

# Angewandte Informatik 2 - Tutorium 3

## Besprechung: Übungsblatt 3

Götz Bürkle, Alexander Lenk, Florian Spormann,  
Thomas Strehlow, Christian Rudolf

Institut für Angewandte Informatik und  
Formale Beschreibungsverfahren - AIFB  
Universität Karlsruhe (TH)

KW 24/25

## Übersicht

### XML-Hilfsmittel

### Übungsblatt 3

Aufgabe 1 - DTD

Aufgabe 2 - DTD

Aufgabe 3 - XML-Schema

Aufgabe 4 - XML-Schema

### Zusammenfassung

## Hilfsmittel für die Arbeit mit XML

- ▶ XML-Editor: **Altova XMLSpy Home Edition**, sehr leistungsfähiger mit begrenztem aber meist ausreichendem Funktionsumfang kostenlos erhältlicher Editor, siehe [http://www.altova.com/de/produkte/xmlspy/xml\\_editor.html](http://www.altova.com/de/produkte/xmlspy/xml_editor.html)
- ▶ Online-Validatoren:
  - ▶ XML-Validator: <http://www.validome.org/xml/>
  - ▶ DTD-/Schema-Validator: <http://www.validome.org/grammar/>
- ▶ XSLT-fähige Browser: Opera 9, Firefox, Internet Explorer 6

## DTD erstellen

Es ist eine DTD zu einem gegebenen XML-Dokument zu erstellen, so daß das gegebene Dokument gültig (valide) bezüglich dieser DTD ist.

### Was muß man dabei beachten?

- ▶ Welches ist das **Wurzel-Element**?
- ▶ Welche Elemente sind wie ineinander **geschachtelt**?
- ▶ Welche Elemente dürfen **wie oft** vorkommen?
- ▶ Welche Elemente besitzen **welche Attribute**?
- ▶ Welche Attribute **müssen** gesetzt sein, welche **können** gesetzt sein?

## book.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE book SYSTEM "book.dtd">
<book isbn="0836217462">
  <title>Being a Dog Is a Full-Time Job</title>
  <author>Charles M. Schulz</author>
  <character>
    <name>Snoopy</name>
    <friend-of>Peppermint Patty</friend-of>
    <since>1950-10-04</since>
    <qualification>extrov...</qualification>
  </character>
  <character>
    <name>Peppermint Patty</name>
    <since>1966-08-22</since>
    <qualification>bold, ...</qualification>
  </character>
</book>
```

## DTD

- ▶ **Wurzel-Element:** book
- ▶ **Schachtelung:**
  - book **enthält** title, author **und** character
  - character **enthält** name, friend-of, since **und** qualification
- ▶ **Anzahl:** genau ein title und genau ein author, mehrere character
- ▶ **Attribute:** book besitzt das Attribut isbn
- ▶ Das Attribut isbn **muß gesetzt sein**, jedes Buch hat schließlich eine ISBN.

## book.dtd

```
<!ELEMENT book (title, author, character*)>
<!ATTLIST book
  isbn CDATA #REQUIRED
>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT character
  (name, friend-of*, since, qualification)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT friend-of (#PCDATA)>
  <!ELEMENT since (#PCDATA)>
  <!ELEMENT qualification (#PCDATA)>
```

CDATA („character data“) - Inhalt wird als Text behandelt

PCDATA („parsed character data“) - kann Markup enthalten

## DTD erstellen

DTD für das Element `getraenkekiste` erstellen mit folgenden Eigenschaften:

- ▶ `produktname`
- ▶ `kistenfarbe`
- ▶ `flaschentyp`
- ▶ `flaschenanzahl`
- ▶ `flaschengroesse`

Sowohl in Element- und Attributdeklaration und jeweils ein Beispieldokument dazu schreiben.

## Elementdeklaration - DTD

```
<!ELEMENT getraenkekiste  
  (produktname, kistenfarbe,  
   flaschentyp, flaschenanzahl,  
   flaschengroesse) >  
<!ELEMENT produktname (#PCDATA) >  
<!ELEMENT kistenfarbe (#PCDATA) >  
<!ELEMENT flaschentyp (#PCDATA) >  
<!ELEMENT flaschenanzahl (#PCDATA) >  
<!ELEMENT flaschengroesse (#PCDATA) >
```

## Elementdeklaration - Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE getraenkekiste SYSTEM  
  "getraenkekiste_element.dtd">  
<getraenkekiste>  
  <produktname>Coca Cola</produktname>  
  <kistenfarbe>rot</kistenfarbe>  
  <flaschentyp>PET-Mehrwegflasche</flaschentyp>  
  <flaschenanzahl>12</flaschenanzahl>  
  <flaschengroesse>1 Liter</flaschengroesse>  
</getraenkekiste>
```

## Attributdeklaration - DTD

```
<!ELEMENT getraenkekiste EMPTY>
<!ATTLIST getraenkekiste
  produktname CDATA #REQUIRED
  kistenfarbe CDATA #REQUIRED
  flaschentyp CDATA #REQUIRED
  flaschenanzahl CDATA #REQUIRED
  flaschengroesse CDATA #REQUIRED
>
```

## Attributdeklaration - Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE getraenkekiste SYSTEM
  "getraenkekiste_attribut.dtd">
<getraenkekiste
  produktname="Coca Cola"
  kistenfarbe="rot"
  flaschentyp="PET-Mehrwegflasche"
  flaschenanzahl="12"
  flaschengroesse="1 Liter" />
```

## DTD → XML-Schema

Aus einer gegebenen DTD Vorlesungsverz ein („äquivalentes“) XML-Schema erstellen.

Was hat es mit einem XML-Schemas auf sich?

- ▶ XML-Schemas sind „mächtiger“ als DTDs
- ▶ Abbildung DTD → XML-Schema möglich, jedoch XML-Schema → DTD nicht immer!
- ▶ XML-Schema ist selbst ein XML Format

## DTD - Vorlesungsverz.dtd

```
<!ELEMENT Vorlesungsverz (Vorlesung+, Jahr)>
<!ELEMENT Vorlesung (Dozent, Titel, Hoersaal,
Uebungsleiter?)>
<!ATTLIST Vorlesung
  Kennung ID #REQUIRED
>
<!ELEMENT Dozent (#PCDATA)>
<!ELEMENT Titel (#PCDATA)>
<!ELEMENT Hoersaal (#PCDATA)>
<!ELEMENT Uebungsleiter (#PCDATA)>
<!ELEMENT Jahr (#PCDATA)>
```

## XML-Schema - Vorlesungsverz.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Vorlesungsverz"
    type="VorlesungsverzType"/>
  <xs:complexType name="VorlesungsverzType">
    <xs:sequence>
      <xs:element name="Vorlesung"
        type="VorlesungType" maxOccurs="unbounded"/>
      <xs:element name="Jahr" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
  [...]
</xs:schema>
```

## XML-Schema - Vorlesungsverz.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema [...]
  <xs:complexType name="VorlesungType">
    <xs:sequence>
      <xs:element name="Dozent" type="xs:string"/>
      <xs:element name="Titel" type="xs:string"/>
      <xs:element name="Hoersaal" type="xs:string"/>
      <xs:element name="Uebungsleiter"
        type="xs:string" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Kennung"
      type="xs:ID" use="required"/>
  </xs:complexType>
</xs:schema>
```

## XML-Schema für Fußball-WM-Teilnehmer

Folgende Regeln sollen gelten:

- ▶ `jahr` der Veranstaltung ist als Ganzzahl im Tag `wm-teilnehmer` vorgeschrieben
- ▶ genau 32 `teams` in loser Reihenfolge, jedes `team` muß ein Zeichenkettenattribut `nation` besitzen
- ▶ pro `team` genau ein `trainer`, genau eine `mannschaft` und optional ein Element `sonstige`
- ▶ `trainer` besteht aus einer simplen Zeichenkette
- ▶ `mannschaft` besteht aus minimal 11 und maximal 22 Spielern
- ▶ `spieler` besitzt Attribut `position`

## XML-Schema für Fußball-WM-Teilnehmer

→ Regeln:

- ▶ `position` darf nur folgende Werte annehmen:  
Angriff, Mittelfeld, Abwehr, Tor
- ▶ `spieler` besteht aus einer einfachen Zeichenkette
- ▶ Unter `sonstige` sind Funktionäre und Betreuer in loser Reihenfolge ohne Beschränkungen der Anzahl aufgelistet.
- ▶ Betreuer und Funktionäre sind Zeichenketten als Freitext.

## Beispielausschnitt - wm-teilnehmer.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<wm-teilnehmer jahr="2006"
  xmlns:xsi=
    "http://www.w3.org/2001/XMLSchema-instance">
  <team nation="Deutschland">
    <trainer>Der wo gwinne wellet</trainer>
    <mannschaft>
      <spieler position="Angriff">
        K. Lohse</spieler>
      <spieler position="Tor">
        J. Fielmann</spieler>
      [...]
    </mannschaft>
    [...]
  </team>
  [...]
</wm-teilnehmer>
```

## Beispielausschnitt - wm-teilnehmer.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<wm-teilnehmer jahr="2006" [...]
  <team nation="Deutschland">
    [...]
    <sonstige>
      <betreuer>Mueller-Wallfahrt</betreuer>
      <funktionaer>Mayer-Vordenker</funktionaer>
    </sonstige>
  </team>
  [...]
</wm-teilnehmer>
```

## XML-Schema - wm-teilnehmer.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="wm-teilnehmer"
    type="wm-teilnehmerType"/>
  <xsd:complexType name="wm-teilnehmerType">
    <xsd:sequence>
      <xsd:element name="team" type="teamType"
        minOccurs="32" maxOccurs="32"/>
    </xsd:sequence>
    <xsd:attribute name="jahr"
      type="xsd:positiveInteger" use="required"/>
  </xsd:complexType>
  [...]
</xsd:schema>
```

## XML-Schema - wm-teilnehmer.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema [...]
  <xsd:complexType name="teamType">
    <xsd:sequence>
      <xsd:element name="trainer"
        type="xsd:string"/>
      <xsd:element name="mannschaft"
        type="mannschaftType"/>
      <xsd:element name="sonstige"
        type="sonstigeType" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="nation" type="xsd:string"
      use="required"/>
  </xsd:complexType>
  [...]
</xsd:schema>
```

## XML-Schema - wm-teilnehmer.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema [...]
  <xsd:complexType name="mannschaftType">
    <xsd:sequence>
      <xsd:element name="spieler" type="spielerType"
        minOccurs="11" maxOccurs="22"/>
    </xsd:sequence>
  </xsd:complexType>
  [...]
</xsd:schema>
```

## XML-Schema - wm-teilnehmer.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema [...]
  <xsd:complexType name="sonstigeType">
    <xsd:sequence>
      <xsd:element name="betreuer" type="xsd:string"
        minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="funktionaer"
        type="xsd:string" minOccurs="0"
        maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  [...]
</xsd:schema>
```

## XML-Schema - wm-teilnehmer.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema [...]
  <xsd:complexType name="spielerType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="position"
          type="positionType" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  [...]
</xsd:schema>
```

## XML-Schema - wm-teilnehmer.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema [...]
  <xsd:simpleType name="positionType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Angriff"/>
      <xsd:enumeration value="Mittelfeld"/>
      <xsd:enumeration value="Abwehr"/>
      <xsd:enumeration value="Tor"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

---

## Zusammenfassung

- ▶ **DTD**: DTD lesen, verstehen und erstellen können
- ▶ **XML-Schema**: Transformation DTD → XML-Schema, XML-Schema lesen, verstehen und erstellen können
- ▶ **XML**: XML-Dokument passend zu einer DTD oder einem Schema erstellen können