XML Schema

- using an XML schema is the second possibility to create valid XML documents (see the chapter about DTDs for the first one)
- XML schema files have two advantages compared to DTDs:
  - they don't have an own type of notation but are written in the typical XML syntax
  - they offer more details and functions to describe the content and the structure of a class of documents

    -> but the number of functions can make the creation of a schema file a real challenge
- this lecture will only provide you with a basic introduction due to the complexity of this topic, for more detailed informations use the corresponding literature
- an XML schema is always stored in a separate file

Resources

- the w3 specification can be found here:
  - part 0 (primer): http://www.w3.org/TR/xmlschema-0/
  - part 1 (structures): http://www.w3.org/TR/xmlschema-1/
  - part 2 (data types): http://www.w3.org/TR/xmlschema-2/
- for validating a document using XML schema you need the MSXML package on your machine (windows users only); the last version (MSXML 6.0) can be found here:
  - http://www.microsoft.com/downloads/details.aspx?FamilyID=993C0BCF-3BCF-4009-BE21-27E85E1857B1&displaylang=en
- for more resources, tools and examples, look here:
  - http://www.w3.org/XML/Schema#resources

Basics

- schema files own the file extension *.xsd (abbreviated for "XML schema definition language")
- an XML document which is valid against a schema file is called an instance document (or simply an instance) of that schema
- the basic structure of a schema file is as follows:

  -
    ```
    <?xml version="1.0"?>
    <xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema> *
      < . . . >
    </xsd:schema>
    ```

* some applications use `xmlns:xs=http://www.w3.org/2001/XMLSchema`

Declaring Elements

- elements may be declared with:
  - simple, pre-defined data types
  - simple, self-defined data types
  - complex data types
  - element content
  - mixed content
  - no content

Simple, Pre-Defined Data Types

- declaration
  - `<xsd:element name="`*`name`*`" type="xsd:`*`type`*`" minOccurs="" maxOccurs=""/>`
- example
  - `<xsd:element name="book" type="xsd:string" minOccurs="0"/>`
- `minOccurs` and `maxOccurs` define how often an element may occur (corresponding to `?`, `+` and `*` in DTDs); not defined means "1"; `minOccurs` must be smaller than `maxOccurs`; the latter one can be set to `unbounded`; the declaration of elements that may only occur exactly once may not contain `minOccurs` and `maxOccurs`

- there are lots of pre-defined data types

Pre-Defined Data Types (I)

Data Type	Description	Examples
`xsd:string`	string	This is a string.
`xsd:boolean`	true/false or 1/0	true
`xsd:decimal`	integer or decimal number	-5.2; -3; 726; 2,6
`xsd:integer`	integer	-389; 0; 35
`xsd:positiveInteger`	positive integer, without 0	35
`xsd:negativeInteger`	negative integer, without 0	-389
`xsd:date`	date (yyyy-mm-dd)	2006-11-21

Pre-Defined Data Types (II)

Data Type	Description	Example
`xsd:time`	time (hh:mm:ss.ss)	11:30:00.00 or 11:30:00
`xsd:dateTime`	date & time (yyyy-mm-ddThh:mm:ss.ss)	2006-11-21T11:30:00.00
`xsd:gMonth`	month, gregorian calendar (mm)	11
`xsd:gYear`	year, gregorian calendar (yyyy)	2006
`xsd:gDay`	day, gregorian calendar (dd)	21
`xsd:gYearMonth`	year & month, gregorian calendar (yyyy-mm)	2006-11
`xsd:anyURI`	a uniform resource identifier	urn:loc.gov:books

Simple, Self-Defined Data Types

- declaration

  -
    ```
    <xsd:element name="name">
      <xsd:restriction base="pre-defined_type">
      <. . . restrictions . . .>
      </xsd:restriction>
    </xsd:element>
    ```

Restrictions (I)

- minimum / maximum value
- declaration

  -
    ```
    <xsd:restriction base="xsd:decimal">
      <xsd:minExclusive value="value"/>
      <xsd:maxExclusive value="value"/>
    </xsd:restriction>
    ```

Restrictions (II)

- enumerations
- declaration

  -
    ```
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="string1"/>
      <xsd:enumeration value="string2"/>
      <xsd:enumeration value="string3"/>
    </xsd:restriction>
    ```

Restrictions (III)

- pattern
- declaration
  -
    ```
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d{1}-\d{4}-\d{4}-\d{1}"/>
    </xsd:restriction>
    ```
- example
  - `<ISBN>0-7356-1020-7</ISBN>`

Designated Data Types

- example

  -
    ```
    <?xml version="1.0"?>
    <xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema>
      <xsd:simpleType name="ISBNtype">
      <xsd:restriction base="xsd:string"/>
      . . .
      </xsd:restriction>
      </xsd:simpleType>
      . . .
      <xsd:element name="ISBN" type="ISBNtype"/>
    </xsd:schema>
    ```

Complex Data Types

- the element may contain character data, elements and attributes
- declaration
  - `<xsd:element name="`*`name`*`" type="xsd:anyType"/>`

Element Content: `sequence`

- declaration

  -
    ```
    <xsd:element name="name">
     <xsd:complexType>
            <xsd:sequence>
                    <xsd:element name="name" type="type"> *
                    <xsd:element name="name" type="type">
                    <xsd:element name="name" type="type">
            </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
    ```
- you may alternatively use `<xsd:element ref="name"/>` which means a reference to the corresponding element and its type definition

Element Content: `choice`

- declaration

  -
    ```
    <xsd:element name="name">
      <xsd:complexType>
              <xsd:choice>
                      <xsd:element name="name" type="type">
                      <xsd:element name="name" type="type">
                      <xsd:element name="name" type="type">
              </xsd:choice>
      </xsd:complexType>
    </xsd:element>
    ```

Mixed Content

- declaration

  -
    ```
    <xsd:element name="name">
      <xsd:complexType mixed="true">
              <xsd:sequence>
                      <xsd:element name="name" type="type">
              </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    ```
- -> the outer element may contain character data before or after the inner element

No Content

- declaration 1

  -
    ```
    <xsd:element name="name">
      <xsd:complexType>
      </xsd:complexType>
    </xsd:element>
    ```
- declaration 2

  -
    ```
    <xsd:element name="name" abstract="true"/>
    ```

Declaring Attributes

- declaration 1
  - `<xsd:attribute name="name" type="pre-defined_type" use="value"/>`
- declaration 2
  -
```
<xsd:attribute name="name" use="value">
 <xsd:simpleType>
         <xsd:restriction base="pre-defined_type">
         < . . . >
         </xsd:restriction
 </xsd:simpleType>
</xsd:attribute>
```
- data types and restrictions are the same as with elements

Values For `use`

- `use="optional"` (or no `use` at all)
  - the attribute is optional
- `use="required"`
  - the attribute is required within the document
- `use="prohibited"`
  - the attribute is not allowed to be specified
- `fixed="`*`value`*`"`
  - same as #FIXED in DTDs, no other value will be allowed
- `default="`*`value`*`"`
  - the default value will be used if no other value is specified

Adding Attributes (I)

- the attribute declaration is integrated into the corresponding `<xsd:element></xsd:element>` tag
- it may not occur *before* an `<xsd:sequence>` or an `<xsd:choice>` tag
- for an example, please look at files_04/xsd_full.xsd

Adding Attributes (II)

- declaration for adding an attribute to an empty element

  -
    ```
    <xsd:element name="name">
      <xsd:complexType>
              <xsd:attribute name="name" type="pre-defined_type"/>
      </xsd:complexType>
    </xsd:element>
    ```
- declaration for adding an attribute to an element containing only character data

  -
    ```
    <xsd:element name="name">
      <xsd:complexType mixed="true">
              <xsd:attribute name="name" type="pre-defined_type"/>
      </xsd:complexType>
    </xsd:element>
    ```

Attribute References

- alternatively you can use so-called attribute groups to specify attributes outside the corresponding element
- declaration

  -
    ```
    <xs:element name="name">
      <xs:complexType>
                <xs:attributeGroup ref="attlist.name"/>
      </xs:complexType>
    </xs:element>

    <xs:attributeGroup name="attlist.name">
      <xs:attribute name="name" use="value"/>
    </xs:attributeGroup>
    ```

Special Attribute Types

- ID
  - `type="xsd:ID"`
- IDREF
  - `type="xsd:IDREF"`
- entity
  - `type="xsd:ENTITY"`
- token
  - `type="xsd:token"`
- token used for an enumeration
  -
```
<xsd:simpleType>
  <xsd:restriction base="xs:token">
            <xsd:enumeration value="value1"/>
            <xsd:enumeration value="value2"/>
  </xsd:restriction>
```

Notations

- notations cannot be used directly in a schema file
- for details please look at:
  - http://www.w3.org/TR/xmlschema-2/#NOTATION

Including Other Schema Files

- declaration (after the `<xsd:schema>` element)
  - `<xs:include schemaLocation="`*`path_to_file`*`"/>`

Please note: Many programs offer the possibility to convert an existing and valid DTD into an XML schema file (but without converting notations)!

Testing Your Skills

- Specify an element type that shall include in a certain order: sub1, sub2 and one of sub3, sub4 and sub5, where sub3 is optional. Plus, the element contains the attribute att1, which is of type `positiveInteger` with a maximum value of 20.