Validierung von XML-Dokumenten

Prof. Dr. Christof Schöch

Modul Auszeichnungssprachen
MSc. Digital Humanities, Universität Trier

Überblick

1. Was ist Validieren?
2. Schema-Sprachen im Überblick
3. Syntax von RelaxNG (compact syntax)
4. Beispiel: Weinetiketten

(1) Was ist Validieren?

Eric van der Vlist

Validation can be about checking the structure of XML documents. It can be about checking the content of each text node and attribute independently of each other (datatype checking). It can be about checking constraints on relationships between nodes. It can be about checking constraints between nodes and external information such as lookup tables or links. It can be about checking business rules. Taken liberally, it can be almost anything else, even spell checking. All of these aspects are important for improving the level of quality of XML-based information systems.

(Quelle: Eric van der Vlist, *RelaxNG*, Kapitel 1, Section 3.)

Was ist Validieren?

- Prüfen auf Wohlgeformtheit
- Prüfen, ob das spezifische Vokabular eingehalten wurde
  - Elemente
  - Attribute
  - Werte (Datentypen)
- Prüfen, ob die spezifische Syntax eingehalten wurde
  - hierarchische Anordnung der Elementen
  - Platzierung der Attribute
- Durch Prüfen von Regeln oder Abgleich mit Mustern

Warum Validieren?

- Die systematische, vollständige Eingabe von Daten wird unterstützt (Tooltips, Autocomplete)
- Tippfehler bei Namen von Elementen und Attributen werden erkannt
- Unpassende Datentypen bei Elementinhalt oder Attributwerten werden erkannt
- Nicht sinnvolle Anordnungen von Elementen werden erkannt
- Die Weiterverarbeitung der Dokumente wird erleichtert (Transformationen, Informationsextraktion)

(2) Schema-Sprachen im Überblick

Beispiel Weinetiketten

Kodierung: Weinetiketten

```
4  <etikett>
5    <jahrgang>1921er</jahrgang>
6    <lage>Wöllsteiner Baudenberg</lage>
7  </etikett>
8
```

Welche Schema-Sprachen gibt es?

- DTD (Document Type Definition)
- XSD (XML Schema Definitions; (W3C)
- RelaxNG (REgular LAnguage for XML, New Generation)
  - XML-Syntax
  - compact syntact
- ODD (One Document Does it All; TEI)
- Schematron

DTD: Beispiel

```
1
2 <!ELEMENT weinetikett (jahrgang,lage)>
3 <!ELEMENT jahrgang (#PCDATA)>
4 <!ELEMENT lage (#PCDATA)>
5
6
7
```

DTD (Document Type Definition)

- Vorteile
  - Weit verbreitet
  - sehr kompakt
- Einschränkungen
  - DTD Syntax: an SGML orientiert, nicht an XML
  - XML Namespaces werden nicht gut unterstützt (Präfixe müssen festgelegt werden)
  - Datentypen werden nicht gut unterstützt (nur auf Attributen möglich)
  - Eingeschränkte Möglichkeiten bei den Content Models (bspw.: keine numerischen Quantoren)

XSD: Beispiel

```
1
2  <?xml version="1.0"?>
3  <schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:target=
   "http://www.example.com/name" targetNamespace="http://www.example.com
   elementFormDefault="qualified">
4    <element name="etikett">
5      <complexType>
6        <sequence>
7          <element name="lage" type="string" />
8          <element name="jahrgang" type="string" />
9        </sequence>
10     </complexType>
11   </element>
12 </schema>
13
14
15
```

XSD (XML Schema Declaration)

- Vorteile
  - Die gewohnte XML Syntax gilt
  - XSD kann auf Wohlgeformtheit geprüft werden
  - Namespaces können flexibel eingesetzt werden
  - Datentypen für Elemente (Textinhalt) und Attribute können definiert werden
- Nachteile
  - Etwas komplexer zu schreiben / zu lesen
  - Etwas "more verbose"

Relax NG (compact, nested)

```
1
2
3  start = weinetikett
4  weinetikett = element weinetikett {
5    element jahrgang { text },
6    element lage { text }
7    }
8
9
10
11
```

Relax NG (compact, flat)

```
1
2 start = etikett
3 etikett = element etikett { jahrgang, lage }
4 jahrgang = element jahrgang { text }
5 lage = element lage { text }
6
7
8
9
```

Relax NG (XML Syntax)

rng-xml-example.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0" xmlns:a=
"http://relaxng.org/ns/compatibility/annotations/1.0" datatypeLibrary=
"http://www.w3.org/2001/XMLSchema-datatypes">
  <start>
    <ref name="etikett" />
  </start>
  <define name="etikett">
    <a:documentation xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">
    The root element in a label description using the Wine Label Vocabulary.
    </a:documentation>
    <element>
      <name ns="">etikett</name>
      <ref name="jahrgang" />
      <ref name="lage" />
    </element>
  </define>
  <define name="jahrgang">
    <a:documentation xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">
    The millesime mentioned on the label.</a:documentation>
    <element>
      <name ns="">jahrgang</name>
    </element>
  </define>
  <define name="lage">
    <a:documentation xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">
    The vineyard mentioned on the label.</a:documentation>
    <element>
      <name ns="">lage</name>
    </element>
  </define>
</grammar>
```

Relax NG

- Vorteile
  - Einfach zu lernen, einfacher zu lesen
  - Muster-basierte Syntax
  - Elemente und Attribute funktionieren analog
  - Zwei Varianten: XML Syntax und "compact syntax"
  - Unterstützt XSD Datentypen (int, date, decimal, etc.)
  - Unterstützt auch nutzerdefinierte Datentypen
  - Unterstützt XML Namespaces

Übrigens: Trang

- Mit "Trang" kann man Schemas aus einem Format in ein anderes konvertieren
- https://relaxng.org/jclark/trang.html

(3) Syntax von RelaxNG (compact syntax)

RelaxNG: Bausteine

```
element etikett { jahrgang, lage }

attribut jahr { text }
```

- Muster: `element` und `attribute` patterns
- Elemente und Attribute haben "content patterns"
- Muster sind rekursiv: können verschachtelt werden
- Elemente und Attribute funktionieren fast gleich
  - Aber: Reihenfolge spielt bei Attributen keine Rolle
  - Und: Attribute können keine Elemente oder Attribute enthalten

RelaxNG: Kardinalität

```
element etikett { lage*, jahr+ }

element etikett { lage, jahr? }
```

- default: genau 1x ("once")
- ?: 0x oder 1x ("optional")
- *: 0x, 1x, mehrfach ("zero or more")
- +: 1x oder mehrfach ("once or more")

RelaxNG: Verknüpfungen, Gruppen

```
element etikett { lage & jahr }

element etikett { (lage | jahr), ort* }
```

- , : Sequenz, feste Reihenfolge
- & : "interleave", beliebige Reihenfolge
- | : mehrere Alternativen
- ( ) : Gruppen

RelaxNG: Werte festlegen

```
attribut jahr { "1921" | "1922" | "1923" }
attribut jahr { xsd:integer }
```

- string / token ; mit "enumeration"
- xsd:language
- xsd:decimal, xsd:integer
- xsd:date, xsd:gYear

Details: siehe Liste der Datentypen

Abschluss

Weitere Themen

- co-occurrence patterns (Abhängigkeiten)
- named patterns (mehrfach nutzbar)
- Rekursive Strukturen
- Umgang mit Whitespace
- Namespaces

Und jetzt?

- Anwendungsbeispiel Weinetiketten

Lektürehinweise

Grundlagen

- Georg Vogeler und Patrick Sahle: „XML“, in: Digital Humanities: Eine Einführung, hg. von Fotis Jannidis, Hubertus Kohle und Malte Rehbein. Stuttgart: Metzler, 2017, 128-146.

Referenzlektüre

- David Hunter et al.: "Part II - Validation: Kapitel 7: Relax NG" in: *Beginning XML*, 4th edition. Wiley, 2007.

Weitere Empfehlungen zur Vertiefung

- David Hunter et al.: "Part II - Validation" in: *Beginning XML*, 4th edition. Wiley, 2007, S. 95-246. (DTDs, XML Schema, Relax NG)
- Eric van der Vlist. *Relax NG*. O'Reilly Media, 2003. http://books.xmlschemata.org/relaxng/

Danke!
