XML Schema vs. Relax NG

Semistrukturierten Daten 1

Präsentation der Gruppe 2

Wozu XML Schema?

- W3C Empfehlung
- zur Definition von XML-Dokumentstrukturen
- Selbst in XML Dokumentform
- XML Schema Definition mit Dateiendung XSD

Datentypen

- Einfache Typen Atomare Typen wie xsd:string, xsd:integer... XML Spezifische atomare Typen wie QName, ID, IDREF, language...
- Komplexe Typen Zur Definition von Subelemente und Attribute

Kombination der Subelemente

- xsd:sequence Elemente müssen in der angegebenen Reihenfolge auftretten
- xsd:choice Aus einer Liste von alternativen wird ein Element ausgewählt
- xsd:all Jedes Element kommt in beliebiger Reihenfolge 0 oder 1 Mal vor

Neue Definition

Ein neuer Typ `pc` mit entsprechenden Subelementen

```
<xsd:complexTypename="pc" >
  <xsd:sequence>
    <xsd:elementname="name" type="xsd:string" />
    <xsd:elementname="hersteller" type="xsd:string"/>
    <xsd:elementname="prozessor" type="xsd:string"/>
    <xsd:elementname="mhz" type="xsd:integer" minOccurs="
    <xsd:elementname="kommentar" type="xsd:string" minOcc
                          maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attributename="id" type="xsd:integer"/>
</xsd:complexType>
```

Erweiterung

Erweiterung (extension) des Typs um ein Element `ram`

```
<xsd:complexTypename="myPC">
  <xsd:complexContent>
    <xsd:extensionbase="pc">
      <xsd:sequence>
        <xsd:elementname="ram" type="xsd:integer"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Einschränkung

Einschränkung (restriction) eines Typs

```
<xsd:complexTypename="myPC2">
  <xsd:complexContent>
    <xsd:restrictionbase="pc">
      <xsd:sequence>
        <xsd:elementname="name" type="xsd:string" />
        <xsd:elementname="hersteller" type="xsd:string"/>
        <xsd:elementname="prozessor" type="xsd:string"/>
        <xsd:elementname="mhz" type="xsd:integer"
                        minOccurs="0"/>
        <xsd:elementname="kommentar" type="xsd:string"
                        minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

Relax NG: Einführung I

Relax NG ist eine einfache auf Grammatik basierende Schema Sprache.

Relax NG: Einführung II

RELAX NG Schema ist selbst ein XML Dokument, jedoch bietet es auch eine beliebte kompakte Nicht-XML-Syntax an.

Einführungsbeispiel I

Betrachten wir diese Repräsentation eines auf XML basierten Adressbuchs:

```
<addressBook>
  <card>
    <name>John Smith</name>
    <email>js@example.com</email>
  </card>
  <card>
    <name>Fred Bloggs</name>
    <email>fb@example.net</email>
  </card>
  <card>
    <name>Mirza Ceric</name>
    <email>mirza.ceric@gmail.com</email>
  </card>
</addressBook>
```

Einführungsbeispiel II

Ein DTD könnte so aussehen:

```
<!DOCTYPE addressBook [
<!ELEMENT addressBook (card*)>
<!ELEMENT card (name, email)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
]>
```

Einführungsbeispiel II

Entsprechender XML Schema Code:

```
<xs:schema elementFormDefault="qualified">
  <xs:element name="addressBook">
    <xs:complexType>
    <xs:sequence>
      <xs:element name="card" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
        <xs:sequence>
          <xs:element name="name" type="xs:string"/>
          <xs:element name="email" type="xs:string"/>
        </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Einführungsbeispiel III

Eine Relax NG Definition für das selbe Adressbuch könnte so aussehen:

```
<element name="addressBook" xmlns="http://relaxng.org/ns/
  <zeroOrMore>
    <element name="card">
    <element name="name">
        <text/>
      </element>
      <element name="email">
        <text/>
      </element>
    </element>
  </zeroOrMore>
</element>
```

Highlights

- Syntax ist einfach und elegant

Highlights

- Syntax ist einfach und elegant
- Relax NG kann selbst als XML-Dokument geschrieben werden

Highlights

- Syntax ist einfach und elegant
- Relax NG kann selbst als XML-Dokument geschrieben werden
- Besitzt eine alternative kompakte Syntax

Datentypen I

- Besitzt nur 2 eigene Typen: `string` und `token`

```
<attribute name="available">
  <choice>
    <value>available</value>
    <value type="token">checked out </value>
    <value type="string">on  hold </value>
  </choice>
</attribute>
```

Datentypen II

- RELAX NG erlaubt es extern definierte Datentypen zu verwenden.
- Z.B. die XML-Schema Datentypen (http://www.w3.org/TR/xmlschema-2/).

Datentypen III

```
<element name="number">
  <data type="integer"
datatypeLibrary="http://www.w3.org/XMLSchema-datatypes"/>
</element>
```

Namensräume I

Folgende definition eines Elementes:

```
<element name="foo" ns="http://www.test.com">
  <empty/>
</element>
```

würde auf folgende Elemente zutreffen:

```
<foo xmlns="http://www.test.com"/>
<e:foo xmlns:e="http://www.test.com"/>
<example:foo xmlns:example="http://www.test.com"/>
```

aber auf diese nicht:

```
<foo/>
<e:foo xmlns:e="http://WWW.TEST.COM"/>
<example:foo xmlns:example="http://www.test.net"/>
```

Kompakte Schreibweise

```
element addressBook {
  element card {
    element name { text },
    element email { text }
  }*
}
```

Kompakte Schreibweise II

```
#Das ist ein Kommentar
element addressBook {
  element card {
    element name { text },
    element email { text },
    element note { text }?   #ist optional
  }+  # einmal oder beliebig oft
}
```