PB138 – Markup Languages

Tomáš Pitner

March 8, 2013

Obsah

1 XML Schema

2 Relax NG

XML Schema - basic sources of information

XML Schema Specification - `http://www.w3.org/XML/Schema`

Using W3C XML Schema Tutorial: `http://www.xml.com/pub/a/2000/11/29/schemas/part1.html` - brief

XML Schema Tutorial - `http://www.w3schools.com/schema/default.asp` - more comprehensive

complete tutorial available at `http://www.xfront.com`

How to add XML Schema support to Netbeans IDE (`http://blogs.oracle.com/geertjan/entry/xml_schema_editor_in_netbeans`)

XML Schema - motivation

Stronger tool for XML data model specification than DTD; Offers opportunity how to:

- Separate *type* concept (element type for example) from its *occurrence* (element with particular name) - not possible in DTD
- Offers more *primitive data types.*
- Allows to use *namespaces.*
- Allows to specify *content model* (elements) more accurate way.
- Allows new type *derivation* (*inheritance*).
- Allows *modular* schema design and schema reuse.
- XML syntax of XML Schema

XML Schema - Schema Definition Header

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

XML Schema - assignment of type to element with the given name

```
<xs:element name="ElementName">
... type definition - placed either right here (so called '
</xs:element>
```

XML Schema - Simple Type Definition

- Does not contain any child elements. Can be used like either element or attribute type.
- Possible to define using an existing type restriction

```
<xs:simpleType name="TypeName">
 <xs:restriction base="BaseTypeName"> ... </xs:restriction
</xs:simpleType>
```

XML Schema - simple type definition - Example 1

Content length restriction

```
<xs:simpleType name="nameType">
 <xs:restriction base="xs:string"> <xs:maxLength value="32'
</xs:restriction> </xs:simpleType>
```

XML Schema - simple type definition - Example 2

Content restriction using a regular expression

```
<xs:simpleType name="isbnType">
  <xs:restriction base="xs:string"> <xs:pattern value="[0-
</xs:simpleType>
```

XML Schema - simple types - "union"

Approximately correspond to C "union" concept.
Result is a simple type.
Base type and values enumeration can be merged.
Example:

```
<xs:simpleType name="isbnType">
 <xs:union>
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:pattern value="[0-9]{10}"/>
    </xs:restriction>
   </xs:simpleType>
   <xs:simpleType>
    <xs:restriction base="xs:NMTOKEN">
     <xs:enumeration value="TBD"/>
     <xs:enumeration value="NA"/>
    </xs:restriction>
```

XML Schema - Simple types - values enumeration

Type can be defined as a values list separated by white-spaces. The number of elements list limitation can used as a next derivation type.
Example

```
<xs:simpleType name="isbnTypes">
 <xs:list itemType="isbnType"/>
</xs:simpleType>
<xs:simpleType name="isbnTypes10">
 <xs:restriction base="isbnTypes">
   <xs:minLength value="1"/>
   <xs:maxLength value="10"/>
 </xs:restriction>
</xs:simpleType>
```

XML Schema - complex type definition

```
<xs:complexType name="TypeName">
 <xs:sequence>
   <xs:element ...> ...
      <xs:attribute ...>
   </xs:element>
 </xs:sequence>
</xs:complexType>
```

`<xs:choice>` and `<xs:all>` can be used instead of sequence.

XML Schema - complex type definition - groups

The group element can be used to define complex type.
Group of elements:

```
<xs:group name="GroupName">
 <xs:sequence>
      <xs:element ... /> ...
 </xs:sequence>
</xs:group>
```

`<xs:choice>` and `<xs:all>` can be used instead of sequence.

XML Schema - complex type definition - element groups

Attribute group:

```
<xs:attributeGroup name="AttributesGroupName">
      <xs:attribute ... use="required"/>
      ...
</xs:attributeGroup>
```

The mandatory occurrence may be specified (use=required).

XML Schema - groups use

Example of elements/attributes groups use:

```
<xs:complexType name="bookType">
 <xs:sequence>
  <xs:group ref="mainBookElements"/>
  <xs:element name="character" type="characterType" minOccu
 </xs:sequence>
 <xs:attributeGroup ref="bookAttributes"/>
</xs:complexType>
```

XML Schema - "sequence" compositor

Defines occurrence of elements in the predefined order.

```
<xs:element name="ElementName">
 <xs:complexType>
      <xs:sequence>
       .../...
      </xs:sequence>
      .../...
 </xs:complexType>
</xs:element>
```

sequence is a content model that allows occurrence of the defined sequence of child elements.
xs prefix is bound to NS with URL
http://www.w3.org/2001/XMLSchema
Either ¡xs:choice¿ or ¡xs:all¿ can be used instead of¡xs:sequence¿.

XML Schema - "choice" compositor

Defines the occurrence of only one of the specified child elements or groups of elements.

```
<xs:element name="ElementName">
 <xs:complexType>
   <xs:choice>
     .../...
   </xs:choice>
   .../...
 </xs:complexType>
</xs:element>
```

XML Schema - "all" compositor

Defines occurrence of child elements without definition of their order.
May appear on the definition top level only.
The cardinality of child elements can be one at most.
Example:

```
<xs:complexType name="bookType">
 <xs:all>
   <xs:element name="title" type="xs:string"/>
   <xs:element name="author" type="xs:string"/>
   <xs:element name="character"type="characterType" minOccu
 </xs:all>
 <xs:attribute name="isbn" type="isbnType" use="required"/
</xs:complexType>
```

XML Schema - Element simple content

Example:

```
<xs:element name="book">
 <xs:complexType>
  <xs:simpleContent>
    <xs:extension base="xs:string">
     <xs:attribute name="isbn" type="isbnType"/>
    </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

XML Schema - mixed element content

The text content (textual child nodes) can not be validated.
The child elements can be validated.
Example:

```
<xs:element name="book">
 <xs:complexType mixed="true">
  <xs:all>
   <xs:element name="title" type="xs:string"/>
   <xs:element name="author" type="xs:string"/>
  </xs:all>
  <xs:attribute name="isbn" type="xs:string"/>
 </xs:complexType>
</xs:element>
```

XML Schema - further possibilities

Possibility to specify integrity limitations:

- value is unique - `xs:unique`
- value is a key - `xs:key`
- value is a key reference - `xs:keyref`

XML Schema - Schema annotation

Annotation is a human-readable note (comment) of a schema or its part.
It may contain the processing information (see example - xs:appinfo) as well.
Next content is not specified (limited) - see example (bind, class, ...)
Example

```
<xs:annotation>
 <xs:documentation xml:lang="en">Top level element.</xs:doc
 <xs:documentation xml:lang="fr">Element racine.</xs:docume
 <xs:appinfo source="http://example.com/foo/">
   <bind xmlns="http://example.com/bar/">
    <class name="Book"/>
   </bind>
 </xs:appinfo>
</xs:annotation>
```

XML Schema - Schema definition reuse

Direct:

```
<xs:include schemaLocation="character.xsd"/>
```

With redefinition:

```
<xs:redefine schemaLocation="character12.xsd">
 <xs:simpleType name="nameType">
  <xs:restriction base="xs:string">
   <xs:maxLength value="40"/>
  </xs:restriction>
 </xs:simpleType>
</xs:redefine>
```

XML Schema - abstract and final types

abstract - Type can not be instantiated. Can be used for inheritance derivation only.
final - Type can not be extended/derived by inheritance.

XML Schema - namespaces

Example:

```
<xs:schema targetNamespace="http://example.org/ns/books/"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:bk="http://example.org/ns/books/" elementFormDe
      attributeFormDefault="unqualified">
  .../...
</xs:schema>
```

XML Schema - unspecified elements and attributes

XML Schema allows to use some elements that are not known prior to its use.
Example:

```
<xs:complexType name="descType" mixed="true">
 <xs:sequence>
  <xs:any namespace="http://www.w3.org/1999/xhtml"
          processContents="skip" minOccurs="0" maxOccurs="u
 </xs:sequence>
</xs:complexType>
```

Use `xs:anyAttribute for attributes.`

XML Schema - schema definition reference

```
<book isbn="0836217462"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
 xsi:noNamespaceSchemaLocation="file:library.xsd">

<book isbn="0836217462" xmlns="http://example.org/ns/books
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
 xsi:schemaLocation="file:library.xsd">
```

Relax NG - motivation

XML Schema:

- too complicated (more than a 200 pages of specification)
- May be ambiguous in some situations.
- Tries to cover all applications area (documents, databases and all in between).
- Only hardly to fully implementable.
- See `http://www.xml.com/lpt/a/2002/01/23/relaxng.html` for more.

Relax NG - Primary information sources

Based on RELAX designed by OASIS-OPEN:

- `http://www.oasis-open.org/committees/relax-ng`

Relax NG Tools

- Validators:
  - Jing (http://code.google.com/p/jing-trang/)
  - Libxml2 (http://www.xmlsoft.org/)
  - RNV (http://www.davidashen.net/rnv.html) - supports the compact syntax only
  - See http://relaxng.org/#validators for more.
- XML editors supporting Relax NG
  - Firedocs (http://firedocs.org) - Firefox plug-in
  - XML Operator (www.xmloperator.net) - OSS (BSD Licence)
  - xml editor (http://www.oxygenxml.com/) - commercial
  - ...
  - See http://relaxng.org/#editors for more