Integration Language (gesprochen: "Smile")
- Standardsprache für die koordinierte Kombination von zeitabhängigen Medien zu einer Multimedia-Präsentation
  - zeitliche Abhängigkeiten im Ablauf
  - berücksichtigt auch nicht-zeitabhängige Medientypen (Text, Standbild)
  - auch geeignet für "Streaming", d.h. kontinuierliches Laden von Mediendaten über das Netz
- Standardisierung durch W3C (WWW Consortium)
  - Erster Entwurf November 1997
  - SMIL 1.0 Standard Juni 1998
  - ab 1998: Implementierungen u.a. durch Real, Apple, ambulantplayer.org
  - 1999: Pläne für eine erweiterte und verbesserte Fassung ("Boston SMIL")
  - SMIL 2.0 Standard August 2001
  - SMIL 2.1 Recommendation Dec. 2005 (z.B. Profil für mobile Endgeräte)
  - SMIL 3.0 Recommendation Dec. 2008

Grundstruktur einer SMIL-Datei

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
     <layout>
        <root-layout width="356" height="356"
                backgroundColor="black"/>
        <region id="imgReg" width="256" height="256"
                left="50" top="50"/>
     </layout>
  </head>
  <body>
     <seq>
        <img region="imgReg" src="tiger.jpg" dur="4s"/>
        <img region="imgReg" src="elephant.jpg" dur="4s"/>
        <img region="imgReg" src="butterfly.jpg" dur="4s"/>
      </seq>
  </body>
</smil>
```

Spatiale Struktur (Layout)

Temporale Struktur (Ablauf)

Beispiel: Multimediale Diashow (1)

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <root-layout width="356" height="356"/>
      <region id="brush_region" z-index="1"/>
      <region id="img_region" width="256" height="256"
          left="50" top="50" z-index="2"/>
    </layout>
    <transition id="img_wipe" type="barWipe"
          dur="3s"/>
    <transition id="bkg_wipe" type="barWipe"
          direction="reverse" dur="3s"/>
  </head>
```

slideshow.smil

Beispiel: Multimediale Diashow (2)

```
...
  <body>
     <par>
        <seq>
          <img region="img_region" src="....jpg" ...
             transIn="img_wipe" fill="transition"/>
          ...
        </seq>
        <seq>
          <brush color="green" region="brush_region"
             ... transIn="bkg_wipe" fill="transition"/>
        </seq>
        <audio src....mp3"  end="32s"/>
     </par>
  </body>
</smil>
```

9. Web-Dokumente

Weiterführende Literatur:

Jörg Kantel: RSS und Atom kurz und gut, O'Reilly 2007
http://cyber.law.harvard.edu/rss/rss.html

Really Simple Syndication RSS

- *Syndikation:* Zusammenführen und Integrieren von Informationen (Nachrichten) aus verschiedenen Quellen
- RSS:
  - XML-basiertes Format für Nachrichtenquellen im Internet
  - 1997 von der Firma Userland definiert
  - 1999: my.Netscape.com ("RDF/Rich Site Summary")
  - Heute meistverwendetes Format für Nachrichten, Weblogs, Podcasts
- Konkurrenzformat: *Atom Syndication Format* (ASF) (ebenfalls XML)
- Grundstruktur:
  - *Channel* ist Liste von *Items*
  - Jedes *Item* ist durch einen *Globally Unique Identifier* (guid) definiert, meist ein Link
  - Umfangreiche Möglichkeiten für *Metadaten* zu Items und Channels
- Beispiel:
  - Ein *Podcast* ist in der Regel eine RSS-Datei

Beispiel: RSS Feed zu einer Vorlesung (1)

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">

<channel>
  <title>Digitale Medien News Wintersemester 10/11</title>
  <link>http://www.medien.ifi.lmu.de/lehre/ws1011/dm/</link>
  <description>Site Updates und Mitteilungen für die Vorlesung
  Ditigale Medien im Wintersemester 10/11</description>
  <language>de</language>
  <pubDate>
     Fri, 01 Oct 2010 12:29:32 GMT
  </pubDate>
  <lastBuildDate>
     Tue, 25 Jan 2011 18:37:59 GMT
  </lastBuildDate>
  <docs>http://blogs.law.harvard.edu/tech/rss</docs>
  <managingEditor>xyz@ifi.lmu.de</managingEditor>
  <webMaster>xyz@ifi.lmu.de</webMaster> …
```

Beispiel: RSS Feed zur Vorlesung (2)

```
    <item>
        <title>Übungsblatt 9 verfügbar</title>
        <link>http://www.medien.ifi.lmu.de/lehre/ws1011/dm/</link>
        <description>
                <![CDATA[
                        Das neunte Übungsblatt kann nun heruntergeladen werden.
                        Die Abgabe muss spätestens bis zum 19.01.2011 um 14:00
                        erfolgen.
                ]]>
        </description>
        <pubDate>
                Mon, 10 Jan 2011 11:22:18 GMT
        </pubDate>
        <guid>
                http://www.medien.ifi.lmu.de/lehre/ws1011/dm/index.rss2
        </guid>
    </item>
    …
</channel>
</rss>
```

Beispiel: dm_podcast.rss (Auszug)

```
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd"
   version="2.0">
  <channel>
    <title>Vorlesung Digitale Medien Wintersemester 2010/11</title>
    <itunes:author>Heinrich Hussmann, LMU</itunes:author>
    ...
    <item>
      <title>Informationstheorie, Codierung Teil I</title>
      <description>Es wird eine Einführung  ...</description>
      <guid>http://www.medien.ifi.lmu.de/team/
            heinrich.hussmann/files/dm2a.m4b</guid>
      <enclosure url="http://www.medien.ifi.lmu.de/team/
            heinrich.hussmann/files/dm2a.m4b"
            length="23424928" type="audio/x-m4a"></enclosure>
      <pubDate>Fri, 22 Oct 2010 22:30:00 +0200</pubDate>
      <itunes:explicit>no</itunes:explicit>
      <itunes:duration>01:25:44</itunes:duration>
    </item>
...
```

9. Web-Dokumente

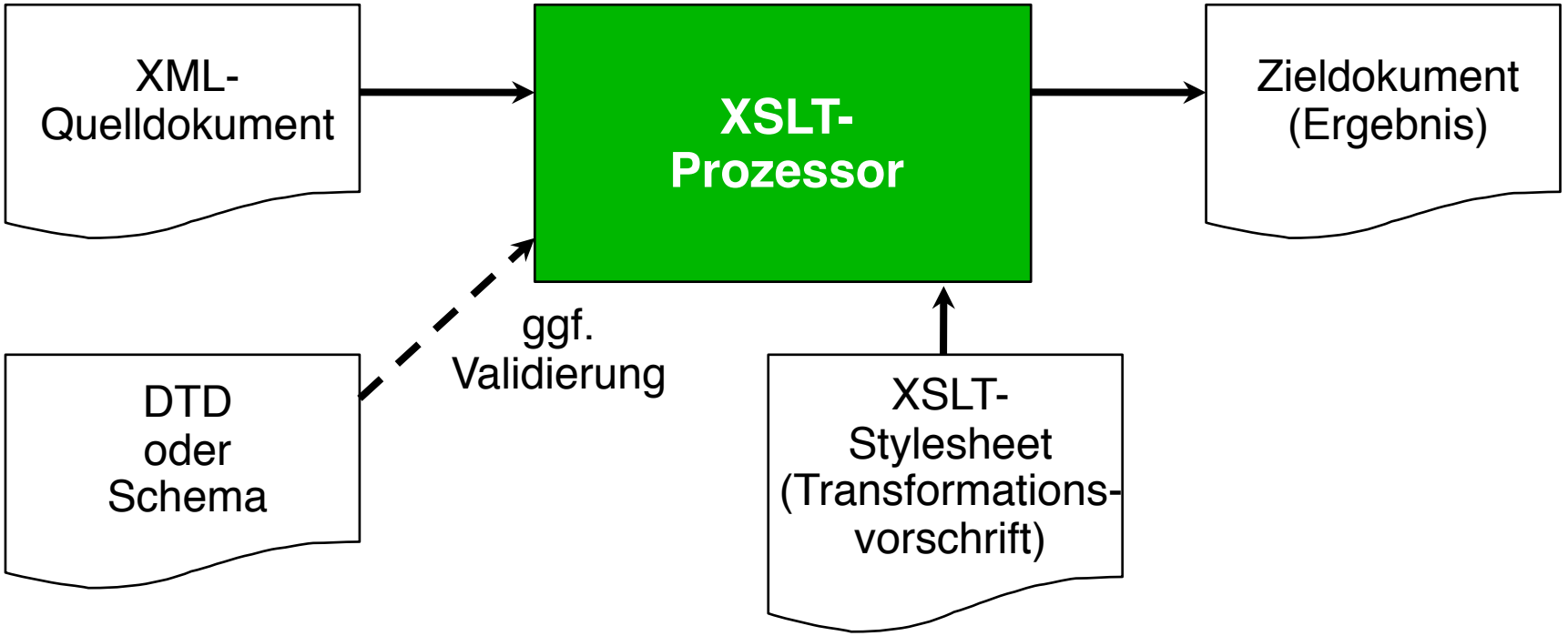
Weiterführende Literatur:

Frank Bongers: XSLT 2.0 und XPath 2.0, Galileo Press, 2. Aufl. 2008

Stylesheets, CSS und XSL

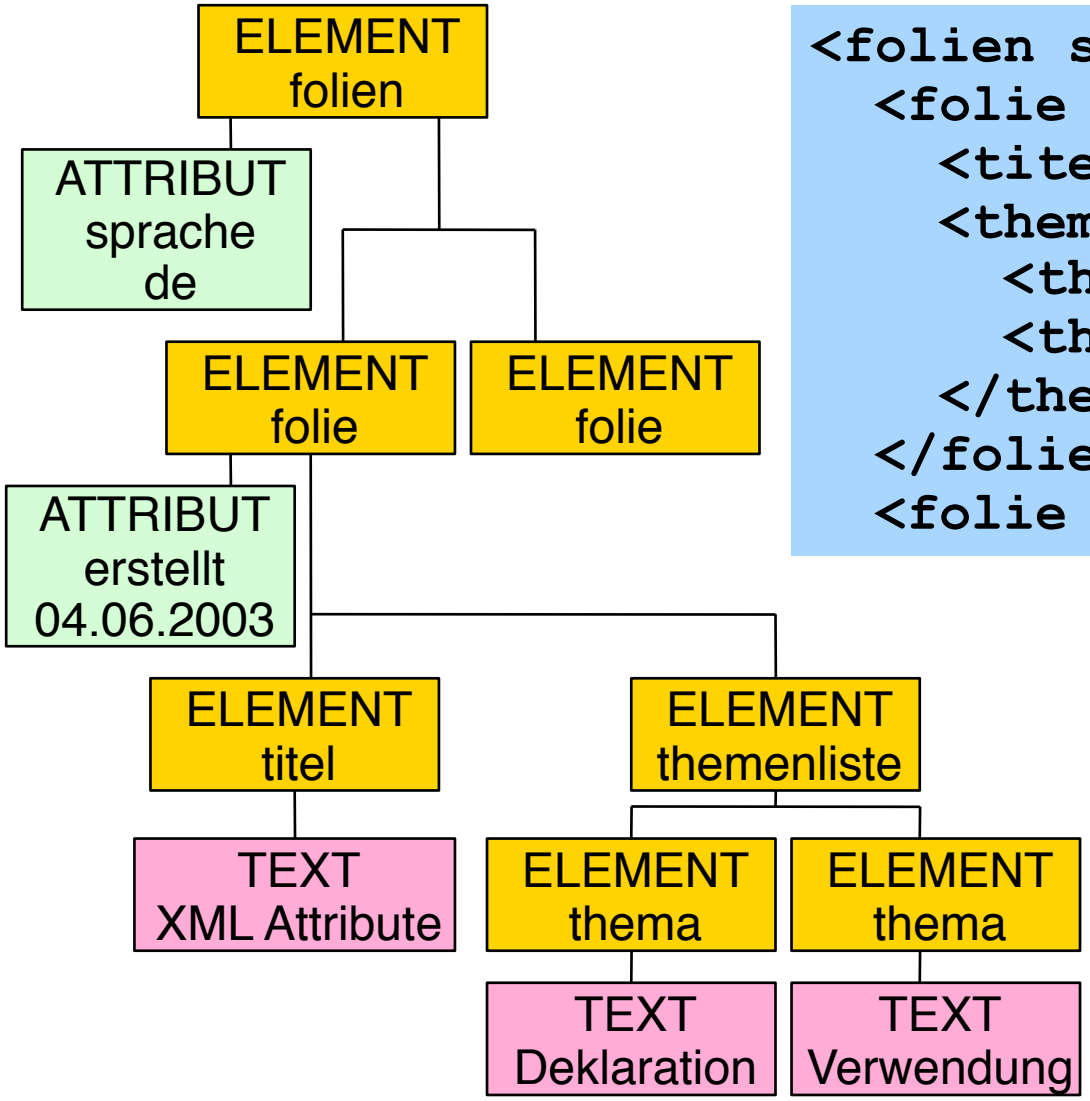
- Zweck von XML + Stylesheets:
  - Trennung des logischen Inhalts von der Präsentation
  - Flexibilität bezüglich der Darstellung auf verschiedenen Plattformen
  - Konsistenzsicherung bei mehrfach dargestellter Information
- Problematische Aspekte von klassischen "Cascading Style Sheets":
  - Verwendet spezielle Syntax ("properties") statt XML
  - *Struktur* der Präsentation muss Struktur des Inhalts folgen
    - » Schwierig: Auslassungen, Reihenfolgeänderungen, Mehrfachdarstellung
  - Keine gute Unterstützung für Druckmedien bzw. entsprechende Darstellung:
    - » Paginierung, Spalten, Kästen, Inhaltsverzeichnis, Index
- eXtensible Style Sheet Language XSL:
  - XSL Formatierungssprache (oft XSL Formatting Objects, XSL-FO genannt)
  - XSL Transformations
  - XPath Navigationssprache

Transformation mit XML



- Mögliche Ergebnistypen:
  - XML-Baum
  - HTML-Baum
  - Text

XML-Dateien als Baumstruktur




```
<folien sprache="de">
  <folie erstellt="04.06.2003">
    <titel>XML Attribute</titel>
    <themenliste>
      <thema>Deklaration</thema>
      <thema>Verwendung</thema>
    </themenliste>
  </folie>
  <folie ...> ...
```

folientrans.xpr

Knotenarten (Auswahl)

- Wurzel-Knoten *(root node)*:
  - Ausgangspunkt des Dokuments
  - Kinder: Dokument-Element, Processing Instructions
- Element-Knoten *(element node)*:
  - Entspricht *tag* im Dokument
  - Name = Tag-Name, evtl. Attribut-Knoten vorhanden
  - Kinder: Element-Knoten, Text-Knoten
- Attribut-Knoten *(attribute node)*:
  - Spezielle Art der "Verwandtschaft" zum Element-Knoten
  - Name = Attribut-Name, Wert = Attribut-Wert, keine Kinder
- Text-Knoten *(text node)*:
  - Zeichenkette aus dem Stylesheet-Dokument
  - kein Name, Wert = Zeichenkette, keine Kinder
- Kommentar-Knoten *(comment node)*:
  - kein Name, Wert = Kommentartext, keine Kinder

ELEMENT
folien

ATTRIBUT
sprache
de

TEXT
XML Attribute

Einfaches Beispiel für Matching

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://
  www.w3.org/1999/XSL/Transform">
    <xsl:template match="folie">
        Folie
    </xsl:template>
</xsl:stylesheet>
```

- Ergebnis:
  - Folie-Elemente werden gefunden

Templates und Matching

`<xsl:template match="`*`type`*`">`

- Definiert eine *Schablone (template)*, die unter genau definierten Bedingungen auf einen oder mehrere Knoten paßt.
  - Wichtigste Typen (Werte von *type*):

`/`	Wurzelknoten
`*`	Elementknoten
`xyz`	Elementknoten des Tags *xyz*
`text()`	Textknoten
`@*`	Attributknoten
`@abc`	Attributknoten mit Namen *abc*
`node()`	Beliebiger Knoten außer Attributknoten und Wurzelknoten

  - Weitere Einschränkungen, z.B.
    - » über Pfade (siehe *XPath*-Syntax)
    - » über Bedingungen, z.B. für die relative Position
  - Alternativen mit "`|`", z.B. "**`*|@*`**"

Rekursion

- Dokumentengetriebener Aufruf von Templates:
  - Rekursion explizit angestossen mit
    ```
    <xsl:apply-templates select="pfad">
    ```
  - *pfad*-Attribut kann fehlen, dann:
    Standard-Rekursion über alle Nachfolge-Knoten *ohne* Attributknoten
- Eingebaute Standard-Templates in XSLT
  - Stellen Text-Inhalte (hintereinander verkettet) lesbar dar
  - Sind überall (zusätzlich) wirksam, wo nicht durch eigene Templates "überschrieben"
- Direkter Aufruf von (benannten!) Templates:

  ```
  <xsl:call-template name="templateName">
  ```

Auslesen von Information

<xsl:value-of select="*expression*">

- *expression* liefert Zeichenreihe, Knotenmenge oder Teilbaum
- *expression* erlaubt Navigation im Baum
  - mit *XPath*-Syntax:
    - » **`xyz`** Wert der Element-Unterknoten mit Name *xyz*
    - » **`@xyz`** Wert der Attribut-Unterknoten mit Name *xyz*
  - relativ zum aktuellen Knoten *(current node)* und einer aktuellen Knotenmenge *(current node set)* ausgewertet
- Wichtigste Funktionen in *expression*:
  - **`current()`** oder **`.`** Wert des aktuellen Knotens
  - **`name()`** Name des aktuellen Knotens

Beispiel: XSLT-Stylesheet (1)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://
  www.w3.org/1999/XSL/Transform">
    <xsl:output method="html"/>
    <xsl:template match="/">
        <html>
            <head>
                <title>Transformationsdemo</title>
            </head>
            <body>
                <xsl:apply-templates/>
            </body>
        </html>
    </xsl:template>
```

Beispiel: XSLT-Stylesheet (2)

```
<xsl:template match="folie">
    <h2>
        <xsl:value-of select="titel"/>
    </h2>
    <ul>
        <xsl:apply-templates
                select="themenliste/thema"/>
    </ul>
    <p>
        <i>Folienname:
           <xsl:value-of select="@ident"/>,
           Erstellt am:
           <xsl:value-of select="@erstellt"/>
        </i>
    </p>
</xsl:template>
```

Beispiel: XSLT-Stylesheet (3)

```
    <xsl:template match="thema">
        <li>
            <xsl:value-of select="current()"/>
        </li>
    </xsl:template>
</xsl:stylesheet>
```

Beispiel: Transformationsergebnis

- Für obige Beispielfolie:

```
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
        charset=UTF8">
      <title>Transformationsdemo</title>
   </head>
   <body>
      <h2>Attribute in XML</h2>
      <ul>
         <li>Deklaration in DTD</li>
         <li>Verwendung in XML-Dokument</li>
      </ul>
      <p><i>Folienname: f1, Erstellt am: 04.06.2003</i></p>

      <h2>Identifikatoren</h2>
      <ul>
         <li>Eindeutigkeit</li>
      </ul>
      <p><i>Folienname: f2, Erstellt am: 03.06.2003</i></p>
   </body>
</html>
```