Hipertekst i Hipermedia
Projekt

Temat:

MOJE HOBBY

Etapy:

Etap	Punktacja [pkt]
HTML, Dokument XML, XML Schema, DTD	25
XSLT	15

Etap 1: HTML (8pkt), Dokument XML, XML Schema (12pkt), DTD (5pkt)

HTML: (8 pkt)
Wymagania:
- zawartość strony zgodna z tematem projektu
- HTML5,
- strona responsywna (minimum to obsługa dwóch różnych wielkości ekranu) (**0,4pkt**)
- walidowanie strony **(0,9pkt) (**https://validator.w3.org/**)**
- układ strony:
    - podział strony na kilka elementów (nagłówek, menu, stopka, pole z treścią), wykorzystanie znaczników semantycznych (header, footer, nav, …) **(1pkt)**
- rozdzielenie treści na kilka plików (przynajmniej trzy) **(0,6pkt)**
- menu zawierające przynajmniej trzy opcje, a jedna z nich z dodatkowymi opcjami podrzędnymi; zaznaczanie wybranej opcji **(0,5pkt)**
- umieszczenie na stronie multimediów:
    - grafika
        - galeria zdjęć (grafika rastrowa) (przynajmniej 5) ma być zorganizowana w postaci miniaturek, które można obejrzeć powiększone **(0,6pkt)**
        - grafika wektorowa SVG **(0,2 pkt)**
        - animacja (wykorzystanie mechanizmu CSS3) **(0,5pkt)**
- umieszczenie na stronie:
    - tabeli **(0,3pkt)**
    - odsyłaczy do innych stron internetowych **(0,3pkt)**
    - odsyłaczy (min 2) do wybranego miejsca w tekście lub do początku strony (wyświetlony tekst powinien być odpowiednio długi, aby była możliwość zademonstrowania tej opcji) **(0,3pkt)**
- style należy zdefiniować w oddzielnym arkuszu stylów, wykorzystać mechanizm CSS
    - różne style dla przynajmniej 4 selektorów (grup selektorów) **(0,6 pkt)**
    - klasy (przynajmniej 3) **(0,6pkt)**
    - identyfikator (przynajmniej 1) **(0,2pkt)**
    - wykorzystanie pseudoklasy **(0,1pkt)**
    - wykorzystanie pseudoelementu **(0,1pkt)**
- stworzenie prostej ankiety-formularza **(0,8pkt)**
    - przynajmniej 7 pól do wprowadzania danych
    - przynajmniej 5 różnych rodzajów pól umożliwiających wprowadzanie danych,
    - przyciski do czyszczenia zawartości formularza oraz wysyłania danych (w atrybucie *action* formularza nie używać mailto)
- dbałość o estetyczny wygląd strony

XML: (0,6pkt)

Wymagania:

- utworzyć plik w formacie XML zawierający **dane** związane z tematem projektu. Dane zawarte w dokumencie XML powinny odpowiadać danym prezentowanym na stronie HTML. Błędem jest tworzenie dokumentu XML zawierającego znaczniki odpowiedzialne za prezentację danych – odzwierciedlające rozmieszczenie danych na stworzonej w poprzednim punkcie stronie HTML. Plik XML należy tworzyć zwracając uwagę na umieszczane w nim dane, relacje pomiędzy nimi a nie na to jak były umieszczone na stronie HTML.

W pliku XML:

- muszą istnieć co najmniej 4 poziomy zagłębienia nie licząc korzenia (**0,1pkt**)
- należy wykorzystać przynajmniej 5 różnych atrybutów (**0,1pkt**)
- dokument XML-owy ma zawierać przynajmniej 10 różnych nazw elementów (**0,1pkt**)
- trzeba umieścić dane dla przynajmniej trzech podelementów korzenia (**0,1pkt**)
- muszą znajdować się zdjęcia (przynajmniej 3) (**0,1pkt**)
- muszą znajdować się linki (przynajmniej 3) (**0,1pkt**)

XML SCHEMA: (12pkt)

Wymagania:

- Wykorzystać plik XML z pierwszego etapu. Jeśli niezbędne są w nim pewne zmiany, należy je wprowadzić.
- W pliku XML muszą istnieć przynajmniej 3 wypełnione podelementy korzenia
- W pliku XML muszą istnieć zdjęcia oraz linki
- Dla pliku XML, aby wymusić jego odpowiednią składnię, należy zaprojektować i utworzyć plik XML Schema.
- Plik XML musi być poprawny składniowo i semantycznie. Struktura pliku XML musi być zgodna z podaną w XML Schema. Do sprawdzenia poprawności należy użyć walidatora (**https://www.corefiling.com/opensource/schemavalidate/**).
- Dla stworzonego pliku XML wygenerować XML Schema przy użyciu Visual C++. Na zaliczenie projektu należy przynieść zarówno XML Schema stworzony przez siebie, jak i wygenerowany automatycznie.
- Należy również zwrócić uwagę na postać dokumentu, czyli sposób zapisu, stosowanie wcięć obrazujących strukturę danych, odpowiednie (adekwatne do zawartej w nich treści) nazywanie znaczników, atrybutów.

Wymagania szczegółowe:

W pliku XML Schema należy zadeklarować i wykorzystać:

- co najmniej 6 definicji globalnych typów złożonych **(1,6pkt)**
- przynajmniej 5 definicji globalnych typów prostych **(1,6pkt)**
- co najmniej 2 definicje lokalnych typów złożonych **(0,8pkt)**
- przynajmniej 2 definicje lokalnych typów prostych **(0,8pkt)**
- stosowanie różnych modeli wyboru, mieszanego typu zawartości **(0,4pkt)**
- przynajmniej jedna definicja grupy (elementów lub atrybutów) **(0,4pkt)**
- istnienie przynajmniej 4 poziomów zagłębienia w strukturze dokumentu xml **(0,4pkt)**
- deklaracja przynajmniej 6 atrybutów z czego przynajmniej 1 zdefiniowany globalnie i użyty przynajmniej 2 razy **(1,2pkt)**
- różnorodne deklaracje przynajmniej 12 różnych elementów (0,6pkt), w tym przynajmniej 5 z nich powinno zawierać podelementy(1pkt) **(1,6pkt)**
- stosowanie aspektów (ograniczeń na elementy i atrybuty)
  - *length, minLength, maxLength, maxInclusive, minInclusive,maxExclusive, minExclusive,* (wybrane min 4) **(0,4pkt)**
  - *pattern* i *enumeration* **(0,8pkt)**
- wyprowadzanie typów **(0,4pkt)**
  - *extension* - rozszerzenie istniejącego typu o dodatkowe elementy

- przynajmniej 3 odnośniki do elementów i/lub atrybutów (ma być odniesienie i do atrybutu i do elementu) **(0,6pkt)**
- użycie listy (0,1pkt), należy określić długość listy (liczbę elementów listy) ((0,1pkt) oraz ograniczyć wartości, jakie mogą wystąpić na liście (0,1pkt), lista musi być również wykorzystana w pliku XML (0,1pkt) **(0,4pkt)**
- wykorzystanie kombinacji (*union*) oraz użycie elementu tego typu w pliku XML **(0,4pkt)**
- wykorzystanie przynajmniej 4 różnych typów wbudowanych (**0,2pkt**)
- walidowanie pliku
- w pliku XML przynajmniej 3 wypełnione podelementy korzenia

XML, DTD: (4,4pkt)

Wymagania:

- Dla pliku XML (wykorzystanego w zadaniu z XML Schema), aby wymusić jego odpowiednią składnię, należy zaprojektować i utworzyć plik DTD.
- Plik XML musi być poprawny składniowo i semantycznie. Struktura pliku XML musi być zgodna z podaną w DTD. Do sprawdzenia poprawności należy użyć walidatora umieszczonego na stronie enauczanie.
- Należy również zwrócić uwagę na postać dokumentu, czyli sposób zapisu, stosowanie wcięć obrazujących strukturę danych, odpowiednie (adekwatne do zawartej w nich treści) nazywanie znaczników, atrybutów.

Wymagania szczegółowe:

W pliku DTD należy wykorzystać:

- deklaracje elementów, w deklaracji elementów wykorzystać zarówno model sekwencji jak i wyboru (**1,1pkt**)
- deklaracje przynajmniej 6 atrybutów (**1pkt**)
- w deklaracjach atrybutów wykorzystać:
  - typ wyliczeniowy (**0,5pkt**)
  - nadanie wartości domyślnej (**0,5pkt**)
  - zapewnienie wymagania wystąpienia atrybutu (**0,5pkt**)
- encję zewnętrzną (**0,2pkt**)
- encję (**0,2pkt**)
- encję parametryczną przynajmniej 2 razy (**0,4 pkt**)
- niedozwolone jest używanie konstrukcji ANY

Wybrane przykładowe błędy występujące w schematach:

- tworzenie dokumentu XML zawierającego znaczniki odzwierciedlające sposób prezentacji danych na stronie np. <podstrona> <akapit></akapit></podstrona> a nie same dane (**-1,5pkt**)
- błędy walidacji XML(plik się nie waliduje) (**do -10pkt**)
- trywialna definicja typu prostego (np. typ prosty, który jest zwykłym typem string) (**-2pkt**)
- powtarzanie definicji typów (wielokrotne definiowanie takich samych typów) (**-2pkt**)
- wykorzystanie anyType (**do -10pkt**)
- nieznacznie przerobiony, wygenerowany plik xsd (**do -10pkt**)
- niepoprawne definiowanie elementów, atrybutów, struktury (**do -6pkt**)
  - np. zamiast używać maxOccurs=4, czterokrotne deklaracje takiego samego elementu
- brak zdjęć (w XML (min 4) oraz w Schema (**-1pkt**)
- brak linków (w XML (min 4) oraz w Schema (**-1pkt**)
- brak w pliku XML przynajmniej 3 wypełnionych podelementów korzenia (**-2pkt**)
- wykorzystanie ANY w DTD (**-1pkt**)

Uwaga

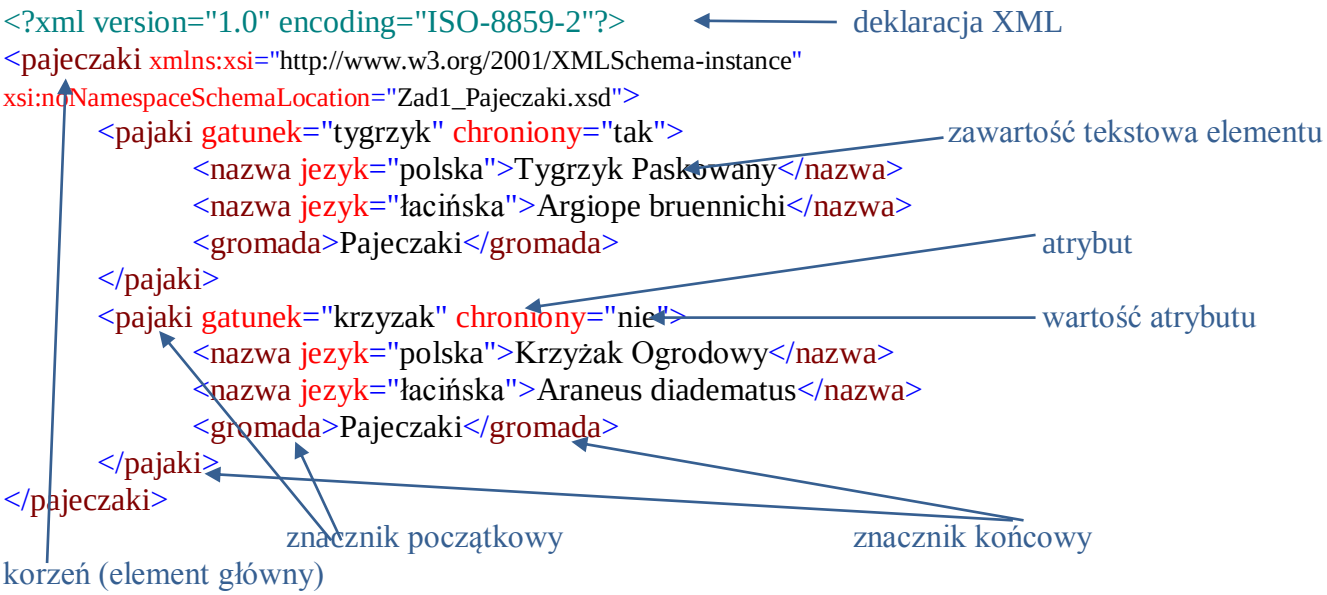
- Ostateczna liczba punktów za projekt jest uzależniona od odpowiedzi udzielanych podczas oddawania projektu, orientacji w projekcie i obowiązujących zagadnieniach.
- Podczas oddawania projektu **kod** ma być **pozbawiony** wszelkich **komentarzy**

Oddawanie projektów

- każda osoba ma wyznaczony na odbiór projektu termin: dzień, godzinę, oraz prowadzącego. Odbiór projektu odbywa się tylko w wyznaczonym terminie.
- Odbiory projektów będą odbywały się online
- nie ma możliwości poprawiania oddanych projektów

- **XML, DTD 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 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 atrybutami
  - typ złożony
- element z atrybutami i zawartością prostą
  - typ złożony z simpleContent

1) Definicja typu prostego nazwanego

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

2) Definicja elementu

```
<xs:element name="pajaki" maxOccurs="unbounded">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="nazwa" maxOccurs="unbounded">
            <xs:complexType >
               <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 atrybutu

definicja atrybutu (zawsze po definicjach elementów)

3) Wyliczenia - lista predefiniowanych wartości

```
<xs:simpleType name="nazwa_typu" >
  < xs:restriction base="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 dodać atrybuty do 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

DTD

6) Deklaracja elementu

<!ELEMENT nazwa_elementu *typ_zawartości_elementu*>

7) Wskaźniki liczby wystąpień

? 0 lub 1 raz
+ 1 lub dowolnie wiele razy
* 0 lub dowolnie wiele razy

8) Deklaracja atrybutu

<!ATTLIST nazwa-elementu
nazwa-atrybutu1 typ-zawartości-atrybutu1 opis1
nazwa-atrybutu2 typ-zawartości-atrybutu2 opis2
...>

Typ zawartości atrybutu

CDATA dowolny tekst,
ID nazwa unikalna w całym dokumencie, dla danego typu elementu tylko jeden atrybut może być zadeklarowany jako ID,
IDREF nazwa występująca gdzieś w dokumencie jako wartość typu ID

Opis: określa czy atrybut jest wymagany i jaką ma wartość domyślną

#REQUIRED	atrybut wymagany,	#IMPLIED	atrybut opcjonalny,
"wartość-domyślna"	wartość domyślna atrybutu	#FIXED "wartość"	wartość ustalona.

9) Deklaracja encji

- wewnętrznych
  <!ENTITY *nazwa_encji* ‘*reprezentowany tekst*’**>**
- zewnętrznych
  <!ENTITY *nazwa_encji* **SYSTEM ‘***nazwa.pliku***’>**
- parametrów
  <!ENTITY % *nazwa_encji* **‘***reprezentowany tekst***’>**