XML Schemas

Simon Mahony
From an original document by Susan Hockey

This document is part of a collection of presentations and exercises on XML. For full details of this and the rest of the collection see the cover sheet at:

XML Schemas

- New way of defining document structures
- More powerful than DTDs
- W3C Schema definitions finalized in May 2001
- Several other methods of defining schemas

Advantages of Schemas

- Use XML syntax
- Can place constraints on type of data for an element
  - Must be a string, or number etc, or match a pattern
- Can place constraints on number of occurrences of an element
- Can determine order that elements must appear

Functions of a Schema

- Defines elements that can appear in a document
- Defines attributes that can appear in a document
- Defines which elements are child elements
- Defines the order of child elements

Functions of a Schema

- Defines the number of child elements
- Defines whether an element is empty or can include text
- Defines data types for elements and attributes
- Defines default and fixed values for elements and attributes

Datatypes in Schemas

- A description of an object
- Easier to
  - describe permissible document content
  - validate the correctness of data
  - work with data from a database
  - define data facets (restrictions on data)
  - define data patterns (data formats)
  - convert data between different data types
- Overall more like a database

Element Types in an XML Schema

- Simple element types – contain only text with no other elements or attributes
- Complex types – contain other elements and/or attributes
- New types are defined by derivation from existing types

Namespaces

- Identify a collection of related element names
- Used particularly when a schema takes element definitions from several sources
- Avoid element name conflicts

Namespace Declaration

- Namespaces look like a URL
- Declare with the xmlns attribute

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

- Elements in this schema which come from this definition should be prefixed with xs

Simple Types

- For elements that contain only text – no attributes or other elements
- Declaration gives
  - the name of the element
  - the type of the element

Some Element Types

- String
- Decimal – positive or negative number with optional decimal point
- Integer – whole number
- Date
- Time

Example of a String Element

<xs:element name = "sender" type ="xs:string"/>

<sender>Janet</sender>

Content of <sender> contains some text (letters, numbers, punctuation etc)

Example of a Date Element

```
<xs:element name = "datesent" type ="xs:date"/>

<datesent>2004-02-26</datesent>
```

- Note the year-month-day format of the date
- There are other forms of dates for month or year only

Example of an Integer Element

<xs:element name = "servings" type ="xs:integer"/>

<servings>4</servings>

<servings>4.5</servings>

would be incorrect

Restrictions on Values: Custom Simple Types

- Can make a new simple type derived from an existing one
- Use an existing simple type and put some restrictions on it
- Restriction can be a pattern or a list of acceptable values

Example of Restriction

```
<xs:element name = "language">
   <xs:simpleType>
        <xs:restriction base="xs:string">
        <xs:pattern value="English|Latin"/>
        </xs:restriction>
   </xs:simpleType>
</xs:element>
```

Modified from Castro (2000), *XML for the World Wide Web* p.85

Example of a Restriction

- Defines element <language> to contain either English or Latin

Correct:
<language>English</language>
<language>Latin</language>

Incorrect:
<language>French</language>

Example of a Restriction (modified fromw3schools.com)

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
                <xs:minInclusive value="16"/>
          <xs:maxInclusive value="34"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Example Markup

Correct:

<age>20</age>

<age>16</age>

Incorrect:

<age>15</age>

<age>3</age>

Regular Expressions in Patterns

.	any character
\d	any digit
\D	any non-digit
\s	any white space
\S	any character that is not white space

Regular Expressions in Patterns

*	zero or more
?	one or zero
+	one or more
[ ]	one of a group of values contained within [ ]
[-]	range of values separated by -
{ }	exactly the number within { }

Examples of Regular Expressions

[0-9]	Range of values from 0 to 9
A\d+C	A followed by one or more digits followed by C
\d{5}	5 digits (e.g. US zip code)
\d{5}(-\d{4})?	5 digits followed optionally by – and 4 more digits e.g. 07076-2918

Example Definition for a Zip Code

```
<xs:simpleType name="zipcodeType">
    <xs:restriction base="xs:string">
        <xs:pattern value="\d{5}(-\d{4})?"/>
    </xs:restriction>
</xs:simpleType>
```

Modified from Castro (2000), p81

Defines the structure of a something called a zipcodeType

Example Use of a zipcodeType

<xs:element name="zipcode" type "zipcodeType"/>

<xs:element name="zip" type="zipcodeType"/>

Defines <zipcode> and <zip> both of which have the structure of a zipcodeType

Example Encoding

Elements defined to be zipcodeType

<zipcode>07076</zipcode>
<zipcode>07076-2918</zipcode>
<zip>33844</zip>

All match the definition of zipcodeType

Specifying a List of Possible Values

```
<xs:element name="language">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:enumeration value="English"/>
         <xs:enumeration value="French"/>
         <xs:enumeration value="German"/>
         <xs:enumeration value="Spanish"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

Example Markup

Correct:

<language>English</language>

<language>French</language>

<language>German</language>

Incorrect:

<language>Latin</language>

Specifying the Format of a Decimal number

```
<xs:element name="average">
   <xs:simpleType>
      <xs:restriction base="xs:decimal">
         <xs:fractionDigits value="2"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

Allows up to 2 digits after the decimal point

Example Markup

Correct:

<average>76.26</average>

<average>7.2</average>

<average>7025.26</average>

Incorrect:

<average>76.265</average>

Other Features in Simple Types

- Lowest possible value
- Length (number of characters)
- List of items, e.g. several zipcodes
- Predefine content

Complex Types

- Elements that contain only other elements
- Elements with attributes
- Empty elements that possibly have attributes
- Mixed content – elements and text

Elements Containing Other elements

- Other elements must be given as a sequence
- Order is important

Example of a Sequence

```
<xs:complexType name="memoType">
   <xs:sequence>
        <xs:element name="sender" type="xs:string"/>
        <xs:element name="recipient" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
   </xs:sequence>
</xs:complexType>

<xs:element name="memo" type="memoType"/>
```

Example Markup

```
<memo>
<sender>Claire</sender>
<recipient>Kerstin</recipient>
<body>
This is the text of the memo.
</body>
</memo>
```

Choices / Alternatives

```
<xs:complexType name="bodyType">
    <xs:choice>
        <xs:element name="section" type="xs:string"/>
        <xs:sequence>
            <xs:element name="heading" type="xs:string"/>
            <xs:element name="p" type="xs:string"/>
        </xs:sequence>
    </xs:choice>
</xs:complexType>

<xs:element name ="body" type="bodyType"/>
```

Example Markup

<body> contains <section>, or <heading> followed by

<body><section> …. </section></body>

<body><header>…. </header> …. </body>

Elements Appearing in Any Order

<xs:all>

unordered set of elements

</xs:all>

Can only validate loosely

Number of Occurrences

Use attributes minOccurs and maxOccurs

```
<xs:element name="p" type="xs:string"
  minOccurs="3" maxOccurs="7"/>
```

- Element <p> occurs between 3 and 7 times
- Use the value "unbounded" for any number of times

Elements with Attributes

- Attributes are declared with <xs:attribute>
- Give
  - the name of the attribute
  - the type of content for the attribute

Attribute Example – For an Element Containing More Elements

```
<xs:element name ="memo">
   <xs:complexType>
   <xs:sequence>
   <xs:element name="heading" type="xs:string"/>
   <xs:element name="body" type="xs:string"/>
   </xs:sequence>
   <xs:attribute name="kindofmemo" type="xs:string"
   use="required"/>
   </xs:complexType>
</xs:element>
```

Example Markup

```
<memo kindofmemo="letter">
<heading>heading text</heading>
<body>body text</body>
</memo>
```

The attribute kindofmemo is required.

Attribute Example – for Element Containing Only Text

```
<xs:complexType name="headingform"
   type="xs:string">
   <xs:attribute name="headingid" type="xs:string"/>
</xs:complexType>

<xs:element name="heading" type ="headingform"/>

<heading headingid="one">First heading</heading>
```

Elements with Mixed Content

Element declaration must have an attribute called mixed with value "true"

```
<xs:complexType name="p" mixed="true">
<xs:choice minOccurs="0" maxOccurs="unbounded">
        definitions of elements inside an element of type p
</xs:choice>
</xs:complexType>
```

Empty Elements

- Can have attributes but no content
- Must define a restriction base which is any type – there is no type on which this element is based
- Define attributes

Example of Empty Element Declaration

```
<xs:element name="page">
   <xs:complexType>
      <xs:complexContent>
    <xs:restriction base="xs:anyType">
    <xs:attribute name="pageid" type="xs:string"/>
    <xs:attribute name="pagenum" type="xs:integer"/>
          </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
```

Empty Element Examples

```
<page pageid="Part" pagenum="2"/>

<page pageid="Chapter" pagenum="4"/>
```

Putting it All Together – Schema Definition File

- The root element of a W3C schema document must reference the W3C schema specfication

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


</xs:schema>
```

Prefix your definitions with xs:

Putting it All Together: Document File

In the document file, the root element carries an attribute indicating the location of the schema

```
<memo xmlns:xsi =
   "http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation
   ="schemafilename.xsd">
```

For a schema stored locally in *schemafilename.xsd*

Further Reading

- W3C XML Schema: http://www.w3.org/XML/Schema.html
  and especially the W3C XML Schema primer:
  http://www.w3.org/TR/xmlschema-0/

- O'Reilly: Using W3C XML Schema
  http://www.xml.com/pub/a/2000/11/29/schemas/part1.html
- W3schools XML Schema Tutorial:
  http://www.w3schools.com/schema

- **XML Schema, a brief introduction**
  http://lucas.ucs.ed.ac.uk/xml-schema/