



Unweit von hier befand sich das Basislager von dem aus
die wagemutigen BRÜDER FLORIAN UND MICHAEL BLOCH
mit ihren unerschrockenen Sherpas am
Sonntag den 28.6.2009 um 11:44:17 MEZ
zur ERSTBESTEIGUNG DES PATSCHERKOFEL MIT SAUERSTOFFMASKEN aufbrachen.
Um 13:35:02 MEZ konnte am Fuß des Gipfelkreuzes die Pfeifnudel-Flagge gehisst werden.

www.pfeifnudel.de

07-Input/Output-3-XML

Objektorientierte Programmierung | Matthias Tichy



Software Engineering
Programming Languages



universität
uulm

Lernziele

- Einführung in XML
- XML-Schema

XML

- Ein bisschen zwischen Text- und Binärformat
 - gute automatisierte Verarbeitung
 - gute Lesbarkeit für den Menschen
- Beispiele:
 - SVG
 - Microsoft Office (.docx, .pptx, .xlsx)
 - OpenOffice (.odt, .odp, .ods)
 - HTML
- XML ist neben JSON ein Standardaustauschformat im Web und vielen Softwarewerkzeugen
 - Damit auch Zugriff auf viele Datenbanken
 - z.B. Google-APIs

Einführung in XML

XML

- Ziel: Plattform- und Implementationsunabhängiger Datenaustausch im aufkommenden World Wide Web
- Vorgänger: SGML, einfachere Variante gesucht
- erste Version 1998 veröffentlicht von W3C
- mittlerweile 5. Fassung der Version 1.0 (2008)
und 2. Fassung der Version 1.1 (2006)

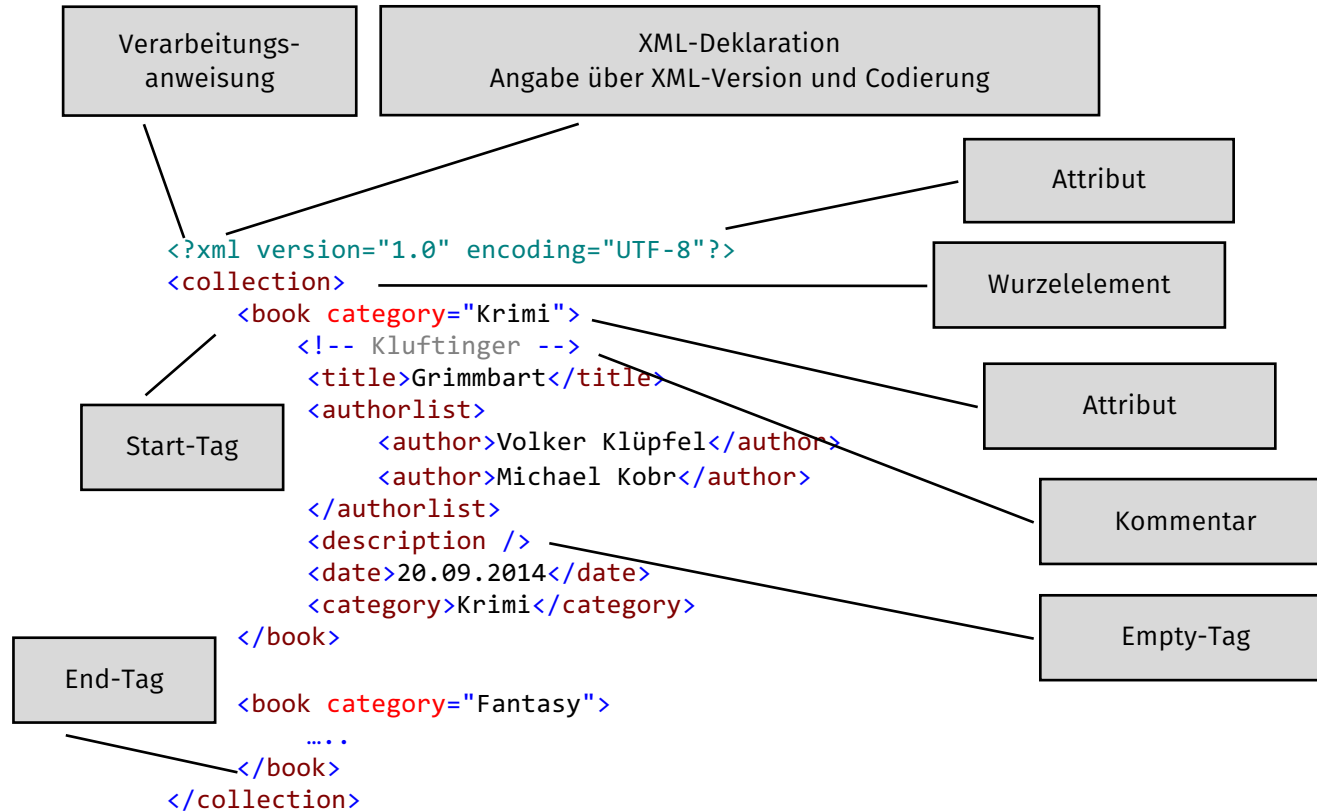
Ziele

- Aufbau von XML-Dateien kennen und verstehen
- (fremde) XML-Dateien lesen können
- eigene XML-Dateien definieren können

- Vorteile einer Strukturdefinition verstehen

- DTDs lesen und definieren können
- XSDs lesen und definieren können
- Unterschiede zwischen DTD und XSD kennen

XML

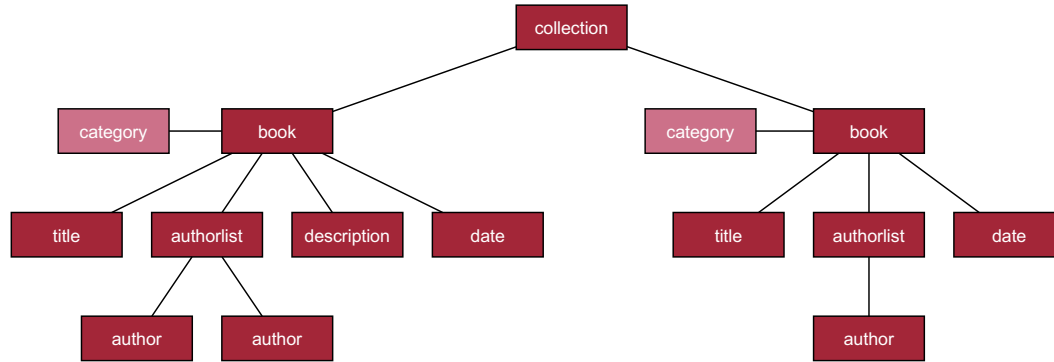


XML als Baum

```
<?xml version="1.0" encoding="UTF-8"?>
<collection>
  <book category="Krimi">
    <!-- Kluftinger -->
    <title>Grimmbart </title>
    <authorlist>
      <author>Volker Klüpfel</author>
      <author>Michael Kobr</author>
    </authorlist>
    <description />
    <date>20.09.2014</date>
  </book>
```

```
<book category="Fantasy">
  <title>
    Die 13½ Leben des Käpt'n
    Blaubär
  </title>
  <authorlist>
    <author>Walter Moers</author>
  </authorlist>
  <date>01.12.2002</date>
</book>
</collection>
```


XML als Baum



```
<?xml version="1.0" encoding="UTF-8"?>
<collection>
  <book category="Krimi">
    <!-- Kluftinger -->
    <title>Grimmbart </title>
    <authorlist>
      <author>Volker Klüpfel</author>
      <author>Michael Kobr</author>
    </authorlist>
    <description />
    <date>20.09.2014</date>
  </book>
```

```
<book category="Fantasy">
  <title>
    Die 13½ Leben des Käpt'n
    Blaubär
  </title>
  <authorlist>
    <author>Walter Moers</author>
  </authorlist>
  <date>01.12.2002</date>
</book>
</collection>
```

XML

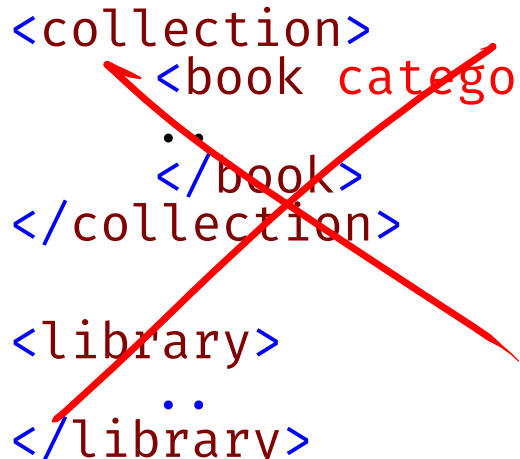
- strukturierte Daten
- Tags
 - werden selbst definiert
 - sind typischerweise selbstbeschreibend
 - sollten prägnant sein
 - sind Case Sensitive
 - sind üblicherweise klein geschrieben
 - enthalten keine Leerzeichen, sondern nur Buchstaben, Ziffern, -, _, .
 - sind != "xml" (einziges reserviertes Wort)

XML Element

- Element: Start-Tag bis einschließlich Ende-Tag
- kann enthalten:
 - andere Elemente
 - Text
 - Attribute
 -
 - oder eine Mischung von allem
- leere Elemente: `<description/>` oder
- `<description></description>`

wohlgeformtes XML

- Genau ein Wurzelement



```
<collection>  
  <book category="Krimi">  
    .  
  </book>  
</collection>  
  
<library>  
  .  
</library>
```

wohlgeformtes XML

- Genau ein Wurzelement
- Schließendes Ende-Tag

```
<collection>  
  <book category="Krimi">  
    ..  
    <!-- hier fehlt das schließende book -->  
</collection>
```

wohlgeformtes XML

- Genau ein Wurzelelement
- Schließendes Ende-Tag
- keine Überschneidung der Elemente

```
<book category="Krimi">  
  <!-- Klurtinger -->  
  <title>Grimmbart  
  ..  
  </book>  
</title>
```

wohlgeformtes XML

- Genau ein Wurzelement
- Schließendes Ende-Tag
- keine Überschneidung der Elemente
- Attribut-Werte müssen in Hochkomma stehen

~~<book category=Krimi>~~

~~..~~

~~</book>~~

wohlgeformtes XML

- Genau ein Wurzelement
- Schließendes Ende-Tag
- keine Überschneidung der Elemente
- Attribut-Werte müssen in Hochkomma stehen
- Keine zwei Attribute mit gleichem Namen innerhalb eines Elements

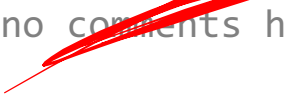
```
<book category="Krimi" category="Regional">
```

```
..  
</book>
```


wohlgeformtes XML

- Genau ein Wurzelement
- Schließendes Ende-Tag
- keine Überschneidung der Elemente
- Attribut-Werte müssen in Hochkomma stehen
- Keine zwei Attribute mit gleichem Namen innerhalb eines Elements
- Kommentare und Steueranweisungen dürfen nicht innerhalb von Tags stehen

```
<book category="Krimi" <!-- no comments here! -->
</book>
```



weitere Syntaxregeln

- folgende Zeichen dürfen nicht in Text vorkommen bzw. müssen umgeformt werden:

▪ <	➔	<
▪ >	➔	>
▪ &	➔	&
▪ '	➔	'
▪ "	➔	"

- whitespaces bleiben erhalten

`<title>Grimmbart</title>`

`!=`

`<title>Grimmbart </title>`

- Zeilenendezeichen ist immer <LF>

eXtensible Markup Language

■ Was heißt hier Erweiterbarkeit?

```
<book category="Krimi">
  <title>Grimmbart</title>
  <authorlist>
    <author>Volker Klüpfel</author>
    <author>Michael Kobr</author>
  </authorlist>
  <publisher>Droemer</publisher>
  <description />
  <date>20.09.2014</date>
  <price>19.99</price>
</book>
```

Attribute

```
<book category="Fantasy">  
  <title>Die 13½ Leben des Käpt'n Blaubär</title>  
  <authorlist>  
    <author>Walter Moers</author>  
  </authorlist>  
  <date>01.12.2002</date>  
</book>
```

```
<book>  
  <title>Die 13½ Leben des Käpt'n Blaubär</title>  
  <authorlist>  
    <author>Walter Moers</author>  
  </authorlist>  
  <date>01.12.2002</date>  
  <category>Fantasy</category>  
</book>
```

Attribute

- Attribute sind nicht leicht erweiterbar
- Faustregeln:
 - **Attribute** für Metadaten
 - **Elemente** für konkrete Daten
- Beispiel:
`<book id="123">`
`..`
`</book>`

Namespaces

```
<collection>  
  <book category="Krimi">  
    ..  
  </book>  
</collection>
```

```
<collection>  
  <dvd genre="Tierfilm">  
    ..  
  </dvd>  
</collection>
```

Namespaces

```
<collections xmlns:b="Bücher" xmlns:dvd="DVDs">
  <b:collection>
    <book category="Krimi">
      ..
    </book>
    <book category="Fantasy">
      ..
    </b:collection>

  <dvd:collection>
    <dvd genre="Tierfilme">
      ..
    </dvd>
  </dvd:collection>

</collections>
```

Default-Namespaces

```
<collections>
  <collection xmlns="Bücher">
    <book category="Krimi">
      ..
    </book>
    <book category="Fantasy">
      ..
    </collection>

  <collection xmlns="DVDs">
    <dvd genre="Tierfilme">
      ..
    </dvd>
  </collection>

</collections>
```


Murmelgruppe – 5 Minuten

Bearbeiten Sie mit Ihrem Nachbarn (~2-3 Personen) folgende Aufgabe:

- Definieren Sie eine XML-Datei (mit ausgedachten konkreten Daten) für die Verwaltung von Übungsblättern:
 - Jedes Übungsblatt hat eine Nummer, ein Abgabedatum und mehrere Aufgaben
 - Jede Aufgabe hat eine Nummer, ein Thema und Punkte
 - Eine Aufgabe kann auch Unteraufgaben haben, die wiederum Nummern und (Teil-) Punkte haben

Lösungen?

XML-Schema

wohlgeformt vs. gültig

- wohlgeformt:
Syntax korrekt
- gültig:
Struktur entspricht einem vorgegebenen Aufbau → Schema
- Wie soll Schema beschrieben werden?
 - natürlichsprachlich
"Buch hat eine Kategorie, einen Titel, ..."
 - formale Schemasprache: DTD oder XSD

Document Type Definition (initialer Standard)

Beispiel

- Verweis auf DTD in XML-Datei:

```
<!DOCTYPE collection SYSTEM "booklist.dtd">
```

- Inhalt von booklist.dtd:

```
<!ELEMENT collection (book+)>
<!ELEMENT book (title, authorlist,
                description?, date)>
<!ATTLIST book category CDATA #IMPLIED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT authorlist (author+)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT description EMPTY>
<!ELEMENT date (#PCDATA)>
```

Probleme mit DTD

- DTD kann nicht ...
 - Anzahl der Instanzen eines Elements angeben
 - Aussehen der Zeichendaten innerhalb eines Elements spezifizieren
 - semantische Bedeutung eines Elements beschreiben
- → XML Schema Definition (XSD)

XSD – Allgemein

- XSD ist selbst in XML definiert → Validierung
- XSD unterstützt mehr und komplexere Datentypen
- Es können eigene Datentypen konstruiert und wiederverwendet werden
- Unterstützung von Namespaces
- XSD sehr umfangreich → nur grober Überblick

XSD – Definition und Referenz

- booklist.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="bookcollection"
            xmlns="bookcollection"
            xmlns:xs="http://www.w3.org/2001/XMLSchema">

    ..
</xs:schema>
```

- booklist.xml:

```
<collection
    xmlns="bookcollection"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="bookcollection booklist.xsd">

    ..
</collection>
```


XSD – Types

- Unterscheidung zwischen
 - SimpleTypes
Basistyp oder einfaches Element
 - ComplexTypes
Zusammengesetzte Typen/Elemente evtl. mit Attributen

XSD – Simple Types

- Element

```
<xs:element name="author" type="xs:string"/>
```

- Attribut

```
<xs:attribute name="category" type="xs:string"/>
```

- mit Defaultwert:

```
<xs:attribute name="category" type="xs:string"  
              default="unsorted"/>
```

- zwingend erforderlich:

```
<xs:attribute name="category" type="xs:string"  
              use="required"/>
```

XSD – Einschränkungen

■ Wertebereich:

minInclusive, maxInclusive, minExclusive, maxExclusive

```
<xs:element name="speed">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="350"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

XSD – Einschränkungen

- "named types" und Referenz

```
<xs:simpleType name="speedType">  
  <xs:restriction base="xs:int">  
    <xs:minInclusive value="0"/>  
    <xs:maxInclusive value="350"/>  
  </xs:restriction>  
</xs:simpleType>  
  
<xs:element name="speed" type="speedType"/>
```

XSD – Einschränkungen

- Aufzählungen: `enumeration`

```
<xs:element name="RegierungsbezirkeBW" type="regBWType"/>
<xs:simpleType name="regBWType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Freiburg"/>
    <xs:enumeration value="Karlsruhe"/>
    <xs:enumeration value="Stuttgart"/>
    <xs:enumeration value="Tübingen"/>
  </xs:restriction>
</xs:simpleType>
```

XSD – Einschränkungen

- reguläre Ausdrücke: `pattern`
- Längen: `length`, `maxLength`, `minLength`
- Ziffern: `fractionDigits`, `totalDigits`
- Behandlung von Whitespace: `whiteSpace`
 - `preserve`: Whitespaces bleiben unberührt
 - `replace`: Tabs, Zeilenumbrüche etc. werden zu Leerzeichen
 - `collapse`: wie `replace`, mehrere aufeinanderfolgende werden zu einem

XSD – Kombination

- Einfache Datentypen können zu neuen Datentypen kombiniert werden

- Listen:

```
<xs:simpleType name='listOfIntegers'>  
  <xs:list itemType='xs:integer' />  
</xs:simpleType>
```

- Vereinigung:

```
<xs:attribute name="farbe">  
  <xs:simpleType>  
    <xs:union memberTypes="xs:string farbType" />  
  </xs:simpleType>  
</xs:attribute>
```

XSD – Complex Types

■ Complex Element

```
<xs:element name="author" type="xs:string">
<xs:element name="authorlist">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="author" minOccurs="1"
                  maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```


XSD – Complex Types

- Mögliche Kindelemente von `<complexType>`:
 - `sequence` alle in der angegebenen Reihenfolge
 - `choice` entweder-/oder
 - `all` alle in beliebiger Reihenfolge
 - `base` nix bedeutet leeres Element

XSD – Data Types

- einige vordefinierte (simple) DataTypes:
 - `string`
 - `normalizedString` (ohne Leerzeichen an den "Rändern")
 - `int`
 - `byte`
 - `long`
 - `double`
 - `float`
 - `boolean`
 - `time`
 - `data`
 - `ID`, `IDREF(s)`, `NMTOKEN(s)`, etc....

XSD – Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="bookcollection" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="bookcollection">
  <xs:element name="collection">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="book" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

XSD – Beispiel

```
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="title"/>
      <xs:element ref="authorlist"/>
      <xs:element ref="description" minOccurs="0"/>
      <xs:element ref="date"/>
    </xs:sequence>
    <xs:attribute name="category" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

XSD – Beispiel

```
<xs:element name="title" type="xs:string"/>

<xs:element name="authorlist">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="author" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="author" type="xs:string"/>

<xs:element name="description" />
<xs:element name="date" type="xs:date"/>
</xs:schema>
```

- Murmelgruppe – 5 Minuten
- Bearbeiten Sie mit Ihrem Nachbarn (~2-3 Personen) folgende Aufgabe:
 - Erstellen Sie eine XSD für obiges Übungsblatt-XML!

Lösungen?

Typische Aufgaben eines XML-Werkzeugs

- Syntaxhighlighting
- Editierhilfe
- Prüfung der Wohlgeformtheit
- Validierung
- Baumdarstellung
- Codevervollständigung über Schema
- Generierung von DTD oder XSD
- XSLT-Verarbeitung
- Definition von Web-Schnittstellen
- ...

Lernziele

- Einführung in XML
- Document Type Definition
- XML-Schema