# Part III. XML Schema

- In XML format
- Includes primitive data types (integers, strings, dates,…)
- Supports value-based constraints (integers > 100)
- Inheritance
- Foreign keys
- …

Resources

http://www.w3schools.com/schema/

## Elements v.s. Types in XML Schema

- Types:
  - Simple types (integers, strings, ...)
  - Complex types (regular expressions, like in DTDs)
- Element-type-element alternation:
  - Root element has a complex type
  - That type is a regular expression of elements
  - Those elements have their complex types...
  - ...
  - On the leaves we have simple types

# XML Schemas

```
<xsd:element name="paper" type="papertype"/>
<xsd:complexType name="papertype">
    <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name="author" minOccurs="0"/>
        <xsd:element name="year"/>
        <xsd: choice> < xsd:element name="journal"/>
                      <xsd:element name="conference"/>
        </xsd:choice>
    </xsd:sequence>
</xsd:complexType>
```

DTD: <!ELEMENT paper (title,author*,year, (journal|conference))>

## Elements v.s. Types in XML Schema

```
<xsd:element name="person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="name"
                      type="xsd:string"/>
      <xsd:element name="address"
                      type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="person"
                      type="ttt"/>
<xsd:complexType name="ttt">
  <xsd:sequence>
    <xsd:element name="name"
                      type="xsd:string"/>
    <xsd:element name="address"
                      type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

DTD: `<!ELEMENT person (name,address)>`

# Local and Global Types in XML Schema

- Local type:

```
<xsd:element name="person">
        [define locally the person's type]
</xsd:element>
```

- Global type:

```
<xsd:element name="person" type="ttt"/>

<xsd:complexType name="ttt">
        [define here the type ttt]
</xsd:complexType>
```

Global types: can be reused in other elements

# Local v.s. Global Elements in XML Schema

- Local element:

```
<xsd:complexType name="ttt">
    <xsd:sequence>
        <xsd:element name="address"  type="..."/>...
    </xsd:sequence>
 </xsd:complexType>
```

- Global element:

```
  <xsd:element name="address"  type="..."/>

  <xsd:complexType name="ttt">
      <xsd:sequence>
        <xsd:element ref="address"/>  ...
      </xsd:sequence>
 </xsd:complexType>
```

Global elements: like in DTDs

# Regular Expressions in XML Schema

Recall the element-type-element alternation:

```
<xsd:complexType name="….">
        [regular expression on elements]
 </xsd:complexType>
```

Regular expressions:

| | |
|---|---|
| • <xsd:sequence> A B C </...> | = A B C |
| • <xsd:choice> A B C </...> | = A \| B \| C |
| • <xsd:group> A B C </...> | = (A  B  C) |
| • <xsd:... minOccurs="0" maxOccurs="unbounded"> ..</...> | = (...)* |
| • <xsd:... minOccurs="0" maxOccurs="1"> ..</...> | = (...)? |

## Local Names in XML-Schema

**name** has different meanings in **person** and in **product**

```
<xsd:element name=“person”>
 <xsd:complexType>
       . . . . .
     <xsd:element name=“name”>
           <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name=“firstname” type=“xsd:string”/
>
                    <xsd:element name=“lastname” type=“xsd:string”/
>
                </xsd:sequence>
           </xsd:complexType>
     </xsd:element>
      .   .  .  .
 </xsd:complexType>
</xsd:element>
<xsd:element name=“product”>
 <xsd:complexType>
       . . . . .
     <xsd:element name=“name”   type=“xsd:string”/>

 </xsd:complexType>
</xsd:element>
```

## Attributes in XML Schema

```
<xsd:element name="paper" type="papertype"/>
<xsd:complexType name="papertype">
    <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        . . . . . .
    </xsd:sequence>
 <xsd:attribute name="language" type="xsd:NMTOKEN" fixed="English"/>
</xsd:complexType>
```

Only

Attributes are associated to the *type*, not to the element
Only to *complex types*; more trouble if we want to add attributes to *simple types*.

## “Mixed” Content, “Any” Type

```
<xsd:complexType mixed="true">
  . . . .
```

- Better than in DTDs: can still enforce the type, but now may have text between any elements

```
<xsd:element name="anything" type="xsd:anyType"/>
  . . . .
```

- Means anything is permitted there

## “All” Group

```
<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>
```

- A restricted form of & in SGML
- Restrictions:
  - Only at top level
  - Has only elements
  - Each element occurs at most once
  - Allows elements to appear in any order
- E.g. “comment” occurs 0 or 1 times

## Derived Types by Extensions

```
<complexType name="Address">
 <sequence>  <element name="street" type="string"/>
             <element name="city"   type="string"/>
 </sequence>
</complexType>

<complexType name="USAddress">
 <complexContent>
   <extension base="Address">
   <sequence>  <element name="state" type="USState"/>
               <element name="zip"   type="positiveInteger"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

Corresponds to inheritance

## Restrictions, or “facets”

- Restrictions are used to define acceptable values for XML elements or attributes. Restrictions on XML elements are called facets.
- The general form for putting a restriction on a text value is:

  -
    ```
    <xs:element  name="name">                    (or xs:attribute)
        <xs:restriction base="type">
            ... the restrictions ...
        </xs:restriction>
    </xs:element>
    ```
- For example:

  -
    ```
    <xs:element  name="age">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0">
            <xs:maxInclusive value="140">
        </xs:restriction>
    </xs:element>
    ```

## Derived Types by Restrictions

```
<complexContent>
    <restriction base="ipo:Items“>
     … [rewrite the entire content, with restrictions]...
    </restriction>
 </complexContent>
```

- (*): may restrict cardinalities, e.g. (0,infty) to (1,1); may restrict choices; other restrictions…

Corresponds to set inclusion

## Simple Types

- String
- Token
- Byte
- unsignedByte
- Integer
- positiveInteger
- Int (larger than integer)
- unsignedInt
- Long
- Short
- …

- Time
- dateTime
- Duration
- Date
- ID
- IDREF
- IDREFS

## Facets of Simple Types

- Facets = additional properties restricting a simple type
- 15 facets defined by XML Schema

Examples

- length
- minLength
- maxLength
- pattern
- enumeration
- whiteSpace
- maxInclusive
- maxExclusive
- minInclusive
- minExclusive
- totalDigits
- fractionDigits

- Can further restrict a simple type by changing some facets
- Restriction = subset

# Restrictions on numbers

- minInclusive -- number must be ≥ the given *value*
- minExclusive -- number must be > the given *value*
- maxInclusive -- number must be ≤ the given *value*
- maxExclusive -- number must be < the given *value*
- totalDigits -- number must have exactly *value* digits
- fractionDigits -- number must have no more than *value* digits after the decimal point

# Restrictions on strings

- length -- the string must contain exactly *value* characters
- minLength -- the string must contain at least *value* characters
- maxLength -- the string must contain no more than *value* characters
- pattern -- the *value* is a regular expression that the string must match
- whiteSpace -- not really a "restriction"--tells what to do with whitespace
  - value="preserve" Keep all whitespace
  - value="replace" Change all whitespace characters to spaces
  - value="collapse" Remove leading and trailing whitespace, and replace all sequences of whitespace with a single space

## Enumeration

- An enumeration restricts the value to be one of a fixed set of values
- Example:

  -
    ```
    <xs:element name="season">
        <xs:simpleType>
            <xs:restriction  base="xs:string">
                <xs:enumeration value="Spring"/>
                <xs:enumeration value="Summer"/>
                <xs:enumeration value="Autumn"/>
                <xs:enumeration value="Fall"/>
                <xs:enumeration value="Winter"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    ```

## Defining Namespaces in XSchema

Placing the **targetNamespace** attribute at the top of your XSD schema means that all entities defined in it are part of this namespace.

```
<?xml version="1.0" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                targetNamespace="myNamespace">
...
</xs:schema>
```

# Reference XSD in XML Documents

- http://www.w3schools.com/xml/schema_howto.asp
- https://msdn.microsoft.com/en-us/library/ms757863(v=vs.85).aspx

## Summary on XSchema

- In XML format
- Includes primitive data types (integers, strings, dates,….)
- Supports value-based constraints (integers > 100)
- Inheritance
- Foreign keys
- …

Resources:

http://www.w3schools.com/schema/