Ch. 10: Defining Simple Types

Seoul National University, Internet Database Laboratory

July, 2015

Contents

- Defining a Simple Type Element
- Using Date and Time Types
- Using Number Types
- Predefining an Element's Content
- Deriving Custom Simple Types
- Deriving Named Custom Types
- Specifying a Range of Acceptable Values
- Specifying a Set of Acceptable Values
- Limiting the Length of an Element
- Specifying a Pattern for an Element
- Limiting a Number's Digits
- Deriving a List Type
- Deriving a Union Type

Defining a Simple Type Element

- Simple type can only contain a value
  - May not contain any child elements and attributes
- Typical simple data types
  - **xs:string / xs:decimal / xs:Boolean / xs:date / xs:time / xs:anyURI**
- The entire built-in simple type list at
  - *www.w3.org/TR/xmlschema-2/#built-in-datatypes*
- Built-in simple types always begin with **xs:**
  - **xs:** XML Schema namespace prefix
- Can create custom simple types

< xsd >

```
<xs:element name="height" type="xs:string"/>

<xs:element name="year_built" type="xs:integer"/>
```

Defining a Simple Type Element

< xsd >

```
<xs:element name="height" type="xs:string"/>
<xs:element name="year_built" type="xs:integer"/>
```

< xml >

```
<height>39 feet</height>
<year_built>430</year_built>
```

```
<height>39</height>
<year_built>long ago</year_built>
```

< xsd >

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

< xml >

```
<last_modified> 2008-05-23 </last_modified>
```

```
<last_modified> May 23, 2008 </last_modified>
```

In XML Schema, the *data format* is **YYYY-MM-DD**

- It's important to know the format of all built-in simple types

Using Date and Time Types [1/4]

- **xs:date**
  - Format ***YYYY-MM-DD***
  - e.g. 2008-05-23

< xsd >

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

< xml >

```
<birth>1879-03-14</birth>
```

- **xs:time**
  - Format ***hh:mm:ss***
  - e.g. 16:21:00

< xsd >

```
<xs:element name="time_painted"
                        type="xs:time"/>
```

< xml >

```
<time_painted>21:08:00</time_painted>
```

- **xs:dateTime**
  - Format ***YYYY-MM-DDThh:mm:ss***
  - e.g. 2008-05-23T16:21:00

< xsd >

```
<xs:element name="when_shot"
                        type="xs:dateTime"/>
```

< xml >

```
<when_shot>1968-04-04T18:01:00-05:00</when_shot>
```

Using Date and Time Types [2/4]

- **xs:duration** an amount of time
  - Format ***PnYnMnDTnHnMnS***
    - ***P*** : Always required
    - ***T*** : Only required if you have any time units
    - ***n*** : How many of the following units
      - **Y**ears, **M**onths, **D**ays, **H**ours, **M**inutes, **S**econds
  - e.g. P3M4DT6H17M ➔ 3months+4days+6hrs+17mins

< xsd >

```
<xs:element name="strike_length" type="xs:duration"/>
```

< xml >

```
<strike_length>P5D</strike_length>
```

- **xs:gYear**
  - Format ***YYYY***
  - e.g. 2011

< xsd >

```
<xs:element name="tribute_year" type="xs:gYear"/>
```

< xml >

```
<tribute_year>1995</tribute_year>
```

Using Date and Time Types [3/4]

- **xs:gYearMonth**
  - Format ***YYYY-MM***
  - e.g. 2003-05

< xsd >

```
<xs:element name="birth_year_month"
                          type="xs:gYearMonth"/>
```

< xml >

```
<birth_month>1986-12</birth_month>
```

- **xs:gMonth**
  - Format ***--MM***
  - 2 initial dashes
    - The "missing" year
    - A separator
  - e.g. --04

< xsd >

```
<xs:element name="birth_month"
                          type="xs:gMonth"/>
```

< xml >

```
<birth_month>--12</birth_month>
```

Using Date and Time Types [4/4]

- **xs:gMonthDay**
  - Format ***--MM-DD***
  - e.g. --09-14

< xsd >

```
<xs:element name="leap_day"
                        type="xs:gMonthDay"/>
```

< xml >

```
<leap_day>--02-29</leap_day>
```

- **xs:gDay**
  - Format ***---DD***
  - e.g. ---07

< xsd >

```
<xs:element name="ides" type="xs:gDay"/>
```

< xml >

```
<ides>---15</ides>
```

- Optional time zone indicator
  - All time types can end with it
  - ***Z*** for UTC
  - ***-hh:mm*** / ***+hh:mm*** offset from UTC
  - UTC(Universal Time Coordinated) = GMT(Greenwich Mean Time)
- Time types can include fractional seconds
  - Format ***hh:mm:ss.sss***

Using Number Types [1/3]

- **xs:decimal**
  - Positive, negative, or zero numbers
  - Finite number of digits
  - Optional decimal point
  - e.g. 4.26, -100, 0
- **xs:integer**
  - Positive, negative, or zero numbers
  - No fractional part
  - e.g. 542, -7
- **xs:positiveInteger** (1, 2, etc.)
  xs:negativeInteger (-1, -2, etc.)
  xs:nonPositiveInteger (0, -1, -2, etc.)
  xs:nonNegativeInteger (0, 1, 2, etc.)

Using Number Types [2/3]

- **xs:int**
  - A signed 32-bit integer
  - Often used for database ID fields
- **xs:float**
  - Single precision, 32-bit floating point
  - Positive and negative zero (*0* and *-0*)
  - Positive and negative infinity (*INF* and *-INF*)
  - Not a number (*NaN*)

Using Number Types [3/3]

< xsd >

```
<xs:element name="years_standing"
                type="xs:positiveInteger"/>

<xs:element name="height" type="xs:decimal"/>
```

< xml >

```
<years_standing>1602</years_standing>

<height>384.25</height>
```

```
<years_standing>1602.5</years_standing>

<height>384</height>
```

- more number types explained at
  - *www.w3.org/TR/xmlschema-2/*

Predefining an Element's Content [1/2]

- Set an element's value: **fixed="*value*"** (attribute)
  - ***value*** : the element must be equal to
  - Unless the element is omitted from the XML document

< xsd >

```
<xs:element name="how_destroyed"
                    type="xs:string" fixed="fire"/>
```

< xml >

```
<how_destroyed>fire</how_destroyed>
```

```
<how_destroyed></how_destroyed>
```

```
<how_destroyed>earthquake</how_destroyed>
```

Predefining an Element's Content [2/2]

- Set an element's default value: **default="*value*"** (attribute)
  - The element will be equal to value if the element is empty or omitted

< xsd >

```
<xs:element name="how_destroyed"
                    type="xs:string" default="fire"/>
```

< xml >

```
<how_destroyed>fire</how_destroyed>
```

```
<how_destroyed></how_destroyed>
```

```
<how_destroyed>earthquake</how_destroyed>
```

Deriving Custom Simple Types

you can also place restrictions on what would be considered valid content. These restrictions are called *facets*

< xsd >

```
<xs:element name="story">

   <xs:simpleType>

      <xs:restriction base="xs:string">

         <xs:length value="1024"/>

      </xs:restriction>

   </xs:simpleType>

</xs:element>
```

Identify the name of the XML element

Start deriving custom simple type

Any one of the built-in simple types upon which you'd like to base your custom type

Resrictions or facets

Deriving Named Custom Types

- Can use a custom type more than one element by naming it
- Do not type namespace prefix **xs:** in your custom type
  - Because new custom type is not part of **xs** namespace

< xsd >

named custom simple type

```
<xs:simpleType name="story_type“ >
   <xs:restriction base="xs:string">
      <xs:length value="1024"/>
   </xs:restriction>
</xs:simpleType>
```

< xsd >

```
<xs:element name="story" type="story_type“ />

<xs:element name="summary" type="story_type“ />

<xs:element name="another_story" type="story_type“ />
```

Specifying a Range of Acceptable Values [1/5]

- Highest possible value: **< xs:maxInclusive value="*n*" />**
  - ***n*** : content must be less than or equal to n

< xsd >

```
<xs:element name="total_bases">
   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:maxInclusive value="6856"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

< xml >

```
<total_bases>6855</total_bases>
```

```
<total_bases>6856</total_bases>
```

Specifying a Range of Acceptable Values [2/5]

- Highest possible value: **<xs:maxExclusive value="*n*"/>**
  - ***n*** : content must be less than (but not equal to) n

< xsd >

```
<xs:element name="total_bases">
   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:maxExclusive value="6856"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

< xml >

```
<total_bases>6855</total_bases>
```

```
<total_bases>6856</total_bases>
```

Specifying a Range of Acceptable Values [3/5]

- Lowest possible value
  - **< xs:minInclusive value="*n*" /> :** grater than or equal to n
  - **< xs:minExclusive value="*n*" /> :** grater than (but not equal to) n

< xsd >

```
<xs:element name="game_day">
   <xs:simpleType>
      <xs:restriction base="xs:date">
         <xs:minInclusive value="1954-04-13"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

< xml >

```
<game_day>1954-04-13</game_day>
```

```
<game_day>1954-04-14</game_day>
```

Specifying a Range of Acceptable Values

- Can use both min & max limits (but only 1 min & 1 max)

< xsd >

```
<xs:element name="game_day">
    <xs:simpleType>
        <xs:restriction base="xs:date">
            <xs:minInclusive value="1954-04-13"/>
            <xs:maxInclusive value="1976-10-03"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

< xml >

```
<game_day>1976-07-20</game_day>
```

```
<game_day>2008-07-04</game_day>
```

- Can use the min & max facets with date, time, and numeric simple types

- **<xs:enumeration value="*choice*"/>**
  - One acceptable value
  - ***choice*** : must be unique
- Enumeration values may contain white space
- Can use the **xs:enumeration** facet with all simple types
  - Except *boolean* type

Specifying a Set of Acceptable Values

<xs:enumeration value="zzz" />

< xsd >

```
<xs:element name="wonder_name">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="Colossus of Rhodes"/>
            <xs:enumeration value="Great Pyramid of Giza"/>
            <xs:enumeration value="Hanging Gardens of Babylon"/>
            <xs:enumeration value="Statue of Zeus at Olympia"/>
            <xs:enumeration value="Temple of Artemis at Ephesus"/>
            <xs:enumeration value="Mausoleum at Halicarnassus"/>
            <xs:enumeration value="Lighthouse of Alexandria"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

< xml >

```
<wonder_name>Great Pyramid of Giza</wonder_name>
```

```
<wonder_name>Great Pyramid</wonder_name>
```

```
<wonder_name>
        Lighthouse of Alexandria
        Hanging Gardens of Babylon
</wonder_name>
```

Limiting the Length of an Element [1/3]

- **<xs:length value="*g*"/>**
  - ***g*** : the number of characters that the element must have

< xsd >

```
<xs:element name="wonder_code">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:length value="5"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

< xml >

```
<wonder_code>w_285</wonder_code>
```

Limiting the Length of an Element [2/3]

- <xs:minLength value="*n*"/> / <xs:maxLength value="*x*"/>
  - ***n*** : the minimum length
  - ***x*** : the maximum length

< xsd >

```
<xs:element name="brief_description">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:maxLength value="256"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

< xml >

```
<brief_description>In 294 BC, a huge statue was built honoring
the god Helios. This Colossus of Rhodes,
often depicted straddling the harbor, likely stood by it. The
statue was toppled by earthquake, and wasn't rebuilt. Even broken,
many still traveled to see it.</brief_description>
```

Limiting the Length of an Element [3/3]

- Can use the *length* facet with
  - String, and other string-based XML Schema simple data types
    - Such as anyURI or hexBinary
- Binary type
  - Limits the number of octets of binary data
- List type
  - Limits the number of list items

Specifying a Pattern for an Element [1/4]

- To construct a pattern, you use a regular expression language
  - Regex : based on Perl's regex language
    - But there are no **^** or **$** chars to limit a match to the beginning or end
    - Information about XML Schema regular expression at
      - www.w3.org/TR/2004/REC-xmlschema-2-20041028/datatypes.html#regexs
- **<xs:pattern value="*regex*"/>**
  - ***regex*** : regular expression that content must match
    - **.** (a period) any character
    - **\d** digit
    - **\D** non-digit
    - **\s** white space (space, tab, newline, return)
    - **\S** any character that is not white space
    - **x*** have zero or more *x's*
      (xy)* have zero or more *xy's*

Specifying a Pattern for an Element [2/4]

- **<xs:pattern value="*regex*"/>**
  - ***regex*** : regular expression that content must match
    - **x?** have zero or one *x*
      (xy)? Have zero or one *xy*
    - **x+** have one or more *x's*
      (xy)+ have one or more *xy's*
    - **[abc]** *a*, *b*, or *c*
    - **[0-9]** range of values *from 0 to 9*
    - **this | that** have *this* or *that* in the content
    - **x{5}** have exactly 5 *x's*
    - **x{5,}** have at least 5 *x's*
    - **x{5,8}** have at least 5 and at most 8 *x's*
    - **(xyz){2}** have exactly 2 *xyz's*
    - **…**

Specifying a Pattern for an Element [3/4]

< xsd >

```
<xs:element name="wonder_code">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:pattern value="w_\d{3}"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

< xml >

```
<wonder_code>w_285</wonder_code>
```

```
<wonder_code>285_w</wonder_code>
```

Specifying a Pattern for an Element [4/4]

< xsd >

```
<xs:element name="race_time">
   <xs:simpleType>
      <xs:restriction base="xs:duration">
         <xs:pattern value="PT\d+H\d+M\d+S"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

< xml >

```
<race_time>PT2H4M26S</race_time>
```

```
<race_time>PT2H15M25S</race_time>
```

Limiting a Number's Digits [1/2]

- Total number of digits in a number
  - <xs:totalDigits value="n"/>
    - n : the maximum number of digits
  - Must be a positive number
- The number of digits after the decimal point
  - <xs:fractionDigits value="n"/>
    - n : the maximum number of digits after the decimal
  - Must be a non-negative integer
- Can use both facets

Limiting a Number's Digits [2/2]

< xsd >

```
<xs:element name="atomic_weight">
   <xs:simpleType>
      <xs:restriction base="xs:decimal">
         <xs:totalDigits value="6"/>
         <xs:fractionDigits value="4"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

< xml >

```
<atomic_weight>12.0107</atomic_weight>
```

```
<atomic_weight>55.845</atomic_weight>
```

```
<atomic_weight>196.9665</atomic_weight>
```

```
<atomic_weight>1.00794</atomic_weight>
```

Deriving a List Type [1/2]

- Can derive a list type from a simple type
- Comparison with Enumerations
  - Enumerations : provide a set of optional values for an element
  - Lists : sequences of values within the element itself
- Spaces separate one item from the next
  - *"Colossus of Rhodes Lighthouse of Alexandria"*
    - Six items, not two

Deriving a List Type [2/2]

< xsd >

```
<xs:element name="recent_eclipses">
   <xs:simpleType>
      <xs:list itemType="xs:dateTime"/>
   </xs:simpleType>
</xs:element>
```

< xml >

```
<recent_eclipses>
   2008-02-21T03:26:00Z
   2007-08-28T10:37:00Z
</recent_eclipses>
```

- If you are going to reuse the list, create a named list type

< xsd >

```
<xs:simpleType name="dateTime_list">
   <xs:list itemType="xs:dateTime"/>
</xs:simpleType>
...
<xs:element name="recent_eclipses" type="dateTime_list"/>
```

Deriving a Union Type [1/2]

- Union
  - Can define an element to be one of two(or more) different simple types

< xsd >

```
<xs:simpleType name="isbn10">
    <xs:restriction base="xs:string">
        <xs:pattern value="\d{9}[\d|X]"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="isbn13">
    <xs:restriction base="xs:string">
        <xs:pattern value="\d{3}-\d{10}"/>
    </xs:restriction>
</xs:simpleType>
```

```
<xs:element name="book">
    <xs:simpleType>
        <xs:union memberTypes="isbn10 isbn13"/>
    </xs:simpleType>
</xs:element>
```

White-space-separated group of simple types

Deriving a Union Type [2/2]

< xsd >

```
<xs:simpleType name="isbn10">
    <xs:restriction base="xs:string">
        <xs:pattern value="\d{9}[\d|X]"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="isbn13">
    <xs:restriction base="xs:string">
        <xs:pattern value="\d{3}-\d{10}"/>
    </xs:restriction>
</xs:simpleType>
```

< xml >

```
<!-- The Fountainhead, by Ayn Rand -->
<book>0452286751</book>
```

```
<!-- The Kill a Mockingbird, by Harper Lee-->
<book>044508376X</book>
```

```
<!-- XML: Visual QuickStart Guide (2nd Edition),
by Kevin Howard Goldberg -->
<book>978-0321559678</book>
```