PB138 — XML Schema

Basic sources of information

- **XML Schema Specification**
- Using W3C XML Schema **Tutorial** brief
- XML Schema Tutorial at W3Schools
- More comprehensive complete tutorial at xfront.com
- How to add XML Schema support to Netbeans IDE available at Geertjan's Blog
- Try this XML Schema online validator or
- Similar but more general Validome validator (not only XML schema) alternativelly available in web archive.

Motivation

Stronger tool for XML data model specification **than DTD**, it allows:

- Separate **type** (e.g. *element type*) from its **occurrence** (i.e. element with particular name)
- More **primitive data types**.
- Allows to use **namespaces**.
- Allows to specify content model (elements) more **accurate way**.
- Allows new **type inheritance**.
- Allows modular schema design and schema **reuse**.
- XML Schema has an **XML syntax**.

XML Schema Definition Header

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

Assignment of type to element with the given name

```
<xs:element name="element_name">
 <!-- here comes the type definition --
 placed either right here (so called "local")
 or as a referenced one (so called "global") -->
</xs:element>
```

short type reference:

```
<xs:element name="element_name" type="element_type"/>
```

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>
```

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>
```

Simple type definition (Example 2)

Content restriction using a regular expression

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

Simple types — `union`

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

Simple types — `union`

```
<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>
   </xs:simpleType>
 </xs:union>
</xs:simpleType>
```

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.

Simple types - values enumeration

```
<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>
```

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.

Complex type definition — element 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.

Complex type definition — attribute groups

- Attribute group:

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

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

Groups usage

- Example of elements/attributes groups use:

```
<xs:complexType name="bookType">
 <xs:sequence>
  <xs:group ref="mainBookElements"/>
  <xs:element name="character" type="characterType" minOccurs="0" maxOccurs=
"unbounded"/>
 </xs:sequence>
 <xs:attributeGroup ref="bookAttributes"/>
</xs:complexType>
```

Compositor `sequence`

- Defines occurrence of elements in the predefined order.

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

Compositor `sequence`

- sequence is a content model that allows occurrence of the defined sequence of child elements.
- `xs` prefix is (as usually) bound to the NS with URL http://www.w3.org/2001/XMLSchema
- Either `<xs:choice>` or `<xs:all>` can be used instead of `<xs:sequence>`.

Compositor `choice`

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

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

Compositor `all`

- 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.

Compositor `all`

```
<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" minOccurs="0" maxOccurs="1"/>
 </xs:all>
 <xs:attribute name="isbn" type="isbnType" use="required"/>
</xs:complexType>
```

Element simple content

```
<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>
```

Mixed element content

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

```
<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>
```

Further options

- Possibility to specify integrity limitations:
  - value is unique — `xs:unique`
  - value is a key — `xs:key`
  - value is a key reference — `xs:keyref`

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`, …)

Schema annotation

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

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>
```

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.

Namespaces in XML Schema

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

Unspecified elements and attributes

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

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

Use `xs:anyAttribute` for attributes.

Schema definition reference

```xml
<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">
```