



# XML Schema ( XML Schema Definition → XSD )

# XML Schema



- DTD ist für viele Zwecke ausreichend, bietet aber oft nicht genug Möglichkeiten
- es fehlt zum Beispiel
  - Datentypen
  - genaue Angabe der Anzahl ( „Element soll 2 bis 5 mal vorkommen“)
- XML Schema kann das, ist dafür aber auch aufwendiger und komplizierter

```
<!ELEMENT fortbildner (name,schule,alter?)>  
<!ELEMENT name (#PCDATA)>  
<!ELEMENT schule (#PCDATA)>  
<!ELEMENT alter (#PCDATA)>
```



wohlgeformt,  
gültig, unsinnig

```
<fortbildner>  
  <name>Alfred E. Neumann</name>  
  <schule>Mad Academy</schule>  
  <alter>dreiundreissig komma zwei</alter>  
</fortbildner>
```

# XML Schema Definition (XSD)

---



- die XML Schema Definition wird selbst in XML geschrieben
- XSD bietet die Möglichkeit, Datentypen zu verwenden und eigene Datentypen zu erstellen
- es können „komplexe“ Datentypen erstellt werden (Objekte)
- Datentypen können von anderen Datentypen abgeleitet werden (Vererbung)
- XML Dokumente, die mit einem XSD verbunden sind und dieses erfüllen heißen „schema valid“ oder „Instanzdokument“
- es ist durchaus möglich, einem XML eine DTD und eine XSD zuzuweisen. Es kann dann passieren, dass es die DTD erfüllt und die XSD nicht oder umgekehrt.

# Einsatzbereiche XML Schema ↔ DTD



Unterscheidung von Dokumenten hinsichtlich ihres Aufbaus:

## 1. Erzählende (narrative) Dokumente:

Dokumente, die aus Abschnitten und Unterabschnitten bestehen; die gesamte Struktur ist weitgehend linear: Bücher, Artikel, etc.

→ DTD

## 2. Datensatzartige Dokumente:

Stark typisierte Dokumente; zielen auf Datenaustausch ab.

→ XML Schema

Prof. B. Stein,  
<http://www.uni-weimar.de/en/media/chairs/webis/teaching/lecturenotes/#web-technology>

# XSD – Elemente definieren



```
<xs:element name="fortbildner">
  <xs:complexType>
    <xs:sequence>
      <xs:element name='name' type='xs:string' />
      <xs:element name='schule' type='xs:string' />
      <xs:element name='alter' type='xs:int' minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- XSD verwendet selbst XML, XSD Befehle beginnen mit „xs:“
- Element mit Daten:  

```
<xs:element
  name='...' type='xs:...'
  minOccurs='...' maxOccurs='...' />
```
- für „maxOccurs“ gibt es „unbounded“ → beliebig viele
- minOccurs/maxOccurs weglassen → 1

# XSD – Basisdatentypen

## (Simple Types)



- xs:string
  - xs:decimal
  - xs:integer
  - xs:float
  - xs:boolean
  - xs:date
  - xs:time
  - xs:anyType
- Eigene Basisdatentypen durch Hinzufügen von Einschränkungen (facets) zu vorhandenen Simple Types

```
<xs:simpleType name="t_alter">
  <xs:restriction base='xs:decimal'>
    <xs:totalDigits value="3"/>
    <xs:fractionDigits value="0"/>
    <xs:minInclusive value="1"/>
    <xs:maxInclusive value="130"/>
  </xs:restriction>
</xs:simpleType>
<!-- ... -->
<xs:element name="alter" type="t_alter" />
```

# XSD – Beispiele für Facets



Aufzählung

```
<xs:simpleType name="wochentag">
  <xs:restriction base='xs:string'>
    <xs:enumeration value='Montag' />
    <xs:enumeration value='Dienstag' />
    <xs:enumeration value='Mittwoch' />
    <!-- ... -->
  </xs:restriction>
</xs:simpleType>
```

Regular Expression

```
<xs:simpleType name="email">
  <xs:restriction base='xs:string'>
    <xs:pattern value="^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$"/>
  </xs:restriction>
</xs:simpleType>
```

Längeneinschränkung

```
<xs:simpleType name="lehrerkuerzel">
  <xs:restriction base='xs:string'>
    <xs:minLength value="2"/>
    <xs:maxLength value="3"/>
  </xs:restriction>
</xs:simpleType>
```

- komplette Liste der verfügbaren Simple Types und Ihrer Facets:

<http://www.w3.org/TR/xmlschema11-2/#built-in-primitive-datatypes>

# XSD – Komplexe Elemente



```
<xs:element name="fortbildner">
  <xs:complexType>
    <xs:sequence>
      <xs:element name='name' type='xs:string' />
      <xs:element name='schule' type='xs:string' />
      <xs:element name='alter' type='xs:integer' minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- Elemente die weitere Elemente enthalten
  - „xs:complexType“ mit
    - „xs:sequence“ (verbindliche Reihenfolge)
    - „xs:choice“ (beliebige Reihenfolge)

```
<fortbildner>
  <name>Alfred E. Neumann</name>
  <schule>Mad Academy</schule>
  <alter>33</alter>
</fortbildner>
```



# XSD – Sequenz & Choice



- man kann auch feste und freie Reihenfolge mischen: das Element „test“ soll als erstes „a“ enthalten und danach beliebig oft „a“ oder „b“ ( als Regex:  $a[ab]^+$  )

choice  
innerhalb  
einer sequence

```
<xs:element name="test">
  <xs:complexType>
    <xs:sequence>
      <xs:element name='a' />
      <xs:choice maxOccurs='unbounded'>
        <xs:element name='a' />
        <xs:element name='b' />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<test>
  <a></a>
  <b></b>
  <a></a>
</test>
```

```
<test>
  <a></a>
  <a></a>
  <a></a>
  <b></b>
  <a></a>
</test>
```

# XSD Attribute



```
<xs:element name="tag" maxOccurs='unbounded'>
  <xs:complexType>
    <xs:sequence>
      <xs:element name='termin' minOccurs='0' maxOccurs='unbounded' />
      <xs:element name='thema' type='xs:string' maxOccurs='unbounded' />
    </xs:sequence>
    <xs:attribute name='dow' type='wochentag' use='optional' />
    <xs:attribute name='datum' type='xs:date' use='required' />
  </xs:complexType>
</xs:element>
```

- `<xs:attribute name='...' type='...' use='...' />`
- `use` kann `required` oder `optional` enthalten
- erfordert, dass ein Element als „complexType“ deklariert wird
- hat das Element eine „xs:sequence“, dann *müssen* die Attribute *dahinter* stehen

# XSD Attribute für Elemente ohne Kindelemente



- Elemente, die nur aus Attributen bestehen

```
<xs:element name="termin">
  <xs:complexType>
    <xs:attribute name='zeit' type='xs:time' />
  </xs:complexType>
</xs:element>
```

`<termin zeit='12:43:00' />`

- Für Element mit Wert und Attributen

```
<xs:element name="termin" >
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='xs:string'>
        <xs:attribute name='zeit' type='xs:time' />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

*Lösung:  
complexType mit  
simpleContent*

`<termin zeit='22:43:00'>Party !!</termin>`



# XSD Elemente wiederververwenden

- wenn man genau eine Art von Element immer wieder braucht, kann man es mehrfach verwenden

- durch Referenzierung mit **ref** (einfache Elemente)

- als eigenen Typ (complexType)

```
<xs:element name="ORT" type="xs:string" />
<xs:element name="PLZ" type="xs:int" />
<xs:element name="STRASSE" type="xs:string" />
<xs:complexType name="typADRESSE">
  <xs:sequence>
    <xs:element ref="STRASSE" />
    <xs:element ref="ORT" />
    <xs:element ref="PLZ" />
  </xs:sequence>
</xs:complexType>
<xs:element name="KUNDE">
  <xs:element name="VORNAME" type="xs:string" />
  <xs:element name="NACHNAME" type="xs:string" />
  <xs:element name="LIEFERADRESSE" type="typADRESSE" />
  <xs:element name="RECHNUNGSADRESSE" type="typADRESSE" />
</xs:element>
```

# Die XSD Datei



- die XSD Datei hat als root-Element das Element

```
<xs:schema version="1.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

  <xs:element name='wurzel'>
    ...
  </xs:element>
</xs:schema>
```

- Tools wie Netbeans generieren diesen Header automatisch

# XSD mit XML verknüpfen



- die Einbindung einer lokalen Datei findet als Attribut im root-Element der XML-Datei statt:

```
<fortbildung
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:noNamespaceSchemaLocation='./fobi1.xsd' >
  <!-- ... -->
</fortbildung>
```

- Die Location kann eine lokale Datei oder eine Webadresse sein

```
xsi:noNamespaceSchemaLocation='C:/beispiel/fobi.xsd'
```

```
xsi:noNamespaceSchemaLocation='http://web.wara.de/~mertens/fobi1.xsd'
```

- Alternativ: **xsi:schemaLocation** (→ namespace)  
<http://www.oracle.com/technetwork/articles/srivastava-namespaces-092580.html>

# XML Schema – Weitere Quellen

---



- <http://www.w3.org/TR/xmlschema-0/> (Grundlagen)
- <http://www.w3.org/TR/xmlschema11-1/> (Structures)
- <http://www.w3.org/TR/xmlschema11-12> (Datentypen)
- <http://blog.codeinside.eu/artikel/guide-xml-xml-schema-xsd-teil-1/>
- <http://www.codeproject.com/Articles/18426/XSD-Tutorial-Part-of-Elements-and-Attributes>

# Aufgabe 1 – XSD erstellen (20 min)



- Schreiben Sie ein XSD zur Validierung von Messung-XML Dateien. Zur Erinnerung nochmal die DTD und eine mögliche XML-Datei dazu.

```
<messung>
  <zeitpunkt datum='12.05.2013' zeit='12:53' />
  <ort>
    <stadt>Gelsenkirchen</stadt> <!-- könnte man weglassen-->
    <gps lat='51.4' long='6.9' />
  </ort>
  <messwert>
    <name>Temperatur</name>
    <einheit>°C</einheit>
    <wert>25.23</wert>
  </messwert>
  <messwert>
    <name>Luftdruck</name>
    <einheit prefix='m'>bar</einheit>
    <wert>1055</wert>
  </messwert>
  <messwert>
    <name>Strahlenbelastung</name>
    <fehlermeldung>Sensor nicht kalibriert</fehlermeldung>
  </messwert>
</messung>
```

```
<!DOCTYPE messung [
  <!ELEMENT messung (zeitpunkt,ort,messwert+)>
  <!ELEMENT zeitpunkt EMPTY>
  <!ATTLIST zeitpunkt
    zeit CDATA #REQUIRED
    datum CDATA #REQUIRED
  >
  <!ELEMENT ort (stadt?,gps)>
  <!ELEMENT stadt (#PCDATA)>
  <!ELEMENT gps EMPTY>
  <!ATTLIST gps
    long CDATA #REQUIRED
    lat CDATA #REQUIRED
  >
  <!ELEMENT messwert (name,einheit,wert)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT einheit (#PCDATA)>
  <!ELEMENT wert (#PCDATA)>
  <!ATTLIST einheit
    prefix (f|p|n|m|k|M|G|T) #IMPLIED
  >
]>
```



# Aufgabe 2 – XSD auswerten (15 min)



- Schreiben Sie eine XML Datei die „schema valid“ ist bezüglich des Schemas in der Datei **buechereibestand.xsd**

```
<xs:element name="büchereibestand">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="artikel" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="titel" />
            <xs:choice>
              <xs:element name="buch" type="typ_buch" />
              <xs:element name="cd" type="typ_cd" />
              <xs:element name="film" type="typ_film" />
            </xs:choice>
          </xs:sequence>
          <xs:attribute name="angeschafft"
            use="optional" type="xs:date"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Nur ein Ausschnitt  
aus der Datei !

# Aufgabe 3 - Vererbung (Fortgeschrittene)

---



- Die XSD aus Aufgabe 2 ist eigentlich ungeschickt. Logischer wäre es, einen `complexType` „artikel“ zu definieren und davon die `complexTypes` „cd“, „film“ und „buch“ abzuleiten (wie auf der Folie XSD-Elemente wiederverwenden)

Kopieren und verbessern Sie die Datei entsprechend.  
Schreiben Sie ein passendes Instanz-Dokument zu Ihrer XSD.



**Danke für Ihre Aufmerksamkeit  
Fragen ?**