4 – XML Schema & Validation | Exercise 1: Validation

1. Download the files countries_without_targetns.xsd and countries_without_targetns_with_bugs.xml from ILIAS.
   done

2. Use an online validator to fix the first bug
   http://www.corefiling.com/opensource/schemaValidate.html
   done

3. Fix the remaining bugs with your XML IDE validator.
   done

4. Inspect the repaired countries_without_targetns_with_bugs.xml and find out how XML documents are bound to an XML Schema. Deduce how this binding could work based on your knowledge about namespaces.
   done

Exercise: Qualified Elements and Attributes

Use the files countries_with_targetns.xsd and countries_with_targetns.xml for this exercise.

1. Ensure that the XML document correctly validates against the Schema. Delete `elementFormDefault="qualified"` from the Schema. The XML document does not validate anymore. Why ?

Well Formed: VALID
Schema Validation: INVALID

The following errors were found:

TYPE	LOC	MESSAGE
Validation	7, 9	cvc-complex-type.2.4.a: Invalid content was found starting with element 'name'. One of '{name, population, population_under_15, population_over_64, life_exp_men, life_exp_women}' is expected.
Validation	15, 9	cvc-complex-type.2.4.a: Invalid content was found starting with element 'name'. One of '{name, population, population_under_15, population_over_64, life_exp_men, life_exp_women}' is expected.
Validation	23, 9	cvc-complex-type.2.4.a: Invalid content was found starting with element 'name'. One of '{name, population, population_under_15, population_over_64, life_exp_men, life_exp_women}' is expected.

Die Elemente im XML-Schema gehören nicht mehr zum Namespace

2. Add `attributeFormDefault="qualified"` to the original Schema. The XML document does not validate anymore. Why ?

Well Formed: VALID
Schema Validation: INVALID

The following errors were found:

TYPE	LOC	MESSAGE
Validation	5, 106	cvc-complex-type.3.2.2: Attribute 'year' is not allowed to appear in element 'european_countries'.
Validation	5, 106	cvc-complex-type.4: Attribute 'year' must appear on element 'european_countries'.

Die Attribute müssten nun im XML mit dem Prefix versehen werden.

3. In the Schema file you will find `type="xs:string"` and `type="percentType"`. Explain the effect of namespaces here, respectively why we cannot write `type="string"` or `type="xs:percentType"` instead.

Wir haben unseren Namespace als Default-Namespace definiert. Somit müssen wir für unser eigenes „Format“ kein Prefix angeben. String ist aus dem XML, entsprechend muss es dort angegeben werden.

Exercise: An XML Schema for Volker

1. The HSLU mobile App displays the mensa menu. Technically, the App parses and displays the XML file
   http://mensa-technik-architektur.hslu.ch/mobileappmenu.xml
   done

2. This XML file is updated and uploaded manually at the beginning of every week by chef Volker himself.
   There is no XML Schema or DTD for this file.
   What do you think would happen, if Volker accidentally uploads a file with typos ?
   Help Volker and write an XML Schema for this file. Volker will certainly appreciate if your Schema comes with a target namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="mobileAppMenu">
        <xs:complexType>
            <xs:sequence minOccurs="1" maxOccurs="unbounded">
                <xs:element ref="Date"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="Date">
        <xs:complexType>
            <xs:all>
                <xs:element name="Lunch" type="xs:string"/>
                <xs:element name="Snack" type="xs:string"/>
            </xs:all>
            <xs:attribute name="Value" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Exercise: An XML Schema for James Bond

1. Write an XML Schema for the James Bond movie collection, such that
   - the file bond_movies.xml validates correctly,
   - the file bond_movies_with_bugs.xml reports all 10 bugs.

   We do not need a target namespace here.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="bond_movies">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="movie">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="title" type="xs:string" />
              <xs:element name="bond" type="xs:string" />
              <xs:element name="bond_girl" type="xs:string" />
              <xs:element name="regie" type="xs:string" />
              <xs:element name="year" type="xs:gYear" />
              <xs:element name="duration" type="durationType" />
            </xs:sequence>
            <xs:attribute name="number" type="numberType" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="month" type="monthType" use="required" />
      <xs:attribute name="year" type="xs:gYear" use="required" />
    </xs:complexType>
  </xs:element>

  <xs:simpleType name="monthType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="January"/>
      <xs:enumeration value="February"/>
      <xs:enumeration value="March"/>
      <xs:enumeration value="May"/>
      <xs:enumeration value="June"/>
      <xs:enumeration value="July"/>
      <xs:enumeration value="August"/>
      <xs:enumeration value="September"/>
      <xs:enumeration value="October"/>
      <xs:enumeration value="November"/>
      <xs:enumeration value="December"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="durationType">
    <xs:restriction base="xs:short">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="300"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="numberType">
    <xs:restriction base="xs:string">
      <xs:length value="3"/>
      <xs:pattern value="_\d{2}"/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

Exercise

1. Write an XML Schema that successfully validates the document bond_movies_with_actor_list.xml and further satisfies the following identity constraints:
   1. Every actor has a key attribute
   2. Every movie as an attribute that refers to the bond actor.

```
<bond_movies month="August" year="2013">
  <actor id="1">Sean Connery</actor>
  <actor id="2">George Lazenby</actor>
  <actor id="3">Roger Moore</actor>
  <actor id="4">Timothy Dalton</actor>
  <actor id="5">Pierce Brosnan</actor>
  <actor id="6">Daniel Craig</actor>

  <movie actor="1">
    <title>Dr. No</title>
    <bond_girl>Ursula Andress</bond_girl>
    <regie>Terence Young</regie>
    <year>1962</year>
    <duration>105</duration>
  </movie>
  ...
</bond_movies>
```

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="bond_movies">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="actor" minOccurs="1" maxOccurs="unbounded"/>
				<xs:element ref="movie" minOccurs="1" maxOccurs="unbounded"/>
			</xs:sequence>
			<xs:attribute name="month" type="monthType" use="required"/>
			<xs:attribute name="year" type="xs:gYear" use="required"/>
		</xs:complexType>
		<xs:key name="actorID">
			<xs:selector xpath="actor"/>
			<xs:field xpath="@id"/>
		</xs:key>
		<xs:keyref name="actorRef" refer="actorID">
			<xs:selector xpath="movie"/>
			<xs:field xpath="@actor"/>
		</xs:keyref>
	</xs:element>

	<xs:element name="actor">
		<xs:complexType>
			<xs:simpleContent>
				<xs:extension base="xs:string">
					<xs:attribute name="id" type="xs:integer"/>
				</xs:extension>
			</xs:simpleContent>
		</xs:complexType>
	</xs:element>

  <xs:element name="movie">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="title" type="xs:string"/>
				<xs:element name="bond_girl" type="xs:string"/>
				<xs:element name="regie" type="xs:string"/>
				<xs:element name="year" type="xs:gYear"/>
				<xs:element name="duration" type="durationType"/>
			</xs:sequence>
			<xs:attribute name="actor" type="xs:integer"/>
		</xs:complexType>
	</xs:element>
```

```
  <xs:simpleType name="durationType">
              <xs:restriction base="xs:short">
                     <xs:minInclusive value="0"/>
                     <xs:maxInclusive value="300"/>
              </xs:restriction>
       </xs:simpleType>

  <xs:simpleType name="monthType">
              <xs:restriction base="xs:string">
                     <xs:enumeration value="January"/>
                     <xs:enumeration value="February"/>
                     <xs:enumeration value="March"/>
                     <xs:enumeration value="May"/>
                     <xs:enumeration value="June"/>
                     <xs:enumeration value="July"/>
                     <xs:enumeration value="August"/>
                     <xs:enumeration value="September"/>
                     <xs:enumeration value="October"/>
                     <xs:enumeration value="November"/>
                     <xs:enumeration value="December"/>
              </xs:restriction>
       </xs:simpleType>

  <xs:simpleType name="numberType">
              <xs:restriction base="xs:string">
                     <xs:length value="3"/>
                     <xs:pattern value="_\d{2}"/>
              </xs:restriction>
       </xs:simpleType>
</xs:schema>
```

Exercise

1. Excursions `extend` courses by element `<date>` with attributes `from` and `to`. Write this complex type with extension.

```
<xs:complexType name="excursion">
  <xs:complexContent>
    <xs:extension base="course">
      <xs:sequence>
        <xs:element name="date">
          <xs:complexType>
            <xs:attribute name="from" type="xs:date"/>
            <xs:attribute name="to" type="xs:date"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

2. Put everything together in order to obtain a Schema that successfully validates the file `hslu_courses.xml`

   Siehe unten…

3. Define a new simple type of string constants for `<level>.`

```
<xs:simpleType name="level_type">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Basic" />
    <xs:enumeration value="Intermediate" />
    <xs:enumeration value="Advanced" />
  </xs:restriction>
</xs:simpleType>
```

4. Define professor names as constants and change the data type of `<profs>` into a list of these constants.

```
<xs:simpleType name="prof_type">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Pouly" />
    <xs:enumeration value="Arnold" />
    <xs:enumeration value="Meier" />
    <xs:enumeration value="Koehler" />
    <xs:enumeration value="Diethelm" />
    <xs:enumeration value="Sollberger" />
    <xs:enumeration value="Olnhoff" />
    <xs:enumeration value="Koller" />
    <xs:enumeration value="Kurmann" />
  </xs:restriction>
</xs:simpleType>
```

5. Define a new complex type `introductory_week` by extension or restriction. Introductory weeks have exactly one prof in charge of and do not give credit points. They carry one attribute for the start date.

   …

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

  <xs:element name="hslu_courses">
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded" minOccurs="0">
        <xs:element name="course" type="course"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="course">
    <xs:sequence>
      <xs:element name="profs">
        <xs:simpleType>
          <xs:list itemType="prof_type" />
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="lecture">
    <xs:complexContent>
      <xs:extension base="course">
        <xs:sequence>
          <xs:element name="level" type="level_type"/>
        </xs:sequence>
        <xs:attribute name="ects" type="xs:integer" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="study_week">
    <xs:complexContent>
      <xs:extension base="course">
        <xs:sequence>
          <xs:element name="level" type="level_type"/>
        </xs:sequence>
        <xs:attribute name="ects" type="xs:integer" fixed="3" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="excursion">
    <xs:complexContent>
      <xs:extension base="course">
        <xs:sequence>
          <xs:element name="date">
            <xs:complexType>
              <xs:attribute name="from" type="xs:date"/>
              <xs:attribute name="to" type="xs:date"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:simpleType name="prof_type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Pouly" />
      <xs:enumeration value="Arnold" />
      <xs:enumeration value="Meier" />
      <xs:enumeration value="Koehler" />
      <xs:enumeration value="Diethelm" />
      <xs:enumeration value="Sollberger" />
      <xs:enumeration value="Olnhoff" />
      <xs:enumeration value="Koller" />
      <xs:enumeration value="Kurmann" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="level_type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Basic" />
      <xs:enumeration value="Intermediate" />
      <xs:enumeration value="Advanced" />
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

Control Questions

1. Explain the difference between well-formed and valid.
   Well-formed: Syntax stimmt
   Valid: getestet gegen ein DTD / XML-Schema (XSD-File)

2. Name at least 3 advantages of XML Schema over DTDs.
   - *XSD ist XML-Syntax (DTD ist ein anderes Format)*
   - *XML Schema unterstützt Namespaces*
   - *XML Schema erlaubt die Validation von Elemente und Attribute von Standard und User-DataTypes*
   - *XML Schema ermöglichen auf einfache Art komplexe Strukturen zu entwerfen und wieder zu verwenden*

3. When does your application require an XML Schema ?
   Für die Validierung der XML-Daten / Validierung der Quell-Daten

4. Explain the concept of a target namespace.
   Sprachidentifikation (Damit eine selbst definierte Sprache an ein Identifiert gebunden werden kann und somit von anderen Sprachen erkannt werden kann)

5. What does it mean to qualify an element ?
   Das es zu einem Namespace gehört.

6. What is the key difference of `<all>` with respect to `<sequence>` and `<choice>` ?
   Sequence: genaue Reihenfolge der Elemente
   Liste: beliebige Reihenfolge (0..n)
   Choice: Entweder / oder (eines muss gewählt werden und nur eines davon)

7. Explain extension and restriction in XML Schema.
   Extension: Elemente / Attribute können erweitert werden (Schema erweitern)
   Restriction: Es können nur Werte eingeschränkt werden (Schema bleibt fix)

8. There is no automatic type substitution in XML Schema. Explain this statement in comparison with Java.
   Im XML Schema: Elemente (z.B course) können generischen Inhalt haben (z.B. lecture, study_week) ☺
   Im XML: Für jedes Element mit generischem Inhalt (course) muss mit xsi:type den genauen Type angegeben werden. ☹