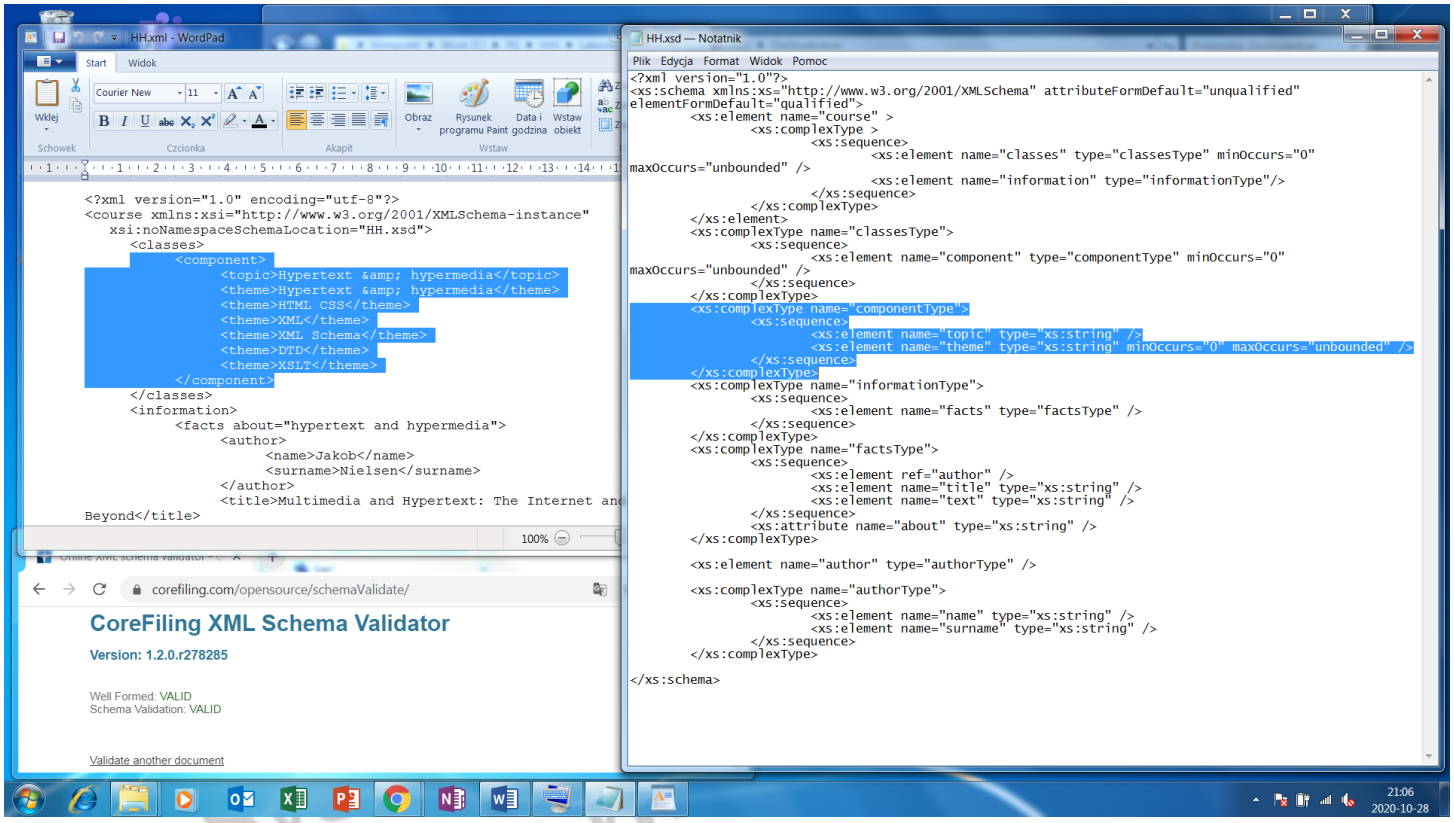
Podstawy XML i XML Schema

Celem ćwiczenia jest zapoznanie z dokumentami XML, XML Schema oraz walidacją.
Do wykonania ćwiczenia potrzebny jest dowolny edytor plików tekstowych, przeglądarka internetowa oraz walidator.

Do walidowania plików należy użyć walidatora http://tools.decisionsoft.com/schemaValidate/.

W zadaniu 2 następuje sprawdzanie czy plik XML jest **poprawnie uformowany** (*ang. well-formed*) tzn. zgodny z zasadami tworzenia dokumentów XML. W pozostałych zadaniach sprawdzamy dodatkowo, czy plik XML jest **poprawny strukturalnie** (*ang. valid document*), tzn. czy zawiera elementy, atrybuty, hierarchię zgodne z gramatyką zawartą w pliku XML Schema.

- Na koniec zajęć swój katalog należy spakować do pliku **ZIP** o nazwie `Nazwisko_Imie_NrIndeksu_XML.ZIP` (bez polskich liter) i przesłać na Moodla.
- W przesyłanym katalogu ma się również znaleźć sprawozdanie z zajęć. Plik sprawozdania ma być nazwany `Nazwisko_Imie_NrIndeksu_XML` i należy w nim umieszczać zrzuty ekranu zawierające jednocześnie widok kodu XML, XML Schema oraz widok walidatora. Na screen-ie ma być widoczna data i czas. Kod odpowiedzialny za dany etap powinien być zaznaczony. Przykładowy zrzut ekranu powinien wyglądać następująco:



1. Stworzyć katalog nazwany własnym imieniem i nazwiskiem. Umieścić w nim pliki ściągnięte z Moodla. Do pracy z plikami można użyć środowiska Visual lub zwykłego notatnika. Należy pamiętać o okresowym zachowywaniu wyników pracy. Po każdym punkcie plik należy **walidować**, aby sprawdzić czy jest zgodny z tworzonym XML Schema. Wykorzystanie Microsoft Visual Studio jako edytora pozwala na bieżąco śledzić błędy.
2. (0,5pkt) W pliku HH0.xml popełniono kilka błędów – plik nie jest poprawnie uformowany (ang. *well-formed*). Znajdź je i popraw tak, aby plik miał poprawną składnię i parser nie zgłaszał błędów.
3. **Umieścić zrzut ekranu w pliku sprawozdania.**

4. Zapoznać się z plikami udostępnionymi na Moodlu. Przeanalizować plik HH.xml, zwrócić uwagę na strukturę dokumentu, wykorzystane znaczniki. Przeanalizować plik HH.xsd, zwrócić uwagę na sposób definiowania znaczników występujących w pliku HH.xml. Wyświetlić pliki xml i xsd w przeglądarce. Zastanowić się, dlaczego są one wyświetlane w taki sposób.
   UWAGA *Dane potrzebne do uzupełniania pliku XML znajdują się w pliku text.docx – należy dodawać je zgodnie z poleceniami zawartymi w instrukcji. Dane te można też znaleźć w pliku z punktu 2 - HH0.xml (oczywiście należy go ewentualnie wykorzystać po poprawieniu w nim wszystkich błędów). Poprawny plik HH0.xml jest docelowym plikiem XML.*
5. (1pkt) W pliku XML (HH.xml) dodać swoje imię i nazwisko. W pliku xsd dodać odpowiednią deklarację. Wykorzystać zadeklarowany już element `author` i referencje.

```
<course>
   <author>
      <name>student's name</name>
      <surname>student's surname</surname>
   </author>
```

6. **Umieścić zrzut ekranu w pliku sprawozdania.**
7. (0,5pkt) Do elementu `component` dodać podelement `score`, w którym przechowywana będzie punktacja. Dodać odpowiednią deklarację w xsd. Podelement `score` powinien zostać zadeklarowany jako ostatni podelement elementu `component`.
8. (0,5pkt) W pliku XML do elementu `component` dodać atrybut `id` typu `byte`. Odpowiednio zmienić definicję typu `componentType`. Atrybut opisuje kolejny numer składowej danych zajęć.

```
<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>
```

9. **Umieścić zrzut ekranu w pliku sprawozdania.**
10. (1pkt) W pliku XML umieścić linki. Zadeklarować element `links`, w którym można umieścić dowolną liczbę elementów `link`. W elemencie `link` atrybut `source` ma zawierać adres, natomiast tekst linku ma być umieszczony jako wartość elementu `link`. W pliku xsd dodać odpowiednie definicje. Element `links` ma być zadeklarowany jako opcjonalny – może nie wystąpić w pliku XML. Element `links` ma być podelementem elementu `information`. Typ dla elementu `link` zdefiniować globalnie.

```
<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>
```

11. **Umieścić zrzut ekranu w pliku sprawozdania.**
12. (0,5pkt) Umieścić w pliku XML element pozwalający na przechowywanie adresu plików zdjęć oraz ich tytułu. W pliku xsd dodać odpowiednią definicję. W deklaracji elementu `image` wykorzystać typ globalny zdefiniowany w poprzednim punkcie dla elementu `link`.

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

13. (1pkt) Stworzyć typ globalny prosty `shortStringType` oparty o typ łańcucha znaków. Określić maksymalną dopuszczalną długość łańcucha znaków na 30. Wykorzystać zdefiniowany typ w deklaracji elementów: `theme`, `name`. Analogicznie stworzyć typ `longStringType` o długości 50 znaków i wykorzystać go w wybranych miejscach (np. `surname`, ….)
14. **Umieścić zrzut ekranu w pliku sprawozdania.**

15. (1pkt) W pliku XML w elemencie `text` wyszukać zdanie „Hypertext, in other words!” i umieścić je w znaczniku `subtitle`. Tak zmienić deklarację elementu `text`, aby plik się walidował.
16. **Umieścić zrzut ekranu w pliku sprawozdania.**
17. Uzupełnić plik XML o dane dotyczące laboratorium i projektu analogicznie jak zrobiono to dla wykładu (najprościej jest zrobić to korzystając z poprawionego pliku HH0.xml – kopiując odpowiednie struktury - lub ewentualnie z pliku text.docx tworząc odpowiednie struktury samodzielnie).
18. (1pkt) W pliku XML w znaczniku `classes` powinien znaleźć się atrybut `kind`, który pozwoli rozróżnić pomiędzy laboratorium, wykładem i projektem. W pliku xsd dodać odpowiednią deklarację. Dla zadeklarowanego atrybutu zdefiniować typ globalny i wykorzystując `enumeration` określić dopuszczalne, możliwe wartości atrybutu (*lecture*, *laboratory*, *project*). Atrybut ma być wymagany (`use="required"`).
19. **Umieścić zrzut ekranu w pliku sprawozdania.**
20. (1pkt) W elemencie `classes` umożliwić opcjonalne umieszczanie atrybutu `obligatory`. Jego wartość może przyjmować tylko wartości ”yes” lub ”no” (wykorzystać `pattern`). Typ dla atrybutu zdefiniować lokalnie. Ustawić dla atrybutu wartość domyślną na ”no”. Dla elementu `classes` związanego z laboratorium wartość atrybutu `obligatory` w pliku XML ma przyjąć wartość ”yes”.
21. **Umieścić zrzut ekranu w pliku sprawozdania.**
22. (2pkt) W pliku XML dodać fragment hierarchii: element z trzema podelementami, do jednego podelementu dodać atrybut. Dla jednego z podelementów wykorzystać nowy, zdefiniowany lokalnie typ, dla drugiego nowy, zdefiniowany globalnie typ. Trzeci podelement ma być wybierany z dwóch określonych elementów (`choice`). Dodawany fragment powinien być związany tematycznie z danymi zawartymi w pozostałych elementach.
23. **Umieścić zrzut ekranu w pliku sprawozdania.**

XML i XML Schema - krótka ściąga ☺

XML

- wszystkie niepuste elementy muszą mieć znacznik początkowy i końcowy
- elementy mogą być zagnieżdżone, nie mogą na siebie zachodzić
- rozróżnianie dużych i małych liter

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<pajeczaki xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="Zad1_Pajeczaki.xsd">
    <pajaki gatunek="tygrzyk" chroniony="tak">
        <nazwa jezyk="polska">Tygrzyk Paskowany</nazwa>
        <nazwa jezyk="łacińska">Argiope bruennichi</nazwa>
        <gromada>Pajeczaki</gromada>
    </pajaki>
    <pajaki gatunek="krzyzak" chroniony="nie">
        <nazwa jezyk="polska">Krzyżak Ogrodowy</nazwa>
        <nazwa jezyk="łacińska">Araneus diadematus</nazwa>
        <gromada>Pajeczaki</gromada>
    </pajaki>
</pajeczaki>
```

deklaracja XML

zawartość tekstowa elementu

atrybut

wartość atrybutu

znacznik początkowy

znacznik końcowy

korzeń (element główny)

XML Schema
Jeśli chcemy stworzyć:

- tylko element z zawartością tekstową
  - typ prosty
- element z podelementami
  - typ złożony
- element z podelementami i atrybutami
  - typ złożony
- element z zawartością mieszaną (podelementy i tekst)
  - typ złożony z atrybutem mixed=true
- element z atrybutem i zawartością tekstową
  - typ złożony z zawartością prostą (complexType simpleContent)

1) Definicja typu prostego nazwanego

```
<xs:simpleType name=”krotki_string”>          ← typ prosty nazwany
    < xs:restriction base=“string”/>
        < xs:maxLength value=“20” />          ← ograniczenie
    </ xs:restriction>
</ xs:simpleType>
```

2) Definicja elementu

```
<xs:element name="pajaki" maxOccurs="unbounded">      ← liczba wystąpień; definicja elementu
  <xs:complexType>                                    ← typ złożony, lokalny
    <xs:sequence>                                     ← sekwencja, elementy w ściśle określonej kolejności
      <xs:element name="nazwa" maxOccurs="unbounded">
        <xs:complexType mixed="true">                 ← typ złożony z zawartością mieszaną
          <xs:attribute name="jezyk" type="xs:string" />
        </xs:complexType>
      </xs:element>
      <xs:element name="gromada" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="gatunek" type="xs:string" />  ← typ atrybutu
    <xs:attribute name="chroniony" type="xs:string" />
  </xs:complexType>                                   ← deklaracja atrybutu (zawsze po deklaracjach elementów)
</xs:element>
```

3) Wyliczenia - lista predefiniowanych wartości

```
<xs:simpleType name=“nazwa_typu” >
```

```
  < xs:restriction base=“ xs:string”>
     < xs:enumeration value=“wartosc1” />
     < xs:enumeration value=“ wartosc2” />
     < xs:enumeration value=“ wartosc3” />
  </ xs:restriction>
</ xs:simpleType>
```

4) **SimpleContent**

Gdy tworzymy pochodny typ złożony na podstawie typu prostego lub innego typu złożonego o zawartości prostej. Można w ten sposób np. dodać atrybuty do prostego typu bazowego.

```
<xs:complexType name="nazwa-typu">
   <xs:simpleContent>
      <xs:extension base="xs:string">
         <xs:attribute name="nazwa_atrybutu" type="xs:string"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

5) Odniesienia do elementu

```
<xs:element name="data" type="xs:date"/>          globalna definicja elementu

<xs:element ref="data" minOccurs="0"/>            odniesienie do elementu zdefiniowanego globalnie
```