27.8 Exkurs XSD-Schema

Daten kann man auf 1000 und eine Art verpacken und weitergeben. Aber nur XML und XSD-Schema bilden das moderne Duo, mit dem man gültige Daten weitergeben kann und ebenso die bearbeiteten Daten als gültige Daten zurück bekommt. In einem Schema – geschrieben in XML – wird die Struktur der Daten im korrespondierenden XML-Dokument beschrieben und Aussagen zu den Daten-Typen inklusive Werte-Bereich notiert. Das bieten m.W. andere Daten-Formate so nicht – zumal man über das DOM die Daten gut bearbeiten (lassen) kann → auslesen aller Daten, auslesen einer bestimmten Untermenge von Daten über Filter, Daten in Elementen und in Attributen ändern, löschen und einfügen.

Zur Überprüfung der korrekten Kodierung von XML-Dateien gibt es zwei Methoden, die sich grundsätzlich voneinander unterscheiden. Die eine Prüfung zielt darauf ab, ob eine XML-Datei wohlgeformt ist, während bei der anderen eine XML-Datei gegen ein XSD-Schema geprüft wird.

- Wohlgeformtheit – Die Syntax des XML-Textes muss korrekt sein.
- Gültigkeit – Wurde der XML-Datei ein XSD-Schema in einer Datei *.xsd zugeordnet, müssen die Elemente der im Schema definierten Struktur folgen und der Inhalt muss den im Schema festgelegten Daten-Typen der einzelnen Elemente und Attribute entsprechen. Es gilt: Jede gültige XML-Datei ist auch wohlgeformt. Die Umkehrung gilt nicht.

27.8.1 XSD-Schema

Ein XSD-Schema:

- beschreibt, welche Elemente und Attribute in einem Dokument enthalten sind,
- beschreibt die Daten-Typen für Elemente und Attribute, die einfach oder komplex sein können,
- gibt die Anzahl und die Reihenfolge der Kind-Elemente vor (optional),
- legt Restriktionen für den Text von Elementen und die Werte von Attributen fest (optional) und
- legt die Standard- und Festwerte für Elemente und Attribute fest (optional).

Um eine XML-Datei anhand eines Schemas zu prüfen, müssen Sie zwischen der XML-Datei und dem externen Schema eine Verknüpfung erzeugen. Diese Verknüpfung wird im Root-Element der XML-Datei definiert. Über den Verweis wird der Inhalt des XSD-Schemas in das XML-Dokument eingefügt:

```
<data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="test.xsd">
```

27.8.2 XSD-Schema erzeugen

Ein XSD-Schema wird stets auf der Basis einer vorhandenen XML-Datei erzeugt. Eine einfache, aber komfortable Möglichkeit bieten XSD-Schema-Generatoren im Internet an, da Sie dann auch das XSD-Design vorgeben können:

- https://www.freeformatter.com/xsd-generator.html
- https://devutilsonline.com/xsd-xml/generate-xsd-from-xml

27.8.3 Eine XML-Datei gegen ein XSD-Schema prüfen

Um eine XML-Datei gegen ein XSD-Schema zu prüfen, können Sie geeignete XML-Editoren einsetzen oder XML-Validatoren (https://www.freeformatter.com/xml-validator-xsd.html) verwenden. Gute Erfahrungen hat der Autor mit dem XML-Editor 'XMLCopyEditor' und dem Konsolen-Programm 'xmllint' gemacht. So installieren Sie den genannten XML-Editor über eine Konsole:

```
$ sudo apt-get install xmlcopyeditor
```

Das Konsolen-Programm *xmllint* steckt in *libxml2-utils* und ist schnell installiert:

```
$ sudo apt-get install libxml2-utils
```

Für die Prüfung der Gültigkeit (Validität) einer XML-Datei benötigen Sie ein passendes XSD-Schema. Die beiden Dateien test.xml und test.xsd – deren Inhalt Sie im letzten Abschnitt nachlesen können – werden für alle weiteren Prüfungen verwendet.

Prüfung im XMLCopyEditor

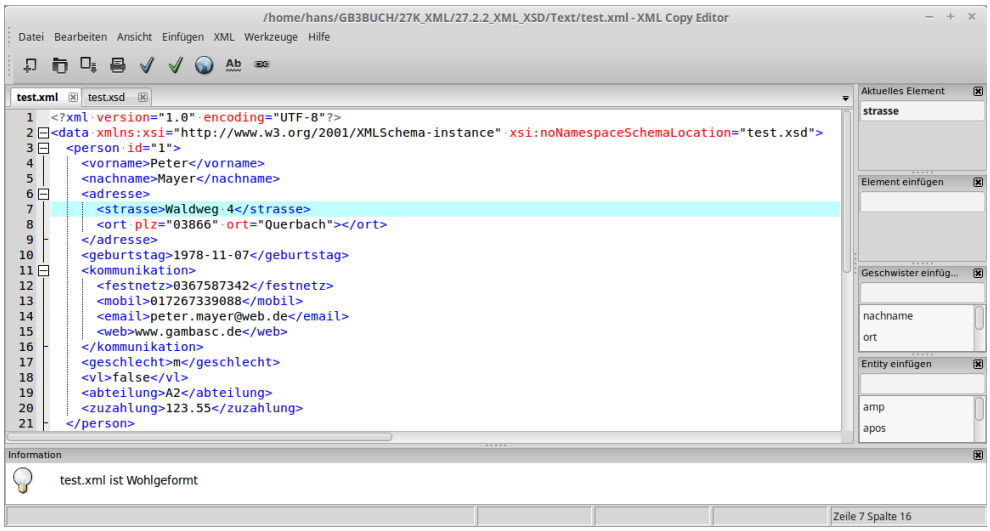

Abbildung 27.8.3.1: Erfolgreiche Prüfung der Syntax der XML-Datei

Die XML-Datei *test.xml* ist gültig:

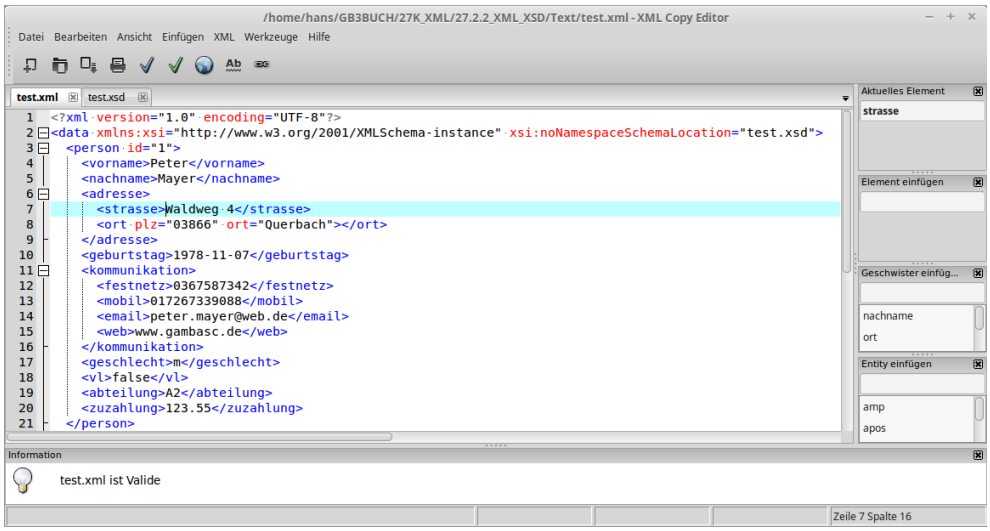

Abbildung 27.8.3.2: Erfolgreiche Prüfung der XML-Datei gegen das XSD-Schema test.xsd

Prüfung mit dem Programm xmllint

Wenn eine gültige XML-Datei vorliegt, dann wird das kurz und knapp mitgeteilt:

```
$ xmllint --noout --schema test.xsd test.xml
test.xml validates
```

Fällt die Prüfung dagegen negativ aus, dann ist wesentlich mehr zu lesen, weil auch Hinweise zum Fehler in der XML-Datei gegeben werden:

```
$ xmllint --noout --schema test.xsd test.xml
test.xml:17: element geschlecht: Schemas validity error : Element 'geschlecht': [facet 'pattern'] The value
'W' is not accepted by the pattern 'm|w'.
test.xml:17: element geschlecht: Schemas validity error : Element 'geschlecht': 'W' is not a valid value of
the local atomic type.
test.xml fails to validate
```

27.8.4 Gambas-Projekt

In einem Gambas-Projekt wird eine XML-Datei gegen ein XSD-Schema geprüft. Für die Prüfung wird

intern das Programm *xmllint* in einer Shell-Instruktion eingesetzt:

```
Public Sub btnValidateXML_Click()

  Dim sResult, sCommand As String

  sCommand = "xmllint --noout --schema " & File.SetExt($sXMLFilePath, "xsd")
  sCommand &= " " & $sXMLFilePath & " 2>&1"
  If Exist($sXMLFilePath) Then
     Shell sCommand To sResult
  Endif
  If sResult Like "*validates*" Then
     Message.Title = "Prüfergebnis XML <<->> XSD"
     Message.Info("Die XML-Datei '" & File.Name($sXMLFilePath) & "' ist gültig!")
  Else
     txaOriginal.Background = &HFF9F9F
     txaOriginal.Text = "\n" "FEHLER:" & "\n\n" & sResult
     btnValidateXML.Enabled = False
     txaOriginal.Pos = 0
  Endif
End
```

Das sind die möglichen Prüfungsergebnisse:

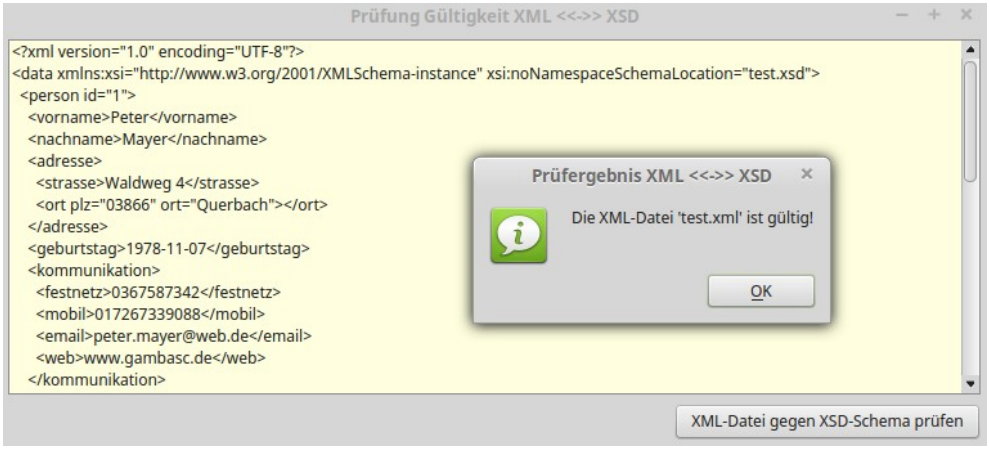

Abbildung 27.8.4.1: Positives Prüfungsergebnis

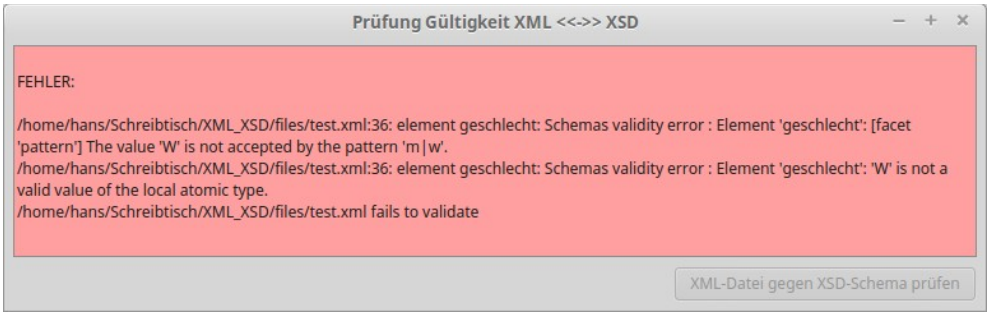

Abbildung 27.8.4.2: Negatives Prüfungsergebnis mit Hinweisen

Das vollständige Projekt sowie die beiden Dateien test.xml und test.xsd werden Ihnen im Download-Bereich zur Verfügung gestellt.

27.8.5 Inhalt der Dateien test.xml und test.xsd

Datei test.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="test.xsd">
  <person id="1">
    <vorname>Peter</vorname>
    <nachname>Mayer</nachname>
    <adresse>
```

```
      <strasse>Waldweg 4</strasse>
      <ort plz="03866" ort="Querbach"></ort>
    </adresse>
    <geburtstag>1978-11-07</geburtstag>
    <kommunikation>
      <festnetz>0367587342</festnetz>
      <mobil>017267339088</mobil>
      <email>peter.mayer@web.de</email>
      <web>www.gambasc.de</web>
    </kommunikation>
    <geschlecht>m</geschlecht>
    <vl>false</vl>
    <abteilung>A2</abteilung>
    <zuzahlung>123.55</zuzahlung>
  </person>
  <person id="2">
    <adresse>
    <vorname>Julie</vorname>
    <nachname>O'Bryan</nachname>
      <strasse>Querstrasse 22a</strasse>
      <ort plz="07381" ort="Pößneck"></ort>
    </adresse>
    <geburtstag>1982-09-17</geburtstag>
    <kommunikation>
      <festnetz>0438776542</festnetz>
      <mobil>01599099088</mobil>
      <email>js_obryan@aol.com</email>
      <web></web>
    </kommunikation>
    <geschlecht>w</geschlecht>
    <vl>true</vl>
    <abteilung>B1</abteilung>
    <zuzahlung>92</zuzahlung>
  </person>
</data>
```

Datei test.xsd:

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

<!-- 0. Ebene: Deklaration Root 'data' -->
<xs:element name="data">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="person" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- 1. Ebene: Deklaration der Datensätze vom Typ 'person' -->
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="vorname" minOccurs="0"/>
      <xs:element ref="nachname" minOccurs="0"/>
      <xs:element ref="adresse"/>
      <xs:element ref="geburtstag"/>
      <xs:element ref="kommunikation"/>
      <xs:element ref="geschlecht"/>
      <xs:element ref="vl"/>
      <xs:element ref="abteilung"/>
      <xs:element ref="zuzahlung"/>
    </xs:sequence>
    <xs:attribute type="xs:positiveInteger" name="id" use="optional"/>
  </xs:complexType>
</xs:element>

<!-- 2. Ebene: Deklaration aller komplexen Elemente -->
<xs:element name="adresse">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="vorname" minOccurs="0"/>
      <xs:element ref="nachname" minOccurs="0"/>
      <xs:element ref="strasse"/>
      <xs:element ref="ort"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>

  <xs:element name="kommunikation">
    <xs:complexType>
      <xs:sequence>
```

```
        <xs:element ref="festnetz"/>
        <xs:element ref="mobil"/>
        <xs:element ref="email"/>
        <xs:element ref="web"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

<!-- 3. Ebene: Deklaration der einfachen Elemente  -->
<xs:element name="ort">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="postleitzahl" name="plz" use="optional"/>
        <xs:attribute type="xs:string" name="ort" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="vorname" type="xs:string"/>
<xs:element name="nachname" type="xs:string"/>
<xs:element name="strasse" type="xs:string"/>
<xs:element name="festnetz" type="telefon"/>
<xs:element name="mobil" type="telefon"/>
<xs:element name="web" type="xs:anyURI"/>
<xs:element name="geburtstag" type="xs:date"/>
<xs:element name="vl" type="xs:boolean"/>
<xs:element name="zuzahlung" type="geldwert"/>

<!-- Elemente mit Restriktionen -->
<xs:element name="abteilung">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="A1|A2|B1|B2|C1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="geschlecht">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="m|w"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="email">
  <xs:simpleType>
    <xs:restriction base="xs:string">
    <!-- https://stackoverflow.com/questions/2147780/how-to-validate-an-email-id-in-xml-schema -->
      <xs:pattern value="[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<!-- Eigene Datentypen -->
<xs:simpleType name="geldwert">
  <xs:restriction base="xs:decimal">
  <xs:totalDigits value="5"/>
  <xs:fractionDigits value="2"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="telefon">
    <xs:restriction base="xs:integer">
      <xs:pattern value="[0-9]*"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="postleitzahl">
    <xs:restriction base="xs:integer">
    <!-- <xs:pattern value="[0-9]{5}"/>  Trivialer Ansatz: Erkennt z.B. PLZ 62345 nicht als fehlerhaft -->
    <!-- Zu den Ziffernkombinationen 00, 05, 43, 62 am Anfang ist keine Post-Leitregion definiert! -->
    <xs:pattern value="0[1-46-9][0-9]{3}|[1-357-9][0-9]{4}|4[0-24-9][0-9]|{3}|6[013-9][0-9]{3}"/>
    </xs:restriction>
</xs:simpleType>

</xs:schema>
```

Gut nachvollziehbar wird das Zusammenspiel zwischen XML-Datei und XSD-Datei, wenn man in der Erprobung des vorgestellten Gambas-Projektes einige Daten fehlerhaft eingibt und dann prüft:

```
PLZ             3960              ' Es fehlt eine Ziffer
PLZ             43966             ' Diese PLZ existiert nicht, da die Post-Leitregion 43 nicht existiert
EMail           js_o'bryan@...    ' Unzulässiges Zeichen '
Abteilung       A4                ' Die Aufzählung enthält keine Abteilung A4
Datum           2015-31-12        ' Datumsformat nach ISO 8601:2004 (Erweiterte Syntax: YYYY-MM-DD) falsch
vl              True              ' Fehler - weil standardmäßig nur true und false zulässig sind
Telefonnummer   +49 03937868686   ' Das Pluszeichen ist keine Ziffer - wie auch das Leerzeichen
```