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 https://www.corefiling.com/opensource/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.

1. Na dysku wskazanym przez prowadzącego 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. Po zajęciach własny katalog należy **SKASOWAĆ**. 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. (1pkt) 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. **Poprosić prowadzącego o sprawdzenie pracy**
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 deklarowania znaczników występujących w pliku HH.xml oraz sposób definiowania typów. Komentarze umieszczone w pliku xsd powinny pomóc w zorientowaniu się w nazewnictwie.
   Wyświetlić pliki xml i xsd w przeglądarce. Zastanowić się, dlaczego są one wyświetlane w taki sposób.
   Zwalidować pliki – zobaczyć, że dane zawarte w pliku xml są zgodne z regułami podanymi w pliku xsd.
5. UWAGA *Dane zawarte w pliku XML są tematycznie zgodne z danymi na stronie HTML z pierwszego laboratorium. Należy tutaj zwrócić uwagę, że nie są to dokładnie takie same dane oraz, że plik XML nie odzwierciedla położenia i wyglądu tych danych na stronie internetowej ale niesie informacje o samych danych i zależnościach pomiędzy nimi.*.
6. (0,5pkt) W pliku XML (HH.xml) dodać swoje imię i nazwisko, jako zawartość elementów `name` oraz `surname`. Swoje imię i nazwisko wpisać również w komentarzu w pliku xsd.
7. (0,5pkt) W pliku HH.xml, w elemencie `system` dodać podelement `description` (zawartość elementu przekopiować z poprawionego pliku HH0.xml). W pliku xsd, odpowiednio uzupełnić definicję typu dla elementu `system`, aby była ona zgodna ze strukturą pliku xml.

```
<systems>
   <system>
      <name>HyperCard</name>
      <description>
         HyperCard was a powerful, yet extremely easy to use tool for creating “stacks” -
      </description>
   </system>
   <system>
      <name>World Wide Web</name>
      <description>
         The World Wide Web is a collection of linked resources and documents, connected
      </description>
   </system>
</systems>
```

8. **Poprosić prowadzącego o sprawdzenie pracy**

9. Dla ułatwienia przygotowane zostały kolejne wersje plików XML. Do pracy należy zgodnie z instrukcją brać kolejne pliki XML, plik xsd jest zawsze ten sam, sukcesywnie uzupełniany podczas laboratorium. Deklaracje i definicje w pliku xsd powinny odpowiadać temu, co znajduje się w pliku XML, aby walidacja przebiegała bez błędów.
10. (1pkt) Do dalszej pracy wziąć plik HH_10.xml. W pliku xsd zadeklarować dwa globalne elementy: `image` oraz `link`. Jako typu dla tych elementów użyć typu `adressType` zdefiniowanego globalnie na końcu pliku xsd. W definicji typu dla elementu `system` dodać deklaracje elementów `link` oraz `image` wykorzystując referencję do zadeklarowanych elementów globalnych. Dla obu elementów ustawić wartości atrybutów `minOccurs="0" maxOccurs="unbounded"`.
11. (1pkt) Stworzyć typ globalny prosty o nazwie `shortStringType` oparty o typ łańcucha znaków (`xs:string`). Określić minimalną dopuszczalną długość łańcucha znaków na 1 a maksymalną dopuszczalną długość łańcucha znaków na 30. Wykorzystać zdefiniowany typ w deklaracji elementów: `name` oraz `surname` w typie `authorType` (zamiast typu wbudowanego `xs:string`). Analogicznie stworzyć typ `longStringType` o długości 70 znaków. Typ ten zostanie wykorzystany w dalszej części laboratorium.
12. **Poprosić prowadzącego o sprawdzenie pracy**
13. (3pkt)Do dalszej pracy wziąć plik HH_13.xml. W pliku xsd zadeklarować w typie dla elementu `hipertext` element `persons`. Dla elementu `persons` stworzyć globalny typ `personsType`, w którym zadeklarować elementy zgodnie z tym, co znajduje się w pliku xml.
    Dla elementu `person` stworzyć globalny typ złożony, w którym zostaną zadeklarowane występujące w nim podelementy, zgodnie z plikiem xml.
    W deklaracji elementów `birth` i `death` wykorzystać typ wbudowany `xs:gYear`. Należy pamiętać o ustawieniu w odpowiednich deklaracjach atrybutów `minOccurs, maxOccurs.`
    Podobnie jak w pk.10 dla elementów `image` i `link` wykorzystać referencje do istniejących elementów globalnych.
    W typie dla elementu `person` (`personType`) zadeklarować atrybut `alive`. Typ dla atrybutu zdefiniować lokalnie i wykorzystując `enumeration` zdefiniować tylko dwie dopuszczalne wartości: `yes` oraz `no`. Atrybut ma być wymagany (`use="required"`). Należy pamiętać, że deklaracje atrybutów następują po wszystkich deklaracjach elementów.
14. **Poprosić prowadzącego o sprawdzenie pracy**
15. (2pkt) Do dalszej pracy wziąć plik HH_15.xml. W pliku xsd, w typie dla elementu `person` dodać deklarację elementu `achievements`. Typ dla tego elementu zdefiniować lokalnie.
    Typ dla elementu `achievement` zdefiniować globalnie i nazwać go `achievementType`. Podobnie jak w pk.10 oraz 13 dla elementów `image` i `link` wykorzystać referencje do istniejących elementów globalnych. Dla elementu `title` wykorzystać zdefiniowany wcześniej typ `longStringType`.
16. (0,5pkt) Typ dla elementu `definition` zdefiniowano lokalnie. Należy ten typ lokalny zamienić na typ globalny i zmienić deklarację elementu `definition` używając w niej nowo utworzonego typu globalnego.
17. (0,5pkt) zamienić definicję typu `personType` tak, aby była ona rozszerzeniem typu `authorType`.
18. **Poprosić prowadzącego o sprawdzenie pracy**

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 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”>
    < xs:restriction base=“string”/>
        < xs:maxLength value=“20” />
    </ xs:restriction>
</ xs:simpleType>
```

typ prosty nazwany

ograniczenie

2) Definicja elementu

```
<xs:element name="pajaki" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nazwa" maxOccurs="unbounded">
        <xs:complexType mixed="true">
          <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" />
    <xs:attribute name="chroniony" type="xs:string" />
  </xs:complexType>
</xs:element>
```

liczba wystąpień

definicja elementu

typ złożony, lokalny

sekwencja, elementy w ściśle określonej kolejności

typ złożony z zawartością mieszaną

typ atrybutu

deklaracja atrybutu (zawsze po deklaracjach elementów)

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 (referencje) do elementu

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

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

dr inż. Wioleta Szwoch
Katedra Inteligentnych
Systemów Interaktywnych
WETI, PG