XML Schema

Prof. Vipul Dabhi
Department of Information Technology,
D. D. University.

Definition and Declaration

- Definition

  Create new types (both simple and complex types)

- Declaration

  Enable elements and attributes with specific names and types (both simple and complex) to appear in document instances

Declaration and Definition

```
<!-- Definition-->
<xsd:simpleType name="TitleType">
<xsd:restriction base="xsd:string">
<xsd:enumeration value="Mr."/>
<xsd:enumeration value="Prof."/>
<xsd:enumeration value="Dr."/>
<!-- and so on ... -->
</xsd:restriction>
</xsd:simpleType>
```

```
<!-- Declaration-->
<element name=“title” type=“TitleType”/>
```

Schema Data Types

- Simple type
  - Do not have sub-elements
    - Do not have “element” sub-elements
    - Do not have “attribute” sub-elements
  - Predefined type or derived from predefined type
- Complex type
  - Have either “element” sub-elements or “attribute” sub-elements

SimpleTypes

- Pre-Defined SimpleTypes
  - String, CDATA, token, byte, unsignedByte, binary, integer, positiveInteger, negativeInteger, nonNegativeInteger, nonPositiveInteger, int, unsignedInt, long, unsignedLong, short, unsignedShort, decimal, float, double, boolean, time, timeInstant, timePeriod, timeDuration, date, month, year, century, recurringDay, recurringDate, recurringDuration, Name, Qname, NCName, uriReference, language, ID, IDREF, IDREFS, ENTITY, ENTITIES, NOTATION, NMTOKEN, NMTOKENS

Example

<element name=“Title” type=“string”/>

<element name=“Heading” type=“string”/>

<element name=“Topic” type=“string”/>

<element name=“Price” type=“decimal”/>

<attribute name=“focus” type=“string”/>

Derived Simple Type

- Derived from existing simple types (predefined or derived)
- Typically restricting existing simple type
  - The legal range of values for a new type is subset of the ones of existing type
  - Existing type is called base type
  - Use restriction element along with facets to restrict the range of values

Facets are rules of restriction

Example

```
<xsd:simpleType name="RollNoType">
    <xsd:restriction base="xsd:integer">
        <xsd:minInclusive value="1"/>
        <xsd:maxInclusive value="130"/>
    </xsd:restriction>
</xsd:simpleType>
```

Example

```
<xsd:simpleType name="IDType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="\d{2}[A-Z]{2}\d{3}"/>
    </xsd:restriction>
</xsd:simpleType>
```

Two digits followed by two alphabets followed by three digits.

Example

```
<xsd:simpleType name="IndiaStateType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Gujarat"/>
        <xsd:enumeration value="Maharashtra"/>
        <xsd:enumeration value="Rajasthan"/>
    <!-- and so on ... -->
    </xsd:restriction>
</xsd:simpleType>
```

Enumeration facet limits a simple type to a set of distinct values.

ComplexType

- Defined using “complexType” element
- Typically contain
  - element declarations
  - element references
  - attribute declarations

Example

```
<xsd:complexType name="StudentType" >
    <xsd:sequence>
        <xsd:element  name="name" type="xsd:string" />
        <xsd:element  name="rollno" type="xsd:int" />
        <xsd:element  name="semester" type="xsd:int" />
        <xsd:element  name="email" type="xsd:string" />
    </xsd:sequence>
<xsd:attribute name="category"type="xsd:string"/>
</xsd:complexType>
```

Element Vs Attribute

- Element declarations can reference both simple types or complex types
- All attribute declarations can reference only simple types
  - Because they cannot contain other sub-elements

Occurrence

- Occurrences of Elements
  - minOccurs
  - maxOccurs
- <element name="email" type="string" minOccurs="1" maxOccurs="5"/>
- Occurrences of Attributes
  - Attributes can occur once or not at all
  - "use" attribute
    - required
    - optional
    - fixed
    - default

Example

<attribute name=“test” type=“string” use=“required”
value=“37” use=“optional” use=“fixed”, value=“37”
use=“default” value=“37” use=“prohibited” >

- Attributes Enumeration
  - simpleType element with base attribute
- base attribute specifies the type

Attributes Enumeration

- Attributes Enumeration
  - simpleType element with base attribute
  - base attribute specifies the type

```
<xsd:attribute name="category" default="BE"/>
<xsd:simpleType base="xsd:string">
    <xsd:enumeration value="BE" />
    <xsd:enumeration value="ME" />
    …
</simpleType>
</attribute>
```

ListType

- Comprised of sequences of ***atomic simple types***
- Three built-in list types
  - NMTOKENS, IDREFS, ENTITIES
- User defined List type
  - Derive from atomic types
- **facets**
- length, minLength, maxLength, enumeration

Example of ListType

- Schema

  <xsd:simpleType name="IDSList">

  <xsd:list itemType="IDType"/>

  </xsd:simpleType>

- Instance Document

<IDSList>09IT110 09IT100</IDSList>

ListType with Facet

```
<xsd:simpleType name= “IndiaStateListType">
   <xsd:list  itemType="IndiaStateType"/>
</xsd:simpleType>

<xsd:simpleType name="FourIndiaStates">
    <xsd:restriction base=“IndiaStateListType">
    <xsd:length value="4"/>
    </xsd:restriction>
</xsd:simpleType>
```

ListType

- <element name="fourStates" type="FourIndiaStates">

- Define a list of exactly four India states (FourIndiaStates), we first define a new list type called IndiaStateListType from IndiaStateType, and then we derive FourIndiaStates by restricting IndiaStateListType to only four items

- <fourStates> Gujarat Maharashtra Rajasthan Madhyapradesh </fourStates>

UnionType

- Enables an element or attribute value to be ***one or more instances of one type*** drawn from the union of multiple atomic and list types
- facets: ***pattern*** and ***enumeration***

<xsd:simpleType name="CityUnion">
<xsd:union memberTypes="CityType PincodeType"/>
</xsd:simpleType>

<element name=cities type=“CityUnion”>

- Example 1

<cities>CA</cities>

- Example 2

<cities>387001 410001 380001</cities>

Explicit and Implicit Type

- Explicit type
  - One in which a ***name*** is given to the type
  - Element that uses the type is generally defined in a different section of the file
  - Object-oriented in that same explicit type is used as the type for several different elements
- Implicit type (nameless type)
  - Use when the type is not needed by multiple elements

ExplicitType

```
<!-- declare type-->
<xsd:complexType name="StudentType" >
    <xsd:sequence>
       …
    </xsd:sequence>
</xsd:complexType>

<!-- define element-->
<element name=student type=“StudnetType”/>
```

ImplicitType

```
<element name=student>
    <xsd:complexType name="StudentType" >
    <xsd:sequence>
                …
    </xsd:sequence>
   </xsd:complexType>
</element>
```

Choice and Group

- choice
  - Only one of its children to appear in an instance
- group
  - Grouping a group of elements
  - Further constraints
    - sequence
- all
  - Appear zero or once
  - In any order

ComplexType

```
<xsd:complexType name="PurchaseOrderType">
    <xsd:sequence>
    <xsd:choice>
        <xsd:group ref="shipAndBill" />
        <xsd:element name="singleUSAddress" type="USAddress" />
    </xsd:choice>
    <xsd:element ref="comment" minOccurs="0"/>
    <xsd:element name="items" type="Items" />
    </xsd:sequence>
<xsd:attribute name="orderDate" type="xsd:date" />
</xsd:complexType>
```

As choice allows a single selection, either shipAndBill (where user has to specify two addresses. Shipto and billto if both are different) or singleUSAddress (where both addresses are same).

ComplexType

```
<xsd:group name="shipAndBill">
    <xsd:sequence>
        <xsd:element name="shipTo" type="USAddress" />
        <xsd:element name="billTo" type="USAddress" />
    </xsd:sequence>
</xsd:group>
```

Example of all

```
<xsd:complexType name="PurchaseOrderType">
    <xsd:all>
    <xsd:element name="shipTo" type="USAddress"/>
    <xsd:element name="billTo" type="USAddress"/>
    <xsd:element ref="comment" minOccurs="0"/>
    <xsd:element name="items" type="Items" />
    </xsd:all>
<xsd:attribute name="orderDate" type="xsd:date" />
</xsd:complexType>
```

Schema Namespaces

- Two namespaces to deal with
  - Namespace for XML Schema document itself
  - http://www.w3.org/2000/08/XMLSchema
  - In XML Schema document, this is set as default namespace
  - Prefix string convention is schema
- Namespace for XML document being Constrained
- targetNamespace
  - Is the namespace that is going to be assigned to the schema you are creating.
  - It is the namespace an instance is going to use to access the types it declares

XML <schema> element

- <? Xml version="1.0"?>
  - <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  - targetNamespace=http://www.w3schools.com
  - xmlns=http://www.w3schools.com
  - elementFormDefault="qualified">
  - …..
  - …
  - </xs:schema>

- xmlns:xs – indicates that elements and data types used in the schema come from the namespace and should be prefixed as xs.
- targetNamespace – indicates that elements defined belongs to this schema
- xmlns – indicates the default namespace
- elementFormDefault = "qualified" – indicates that any elements used by XML instance document which were declared in this schema must be namespace qualified.

Schema Location

In an instance document, the attribute xsi:schemaLocation

<purchaseReport
xmlns="http://www.example.com/Report"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xsi:schemaLocation="http://www.example.com/Report
http://www.example.com/Report.xsd">
<!-- etc -->
</purchaseReport>

SimpleType Restriction - Examples

```
<simpleType name="smsText">
   <restriction base="string">
   <length value="160" fixed="true"/>
   </restiction>
</simpleType>

<simpleType name="safeDrivingSpeed">
   <restriction base="decimal">
   <minInclusive value="20"/>
   <maxExclusive value="121"/>
   </restiction>
</simpleType>
```

SimpleType Restriction - Examples

- Pattern: For data formatting and is based on regular expression
- Examples
  - Chapter \d -> Chapter 0, Chapter 1,….
  - a*x -> x, ax, aax, aaax, ….
  - a+x -> ax, aax, aaax …..
  - (a | b) + x -> ax, bx, aax, abx, bax, bbx, aabx

SimpleType Restriction - Examples

```
<simpleType name="emailType">
    <restriction  base="string">
        <pattern  value=*@*\.(com|org|net)>
    </restriction>
</simpleType>
```