Laboratorium Hipertekst i Hipermedia

1. CEL ĆWICZENIA

Ćwiczenie ma na celu zapoznanie studentów ze standardami XML oraz XML Schema.

2. WPROWADZENIE

2.1. Co to jest XML

XML (ang. *Extensible Markup Language*) – uniwersalny język formalny przeznaczony do reprezentowania różnych danych w strukturalizowany sposób.

XML jest niezależny od platformy, co umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami i znacząco przyczyniło się do popularności tego języka w dobie Internetu. XML jest standardem rekomendowanym oraz specyfikowanym przez organizację W3C.

Dokument XML powinien być poprawny składniowo. Podstawowe zasady:

- XML rozróżnia wielkie i małe litery, zatem znacznik <abc> nie jest tym samym, co <Abc>,
- Znaczniki muszą być domknięte, zatem musimy pisać znacznik otwierający i zamykający, czyli <abc> </abc>; znaczniki te muszą być oczywiście identyczne,
- Nazwa znacznika musi się rozpoczynać od znaku litery lub znaku podkreślenia (_); w nazwie mogą wystąpić litery, cyfry, kropki (.), podkreślenia (_) i łączniki (-),
- Elementy muszą być poprawnie zagnieżdżane - jeśli otworzysz znacznik A, a potem niższy w hierarchii znacznik B, to najpierw musisz zamknąć B, zanim zamkniesz A,
- Wartości atrybutów muszą być ujęte w cudzysłów.

Przykładowy plik XML

```
<?xml version="1.0"?>
<zamowienie>
  <wyslac_do kraj="PL">
    <osoba>Krzysztof Nowak</osoba>
    <ulica>Niepodleglosci</ulica>
    <numer>799A</numer>
    <kod_pocztowy>81-100</kod_pocztowy>
    <miasto>Sopot</miasto>
  </wyslac_do>
  <rachunek>
    <osoba>Krzysztof Nowak</osoba>
    <ulica>Niepodleglosci</ulica>
    <numer>799A</numer>
    <kod_pocztowy >81-100</kod_pocztowy>
    <miasto>Sopot</miasto>
  </rachunek>
  <komentarz>Pilnie potrzebne</komentarz>
  <towary>
    <towar>
      <nazwa_towaru>Silnik iFlight 2216</nazwa_towaru>
      <ilosc>6</ilosc>
      <cena_jednostkowa>58.50</cena_jednostkowa>
    </towar>
    <towar>
      <nazwa_towaru>Kontroler ESC 30A iFlight</nazwa_towaru>
      <ilosc>6</ilosc>
      <cena_jednostkowa>62.00</cena_jednostkowa>
    </towar>
```

```
    </towary>
  </zamowienie>
```

2.1.1. Prolog dokumentu.

Każdy dokument XML rozpoczyna się od prologu, w którym zawarta jest deklaracja XML.

```
<?xml version="1.0" ?>
```

W prologu podajemy też zwykle stronę kodową dokumentu.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml version="1.0" encoding="iso-8859-2"?>
<?xml version="1.0" encoding="Windows-1250"?>
```

2.1.2. Element

Podstawowymi "klockami" budulcowym drzewa danych są elementy. Analizując przykładowy plik z punktu 2.1, możemy zauważyć, że mamy dokładnie jeden element główny *zamowienie*, który jest obowiązkowym składnikiem poprawnego składniowo dokumentu. Element główny zawiera inne elementy: *wysłać_do, rachunek, komentarz, towary* to podelementy.

Element składa się ze znacznika otwierającego, treści i znacznika zamykającego, np.

```
<ulica>Niepodleglosci</ulica>
```

Elementy zawierające podelementy lub atrybuty są nazwane **typami złożonymi** (*complex types*), natomiast elementy zawierające liczby, łańcuchy, daty, ale nie zawierające podelementów są **typami prostymi** (*simple types*).

2.1.3. Atrybuty

W elementach możemy stosować atrybuty, które precyzują informacje. Wartości atrybutów są zawsze umieszczane w cudzysłowach, co jest warunkiem poprawności składniowej dokumentu.

W dokumencie XML możemy napisać:

```
<jezyk rodzaj="sztuczny" rok_powstania="1887">Esperanto</jezyk>
```

Tutaj mamy dwa atrybuty: rodzaj języka i rok jego powstania.

2.2. Co to jest XML Schema

XML Schema (Schemat XML, Schemat Rozszerzalnego Języka Znaczników) to opracowany przez W3C standard służący do definiowania struktury dokumentu XML.

Przykładowy plik XML Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:annotation>
    <xsd:documentation xml:lang="pl">
    XML Schema do pliku zamowienia.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="zamowienie" type="zamowienie_typ"/>

  <xsd:element name="komentarz" type="xsd:string"/>

  <xsd:complexType name="zamowienie_typ ">
    <xsd:sequence>
      <xsd:element name="wyslac_do" type="adres_typ"/>
      <xsd:element name="rachunek" type="adres_typ"/>
      <xsd:element ref="komentarz" minOccurs="0"/>
      <xsd:element name="towary" type="towary_typ"/>
    </xsd:sequence>
```

```
  </xsd:complexType>

  <xsd:complexType name="adres_typ">
    <xsd:sequence>
      <xsd:element name="osoba" type="xsd:string"/>
      <xsd:element name="ulica" type="xsd:string"/>
      <xsd:element name="kod" type="xsd:decimal"/>
      <xsd:element name="miasto" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="kraj" type="xsd:NMTOKEN" fixed="PL"/>
  </xsd:complexType>

</xsd:schema>
```

Nowe typy złożone są definiowane za pomocą elementu *complexType*. Elementy są deklarowane z użyciem elementów *element*, atrybuty są definiowane z użyciem elementów *attribute*.

Na przykład typ Adres jest zdefiniowany jako typ złożony i wewnątrz zawiera cztery deklaracje elementów oraz jedną deklarację atrybutu. Konsekwencją tej definicji jest fakt, że każdy element pojawiający się w dokumencie jako *Adres* musi składać się z czterech elementów i jednego atrybutu. Te elementy muszą być nazwane: *osoba, ulica, kod i miasto* tak jak zostało określone w deklaracjach za pomocą atrybutu *name.* Dodatkowo elementy te muszą pojawiać się dokładnie w tej kolejności w jakiej zostały zadeklarowane (*sequence*). Element typu *Adres* może pojawić się z atrybutem *kraj*, który musi zawierać łańcuch *PL*.

```
      <xsd:element ref="komentarz" minOccurs="0"/>
```

Ta deklaracja tworzy referencję do istniejącego elementu *komentarz,* który został zadeklarowany w innym miejscu schematu. Ogólnie, wartość atrybutu *ref* musi wskazywać na element globalny, tzn. zadeklarowany pod elementem *schema*, a nie wewnątrz definicji typu złożonego.

Element ten jest zadeklarowany wewnątrz *typzamowienia* jako opcjonalny, ponieważ wartość atrybuty minOccurs wynosi 0. Wystąpienie jest wymagane jeżeli wartość ta wynosi 1 lub więcej.

2.3. Typy proste wbudowane w XML Schema

Typ prosty	Przykład		Typ prosty	Przykład
string	Tutaj jest tekst		float	-1E4, 12.45E-4
byte	-1, 126		double	-1E4, 12.45E-4
integer	-12345, -1, 0, 1, 12345		boolean	true, false, 0, 1
int	-1, 1234567		time	17:15:12.021, 16:12:00.000-05:00
long	-1, 1234567890		dateTime	1999-05-31T16:17:00.000-05:00
decimal	-1.23, 0, 123.4, 1000.00		date	2013-10-10

Nowe typy proste są definiowane przez wyprowadzenie ich z już istniejących typów prostych. Np. możemy wprowadzić nowy typ prosty poprzez ograniczenie istniejącego typu prostego, np.

```
<xsd:simpleType name=”liczba_calkowita_typ”>
  <xsd:restriction base=xsd:integer”>
    <xsd:minInclusive value=”999”/>
    <xsd:maxInclusive value=”99999”/>
  </xsd:restriction>
</xsd:simpleType>
```

Stworzony został nowy typ prosty na podstawie typu prostego *Integer*, którego wartości zawierają się w przedziale <999,99999> - przedział zamknięty, czyli: 999, 1000, 1001, …., 99998, 99999.

Poniższy przykład

```
<xsd:simpleType name=”liczba_calkowita_typ”>
  <xsd:restriction base=xsd:integer”>
    <xsd:minExclusive value=”999”/>
    <xsd:maxExclusive value=”99999”/>
  </xsd:restriction>
</xsd:simpleType>
```

Prawie identyczny z poprzednim. Różnica polega na użyciu zwrotów minExclusive i maxExclusive (zamiast minInclusive i maxInclusive). Ten typ oznacza wartości całkowite z zakresu (999;99999) – przedział otwarty, czyli: 1000, 1001, ……, 99998.

Stworzenie typu prostego jest możliwe także przy użyciu aspektu *pattern*.

```
<xsd:simpleType name=”SKU_typ”>
  <xsd:restriction base=xsd:string”>
    <xsd:pattern value=”\d{3}-[A-Z]{2}”/>
  </xsd:restriction>
</xsd:simpleType>
```

Oznacza to: trzy cyfry, potem myślnik, potem dwie wielkie litery ASCII (np. 123-AB).

Aspekt *enumeration* ogranicza typ prosty do zbioru konkretnych wartości, np.:

```
<xsd:simpleType name=”wojewodztwo_typ”>
  <xsd:restriction base=xsd:string”>
    <xsd:enumeration value=”pomorskie”/>
    <xsd:enumeration value=”zachodniopomorskie”/>
    <xsd:enumeration value=”kujawsko-pomorskie”/>
  </xsd:restriction>
</xsd:simpleType>
```

Należy pamiętać, że wartości wyliczenia muszą być unikalne.

Długość elementu.

```
<xsd:element name="haslo">
  <xsd:simpleType>
     <xsd:restriction base="xsd:string">
       <xsd:minLength value="5"/>
       <xsd:maxLength value="8"/>
     </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
```

Dzięki poleceniom minLenght oraz maxLength można podać przedział długości znaków użytych do opisu.

3. Zadania do wykonania

I. Rozgrzewka

L.p.	Zadanie	Punkty
1)	Sprawdź poprawność pliku 00_XML.xml za pomocą: a) Otwierając plik za pomocą przeglądarki WWW (co najmniej dwóch, np. IE i FireFox) b) Korzystając z serwisu sprawdzającego poprawność pliku (np. http://www.corefiling.com/opensource/schemaValidate.html	-
2)	Wykonaj te same kroki co w poprzednim punkcie otwierając plik 01_XML.xml	-
3)	napraw WSZYSTKIE błędy w pliku 01_XML.xml, następnie sprawdź poprawność pliku	5

II. Edytuj samodzielnie pliki XML i XSD (02_XML.xml oraz 02_XML.xsd)

L.p.	rozsz.	Zadanie	Punkty
1)	XML	Wprowadź kolejny zamówiony towar,	5
	XSD	Nic	
2)	XML	Nic	5
	XSD	Stwórz nowy typ prosty ceny, tak aby były to wartości większe od 0 i mniejsze od 10 tyś, zastosuj nowy typ	
3)	XML	Dodaj pozycję producenta towaru (do wszystkich towarów)	5
	XSD	Uzupełnij plik o wprowadzoną pozycję producenta towaru	
4)	XML	Bez zmian	5
	XSD	Zdefiniuj nowy typ prosty nazwy towaru tak aby jego długość była nie mniejsza niż 3 znaki i nie dłuższa niż 30, zastosuj nowy typ	
5)	XML	Do znacznika <zamowienie> dodaj atrybut określający datę przyjęcia zamówienia	5
	XSD	Dodaj odpowiedni wpis	
6)	XML	Dodaj datę realizacji zamówienia do ostatniego produktu w zamówieniu,	10
	XSD	Uzupełnij plik o pozycję daty realizacji zamówienia (ma to być pozycja opcjonalna do wypełnienia),	
7)	XML	Do znacznika <wyslac_do> oraz <rachunek> dodaj atrybut reprezentujący kraj, wpis w pliku XSD jest już wykonany, użyj poprawnej nazwy atrybutu, która jest użyta w pliku XSD	10
	XSD	Odpowiedni wpis jest już wykonany,	
8)	XML	Do znacznika <towar> dodaj atrybut *kod_produktu*, o strukturze: XYZ-234-56 (do wszystkich towarów)	15
	XSD	Kod produktu wprowadź jako aspekt *pattern*	
9)	XML	Rozszerz adres wysyłki (tylko adres do wysyłki) o pozycję województwa	15
	XSD	Wprowadź aspekt *enumeration*, do wyliczenia województw (wprowadź wszystkie województwa),	
10)		Przeprowadź walidacje końcowych plików	20
		SUMA	100

UWAGA!!! Po każdym wykonaniu zadania z podpunktu, sprawdź czy pliki przechodzą proces walidacji.

III. Dla stworzonego pliku XML w poprzednim zadaniu wygeneruj plik XSD (XLM Schema) przy użyciu Visual Studio. Zaobserwuj różnice w pliku wygenerowanym automatycznie a edytowanym ręcznie.