METAJĘZYKI

3

mgr inż. Robert Berezowski

e-mail: beny@ie.tu.koszalin.pl

Ul. Śniadeckich 2

Pokój 223A

Politechnika Koszalińska

Wydział Elektroniki i Informatyki

Katedra Inżynierii Komputerowej

Spis treści

Przestrzenie nazw

- Za pomocą XML można deklarować języki (schematy) w różnych zastosowaniach.
- W różnych zastosowaniach te same **<znaczniki>** mogą odpowiadać różnym danym:
  - **<lekarz>** – **<imie/>**, **<nazwisko/>**
  - **<pacjent>** – **<imie/>**, **<nazwisko/>**
- Jeżeli mamy wpływ na projektowanie znaczników to możemy zmienić nazwy (**<imie.lekarz>**, **<nazwisko.pacjent>**)
- ale
  - Często korzystamy z gotowych schematów, gdzie nie można zmienić **<znaczników>** i powstaje „konflikt nazw”
- **Rozwiązanie – Przestrzeń nazw (XML namespaces)**

Przestrzenie nazw

- <lekarz> – <imie/>, <nazwisko/>

  lekarz:http://qqq.url.pl/lekarz.dtd

  <lekarz:imie/>, < lekarz:nazwisko/>

- <pacjent> – <imie/>, <nazwisko/>

  pacjent:http://qqq.url.pl/lekarz.dtd

  <pacjent:imie/>, <pacjent:nazwisko/>

- **Specyfikacja:**

  http://www.w3.org/TR/REC-xml-names

Przestrzenie nazw

- **Deklaracja:**
  - **<przedrostek:nazwa.elementu xmlns:przedrostek= ”adres”>**
    - **nazwa.elementu** – najczęściej jest to element główny dokumentu,
    - **przedrostek** – dodatkowy skrót z jakim będą używane kolejne elementy
      - **dowolna, poprawna nazwa XML,**
      - **nie może zawierać znaku ‘:’,**
      - **nie można użyć słowa xml, xmlns,**
    - **adres** – zaleca się użycie adresu URL gdzie znajduje się opis struktury, której chcemy użyć.
  - **<xsl:stylesheet xmlns:xsl=”http://www.w3.org/XSL/Transform/1.0”>**
  - **<xsd:schema xmlns:xsd=”http://www.w3.org/1999/XMLSchema”>**
- **Domyślna przestrzeń nazw:**
  - **<schema xmlns=”http://www.w3.org/1999/XMLSchema”>**
- **Użycie**
  - **<przedrostek:nazwa.elementu>**
    - **<xsl:template>, <xsl:apply-templates>, <xsl:value-of>**
    - **<xsd:schema>, <xsd:element>, <xsd:complexType>**

Przestrzenie nazw

- **Przestrzenie standardowe - przedrostki:**
  - **html** -> HTML lub XHTML;
  - **xlink** -> XLink;
  - **xsd** -> XML Schema;
  - **xsl lub xs** -> XSLT;
  - **fo** -> XSL;
  - **mml** -> MathML;
  - **svg** -> SVD;
  - **rdf** -> RDF;
  - **rdfs** -> RDF Schema;
  - **ds** -> XML Signature
- **W jednym dokumencie można stosować kilka przestrzeni nazw.**

Przestrzenie nazw

- Przestrzeń nazw obowiązuję w elemencie, w którym został zadeklarowany i wszystkich elementach należących do zawartości danego elementu.

```
<liga:ekstraklasa xmlns:liga=„http://ie.pl/liga”>
    <team:druzyna xmlns:team=„http://ie.pl/druzyna”>
            …
    </team:druzyna>
    <tab:wyniki xmlns:tab=„http://ie.pl/wyniki”>
            …
    </tab:wyniki>
</liga:ekstraklasa>
```

Idea schematów

- Na podstawie badań stwierdzono, że około 60% tworzonego kodu (w aplikacjach) dotyczy weryfikacji poprawności danych.
- Przeniesienie zadania sprawdzania poprawności z tworzonej aplikacji na narzędzie walidujące daje spore oszczędności.

DTD raz jeszcze

- **Zalety DTD**
  - Prostota
- **DTD a przetwarzanie danych**
  - Do wyświetlania zwykle wystarczy poprawność *well formed*
  - Do przetwarzania stosuje się poprawność *valid*
  - *Elementy nie mogą mieć wartości domyślnych*
  - *Nie ma możliwości zmiany kolejności elementów*
  - DTD wystarcza do definiowania dokumentów „tekstowych”
    - Pełna dowolność zawartości
      - DTD nie wystarcza do definiowania złożonych struktur;
      - DTD nie daje możliwości sprawdzania poprawności typów danych za pomocą standardowych narzędzi

DTD raz jeszcze

- **Niedostatki DTD**
  - Brak definicji typów danych dla zawartości elementów i dla atrybutów
    - możliwe ograniczenia
      - wyliczanie dopuszczalnych wartości atrybutu
      - zasady tworzenia nazw i identyfikatorów
    - zawartość elementu: zawsze tekst
  - Brak możliwości efektywnego wyrażenia identyczności kilku części dokumentu
  - Brak dobrego wykorzystania przestrzeni nazw
    - przedrostki wpisane „na sztywno” do DTD
  - Język zapisu DTD
    - zupełnie różny od XML
    - nie daje możliwości przetwarzania narzędziami XML- owymi

XMLSchema – W3C

- Prace nad rekomendacją rozpoczęto w 1999r, zakończono w maju 2001r.
- Rekomendacja się z 3 części:
  - Część wstępna zawiera wprowadzenie do schematów,
  - Część druga zawiera formalny opis tworzenia schematów,
  - Część trzecia zawiera typy danych, jakie można stosować w schematach.

Cechy XMLSchema

- Dokładniejsze deklarowanie typów danych
  - możliwość deklaracji typów podstawowych
  - możliwość definiowania własnych typów
- Możliwość definiowania zbiorów elementów, w których kolejność jest dowolna
- Możliwość deklarowania wielu elementów o takiej samej nazwie, ale innym położeniu w dokumencie i innej budowie
- Możliwość jednokrotnego definiowania powtarzających się fragmentów struktury i ich wielokrotnego wykorzystywania w dalszej części schematu
- Mechanizmy pozwalające na rozszerzanie i uszczegóławianie struktury dokumentów;
- Korzystanie z wielu schematów w jednym dokumencie
- Uwzględnienie przestrzeni nazw;
- Komponowanie nowych modeli z kilku przestrzeni nazw

Dokument XML

```
<?xml version="1.0"?>
<!DOCTYPE auto SYSTEM "auto.dtd">
<auto>
    <fabryka>ford</fabryka >
    <kolor>czerwony</kolor>
     <rok>2004</rok>
</auto>
```

Dokument DTD

```
<!ELEMENT auto (fabryka, kolor, rok)>
<!ELEMENT fabryka (#PCDATA)>
<!ELEMENT kolor (#PCDATA)>
<!ELEMENT rok (#PCDATA)>
```

```
<?xml version="1.0"?>
 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="auto">
          <xsd:complexType>
          <xsd:sequence>
                    <xsd:element name="fabryka" type="xsd:string"/>
                    <xsd:element name=„kolor" type="xsd:string"/>
                    <xsd:element name=„rok" type="xsd:string"/>
          </xsd:sequence>
          </xsd:complexType>
 </xsd:element>
 </xsd:schema>
```

Dokument XMLSchema

Dokument XML

położenie naszej struktury XMLSchema

```
<?xml version="1.0"?>
<auto    xsd:noNamespaceSchemaLocation= ”auto.xsd”
         xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance">
    <fabryka>ford</fabryka >
    <kolor>czerwony</kolor>
     <rok>2004</rok>
</auto>
```

Odwołanie do deklaracji XMLSchema

```
<?xml version="1.0"?>
 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="auto">
          <xsd:complexType>
          <xsd:sequence>
                    <xsd:element name="fabryka” type="xsd:string"/>
                    <xsd:element name=„kolor” type="xsd:string"/>
                    <xsd:element name=„rok” type="xsd:int"/>
          </xsd:sequence>
          </xsd:complexType>
 </xsd:element>
 </xsd:schema>
```

Dokument ”auto.xsd”

Deklaracja elementów

- **Elementy można deklarować**
  - **Lokalnie - wewnątrz deklaracji typu - <xsd:complexType/>**
  - **globalnie - wewnątrz elementu <xsd:schema/>**
- Deklaracja bez nazwania typu (typ wykorzystany jednokrotnie):

```
<xsd:element name=”auto">
        <xsd:complexType>
                ...
        </xsd:complexType>
</xsd:element>
```

- Deklaracja z nazwaniem typu (typ wykorzystany wielokrotnie):

```
<xsd:element name=„auto" type=”typAuto”/>
<xsd:complexType name=”typAuto”>
                ...
</xsd:complexType>
```

Elementy

- Odwoływanie się do elementów zdefiniowanych wcześniej:

```
<xsd:element name=”auto” type=”xsd:string”/>…
<xsd:complexType>
        ...
        <xsd:element ref=„auto”/>
        ...
</xsd:complexType>
```

- Kontrolowanie liczby wystąpień:
  - Deklaracja:
    - minOccurs=„n” – minimalna liczba wystąpień (domyślnie 1);
    - maxOccurs=„n” – maksymalna liczba wystąpień (domyślnie 1)
      unbounded – dowolna liczba;
  - Deklarować można jedynie w typach lokalnych;
  - Przykład:
    - `<xsd:element name=”auto” type=”xsd:string” maxOccurs=”10”/>`

Typy złożone

- Typ złożony składa się z deklaracji elementów oraz atrybutów:

```
<xsd:complexType name=„nazwa”>
    Sekwencja elementów
</xsd:complexType>
```

  - **Sekwencja elementów może występować w jednej z trzech postaci:**
    - **sequence – elementy muszą wystąpić w określonym porządku (sekwencji)**
      - **W DTD odpowiada zapisowi (el1, el2, el3)**
    - **choice – może wystąpić tylko jeden z elementów (wybór)**
      - **W DTD odpowiada zapisowi (el1 | el2 | el3)**
    - **all – wszystkie elementy mogą wystąpić, ale kolejność jest dowolna**
      - **W DTD nie występuje**

Typ złożony – xsd:sequence

```
<xsd:complexType name="typadres">
    <xsd:sequence>
        <xsd:element name="ulica" type="xsd:string"/>
        <xsd:element name="numer.domu" type="xsd:string" minOccurs="0"/>
        <xsd:element name="numer.mieszkania" type="xsd:int"/>
        <xsd:element name="miejscowosc" type="xsd:string"/>
        <xsd:element name="kod" type="typkod"/>
    </xsd:sequence>
</xsd:complexType>
```

- Określa kolejność w jakiej mają pojawić się elementy;
- Muszą wystąpić wszystkie elementy (chyba, że minOccurs= ”0”)
- Może zawierać inne sekwencje elementów (np. **<xsd:choice>**);
- Element **<xsd:sequence >** jest odpowiednikiem **,** w DTD;

Typ złożony – xsd:choice

```
<xsd:complexType name="typpojazd">
    <xsd:choice>
        <xsd:element name="pociag" type="typpociag"/>
        <xsd:element name="tramwaj" type="typtramwaj"/>
        <xsd:element name="autobus" type="typautobus"/>
    </xsd:choice>
</xsd:complexType>
```

- Możliwy jest wybór dokładnie 1 elementu z pośród podanych;
- **<xsd:choice>** jest odpowiednikiem | w DTD;

Typ złożony – xsd:all

```
<xsd:complexType name="typauto">
    <xsd:all>
        <xsd:element name="kolor" type="typkolor"/>
        <xsd:element name="marka" type="xsd:string"/>
        <xsd:element name="rokprodukcji" type="xsd:int"/>
        <xsd:element name= "bak" type="xsd:int" minOccurs="0”/>
    </xsd:all>
</xsd:complexType>
```

- Wszystkie elementy mogą wystąpić w dowolnej kolejności
- Każdy element może wystąpić nie więcej niż raz
- jeżeli minOccurs= ”0” to może nie wystąpić
- Nie może zawierać modeli **<xsd:sequence>** i **<xsd:choice>**

Typy złożone – grupy elementów

- Grupy używamy jeżeli zestaw elementów pojawia się w kilku miejscach w dokumencie;
  - Grupy odpowiadają encjom parametrycznym w DTD
- Grupa elementów - definicja:

```
<xsd:group name="nazwa">
    <xsd:sequence> lub <xsd:choice> lub <xsd:all>
        <xsd:element ...>
        <xsd:element ...>
    <xsd:sequence> lub </xsd:choice> lub </xsd:all>
<xsd:group>
```

- Odwołanie:

```
<xsd:group ref="nazwa"/>
```

Typ złożony – model mieszany

- Elementy mieszane (tekst i elementy jednocześnie):

```
<xsd:complexType nazwa=„akapit” mixed=„true”>
...
<xsd:complexType>
```

- Elementy do zawartości mieszanej są idealne do przechowywania opisowych fragmentów informacji, w które wplecione są znaczniki;

Typ złożony – model mieszany - przykład

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="notatka">
    <xsd:complexType> <xsd:sequence>
        <xsd:element name="tresc" >
          <xsd:complexType mixed="true"> <xsd:sequence>
<xsd:element name="wyroznic" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>  </xsd:complexType>
        </xsd:element>
      </xsd:sequence>  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

```
<?xml-stylesheet type="text/xsl" href="notka.xsl"?>

<notatka xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="notka.xsd">

<tresc>Dnia <wyroznic>24 marca 2005r. </wyroznic> o godzinie
<wyroznic>17.15</wyroznic> odbeda sie zajecia z przedmiotu
<wyroznic>Metajezyki</wyroznic>. Obecnosc jak zwykle
<wyroznic>wskazana</wyroznic>.    Prowadzacy <wyroznic>mgr
inz. Robert Berezowski</wyroznic></tresc>

</notatka>
```

Zawartość prosta i złożona

- Istnieją dwa sposoby definiowane typów złożonych:
  - zawartość prosta – dane znakowe i atrybuty

```
<xsd:complexType>
        <xsd:simpleContent>...</xsd:simpleContent>
</xsd:complexType>
```

  - zawartość złożona – elementy i atrybuty; (przypadek domyślny)

```
<xsd:complexType>
        <xsd:complexContent>...</xsd:complexContent>
</xsd:complexType>
```

Rozszerzanie i zawężanie typów złożonych

- rozszerzenie typu prostego

```
<xsd:complexType> <xsd:simpleContent>
        <xsd:extension base=„xsd:integer">
          <xsd:attribute name= "przebieg"type="xsd:int"/>
          <xsd:attribute name= "drzwi"type="xsd:int"/>
        </xsd:extension>
</xsd:simpleContent> </xsd:complexType>
```

– rozszerzenie typu złożonego

```
<xsd:complexType>
    <xsd:complexContent>
        <xsd:extension base=„typAuto">
          <xsd:sequence>
            <xsd:element name= "przebieg"type="xsd:int"/>
            <xsd:element name= "drzwi"type="xsd:int"/>
          </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Rozszerzanie i zawężanie typów złożonych

- ograniczenie typu złożonego – wymaga powtórzenia wszystkich elementów, oprócz zbędnych

```
<xsd:complexType> <xsd:complexContent>
        <xsd:restriction base=„typAuto">
          <xsd:sequence>
            <xsd:attribute name= „marka"type="xsd:string"/>
            <xsd:attribute name= „color"type="xsd:string"/>
          </xsd:sequence>
        </xsd:restriction >
</xsd:complexContent> </xsd:complexType>
```

- Elementy puste

```
<xsd:complexType name="empty"/>
```

Treści (typy) w XMLSchema

- Typy treści:
  - Złożone:
    - Zawierające tekst, atrybuty i inne elementy;
      - Określają strukturę dokumentu:
        - Elementy zawierające inne elementy;
        - Elementy zawierające inne elementy i tekst;
        - Elementy zawierające tekst;
        - Elementy puste;

```
<xsd:complexType name="kolor">
    ...
</xsd:complexType>
```

  - Proste:
    - Zawierają tylko tekst i atrybuty (nie zawierają elementów);

```
<xsd:simpleType name="kolor">
    ...
</xsd:simpleType>
```

Treści (typy) w XMLSchema

- Typy proste – predefiniowane:
  <xsd:element name=„nazwa” type=„typ”/>
- typ - przykłady:

xsd:string – tekst;

xsd:boolean – true, false (0 lub 1);

xsd:byte – liczba od –128 do 127;

xsd:integer – liczba całk.– większy zakres

xsd:float – liczba zmiennoprzecinkowa

xsd:double – liczba zmiennoprzecinkowa

xsd:decimal – liczba zapisane dziesiętnie

xsd:date – data (1999-05-05);

xsd:time – czas (13:22:12.000);

xsd:century – wiek

xsd:month – miesiąc

xsd:timeDuration – czas trwania

xsd:year - rok

xsd:uri – adres URL (http://www.prv.pl);

- Przykład:
  - <xsd:element name=„wiek” type=„xsd:unsignedByte”>
    - <wiek>45</wiek>
  - <xsd:element waga> name=„waga” type=„xsd:string”>
    - <waga>79kg</waga>

Definiowanie własnego typu prostego

- Typy proste użytkownika:

  Ograniczenia wskazanego typu prostego

```
<xsd:simpleType name=„nazwaTypu">
    <xsd:restriction base=„jakiśTypProsty">
            ... ograniczenia ...
    </xsd:restriction>
</xsd:simpleType>
```

Ograniczenia zakresów

```
<xsd:element name=„gotowka”>
<xsd:simpleType>
    <xsd:restiction base=„xsd:integer”>
    <xsd:maxInclusive value=„5000”/>
      lub
    <xsd:maxExclusive value=„5000”/>
      lub
    <xsd:minInclusive value=„5000”/>
      lub
    <xsd:minExclusive value=„5000”/>
    </xsd:restiction>
    </xsd:element>
```

<gotowka>3000</gotowka>

- mniej niż 5000 lub równo (nie więcej niż)
- mniej niż 5000
- więcej niż 5000 lub równo (nie mniej niż)
- więcej niż 5000

<gotowka>6000</gotowka>

Ograniczenia zakresów

- Ograniczenia długości typu <xsd:string>:

```
<xsd: simpleType name=„cena”>
   <xsd:restriction base=„xsd:float”>
     <xsd:length value=„6”>
lub
     <xsd:minLength value=„5”>
     <xsd:maxLength value=„7”>
   </xsd:restiction>
</xsd:element>
```

- Ograniczenia cyfr:

```
<xsd: simpleType name=„cena”>
   <xsd:restriction base=„xsd:decimal”>
     <xsd:totalDigits  value=„5”> - ilość cyfr liczby;
     <xsd:fractionDigits  value=„2”> - ilość cyfr po przecinku;
</xsd:restiction>
</xsd:element>
```

Typ prosty wyliczeniowy

```
<xsd: simpleType name=„kontynent”>
  <xsd:restriction base=„xsd:string”>
    <xsd:enumeration value=„azja”/>
    <xsd:enumeration value=„afryka”/>
    <xsd:enumeration value=„europa”/>
    <xsd:enumeration value=„australia”/>
  </xsd:restriction>
</xsd: simpleType>
```

```
<kontynent>azja</kontynent>
```

Lista wartości

- Lista wartości rozdzielona spacjami

```
<xsd:simpleType name=„typMoj”>
  <xsd:list itemType="xsd:int"/>
</xsd:simpleType>
```

```
<lotek>4 5 33 34 36</lotek>
```

Definiowanie własnego typu prostego

- Typ o wartości sumy różnych typów prostych

```
<xsd:simpleType name=„typMoj”>
    <xsd:union  memberTypes=„od1do32 od44do54"/>
</xsd:simpleType>
  <xsd:simpleType name=„od1do32">
    <xsd:restriction base="xsd:int">
          <xsd:minInclusive value="1"/><xsd:maxInclusive value="32"/>
    </xsd:restriction>
  </xsd:simpleType>
<xsd:simpleType name=„od44do54">
    <xsd:restriction base="xsd:int">
          <xsd:minInclusive value="44"/><xsd:maxInclusive value="54"/>
    </xsd:restriction>
  </xsd:simpleType>
```

<liczba>45</liczba>

<liczba>12</liczba>

~~<liczba>36</liczba>~~

Definiowanie własnego typu prostego

- Definicja:

```
<xsd:simpleType name=„kodPocz”>
   <xsd:restriction base=„xsd:string”>
     <xsd:pattern value=„\d{2}-\d{3}”/>
   </xsd:restriction>
</xsd:simpleType>
```

<kodPocz>75-632</ kodPocz>

~~<kodPocz>735-632</ kodPocz>~~

- Wzorce (regularne wyrażenia Perla):
  - Litery, symbole i cyfry występują tak jak zostaną podane;
  - . Zastępuje dowolny znak;
  - \d – dowolną cyfrę \D – przeciwnie do \d
  - \s – dowolny biały znak (spacja, tabulator) \S – przeciwnie do \s
  - abc ciąg a, b, i c
  - [abc] – wystąpienie jednej z wartości z grupy: a, b lub c;
  - [0-9] – zakres wartości – od 0 do 9;
  - x? – opcjonalne wystąpienie x (może ale nie musi
  - Jeden | drugi – występuje jeden lub drugi;
  - X{5} – wystąpi 5 kolejnych X; X{5,} – wystąpi co najmniej 5 kolejnych X;
  - X{5,8} – wystąpi od 5 do 8 kolejnych X;

Deklaracja atrybutów

- Definicja:
  - atrybut o typie nazwanym

    ```
    <xsd:attribute name=„nazwa” type=„typ”/>
    ```

    Typy mogą być tylko „proste”
  - atrybut o typie anonimowym

    ```
    <xsd:attribute name=„lotek">
      <xsd:simpleType>
            <xsd:list itemType=„xsd:int"/>
      </xsd:simpleType>
    </xsd:attribute>
    ```
  - atrybut lokalny odwołujący się do globalnego

    ```
    <xsd:attribute ref="version"/>
    ```
- Atrybuty muszą być deklarowane na samym końcu typu złożonego (tylko i wyłącznie typu złożonego), do którego przynależą, czyli po zadeklarowaniu wszystkich składników typu złożonego;

Deklaracja atrybutów

- Występowanie atrybutów:
  - Atrybut wymagany - required:
    - **<xsd:attribute name=„pozycja” use=„required”... />**
  - Wartość domyślna – default (początkowa atrybutu) – jeżeli atrybut nie zostanie podany to przyjmuje wartość domyślną (value):
    - **<xsd:attribute name=„kolor” use=„default” value=„blue”... />**
  - Wartość ustalona – fixed– nawet jeżeli atrybut zostanie podany to i tak przyjmuje wartość domyślną (value):
    - **<xsd:attribute name=„kolor” use=„fixed” value=„blue”... />**
  - Wartość zabroniona - prohibited:
    - **<xsd:attribute name=„kolor” use=„prohibited” ... />**
- **Jeżeli element nie występuje w dokumencie to nie istnieją również jego atrybuty**

Komentarze

<xsd:annotation>
    <xsd:appinfo>
        Informacje na temat aplikacji XML do której należy schemat
        Przeznaczony dla aplikacji
    </xsd:appinfo>
    <xsd:documentation>
        Informacje o schemacie
        Przeznaczony dla ludzi
    </xsd:documentation>
</xsd:annotation>