

# XML – Grammatiken und XForms

von Astrid Sackel

im Rahmen des Seminars

**XML und intelligente Systeme**

bei Sebastian Wrede und Ingo Lütkebohle

Uni Bielefeld

Wintersemester 2005|06

31. Oktober 2005

## *Wozu eigentlich Grammatiken?*

## *Wozu eigentlich Grammatiken?*

Fallbeispiel:

*„Hans und Peter arbeiten an einer Adressbuch-Datenbank, sie haben die ähnliche Vorstellung von der Struktur, aber nicht exakt dieselbe“*

## Wozu eigentlich Grammatiken?

Fallbeispiel:

*„Hans und Peter arbeiten an einer Adressbuch-Datenbank, sie haben die ähnliche Vorstellung von der Struktur, aber nicht exakt dieselbe“*



## Wozu eigentlich Grammatiken?

Fallbeispiel:

*„Hans und Peter arbeiten an einer Adressbuch-Datenbank, sie haben die ähnliche Vorstellung von der Struktur, aber nicht exakt dieselbe“*

*„Peter muss seine Dateien bearbeiten, Hans muss die Eingaben seiner Freunde korrigieren / validieren“*



## DTD

### Document Type Definition

```

<!ELEMENT addressbook (description,contact*)>

<!ELEMENT description ANY>

<!ELEMENT contact (firstname,lastname,fon*,address,comment
?)>

<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>

<!ELEMENT fon EMPTY?>
<!ATTLIST fon type CDATA #IMPLIED
              number CDATA #REQUIRED>

<!ELEMENT address (street,plz)>

<!ELEMENT comment (#PCDATA)>

```

addressbook.dtd

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE addressbook SYSTEM „addressbook.
dtd“>
<addressbook>
  <description>
    Some addresses.
  </description>
  <contact>
    <firstname>Berta</firstname>
    <lastname>Fischmac</lastname>
    <fon type="mobil" number="030778034" />
    ...
    <address>
      <street>Wallstreet
    </street>
      <plz>90234</plz>
    </address>
    <comment>
      Just available in the evening
    </comment>
  </contact>
  ...
</addressbook>

```

Verweis auf DTD in addressbook.xml

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
„http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd“>
<html>
  <head>
    <title>DTDs in HTML</title>
    <meta http-equiv="content-type" content="text/html;
charset=iso-8859-1" />
  </head>
  <body>
    <h1>DTDs befinden sich im head einer html-Datei.</h1>
  </body>
</html>
```

Verweis auf DTD von XHTML 1.0

## DTD

Document Type Definition  
bekannt aus html-head

### Nachteile

- keine Verwendung eigener Datentypen
- Vorgegebene Datentypen:
  - enumeration | simple kinds of tokens
- Verwendung eigener Syntax
- zu Allgemein

### Stärke

- Kürze (Downloadzeit)

# XML Schema

Ziel: **Beschreibung beliebiger Datenstrukturen ermöglichen**

Datentypen werden in Hierarchien aufgebaut und beschränkt oder erweitert

import- / include-System ermöglicht Modularisierung / Kapselung

## XML Schema Struktur

xsd:schema

  xsd:import (*optional*)

  xsd:include (*optional*)

  xsd:annotation

    xsd:documentation

  (*Reihe von Subelementen, Globale Elemente*)

  xsd:element

  xsd:complexType

  xsd:simpleType

## XML Schema Struktur

xsd:schema

- xsd:import (*optional*)

- xsd:include (*optional*)

- xsd:annotation

  - xsd:documentation

*(Reihe von Subelementen, Globale Elemente)*

- xsd:element

- xsd:complexType

- xsd:simpleType

*jedes Element im Schema hat das Präfix **xsd:***

*referenziert Namespace für xmlns:xsd="http://www.w3.org/2001/XMLSchema"*

## Komplexe Typen

### complexType-Element

Elemente können Subelemente  
enthalten oder Attribute tragen

```
<xsd:complexType name="BestellungTyp">
  <xsd:sequence>
    <xsd:element name="Lieferadresse"
      type="DeAdresse"/>
    <xsd:element name="Rechnungsadresse"
      type="DeAdresse"/>
    <xsd:element ref="Kommentar" minOccurs="0"/>
    <xsd:element name="Waren" type="WarenTyp"/>
  </xsd:sequence>
  <xsd:attribute name="bestelldatum" type="xsd:date"/>
</xsd:complexType>
```

Komplexe Typen in XML Schema: Definition von BestellungTyp

## Komplexe Typen

### complexType-Element

Elemente können Subelemente  
enthalten oder Attribute tragen

```
<xsd:complexType name="BestellungTyp">
  <xsd:sequence>
    <xsd:element name="Lieferadresse"
      type="DeAdresse"/>
    <xsd:element name="Rechnungsadresse"
      type="DeAdresse"/>
    <xsd:element ref="Kommentar" minOccurs="0"/>
    <xsd:element name="Waren" type="WarenTyp"/>
  </xsd:sequence>
  <xsd:attribute name="bestelldatum" type="xsd:date"/>
</xsd:complexType>
```

Komplexe Typen in XML Schema: Definition von BestellungTyp

*Deklarationen selbst sind **Assoziationen** zwischen Namen  
und Beschränkungen, **keine Typen!***

**ref-Attribute** können nur auf **globale Elemente** verweisen  
(d.h. direkte Kindelemente vom schema-Element)

## Häufigkeitsbeschränkungen

```
<xsd:complexType name="WarenTyp">
  <xsd:sequence>
    <xsd:element name="Buch" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="Titel" type="xsd:string"/>
          <xsd:element name="Anzahl">
            <xsd:simpleType>
              <xsd:restriction base="xsd:positiveInteger">
                <xsd:maxExclusive value="100" default="1"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="PreisEUR" type="xsd:decimal"/>
          <xsd:element ref="Kommentar" minOccurs="0"/>
          <xsd:element name="Lieferdatum" type="xsd:date" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="ISBN" type="ISBNTyp" use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

Häufigkeitsbeschränkungen:  
minOccurs, maxOccurs,  
use, default

### minOccurs- / maxOccurs-Attribut

erlaubte Werte von 0-unbounded

## Häufigkeitsbeschränkungen

```
<xsd:complexType name="WarenTyp">
  <xsd:sequence>
    <xsd:element name="Buch" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="Titel" type="xsd:string"/>
          <xsd:element name="Anzahl">
            <xsd:simpleType>
              <xsd:restriction base="xsd:positiveInteger">
                <xsd:maxExclusive value="100" default="1"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="PreisEUR" type="xsd:decimal"/>
          <xsd:element ref="Kommentar" minOccurs="0"/>
          <xsd:element name="Lieferdatum" type="xsd:date" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="ISBN" type="ISBNTyp" use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

Häufigkeitsbeschränkungen:  
minOccurs, maxOccurs,  
use, default

### use-Attribut

erlaubte Werte: optional | required | prohibited

## Häufigkeitsbeschränkungen

```
<xsd:complexType name="WarenTyp">
  <xsd:sequence>
    <xsd:element name="Buch" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="Titel" type="xsd:string"/>
          <xsd:element name="Anzahl">
            <xsd:simpleType>
              <xsd:restriction base="xsd:positiveInteger">
                <xsd:maxExclusive value="100" default="1"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="PreisEUR" type="xsd:decimal"/>
          <xsd:element ref="Kommentar" minOccurs="0"/>
          <xsd:element name="Lieferdatum" type="xsd:date" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="ISBN" type="ISBNTyp" use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

Häufigkeitsbeschränkungen:  
minOccurs, maxOccurs,  
use, default

### default-Attribut

Vorgabe-Werte für Attribut (in Kombi mit use="optional") oder Element

## Häufigkeitsbeschränkungen

```
<xsd:complexType name="DeAdresse">
  <xsd:sequence>
    <xsd:element name="Name" type="xsd:string"/>
    <xsd:element name="Straße" type="xsd:string"/>
    <xsd:element name="Ort" type="xsd:string"/>
    <xsd:element name="PLZ" type="xsd:decimal"/>
  </xsd:sequence>
  <xsd:attribute name="land" type="xsd:NMTOKEN" fixed="DE"/>
</xsd:complexType>
```

Häufigkeitsbeschränkungen:  
fixed

### fixed-Attribut

fester Vorgabe-Wert

default und fixed schließen sich gegenseitig aus

```
<xsd:simpleType name="meinInteger">
  <xsd:restriction base="xsd:integer">
    <xsd:min-inclusive value="10000"/>
    <xsd:max-inclusive value="99999"/>
  </xsd:restriction>
</xsd:simpleType>
```

simpleType: Definition von meinInteger, eingeschränkter Wertebereich

## Einfache Typen

- **simpleType-Elemente** enthalten nur Zahlen, Zeichenketten, Datum, ...  
**keine Attribute, keine Subelemente**
- können z.B. **vordefinierte Basistypen** (z.B. string, integer, unsignedLong, duration, anyURI, language, ...) oder
- **definierte simpleTyp'en** durch Ableitung einschränken (**restriction, base**)

```
<xsd:simpleType name="meinInteger">
  <xsd:restriction base="xsd:integer">
    <xsd:min-inclusive value="10000"/>
    <xsd:max-inclusive value="99999"/>
  </xsd:restriction>
</xsd:simpleType>
```

simpleType: Definiton von meinInteger, eingeschränkter Wertebereich

```
<xsd:simpleType name="ISBNTyp">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{9}[0-9X]"/>
  </xsd:restriction>
</xsd:simpleType>
```

Definiton von ISBNTyp, Muster-Wert-Definition durch regulären Ausdruck

```
<xsd:simpleType name="Bundesland">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Baden-Württemberg"/>
    <xsd:enumeration value="Bayern"/>
    <xsd:enumeratin value="Berlin"/>
    <!-- usw. -->
    <xsd:enumeration value="Thüringen"/>
  </xsd:restriction>
</xsd:simpleType>
```

Definiton von Bundesland, unter Verwendung der Enumeration-Facette

## Einfache Typen

- **simpleType-Elemente** enthalten nur Zahlen, Zeichenketten, Datum, ...  
**keine Attribute, keine Subelemente**
- können z.B. **vordefinierte Basistypen** (z.B. string, interger, unsignedLong, duration, anyURI, language, ...) oder **definierte simpleTyp'en** durch Ableitung **einschränken** (restriction, base)

## Attribute haben immer einfache Typen

Einfache Typen bestehen aus **Atomaren Typen, Listentypen oder Vereinigungstypen**

## Atomare Typen

sind unteilbar (z.B. „DE“)

Einfache Typen bestehen aus **Atomaren Typen**, **Listentypen** oder **Vereinigungstypen**

## Atomare Typen

sind unteilbar (z.B. „DE“)

## Listentypen

Folge von Atomaren Typen (z.B. NMTOKENS, IDREFS, ENTITIES, „de us en“, „Astrid|Basti|Casper“)  
oder neue Listentypen durch Ableitung von Atomaren Typen

```
<xsd:simpleType name="BundeslandListe">  
  <xsd:list itemType="Bundesland"/>  
</xsd:simpleType>  
  
<xsd:simpleType name="SechsBundesländer">  
  <xsd:restriction base="BundeslandListe">  
    <xsd:length value="6"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

Listentypen aus Atomaren Typen

```
<sechsLänder>  
  Sachsen  
  Baden-Württemberg  
  Rheinland-Pfalz  
  Brandenburg  
  Niedersachsen  
  Hessen  
</sechsLänder>
```

SechsBundesländer in XML Instanz

Einfache Typen bestehen aus **Atomaren Typen**, **Listentypen** oder **Vereinigungstypen**

## Atomare Typen

sind unteilbar (z.B. „DE“)

## Listentypen

Folge von Atomaren Typen (z.B. NMTOKENS, IDREFS, ENTITIES. „de us en“, „Astrid|Basti|Casper“)  
oder neue Listentypen durch Ableitung von Atomaren Typen

### Listenfacetten

length-Element | minLength- / maxLength-Element | enumeration-Element

```
<xsd:simpleType name="BundeslandListe">
  <xsd:list itemType="Bundesland"/>
</xsd:simpleType>

<xsd:simpleType name="SechsBundesländer">
  <xsd:restriction base="BundeslandListe">
    <xsd:length value="6"/>
  </xsd:restriction>
</xsd:simpleType>
```

Listentypen aus Atomaren Typen

```
<sechsLänder>
  Sachsen
  Baden-Württemberg
  Rheinland-Pfalz
  Brandenburg
  Niedersachsen
  Hessen
</sechsLänder>
```

SechsBundesländer in XML Instanz

## Inhaltsmodelle

```

<xsd:complexType name="BestellungTyp">
  <xsd:sequence>
    <xsd:choice>
      <xsd:group ref="verschiedeneAdr"/>
      <xsd:element name="eineAdresse" type="DeAdresse"/>
    </xsd:choice>
    <xsd:element ref="Kommentar" minOccurs="0"/>
    <xsd:element name="Waren" type="WarenTyp"/>
  </xsd:sequence>
  <xsd:attribute name="bestelldatum" type="xsd:date"/>
</xsd:complexType>

<xsd:group name="verschiedeneAdr">
  <xsd:sequence>
    <xsd:element name="Lieferadresse" type="DeAdresse"/>
    <xsd:element name="Rechnungsadresse"
type="DeAdresse"/>
  </xsd:sequence>
</xsd:group>

```

Inhaltsmodelle: sequence, choice, group / all

- **choice-Element**  
nur ein Element in der Instanz
- **group-Element**  
zusammengefasste Elemente
- **all-Element**  
Gruppe mit beliebiger Reihenfolge  
Elemente müssen 0 oder 1 mal  
auftauchen  
nur auf oberster Ebene des Inhalts-  
modells möglich
- **sequence-Element**  
bestimmte Reihenfolge einer Gruppe

## Inhaltsmodelle

```

<xsd:complexType name="BestellungTyp">
  <xsd:sequence>
    <xsd:choice>
      <xsd:all ref="verschiedeneAdr"/>
      <xsd:element name="eineAdresse" type="DeAdresse"/>
    </xsd:choice>
    <xsd:element ref="Kommentar" minOccurs="0"/>
    <xsd:element name="Waren" type="WarenTyp"/>
  </xsd:sequence>
  <xsd:attribute name="bestelldatum" type="xsd:date"/>
</xsd:complexType>

<xsd:all name="verschiedeneAdr">
  <xsd:sequence>
    <xsd:element name="Lieferadresse" type="DeAdresse"/>
    <xsd:element name="Rechnungsadresse"
type="DeAdresse"/>
  </xsd:sequence>
</xsd:all>

```

Inhaltsmodelle: sequence, choice, group / all

*theoretisch statt xsd:group im Beispiel xsd:all setzen*

*Frage: Ist das so korrekt?*

- **choice-Element**  
nur ein Element in der Instanz
- **group-Element**  
zusammengefasste Elemente
- **all-Element**  
Gruppe mit beliebiger Reihenfolge  
Elemente müssen 0 oder 1 mal  
auftauchen  
nur auf oberster Ebene des Inhalts-  
modells möglich
- **sequence-Element**  
bestimmte Reihenfolge einer Gruppe

## Vorteil von XML Schema

- unterstützt Datentypen
- in XML Syntax
- Modularisierung
- weite Verbreitung und Integrität von XML Schema

# RelaxNG

*„Abseits vom Mainstream existiert eine Alternative: das ebenso einfach zu lesende wie zu schreibende Relax NG“*

aus dem Artikel

„Entspannung pur – Schema-Sprache Relax NG“  
von Marcel Tily und Stefan Tilkov im Januar 2004

# RelaxNG

vom W3C innerhalb von Spezifikationen verwendet  
(RDF, XHTML2)

*„Abseits vom Mainstream existiert eine Alternative: das ebenso einfach zu lesende wie zu schreibende Relax NG“*

aus dem Artikel

„Entspannung pur – Schema-Sprache Relax NG“  
von Marcel Tily und Stefan Tilkov im Januar 2004

```
<html>
  <head>
    <title>Example Address Book</title>
  </head>
  <body>
    <table class="addressBook">
      <tr class="card">
        <td class="name">
          <span class="givenName">John</span>
          <span class="familyName">Smith</span>
        </td>
        <td class="email">js@example.com</td>
      </tr>
    </table>
  </body>
</html>
```

XML-Instanz für Adressbuch

```

<grammar>

  <start>
    <ref name="html"/>
  </start>

  <define name="html">
    <element>
      <element name="head">
        <element name="title">
          <text/>
        </element>
      </element>
      <element name="body">
        <element name="table">
          <attribute name="class">
            <value>addressBook</value>
          </attribute>
          <oneOrMore>
            <element name="tr">
              <attribute name="class">
                <value>card</value>
              </attribute>
              <element name="td">
                <attribute name="class">
                  <value>name</value>
                </attribute>
                <interleave>
                  <text/>
                </interleave>
              </element>
            </element>
          </oneOrMore>
        </element>
      </element>
    </define>
  </grammar>

  <element name="span">
    <attribute name="class">
      <value>givenName</value>
    </attribute>
    <text/>
  </element>
</optional>
<optional>
  <element name="span">
    <attribute name="class">
      <value>familyName</value>
    </attribute>
    <text/>
  </element>
</optional>
</interleave>
</element>
<element name="td">
  <attribute name="class">
    <value>email</value>
  </attribute>
  <text/>
</element>
</element>
</oneOrMore>
</element>
</element>
</element>
</define>
</grammar>

```

RelaxNG Schema in XML für Adressbuch

```

<grammar>

  <start>
    <ref name="html"/>
  </start>

  <define name="html">
    <element>
      <element name="head">
        <element name="title">
          <text/>
        </element>
      </element>
      <element name="body">
        <element name="table">
          <attribute name="class">
            <value>addressBook</value>
          </attribute>
          <oneOrMore>
            <element name="tr">
              <attribute name="class">
                <value>card</value>
              </attribute>
              <element name="td">
                <attribute name="class">
                  <value>name</value>
                </attribute>
                <interleave>
                  <text/>
                </interleave>
              </element>
            </element>
          </oneOrMore>
        </element>
      </element>
    </define>
  </grammar>

```

```

        <element name="span">
          <attribute name="class">
            <value>givenName</value>
          </attribute>
          <text/>
        </element>
      </optional>
    </optional>
  <element name="span">

```

### Anmerkung zum Aufbau:

#### 3 Basis-Pattern:

- **text**
  - **attribute**: kann nur text enthalten
  - **element**: kann alle drei enthalten
- Wurzelement **grammar**
  - Subelement **start** (mit Definition des Wurzelements der Instanz)

RelaxNG Schema in XML für Adressbuch

```
element html {
  element head {
    element title { text }
  },
  element body {
    element table {
      attribute class { „addressBook“ },
      element tr {
        attribute class { „card“ },
        element td {
          attribute class { „name“ },
          mixed {
            element span {
              attribute class { „givenName“ },
              text
            }?,
            element span {
              attribute class { „familyName“ },
              text
            }?
          }
        },
        element td {
          attribute class { „email“ },
          text
        }
      }+
    }
  }
}
```

## RelaxNG Compact

*Anmerkung zu RelaxNG Compact:*

Problem: „Vor lauter tags sieht man den Inhalt nicht“

Lösung: **alternative kompaktere Syntax**

Vorteil: **einfachere Schreibweise**

Nachteil: **kein XML**

Lösung: Transformations-Tools wie trang

RelaxNG Compact Schema für Adressbuch

## Stärke von RelaxNG

- Vereinigung von Dokumenten verschiedener Typen  
(Zugriff auf versch. Versionen, versch. Schema Varianten möglich)  
im Klartext: **Zugriff auf externe Typsysteme möglich (z.B. XML Schema)**
- Ausdrucksfähigkeit / Mächtigkeit
- unterstützt Seiteneffekt-Bedingungen (Co-Occurrence Constraints)
- durch das einfachere, klarere Modell (Compact Syntax)  
ist eine Schema-Definition von Hand möglich

## Schwäche von RelaxNG

- beschränkte Aufmerksamkeit

# XForms

Grammatiken in der Anwendung

Situation heute: *Formulare ermöglichen Interaktion mit User, allgegenwärtig im Internet*

aber form-Element hat viele Nachteile

zur **Datenvalidierung** wird **Javascript** verwendet (umfangreiche und unübersichtliche Skripte)

**hohe Serverbelastung** bei Validierung über Server

...

```

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://www.w3.org/2002/xforms">
<head>
<title>XForms form</title>
<f:model>
  <f:instance>
    <f:person>
      <f:fname/>
      <f:lname/>
    </f:person>
  </f:instance>
  <f:submission id="form1" method="get"
    action="http://example.com/submit"/>
</f:model>
</head>

<body>
<f:input ref="fname">
<f:label>First Name</f:label></f:input>
<br />
<f:input ref="lname">
<f:label>Last Name</f:label></f:input>
<br />
<br />
<f:submit submission="form1">
<f:label>Submit</f:label></f:submit>
</body>
</html>

```

XForms Aufbau, Einfaches Texteingabe Formular mit Namespace

## Möglichkeiten von XForms

```

<html>
<head><title>html form</title></head>
<body>
  <form action="http://example.com/
    submit" method="get">
    <label>First Name</label>
    <input type="text" name="fname" />
    <label>Last Name</label>
    <input type="text" name="lname" />
    <input type="submit"
      value="Submit">
  </form>
</body>
</html>

```

Einfaches html Such-form ohne Javascript (Datenvalidierung, ...)

```

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://www.w3.org/2002/xforms">
<head>
<title>XForms form</title>
<f:model>
  <f:instance>
    <f:person>
      <f:fname/>
      <f:lname/>
    </f:person>
  </f:instance>
  <f:submission id="form1" method="get"
    action="http://example.com/submit"/>
</f:model>
</head>

<body>
<f:input ref="fname">
<f:label>First Name</f:label></f:input>
<br />
<f:input ref="lname">
<f:label>Last Name</f:label></f:input>
<br />
<br />
<f:submit submission="form1">
<f:label>Submit</f:label></f:submit>
</body>
</html>

```

XForms Aufbau, Einfaches Texteingabe Formular mit Namespace

## Möglichkeiten von XForms

- Trennung zwischen Inhalt und Layout (Model – UserInterface)
- aus XML in XML

```

<person>
  <fname>Hege</fname>
  <lname>Refsnes</lname>
</person>

```

XForms Ausgabe als XML

```
fname=Hege;lname=Refsnes
```

XForms Ausgabe als Text

- Dateneingabevalidierung beim Client, Serverentlastung, weniger Javascript

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://www.w3.org/2002/xforms">
<head>
<title>XForms form</title>
<f:model>
  <f:instance>
    <f:person>
      <f:fname/>
      <f:lname/>
    </f:person>
  </f:instance>
  <f:submission id="form1" method="get"
    action="http://example.com/submit"/>
</f:model>
</head>

<body>
<f:input ref="fname">
<f:label>First Name</f:label></f:input>
<br />
<f:input ref="lname">
<f:label>Last Name</f:label></f:input>
<br />
<br />
<f:submit submission="form1">
<f:label>Submit</f:label></f:submit>
</body>
</html>
```

XForms Aufbau, Einfaches Texteingabe Formular mit Namespace

## Möglichkeiten von XForms

zahlreiche Ereignisse vordefiniert:

- Eingabevalidierung
- Meldungen ausgeben
- Datentypdefinitionen in XML Schema
- neue Datentyp-Elemente wie Datum (Kalender-Funktion), SMS, ...
- Wiederbenutzung in verschiedenen Kontexten
- Wiederbenutzung auf verschiedenen Ausgabegeräten

## Kontrollstrukturen

angeben im **bind-Element** im model  
(instance-Element muss vorhanden sein)

- **required**: Pflichtfeld
- **constraint**: Geburtsjahr liegt vor Todesjahr
- **type**: z.B: integer
- **calculate**: z.B. total sum
- **relevant**: graut Kreditkarte-Nummer Feld aus, wenn nicht per Kreditkarte bezahlt wird

...

```
<model>
  <instance>
    <data xmlns=""><q/></data>
  </instance>
  <bind nodeset="q"
        required="true()"/>
  <submission .../>
</model>
```

required controls, in html form nur über Javascript

```
<bind nodeset="state"
      required="../country='USA'"/>
```

required controls mit XPath

state-Wert nötig, wenn country-Wert USA ist

## Kontrollstrukturen

angeben im **bind-Element** im model

(instance-Element muss vorhanden sein)

- **required**: Pflichtfeld
- **constraint**: Geburtsjahr liegt vor Todesjahr
- **type**: z.B: integer
- **calculate**: z.B. total sum
- **relevant**: graut Kreditkarte-Nummer Feld aus, wenn nicht per Kreditkarte bezahlt wird

```
<html
xmlns:xf="http://www.w3.org/2002/xforms"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...
<xf:instance>
<person xmlns="">
  <fname xsi:type="xsd:string"/>
  <lname xsi:type="xsd:string"/>
  <born xsi:type="xsd:date"/>
  <size xsi:type="xsd:integer"/>
</person>
<bind nodeset="homepage" type="xsd:anyURI"/>
</xf:instance>
```

```
<model>
  <instance>
    <data xmlns=""><q/></data>
  </instance>
  <bind nodeset="q"
    required="true()"/>
  <submission .../>
</model>
```

required controls, in html form nur über Javascript

```
<bind nodeset="state"
  required="../country='USA'"/>
```

required controls mit XPath

state-Wert nötig, wenn country-Wert USA ist

XML Schema Datentyp-Definition in XForms

## Nachteil von XForms

durch umfangreiche Spezifikation langes Warten bis zur verbreiteten Unterstützung erforderlich

## persönliches Fazit

dennoch großer Vorteil (Eingabevalidierung, Vereinfachung)  
durch Verwendung von Grammatiken ermöglicht

*„Hans und Peter müssen Adressen nicht mehr von selbst überprüfen, weniger Serverbelastung, weniger Javascript, weniger Aufwand.“*

# Noch Fragen?

## Noch Fragen?

Vielen Dank für die Aufmerksamkeit.

## Diskussion

*Wir stellen uns vor, wir wollten nun tatsächlich ein Adressbuch bauen, in dem wir unsere gesammelten Adressen verwalten können.*

### 1. Situation

Das Adressbuch hat mehrere Einträge mit jeweils Vorname, Nachname, Straße, Hausnummer, PLZ, Ort, Land, Geburtstag optional, ....

*Frage: Welche Grammatik kann man hier verwenden? Stelle einzelne Vor- / Nachteile heraus.*

## Diskussion

*Wir stellen uns vor, wir wollten nun tatsächlich ein Adressbuch bauen, in dem wir unsere gesammelten Adressen verwalten können.*

### 2. Situation

Firmen-Adressbuch wird mit Eingabe über XForms realisiert.

*Frage: Welche Grammatik kann man hier verwenden? Stelle einzelne Vor- / Nachteile heraus.*

## Diskussion

*Welches Schema wofür?*

*Nach dem Lesen des Artikels von Rick Jelliffe frage ich mich: wieviel Schema braucht die Welt?*

*Wie wichtig sind jetzt Grammatiken für euch persönlich? Pflicht oder Kür?*

## Quellen

Das XML Kompendium

<http://www.w3.org/MarkUp/Forms/2003/xforms-for-html-authors.html>

<http://www.edition-w3c.de/TR/2001/REC-xmlschema-0-20010502/>

<http://www.w3schools.com/xforms/default.asp>

<http://xml.fh-augsburg.de/Schema/Schema.rst/>

```
<xsd:simpleType name="meinInteger">
  <xsd:restriction base="xsd:integer">
    <xsd:min-inclusive value="10000"/>
    <xsd:max-inclusive value="99999"/>
  </xsd:restriction>
</xsd:simpleType>
```

simpleType: Definiton von meinInteger, eingeschränkter Wertebereich

```
<xsd:simpleType name="ISBNTyp">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{9}[0-9X]"/>
  </xsd:restriction>
</xsd:simpleType>
```

Definiton von ISBNTyp, Muster-Wert-Definition durch regulären Ausdruck

```
<xsd:simpleType name="Bundesland">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Baden-Württemberg"/>
    <xsd:enumeration value="Bayern"/>
    <xsd:enumeratin value="Berlin"/>
    <!-- usw. -->
    <xsd:enumeration value="Thüringen"/>
  </xsd:restriction>
</xsd:simpleType>
```

Definiton von Bundesland, unter Verwendung der Enumeration-Facette

## Facetten

- **restriction-Element:**  
um Facetten zu bestimmen
- **minInclusive- / maxInclusive- / minExclusive- / maxExclusive-Attribut:**  
um Wertebereich zu definieren
- **pattern-Element:**  
für Muster-Ausdrücke
- **enumeration-Element:**  
kann Werte jeden Typs beschränken,  
ausser boolean

## Erweiterung

```
<xsd:element name="PreisInternational">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="währung"
          type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

Ableitung eines komplexen Typs von einem einfachen Typ

## Erweiterung

```

<complexType name="Adresse">
  <sequence>
    <element name="Name" type="string"/>
    <element name="Straße" type="string"/>
    <element name="Ort" type="string"/>
  </sequence>
</complexType>
...
<complexType name="USAAdresse">
  <complexContent>
    <extension base="ibest:Adresse">
      <sequence>
        <element name="Bundesstaat" type="ibest:USBundesStaat"/>
        <element name="PLZ" type="positiveInteger"/>
      </sequence>
      <attribute name="exportCode" type="positiveInteger" fixed="1"/>
    </extension>
  </complexContent>
</complexType>

```

## Ableitung durch Erweiterung eines komplexen Typs

```

<ibest:Bestellung
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ibest="http://www.example.com/IBEST"
bestelldatum="1999-12-10">
  <Lieferadresse exportCode="1" xsi:type="ibest:USAAdresse">
    ...

```

## Kennzeichnung des verwendeten Typs in der Instanz