Motivation für XML 2/4

Von HTML zu XML

HTML beschreibt

Layout des Dokumentinhalts

```
<h1>HandyKatalog</h1>
<h2>Nokia 8210</h2>
<table border="1">
 <tr>
  <td>Batterie</td><td>900mAh</td>
 </tr>
 <tr>
  <td>Gewicht</td><td>141g</td>
 </tr> …
</table>
```

XML beschreibt

Struktur u. **Semantik d.** Dokumentinhal

```
<HandyKatalog>
 <Hersteller name="Nokia">
  <Modell name="8210">
   <Batterie>900mAh</Batterie>
   <Gewicht>141g</Gewicht>
   …
  </Modell>
 </Hersteller>
</HandyKatalog>
```

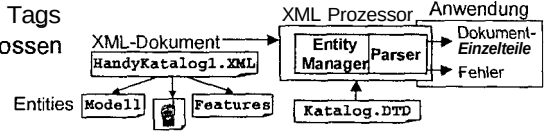
Tim Bray, Co-Editor von XML 1.0:

"XML ***will become the ASCII of the 21st century - basic, essential, unexciting***'

Motivation für XML 4/4

Eigenschaften von XML-Dokumenten und XML-Prozessoren

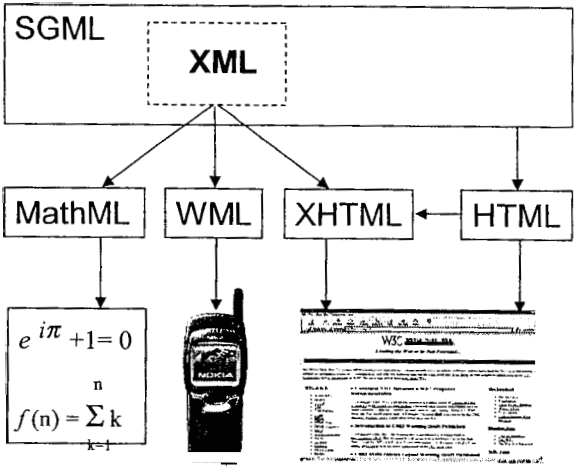
- Wohlgeformtheit (**well-formedness**)
  best. syntaktische Eigenschaften, z.B.:
  - Mindestens 1 Element pro Dokument
  - Exakt 1 Element als Wurzel
  - Keine Überlappungen bei Tags
  - Jedes Tag muß abgeschlossen werden
  - ....
- Gültigkeit (validity)
  - XML Dokument ist wohlgeformt und entspricht einem Schema



- XML-Prozessoren lesen XML-Dokumente ein und überprüfen
  - entweder nur deren Wohlgeformtheit (Nicht-validierende Prozessoren)
  - oder auch deren Validität (validierende Prozessoren)
- Können in Anwendungen (z.B. Browser) eingebunden werden
- Zerlegen ein XML-Dokument in seine Einzelteile und erstellen einen Baum, durch den die Einzelteile für die Anwendung zugreifbar werden

Dokumentbeschreibungssprachen 2/3

Einordnung von XML in OMG's Metadata Architecture



Metaebene **M2**

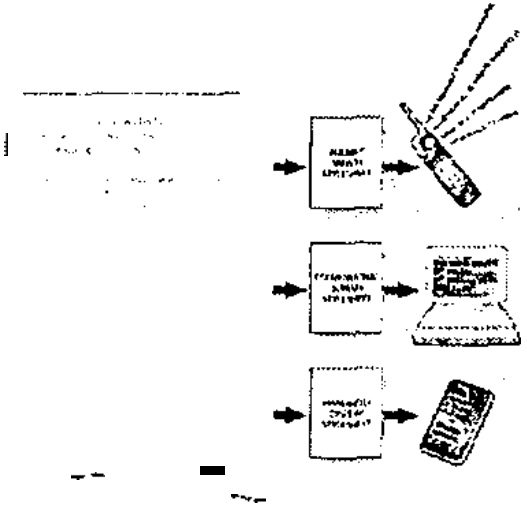
Sprachebene (DTDs) **M1**

Ausprägungs-ebene (Dokumente) **M0**

[www.omg.org]

Anwendungsbereiche für XML 1/3

- **Datenaustausch ("Portable Daten")**
  - über XML als reine Notation oder zusatzlich über gemeinsame Schemata



- **Multi-Delivery**
  - ein und derselbe Inhalt kann auf verschiedenen Endgeraten unterschiedlich präsentiert werden

- **Intelligente Suche**
  - statt einfacher Schlagwortsuche in HTML-Dokumenten, strukturbasierte Suche in XML-Dokumenten möglich

"Mozart" - Komponist oder Kugel?

XML-Dokument 1/3

Beispiel: HandyKatalog

HandyKatalog1.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HandyKatalog>
<!-- NOKIA -->
<Hersteller name="NOKIA">
  <HerstellerNr nr="h1234"/>
  <Modell name="7110">
    <Gewicht>141g</Gewicht>
    <Preis vertrag="ja">999</Preis>
    <Preis vertrag="nein">4999</Preis>
  </Modell>
  <Modell name="8210"
  ...
  </Modell>
 </Hersteller>
</HandyKatalog>
```

Elementname

Kommentar

Subelement

Start Tag

Elementinhalt "element content" von <Hersteller>

End Tag

Prolog (optional) "xml declaration"

Elementwurzel "root element" oder "document element"

Leeres Element "empty element"

Text "character data"

Gemischter Inhalt "mixed content"

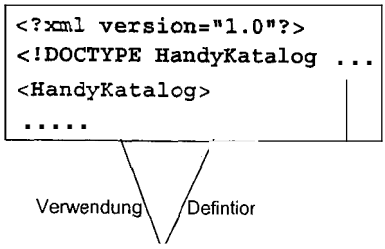
Attribut

Attributwert

DTD 1/8

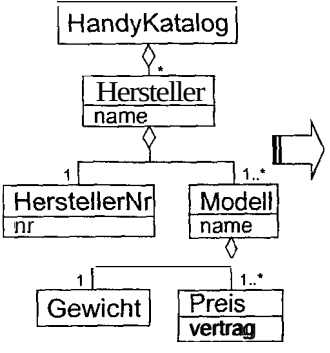
Zweck und Charakteristika

- Eine **DTD** beschreibt **Vokabular** und Grammatik für eine Menge von XML-Dokumenten
- Ein XML-Dokument darf nur eine einzige DTD einbinden ("document type declaration - **DOCTYPE")**
- NACH dem **Prolog**, jedoch VOR der **Elementwurzel** eingebunden werden
- Eine DTD **legt** nicht die **Elementwurzel** eines XML-Dokuments fest



DTD 3/8
Beispiel - Katalog.dtd

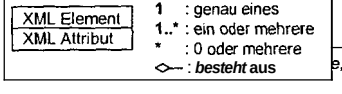
UML Klassendiagramm



XML DTD

```
<!-- Katalog DTD Version 1.0 -->
<!ELEMENT HandyKatalog (Hersteller*)>
<!ELEMENT Hersteller (HerstellerNr, Modell+)>
<!ATTLIST Hersteller name CDATA #REQUIRED>
<!ELEMENT HerstellerNr EMPTY>
<!ATTLIST HerctellerNr nr ID #REQUIRED>
<!ELEMENT Modell (Gewicht, Preis+)>
<!ATTLIST Modell name CDATA #REQUIRED>
<!ELEMENT Gewicht (#PCDATA)>
<!ELEMENT Preis (#PCDATA)>
<!ATTLIST Preis vertrag (ja|nein) "nein">
```

Legende:



DTD 7/8

Attributdeklaration - 10 Typen

- **CDATA**
  - Zeichenkette
  - `<!ATTLIST Hersteller name CDATA #REQUIRED>`

- **ID, IDREF(S)**
  - **ID** gewährleistet Eindeutigkeit von Attributwerten innerhalb eines Dokuments
  - pro Element ist nur 1 Attribut vom Typ **ID** erlaubt

```
<!ATTLIST beispiel
  identität ID    #IMPLIED
  referenz  IDREF #IMPLIED>
```

  - **IDREF** ist eine Referenz auf ein Attribut vom Typ **ID**
  - referentielle Integrität (ungetypt!) wird durch XML-Prozessor geprüft
  - Werte von **ID**-u. **IDREF(S)**-Attributen müssen gültige XML Namen sein, d.h. dürfen z.B. nicht mit Zahlen beginnen

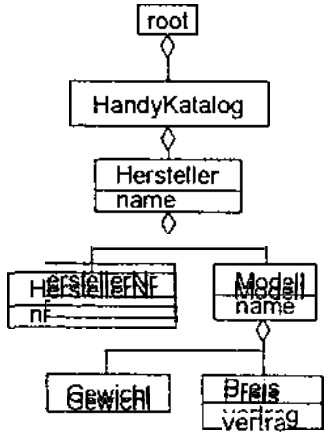
Pfadausdrücke 1/8

XPath - XML Path Language

- Zweck
  - ursprüngliches Ziel: Selektion von Dokumententeilen zum Layoutieren (XSL)
  - mittlerweile auch für **XPointer** verwendet
  - keine XML-Syntax - Proprietär!
  - Selektionskriterien: Element- und Attributnamen, Inhalt, Typ, etc.
- Grundprinzip der Verarbeitung
  - Navigation in einem Baum, ähnlich zur Navigation in einem Dateisystem
  - Ausgangspunkt ist immer ein bestimmter Kontext in Form eines Baumknotens
  - dieser wird entweder von einem XPath-Ausdruck selbst oder von einer Anwendung (z.B. XSLT) vorgegeben
  - Navigation und Filter modifizieren den Kontext
  - Ergebnis eines XPath-Ausdrucks = zuletzt berechneter Kontext
- W3C-Standard "XML Path Language (XPath) **1.0**", 16. Nov. 1999 (44 Seiten)

Pfadausdrücke 6/8

Filter - Beispiele



- **`//Modell[Preis]`**
  - alle `Modell` Elemente, die ein `Preis` Element enthalten
- **`//Hersteller[HerstellerNr]/Modell[Preis]`**
  - alle `Modell` Elemente, die ein `Preis` Element enthalten und in einem `Hersteller` Element enthalten sind, das ein `HerstellerNr` Element enthält
- **`//Hersteller[Modell/Preis]`**
  - alle `Hersteller` Elemente, die ein `Modell` Element enthalten, das ein `Preis` Element als Kind hat
- **`//Modell[Gewicht and Preis]`**
  - alle `Modell` Elemente mit `Gewicht` und `Preis` Subelementen
- **`//Modell[Gewicht = "141g"]`**
  - alle `Modell` Elemente, die ein `Gewicht` Element enthalten, das den Wert `141g` hat
- **`//Modell[@name = "7110"]`**
  - alle `Modell` Elemente, deren Attribut `name` den Wert `7110` hat

Einführung 1/2

- XML Schema
  - Definition der Struktur von XML-Dokumenten
  - W3C **REC** 2.5.2001, ca. 420 Seiten
- Nachteile DTDs
  - eigene Syntax
  - wenige eingeschränkte Datentypen (nur ein Basisdatentyp: String)
  - globale Definition von ETs
  - Parameter Entities für Modularisierung, Vererbung
  - ID: einzelnes Attribut, Eindeutigkeit im gesamten XML-Dokument
  - Referenzen (IDREF(S)): nicht typisiert
- Vorteile XML Schema
  - XML als Syntax
  - zahlreiche vordefinierte Datentypen
  - benutzerdefinierte einfache und komplexe Datentypen
  - Vererbung
  - Key: Eindeutigkeit auf Bereichsebene
  - Referenzen auf Keys
  - …

Einführung 2/2

DTD versus XML Schema

Katalog.xsd

```
<?xml version="1.0"?>
<schema ...>
  <simpleType name="herstellerNr"> ...
  <element name="HandyKatalog">
    <complexType>
```

Katalog.dtd

```
...
<!ELEMENT HandyKatalog (Hersteller*)>
<!ELEMENT Hersteller(HerstellerNr, Modell+)>
<!ELEMENT Modell (Gewicht, Batterie)>
<!ELEMENT Gewicht(#PCDATA)>
<!ELEMENT Batterie (#PCDATA)> ...
```

Elemente und Attribute 3/3

globales Element, lokaler Datentyp

lokales Element, globaler Datentyp

Referenz auf globales Element

lokales Attribut, vordef. Datentyp

globales Element, lokaler Datentyp

lokales Element, vordef. Datentyp

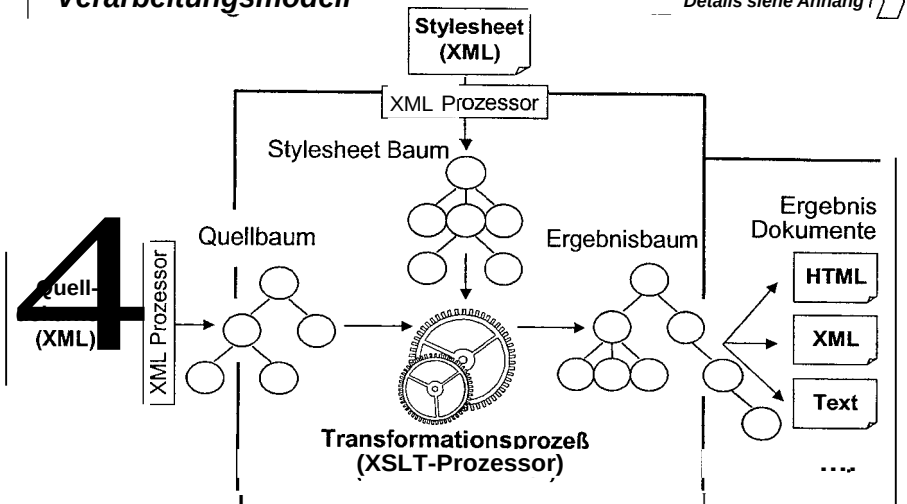
globaler Datentyp

```
<element name="Hersteller">
  <complexType>
    <sequence>
      <element name="HerstellerNr" type="hk:herstellerNr"
               minOccurs="1" maxOccurs="1"/>
      <element ref="hk:Modell" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
  </complexType>
</element>
<element name="Modell">
  <complexType>
    <sequence>
      <element name="Gewicht" type="string"/>
      <element name="Batterie" type="string"/>
    </sequence>
  </complexType>
</element>
<simpleType name="herstellerNr">
  <restriction base="string"> ...
```

XSLT Grundlagen 4/4

Verarbeitungsmodell

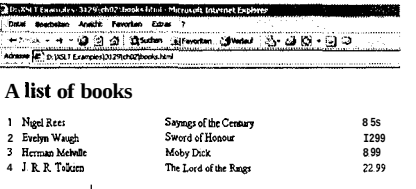
Details siehe Anhang I



XSLT Beispiel 1/3

Quell- und Ergebnisdokument

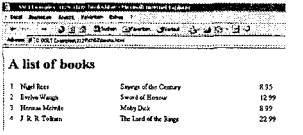
```
<?xml version="1.0"?>
 <books>
   cbook category="reference">
      <author>Nigel Rees</author>
   </book>
      cbook category="fiction">
      <author>Herman Melville</author>
      <title>Moby Dick</title>
      <price>8.99</price>
   </book>
   <book category="fiction">
      <author>J. R. R. Tolkien</author>
      <title>The Lord of the Rings</title>
      <price>22.99</price>
   </book>
  c/books>
```



XSLT Beispiel 3/3

Stylesheet

〉Details siehe Anhang II〈




```
<xsl:stylesheet version="1.0"
```

```
①<xsl:template match="books">
    <html>
    <body>
      <h1>A list of books</h1>
      <table width="640">
        <xsl:apply-templates/>
      </table>
    </body></html>
  </xsl:template>
```

Template Rule

Literale Ergebniselemente

Anwenden weiterer Template Rules auf Kindknoten von <books>

```
②<xsl:template match="book">
    <tr>
      <td><xsl:number/></td>
      <xsl:apply-templates/>
    </tr>
  </xsl:template>
```

Fortlaufende Nummerierung gemäß Position der <book> Knoten irn Quelldokument

```
③<xsl:template match="author | title | price">
    <td><xsl:value-of select="."/></td>
  </xsl:template>
```

Ausgabe des Inhalts der <author> <title> u. <price> Knoten