Vorlesung
Component Ware und Web-Services

- Webservices -

12. Exkurs zu XML

Prof. Dr. Hans-Gert Gräbe, F. Schumacher
Wintersemester 2003/2004

Gliederung

1. **Einleitendes Beispiel**
2. Aufbau von XML Schema
   - Reihenfolge,
   - Datentypen
3. Namensräume
4. Bewertung

1. Einleitendes Beispiel

DTD des Beispiels

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT AdressDB (Adresse)*>
<!ELEMENT Adresse ((Firma | Person), (Strasse | Postfach)?, Ort,
   Telefon, eMail)>
<!ELEMENT Firma (#PCDATA)>
<!ELEMENT Person (Vorname+, Name)>
<!ATTLIST Person Anrede (Herr | Frau) #REQUIRED>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Vorname (#PCDATA)>
<!ELEMENT Strasse (#PCDATA)>
<!ATTLIST Strasse Nummer CDATA #IMPLIED>
<!ELEMENT Postfach (#PCDATA)>
<!ELEMENT Ort (#PCDATA)>
<!ATTLIST Ort PLZ CDATA #REQUIRED>
<!ELEMENT Telefon (#PCDATA)>
<!ELEMENT eMail (#PCDATA)>
```

1. Einleitendes Beispiel

All diese Vorgaben lassen sich mit DTDs nicht realisieren. Aus diesem Grund wurden verschiedene XML-Schemabeschreibungs-sprachen entwickelt (z. B. XDR von Microsoft). Zur Recommendation beim W3C hat es jedoch nur XML Schema gebracht.

1. Einleitendes Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
   <xs:element name="AdressDB">
      <xs:complexType>
         <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="Adresse" type="AdresseType"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:complexType name="AdresseType">
      <xs:sequence>
         <xs:choice>
            <xs:element ref="Firma"/>
            <xs:element name="Person" type="PersonType"/>
         </xs:choice>
         <xs:choice minOccurs="0">
            <xs:element name="Strasse" minOccurs="0">
               <xs:complexType>
                  <xs:simpleContent>
                     <xs:restriction base="StrasseType">
                        <xs:attribute name="Nummer">
                           <xs:simpleType>
                              <xs:restriction base="xs:string">
                                 <xs:pattern value="\d{1,3}[a-z]?"/>
                                 <xs:pattern value="\d{1,3}[a-z]?-\d{1,3}[a-z]?"/>
                              </xs:restriction>
                           </xs:simpleType>
                        </xs:attribute>
                     </xs:restriction>
                  </xs:simpleContent>
               </xs:complexType>
            </xs:element>
```

1. Einleitendes Beispiel

```
            <xs:element name="Postfach" minOccurs="0">
               <xs:simpleType>
                  <xs:restriction base="xs:integer">
                     <xs:totalDigits value="10"/>
                  </xs:restriction>
               </xs:simpleType>
            </xs:element>
         </xs:choice>
         <xs:element name="Ort" type="OrtType"/>
         <xs:element name="Telefon">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:pattern value="\+\d{1,3}-\(0\)\d{2,6}-\d{3,9}"/>
                  <xs:pattern value="\(0\d{2,6}\)\d{3,9}"/>
                  <xs:pattern value="\(0\d{2,6}\)\d{2,6}-\d{1,5}"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="eMail">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:pattern value=".+@.+\..{2,}"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:element name="Firma" type="xs:string"/>
   <xs:element name="Name" type="xs:string"/>
```

1. Einleitendes Beispiel

```
<xs:complexType name="OrtType">
   <xs:simpleContent>
      <xs:extension base="xs:string">
         <xs:attribute name="PLZ" use="required">
            <xs:simpleType>
               <xs:restriction base="xs:integer">
                  <xs:minInclusive value="00000"/>
                  <xs:maxInclusive value="99999"/>
                  <xs:totalDigits value="5"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
<xs:complexType name="PersonType">
   <xs:sequence>
      <xs:element ref="Vorname" maxOccurs="unbounded"/>
      <xs:element ref="Name"/>
   </xs:sequence>
   <xs:attribute name="Anrede" use="required">
      <xs:simpleType>
         <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="Herr"/>
            <xs:enumeration value="Frau"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
</xs:complexType>
```

1. Einleitendes Beispiel

```
    <xs:complexType name="StrasseType">
       <xs:simpleContent>
          <xs:extension base="xs:string">
             <xs:attribute name="Nummer" use="required">
                <xs:simpleType>
                   <xs:restriction base="xs:string">
                      <xs:pattern value="\d{1,3}"/>
                   </xs:restriction>
                </xs:simpleType>
             </xs:attribute>
          </xs:extension>
       </xs:simpleContent>
    </xs:complexType>
    <xs:element name="Vorname" type="xs:string"/>
</xs:schema>
```

Man erkennt sofort, dass XML-Schema-Dokumente zwar aussagekräftiger bzw. präziser sind als DTDs, jedoch einen wesentlich größeren Umfang haben.

Gliederung

1. Einleitendes Beispiel
2. **Aufbau von XML Schema**
   - Reihenfolge,
   - Datentypen
3. Namensräume
4. Bewertung

2. Deklaration vs. Definition

Deklarationen beschreiben Elemente und Attribute, die in einem Instanzdokument auftreten dürfen.

Beispiel:

```
<xs:element name="addressbook" type="Address">...
</xs:element>
```

Definitionen definieren Typen, die in Element- oder Attributdeklarationen verwendet werden können

Beispiel:

```
<xs:complexType name="Address">...
</xs:complexType>
```

Im Folgenden werden die wichtigsten Elemente der XML Schemadefinition vorgestellt!

2. Deklaration von Elementen in Schemas

xs:schema

- Dokumenten- oder Wurzelelement eines jeden XML-Schemas.
- Container für sämtliche Deklarationen und Definitionen sowie Platzhalter für eine Reihe von Standardwerten, die durch Attribute ausgedrückt werden können
- Maximal mit einem Namensraum verknüpft

Beispiel:

```
<xs:schema
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified">
      ...
      Hier folgen alle anderen Elemente
      ...
</xs:schema>
```

2. Deklaration von Elementen in Schemas

xs:element

- Definiert ein Element, in dem ihm ein Datentyp zugewiesen wird,
- Dies ist möglich mittels des Attributes "type" oder "inline" durch eine nachgestellte Typdefinition,
- Alternativ kann auf ein global definiertes Element verwiesen werden

Beispiel:

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

Gültige Instanz:

```
<ausleiher>Hans Mustermann</ausleiher>
```

- Elementgruppen können mit xs:group gebildet werden, wenn diese später an mehreren Stellen zum Einsatz kommen sollen
- Ideal um Bausteine für andere Schemata zu "erzeugen"
- Gruppen können durch xs:redefine geändert werden

2. Deklaration von Elementen in Schemas

- Einige mögliche Attribute von `xs:element`:
  - `id` – Element ID
  - `name` – Lokaler Name des Elementes
  - `type` – qualifizierter Name eines einfachen oder komplexen Typs
  - `ref` – Referenz auf eine globale Elementdefinition
  - `maxOccurs/minOccurs` – Maximale/Minimale Auftretenshäufigkeit
  - `default` – Standardwert des Elements
  - `fixed` – legt den Inhalt eines Elements unveränderlich fest
  - `fixed/default` – schließen sich gegenseitig aus und die Elemente müssen von einfachem Typ sein
  - `form` – gibt an, ob das Element in den Instanzdokumenten qualifiziert werden muss

2. Deklaration von Elementen in Schemas

Beispiel:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:group name="Adresse">
        <xs:sequence>
             <xs:element name="Name" type="xs:string"/>
             <xs:element name="Vorname" type="xs:string"/>
             <xs:element name="Strasse" type="xs:string"/>
        </xs:sequence>
   </xs:group>
   <xs:element name="TopLevel">
        <xs:complexType>
             <xs:group ref="Adresse"/>
        </xs:complexType>
   </xs:element>
</xs:schema>
```

2. Deklaration von Attributen in Schemas

`xs:attribute`

- Globale und lokale Definition möglich,
- Alle Attribute der obersten Ebene eines Schemas werden als global definiert angesehen,
- Lokale Definitionen dienen zwei unterschiedlichen Zwecken, zum einen als Verweis auf ein global definiertes Attribut oder eine lokale Attributdefinition,
- Beide Optionen schließen sich gegenseitig aus.
- Attribute:
  - `fixed` – der Wert des Attributes (wenn vorhanden) wird fixiert und gleich dem hier angegebenen Wert sein.
  - `id, name, type` – analog der vorher gezeigten Elemente
  - `form` – gibt an, ob das Attribut im Instanzdokument qualifiziert sein muss

Gruppen von Attributen werden analog zu Elementgruppen unter Verwendung von `xs:attributeGroup` erzeugt werden.

2. Deklaration von Attributen in Schemas

- Globales Attribut: Deklaration als direktes Sub-Element von "schema"
- Lokales Attribut: Deklaration in tieferer Schachtelungsebene
- Attribute können lokale oder globale Datentypen verwenden oder auf ein bereits spezifiziertes Attribut verweisen

- Globaler Datentyp

```
<attribute name="name" type="type" use="how-it-used" default/fixed="value".../>
```

einfacher Typ

nur in lokalen Attributen; Werte: required, optional, prohibited

nur relevant, wenn use nicht gesetzt

- Lokaler Datentyp

```
<attribute name="name" use="how-it-used" default/fixed="value".../>
   <simpleType>...</simpleType>
</attribute>
```

- Verweis auf Attribut (Zweck: Modularisierung, Wiederverwendung)

```
<attribute ref="name" use="how-its-use" default/fixed="value".../>
```

Quelle: Business Informatics Group, TU Wien

2. Deklaration von Attributen in Schemas


```
<complexType name="Modell">
    <sequence>
        <element name="Gewicht" type="string" minOccurs="1"
            maxOccurs="1"/>
        ...
    </sequence>
    <attribute name="name" use="required">
        <simpleType>
            <restriction base="string">
                <maxLength value="15"/> ...
            </restriction>
        </simpleType>
    </attribute>
    <attribute name="nr" type="hk:herstellerNr"
        use="required"/>
</complexType>
<simpleType name="herstellerNr"> ...
```

Quelle: Uni Würzburg

2. Deklaration von Elementen in Schemas

Kompositoren

- Kompositoren sind Container, in die eine Reihe von Elementen eingebettet werden kann.
- Die Container können als ganzes verarbeitet werden.
- Diese einfachen und flexiblen Strukturen ermöglichen es an mehreren Stellen wiederverwendbar zu sein.
- Die Definition muss benannt und global sein.

`xs:sequence` – Gruppe: Alle Elemente treten genau einmal in der angegebenen Reihenfolge

`xs:choice` - Gruppe: Ein beliebiges Element aus der Gruppe darf auftreten

`xs:all` - Gruppe: Alle Elemente treten höchstens einmal auf, wobei die Reihenfolge unwichtig ist

2. Deklaration von Elementen in Schemas

xs:sequence

- Kann innerhalb oder außerhalb einer Gruppe definiert werden.
- Komposition zur Definition einer geordneten Gruppe von Elementen.
- Attribute:
  - `id` – Element ID
  - `maxOccurs/minOccurs` – Maximale/minimale Auftretenshäufigkeit

Beispiel:

```
<xs:element name="bla">
   <xs:complexType>
        <xs:sequence>
               <xs:element name="blubb" minOccurs="0"
                      maxOccurs="unbounded"/>
               <xs:element name="blobb" minOccurs="10"
                      maxOccurs="unbounded"/>
        </xs:sequence>
   </xs:complexType>
</xs:element>
```

2. Deklaration von Elementen in Schemas

xs:choice

- Definition innerhalb und außerhalb einer Gruppe
- Außerhalb:
  - Ein Kompositor, bildet eine Gruppe sich gegenseitig ausschließender Partikel
- Innerhalb:
  - Analoge Funktion wie außerhalb, nur dass hier die Attribute `maxOccurs` und `minOccurs` verboten sind

Beispiel (lokal):

```
<xs:element name="bla">
   <xs:complexType>
        <xs:choice>
            <xs:element name="blubb"/>
            <xs:element name="blobb"/>
        </xs:choice>
   </xs:complexType>
</xs:element>
```

2. Deklaration von Elementen in Schemas

xs:all

- Kompositor zur Beschreibung einer ungeordneten Gruppe von Elementen
- Wird zur Beschreibung einer ungeordneten Gruppe von Elementen verwendet, die null oder einmal auftreten darf
- Attribute:
  - `id` – Element-ID
  - `maxOccurs` – Maximale Auftretenshäufigkeit, hier ist der Wert auf 1 fixiert

Beispiel:

```
<xs:group name="authorSubElements">
   <xs:all>
        <xs:element ref="name"/>
        <xs:element ref="born"/>
        <xs:element ref="dead" minOccurs="0"/>
   </xs:all>
   <xs:attribute ref="id"/>
</xs:group>
```

Gliederung

1. Einleitendes Beispiel
2. **Aufbau von XML Schema**
   - Reihenfolge,
   - **Datentypen**
3. Namensräume
4. Bewertung

2. Einfache Typen vs. Komplexe Typen

Neben komplexen Elementen lassen in XML-Schema auch einfache und komplexe Datentypen in verschiedenen Formen definieren.

Einfache, vorhandene Typen sind Strings, Zahlen, Zeiteinheiten usw., d.h. sie besitzen keine Kinderelemente oder Attribute.

Beispiel:

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

Komplexe Typen enthalten Kindelemente, Elementreferenzen oder Attribute und können von einem einfachen Typ abgeleitet werden.

Beispiel:

```
<xs:complexType name="Address">
    <xs:sequence>  ...</xs:sequence>
    <xs:attribute name="category" type="xs:string"/>
</xs:complexType>
```

Gliederung

1. Einleitendes Beispiel
2. Aufbau von XML Schema
   - Reihenfolge,
   - Datentypen
3. **Namensräume**
4. Bewertung

3. Namensräume

Im Gegensatz zu DTDs unterstützen XML-Schemata Namensräume.

XML-Schema kennt dabei drei Standard-Namensräume:

- *http://www.w3.org/2001/XMLSchema*
  Der Namensraum für XML-Schema.

- *http://www.w3.org/2001/XMLSchema-instance*
  Für XML-Schema-Komponenten in Instanzdokumenten

- *http://www.w3.org/2001/XMLSchema-datatypes*
  Enthält eine Kopie der eingebauten XML-Schema-Datatypen, diese dienen der Verwendung in alternativen Schemata, die die XML-Schema-Datentypen verwenden möchten

Es lassen sich auch eigene, sogenannte Ziel-Namenräume bilden!

3. Namensräume

Sichtweise:

- Ein Schema ist eine Sammlung von Typdefinitionen und Deklarationen von Elementen, welche zu einem bestimmten Namensraum – dem Ziel-Namensraum – gehören.
- Mittels Ziel-Namensräumen wird zwischen Definitionen und Deklarationen aus verschiedenen Vokabularen unterschieden,
  - z. B. zwischen dem in der XML Schema Sprachdefinition definierten `element` und einem in einem anderen Vokabular (z. B. aus dem Bereich der Chemie) definierten `element`.

Zur Schema-Validierung (d.h. zur Überprüfung, ob ein Instanzdokument einem/mehreren Schemata entspricht) ist es nötig, die im Dokument benutzten Attribute und Elemente den entsprechenden Deklarationen und Definitionen im dazugehörigen Schema zuzuordnen.

Dies geschieht hauptsächlich genau über diese Ziel-Namensräume.

3. Namensräume

Deklaration:

Der Ziel-Namensraum wird durch eine entsprechende Deklaration mittels der Attribute von `<xs:schema>` gebildet.

Beispiel:

```
<xs:schema
   xmlns="http://www.w3.org/2001XMLSchema"
   xmlns:bs="http://bsp.de/bestellung"
   targetNamespace="http://bsp.de/bestellung"
   elementFormDefault="qualified"
   attributeFormDefault="qualified">
   ...
</xs:schema>
```

Quelle: Vonhoegen

3. Namensräume

- `targetNamespace="http://bsp.de/bestellung"`
  gibt an, welcher Namensraum durch das Schema selbst beschrieben wird. Die Angabe eines Ziel-Namensraumes ist optional.

- `xmlns:bs="http://bsp.de/bestellung"`
  deklariert einen Namensraum und weist ihm das Präfix `bs` zu.

- `elementFormDefault="qualified"` und `attributeFormDefault="qualified"`
  legen einen Standardwert für die `form` Attribute von `xs:attribute` und `xs:element` fest. Der Default-Wert hierzu ist `unqualified`.

Die Festlegungen zur Qualifizierung lassen sich auch explizit für einzelne Elemente und Attribute mittels des Attributes `form` treffen:

```
<attribute name="telefonnr" type="integer" form="qualified"/>
```

In der Dokumentinstanz muss dieses Attribut dann mit dem entsprechenden Präfix ausgezeichnet werden.

3. Namensräume

Beispiel: Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://bsp.de/artikel" xmlns:art="http://bsp.de/artikel"
    xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified"
    attributeFormDefault="unqualified">
    <element name="Artikel" type="art:ArtikelTyp"/>
    <complexType name="ArtikelTyp">
            <sequence>
                <element name="Bezeichnung" type="string"/>
                <element name="ArtikelNr" type="integer"/>
                <element name="Preis" type="integer"/>
            </sequence>
    </complexType>
</schema>
```

Valides Instanzdokument dazu:

```
<?xml version="1.0" encoding="UTF-8"?>
<art:Artikel xmlns:art="http://bsp.de/artikel"
            xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
            xsi:schemaLocation="http://bsp.de/artikel/art.xsd">
    <Bezeichnung>Nasenhaarschneider Millennium</Bezeichnung>
    <ArtikelNr>12345</ArtikelNr>
    <Preis>550</Preis>
</art:Artikel>
```

3. Namensräume

Änderungen am Schema:

```
   elementFormDefault="qualified" attributeFormDefault="qualified">
```

Jetzt ist das Instanzdokument auf der vorigen Seite nicht mehr valide.

Valides Instanzdokument:

```
<?xml version="1.0" encoding="UTF-8"?>
<art:Artikel ... >
   <art:Bezeichnung>Nasenhaarschneider Millennium </art:Bezeichnung>
   <art:ArtikelNr>12345</art:ArtikelNr>
   <art:Preis>550</art:Preis>
</art:Artikel>
```

4. Bewertung

- Einschränkungen von DTDs:
  - Unterstützen nur einen Datentyp für Elemente (PCDATA),
  - Ebenso für Attribute (CDATA),
  - Unterstützen keine Namensräume,
  - Benötigen einen eigenen Parser

- Vorteile von Schemas:
  - Verschiedene Datentypen bereits vorgegeben
  - Datentypen sind erweiterbar
  - Schemas sind selbst wieder XML-Dokumente (XML Schema muss irgendwie einmal formal beschrieben worden sein → nicht normative Schema-DTD)
  - Kein zusätzlicher Parser nötig
  - Eingebaute Unterstützung von Namensräumen
  - Sehr flexibel
  - Unterstützen Wiederverwendung

Literatur

- E. van der Vlist: *XML Schema. O'Reilly, 2003*
- *http://www.w3.org/XML/Schema*