Learning Your Way Around XML Schemas

…an Experiment!

(You are the experimentees.)

Regine Heberlein
8/25/2015

https://sites.google.com/site/xmltraining/file-cabinet

→Open schema-addressbook.xml and
schema-address-unnamed.rnc

Two Ways to Validate

In oXygen:

1. Document → Validate → Validate with…

   OR

2. Document → Schema → Associate Schema

```
<?xml-model href="schema_named.rnc" type="application/relax-ng-compact-syntax"?>
```

Relax NG

- REgular LAnguage for XML Next Generation
- XML syntax and Compact syntax

- Eric van der Vlist, RELAX NG (O’Reilly)
- http://relaxng.org/
- http://relaxng.org/tutorial-20011203.html
- http://relaxng.org/compact-tutorial-20030326.html

Some XML Input

```
<addressbook>                                     → root
  <address type="residential">                    → typed, controlled, repeats
    <Title>Mr.</Title>                            → optional
    <FirstName>John</FirstName>                   → required
    <MiddleName></MiddleName>                     → optional
    <LastName>Smith</LastName>                    → required
    <StreetName>Main Street</StreetName>          → required
    <HouseNo>17</HouseNo>                         → required, pattern
    <AptNo></AptNo>                               → optional, pattern
    <City>Fairfield</City>                        → required
    <State>MO</State>                             → required, pattern
    <ZIP>08540</ZIP>                              → required, pattern
  </address>
</addressbook>
```

Relax NG Compact

→ Open schema-address-unnamed.rnc

```
start =
   element addressbook {
      element address {
         attribute type { xsd:string{ pattern="residential|business" }}
         & element Title { text }?
         & element FirstName { text }
         & element MiddleName { text }?
         & element LastName { text }
         & element StreetName { text }
         & element HouseNo { xsd:string{ pattern="\w+" } }
         & element AptNo { xsd:string{ pattern="\w*" } }?
         & element City { text }
         & element State { xsd:string{ pattern="NJ|NY|PA" } }
         & element ZIP { xsd:string{ pattern="\d{5}" } }
      }+
   }
```

Relax NG Compact

- use “,” for ordered, “&” for unordered patterns
- a stated pattern is required exactly once by default
- use metacharacters to speficy repeatable/optional patterns: `element Title { text }?`
- use regex patterns for optional groups:
  `element chronitem { ((Date? | Event?)+ | (Date? | Eventgrp?)+)+ }`
- the order of attributes is not significant
- text matches blanks and whitespace—use regex to normalize, e.g. `[\w+\s*\p{P}*]+`
- use xsd:string for regex: `{ xsd:string{ pattern="\d{5}" }`

Relax NG Compact: Named Patterns

→Open schema-address-named.rnc

```
address =
   element address {
         typeatt
         & Title?
         & FirstName
         & MiddleName?
         & LastName
         & StreetName
         & HouseNo
         & AptNo?
         & City
         & State
         & ZIP
         }+
```

Relax NG XML

→ Open schema-address-unnamed.rng

```
<element name="addressbook"
xmlns="http://relaxng.org/ns/structure/1.0">
      <oneOrMore>
        <element name="address">
          <attribute name="type">
            <choice>
              <value>residential</value>
              <value>business</value>
            </choice>
          </attribute>
          <optional>
            <element name="Title">
              <text/>
            </element>
          </optional>
          <element name="FirstName">
            <text/>
          </element>
          <optional>
            <element name="MiddleName">
              <text/>
            </element>
          </optional>
          <element name="LastName">
            <text/>
          </element>
          <element name="StreetName">
            <text/>
          </element>
          <element name="StreetNo">
            <data type="string">
              <param name="pattern">\w+</param>
            </data>
          </element>
    <element name="HouseNo">
        <data type="string">
          <param name="pattern">\w+</param>
        </data>
      </element>
      <optional>
        <element name="AptNo">
          <data type="string">
            <param name="pattern">\w*</param>
          </data>
        </element>
      </optional>
      <element name="City">
        <text/>
      </element>
      <element name="State">
        <choice>
          <value>NJ</value>
          <value>NY</value>
          <value>PA</value>
        </choice>
      </element>
      <element name="ZIP">
        <data type="string">
          <param name="pattern">\d{5}</param>
        </data>
      </element>
    </element>
  </oneOrMore>
</element>
```

Relax NG XML

- by default, the order of elements is significant. Use <interleave> to allow sibling elements in any order
- use <choice> and <group> to specify optional group patterns
- use <oneOrMore> to specify repeatable patterns
- by default, attributes contain <text/>
- by default, an element cannot be empty AND not have an attribute: use <empty/> to allow
- use data types for regex:

  ```
  <data type="string"><param name="pattern">\d{5}</param></data>
  ```

- for named patterns, use <define name="someElement"> and <ref name="someElement"/>

XSD (XML Schema Definition)

- http://www.w3.org/TR/xmlschema11-1/
- http://www.w3.org/TR/xmlschema11-2/
- http://www.w3schools.com/schema/
- http://www.codeproject.com/Articles/18426/XSD-Tutorial-Part-of-Elements-and-Attributes

XSD

→ Open schema-address.xsd

```
<xs:element name="addressbook">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="address" minOccurs="1"
maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:string" name="Title" />
              <xs:element type="xs:string" name="FirstName"
minOccurs="1"
                maxOccurs="1"/>
              <xs:element type="xs:string" name="MiddleName"
minOccurs="0"
                maxOccurs="1"/>
              <xs:element type="xs:string" name="LastName" />
              <xs:element type="xs:string" name="StreetName"
minOccurs="1"
                maxOccurs="1"/>
              <xs:element name="HouseNo" >
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:pattern value="\w+"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="AptNo" minOccurs="0"
maxOccurs="1">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:pattern value="\w+"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element type="xs:string" name="City" />
              <xs:element name="State" >
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:enumeration value="NJ"/>
                    <xs:enumeration value="NY"/>
                    <xs:enumeration value="PA"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="ZIP" >
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:pattern value="\d{5}"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="type" use="required">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:pattern value="residential|business"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
</xs:element>
```

XSD

- by default, elements allow attributes and content: use xs:complexType/xs:simpleType and xs:restriction to change
- use @minOccurs/@maxOccurs for optional elements (default is “1”)
- use <choice> and <group> for optional group patterns
- by default, attributes are optional: use @use=“required” to change
- by default, the order of elements is enforced, use <xs:all> for unordered sequence

What happens if... And why?

- You swap FirstName and MiddleName?
- You add whitespace to AptNo or MiddleName?
- You add a second LastName element?
- You change the state to MO?
- You add a dash and 4 digits to the ZIP?
- You change the HouseNo to “17a”?\
- You change the Title to “His Lordship”?
- You enter a city that doesn’t exist?
- You have only a FamilyName (no FirstName)?

Let’s Compare RNC, RNG, and XSD:

- How do you make elements optional (zero or one, zero or more)?

  RNC:metacharacter

  RNG:<optional>

  XSD:@minoccurs

- How do you require elements (one exactly, one or more)?

  RNC:default is 1, metacharacters

  RNG:default is 1, <oneOrMore>

  XSD: @maxoccurs (default is 1)

- How do you require attributes or make them optional?

  RNC:default is 1, metacharacters

  RNG:default is 1, use <optional> or <choice>

  XSD: attributes are optional, require with @use

Edit a Schema!

- Don’t allow empty elements or elements with whitespace only
- Add Missouri to the controlled list
- Require the 9-digit ZIP
- Control values for <Title>
- Control values for <City>
- Add <FamilyName> element
- Require either single <FamilyName> OR <FirstName> and <LastName> with optional <MiddleName>

Write a Schema!

- Use Relax NG Compact Syntax
- Use atthegrocers.xml
- Go!

EAD3

- Element lottery!
- Open ead3.rng and ead3.xsd
- Open http://www2.archivists.org/sites/all/files/gammaEAD3TagLibrary.pdf (NB this is a DRAFT and not necessarily in sink with the schema!!)
- Read the schema for your element. Only refer to the tag library if you must.

  What does it say?