**Hypertext & hypermedia**

*Author: dr inż. Wioleta Szwoch, KISI, WETI, PG*

# Fundamentals of XML and XML Schema

The goal of the lab is become familiar with XML and XML Schema. You need an editor (Visual Studio, Notepad++ or other), web browser and validator:
**https://www.corefiling.com/opensource/schemavalidate/**

In **task 2** you only check if the file XML is *well-formed*.


In the **other tasks** we check if the XML file is *valid* with schema


On the local disk, create a directory named with your own name and surname. Place the files downloaded from enauczanie in it. After the laboratory the directory shoul be deleted. Remember to periodically save your work. After each point, the file should be validated.

**There are some helpful tips at the bottom of this document**

1. (0,5pt) There are some mistakes in file HH0.xml. Find and correct them.

**Ask the teacher to check your work - checkpoint 1.**

Familiarize yourself with the downloaded files. Analyze the HH.xml file, pay attention to the structure of the document, the tags used etc. Analyze the HH.xsd file pay attention to the way of declaring tags and attributes and defining types in the HH.xml file. Display the XML and XSD files in the browser. Consider why they are displayed in this way.
*NOTE The data needed to complete the XML file are in the text.docx file - add them according to the instructions in the manual. These data can also be found in the file from point 2 - HH0.xml (of course, it should be used after correcting all errors in it).* ***The correct HH0.xml file is the target XML file****.*

2. (1pt) Add your name and surname in the XML file. In xsd file add appropriate declaration. Use declared element `author` and **references**.

```
<course …>
      <author>
            <name>student’s first name</name>
            <surname> student’s last name </surname>
      </author>
```

3. (0,5pt) Add subelement `score` to element `component`. In this element a numerical grade will be stored. Add appropriate declaration in xsd file. Element `score` should be declared as the last sub-element in element `component`.
4. (0,5pt) In XML file add attribute `id` (type `byte`) to element `component`. Change the definition of `componentType` type. Attribut describes the next number of activities.

```
<component id="1">
      <topic>Hypertext and hypermedia</topic>
      <theme>Hypertext &amp; hypermedia</theme>
      <theme>HTML CSS</theme>
      <theme>XML</theme>
      <theme>XML Schema</theme>
      <theme>DTD</theme>
      <theme>XSLT</theme>
      <score>30</score>
</component>
```

**Ask the teacher to check your work - checkpoint 2.**

5. (1pt) Put links in the XML file. Declare a `links` element in which you can place any number of `link` elements. In the `link` element, the `source` attribute should contain the address, while the link text should be placed as the value of the `link` element. In the xsd file, add the appropriate definitions. The `link` element must be declared as **optional** - it may not exist in the XML file. The `links` element schould be a subelement of the `information` element. The type for the `link` element should be defined **globally**.

```
<links>
      <link source="https://www.w3schools.com/html/">HTML w3schools</link>
      <link source="https://www.w3schools.com/xml/default.asp">XML w3schools</link>
      <link source="https://enauczanie.pg.edu.pl/moodle/">Moodle</link>
</links>
```

6. (0,5pt) Put an element that allows storing the path to the photo files and their title in the XML file. In the xsd file, add the appropriate definition. In the `image` declaration, use the **global type** defined in the previous paragraph for the `link` element.

```
<media>
      <image source="img/Vannevar-Bush.jpg">Vannevar Bush</image>
</media>
```

7. (1pt) Create a simple `shortStringType` type based on the string type. Specify the maximum length of the string at 30. Use the defined type in the element `name` declaration. Similarly, create a `longStringType` type with a length of 50 characters and use it in the element `surname`

**Ask the teacher to check your work - checkpoint 3.**

8. (1pt) In the `text` element, search for a sentence „Hypertext, in other words!” and put them into `subtitle` tag. Change the declaration of the `text` element to make the file valid.

Complete the XML file with data on the laboratory and project in the same way as it was done for the lecture (the easiest way to do this is to use a **corrected HH0.xml file** - copying the appropriate structures - or, alternatively, using the text.docx file to create the appropriate structures yourself).

9. (1pt) In the XML file, add the `kind` attribute to the `classes` tag, which will allow you to distinguish between laboratory, lecture and project. In the xsd file, add the appropriate declaration. For the declared attribute, define the **global type** and use **enumeration** to specify possible values of the attribute (*lecture,laboratory, project*). Attribute should be required. (`use="required"`).

10. (1pt) In the `classes` element, add the `obligatory` attribute. This attribute is **not** required. Its value can only take "yes" or "no" values (use `pattern`). Type for the attribute should be defined **locally**. The default value set to “no”. For the element `classes` for laboratory set in the XML file the value of the attribute to “yes”.

**Ask the teacher to check your work - checkpoint 4.**

11. (2pt) In the XML file, add a fragment of the hierarchy: one element with three sub-elements:
- the first sub-element has an attribute and **locally** defined type,
- the second sub-element has **new, globally** defined type,
- the third sub-element should be chosen from two specific elements (use `choice`).

The added fragment should be thematically related to the data contained in the other elements.

**Ask the teacher to check your work - checkpoint 5.**

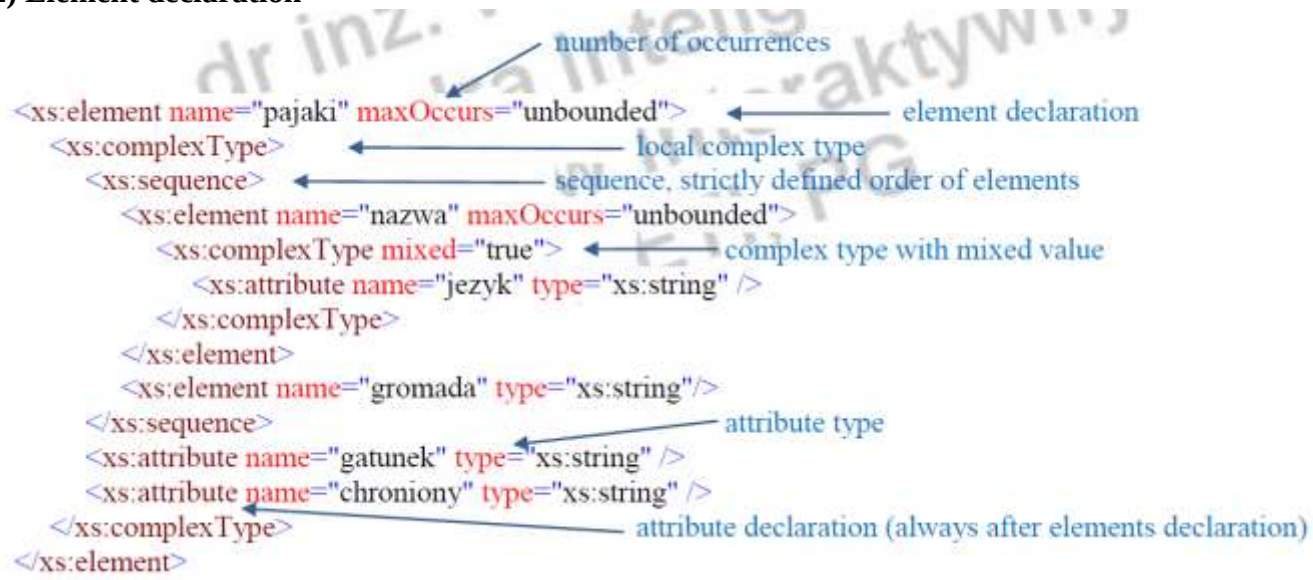
**XML and XML Schema - shortcut ☺**

XML
– Every start tag has a matching end tag
– Elements may nest, but must not overlap
– XML is case sensitive


**1) Simple type definition**

```
<xs:simpleType name=”shortType”>   - global simple type
   <xs:restriction base=”xs:string”>
      <xs:maxLength value=”20”/>    - facet
   </xs:restriction>
</xs:simpleType>
```

**2) Element declaration**




**3) Wyliczenia - lista predefiniowanych wartości**

```
<xs:simpleType name="type_name" >
   <xs:restriction base="xs:string">
      <xs:enumeration value="value1" />
      <xs:enumeration value=" value2" />
      <xs:enumeration value=" value3" />
   </xs:restriction>
</xs:simpleType>
```

4) **SimpleContent**

```
<xs:complexType name="nameType">
   <xs:simpleContent>
      <xs:extension base="xs:string">
         <xs:attribute name="attribute_name" type="xs:string"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

**5) Reference to element**

```
<xs:element name="data" type="xs:date"/>
```
- global element declaration

```
<xs:element ref="data" minOccurs="0"/>
```
- reference to global element