Programowanie komunikacji człowiek-komputer

dr inż. Joanna Ochelska-Mierzejewska

2

Namespace

Przestrzenie nazw

3

Problem …

- Jak połączyć dokumenty

```
<autor>
  <tytuł>prof.</tytuł>
  <imię>Jan</imię>
  <nazwisko>Kowalski</nazwisko>
</autor>
```

```
<rozdział nr="1">
  <tytuł>O XML-u</tytuł>
  ...
</rozdział>
```

4

Rozwiązanie …

- Użycie przestrzeni nazw – powiązania między URI/IRI a wybraną „nazwą kontekstu"
  - Deklarujemy przestrzeń nazw dla poddrzewa atrybutem `xmlns:prefiks` z wartością URI/IRI – np. `xmlns:autorzy="http://example.com/autorzy"`
  - Aby powiedzieć, że nazwa atrybutu `<imię>` pochodzi z przestrzeni nazw `http://example.com/autorzy` używamy zapisu `<autorzy:imię>`
  - Brak prefiksu (`xmlns="http://example.com/autorzy"`) tworzy tzw. domyślną przestrzeń nazw

5

Przestrzenie nazw

- Wykorzystywane są do zapobiegania konfliktom nazewnictwa w sytuacji, kiedy podobnie nazwane elementy czy atrybuty zostają skojarzone z różnymi typami struktur danych wykorzystywanych w tym samum dokumencie
- Zdefiniowane w specyfikacji *Namespace for XML* w 1999 przez W3C

6

Przykład

```
<książka>
  <a:autorzy xmlns:a="http://example.com/authors">
    <a:autor>
      <a:tytuł>prof.</a:tytuł>
      <a:imię>Jan</a:imię>
      <a:nazwisko>Kowalski</a:nazwisko>
    </a:autor>
  </a:autorzy>
  <treść>
    <rozdział nr="1">
      <tytuł>O XML-u</tytuł>
      ...
    </rozdział>
  </treść>
</książka>
```

7

Które nazwy należą do której przestrzeni

```
<a xmlns:b="urn:b">
  <c xmlns="urn:x">
    <d:e xmlns="urn:y"
         xmlns:d="urn:d"
         b:f="g">
      <i xmlns:b="urn:b2">
        <j b:k="1"/>
      </i>
    </d:e>
    <m n="o"/>
  </c>
  <p/>
</a>
```

```
a?
    nie należy do żadnej przestrzeni nazw
c?
    należy do urn:x
e?
    należy do urn:d
f?
    należy do urn:b
i?
    należy do urn:y
j?
    należy do urn:y
k?
    należy do urn:b2
m?
    należy do urn:x
n?
    nie należy do żadnej!
p?
    nie należy do żadnej przestrzeni nazw
```

8

Przestrzenie nazw

```
<JĘZYK_HTML:p xmlns:JĘZYK_HTML="http://www.w3.org/TR/REC-html140">
        Treść traktowana jako paragraf w języki HTML
</JĘZYK_HTML:p>
```

```
<?xml version="1.0" ?>
<ListaStudentów>
 <ftims>
   <JĘZYK_HTML:img xmlns:JĘZYK_HTML="http://www.w3.org/TR/REC-html140" src="obrazek.jpg" />
   <JĘZYK_HTML:a xmlns:JĘZYK_HTML="http://www.w3.org/TR/REC-html140"
        href="stud.ics.p.lodz.pl/~kowalski"> Kowalski </JĘZYK_HTML:a>
</ftims>
</ListaStudentów>
```

```
<?xml version="1.0" ?>
<ListaStudentów xmlns:JĘZYK_HTML="http://www.w3.org/TR/REC-html140" >
 <ftims>
   <JĘZYK_HTML:img src="obrazek.jpg" />
   <JĘZYK_HTML:a href="stud.ics.p.lodz.pl/~kowalski"> Kowalski </JĘZYK_HTML:a>
</ftims>
</ListaStudentów>
```

9

Przestrzenie nazw

```
<?xml version="1.0" ?>
<STUDENCI:ListaStudentów xmlns:JĘZYK_HTML="http://www.w3.org/TR/REC-html140"
                         xmlns:STUDENCI="file://dtd/listaStudentow.dtd">
   <STUDENCI:ftims>
     <JĘZYK_HTML:img src="obrazek.jpg" />
     <JĘZYK_HTML:a href="stud.ics.p.lodz.pl/~kowalski"> Kowalski </JĘZYK_HTML:a>
     <STUDENCI:a href="stud.ics.p.lodz.pl/~kowalski"> Kowalski </STUDENCI:a>
   </STUDENCI:ftims>
</STUDENCI:ListaStudentów>
```

```
<?xml version="1.0" ?>
<ListaStudentów xmlns:JĘZYK_HTML="http://www.w3.org/TR/REC-html140"
                xmlns="file://dtd/listaStudentow.dtd">
   <ftims>
     <JĘZYK_HTML:img src="obrazek.jpg" />
     <JĘZYK_HTML:a href="stud.ics.p.lodz.pl/~kowalski" > Kowalski </JĘZYK_HTML:a>
     <a href="stud.ics.p.lodz.pl/~kowalski"> Kowalski </a>
   </ftims>
</ListaStudentów>
```

10

Przestrzenie nazw a DTD

- Nie ma możliwości umieszczenia
  - Informacji i identyfikatorze URI, z którym dane deklaracje DTD mogłyby być skojarzone
  - Prefiksu dla danego języka (prefiksu przestrzeni nazw), dzięki czemu można zapewnić określony prefiks dla danego DTD
- Ograniczenia dla nazwy
  - Nazwa atrybutu może zawierać co najwyżej jeden dwukropek
  - Nazwa encji, obiektu instrukcji procesora w ogóle nie może zawierać dwukropka
  - Wartość atrybutu `ID`, `IDREF`, `IDREFS`, `ENTITY`, `ENTITIES`, `NOTATION` nie mogą zawierać atrybutu

```
<!ELEMENT prefiks:nazwaElementu (#PCDATA)>
```

```
<!ATTLIST prefiks:nazwaElementu prefiks:nazwaAtrybutu CDATA #REQUIRED>
```

11

Standardowe przestrzenie nazw

Język	Identyfikator URI	Zwyczajowy prefiks
HTML	http://www.w3.org/TR/REC-html140	`html:`
XHTML	http://www.w3.org/1999/xhtml	`html:`
XLink	http://www.w3.org/1999/xlink	`xlink:`
XML Schema	http://www.w3.org/2001/XMLSchema http://www.w3.org/2001/XMLSchema-instance	`xsd:`
XSLT	http://www.w3.org/1999/XSL/Transform	`xsl:`
XSL	http://www.w3.org/1999/XSL/Format	`fo:`
SMIL 2.0	http://www.w3.org/2001/SMIL20/	
SVG	http://www.w3.org/2000/svg	`svg:`

12

XML Schema

Schemat rozszerzalnego języka znaczników

13

Świat bez XML Schema

- Mamy system przetwarzający zamówienia przesyłane jako dokumenty XML-owe zawierające pole odpowiadające liczbie zamawianych produktów

```
<!ELEMENT zamówienie (liczba-produktów)>
<!ELEMENT liczba-produktów (#PCDATA)>
```

- Co się stanie, gdy system odbierze zamówienie

```
<zamówienie>
 <liczba-produktów>no kocham go do szaleństwa</liczba-produktów>
</zamówienie>
```

- A takie?

```
<zamówienie>
 <liczba-produktów>-10000</liczba-produktów>
</zamówienie>
```

- Przydałoby się niskopoziomowe sprawdzenie poprawności – najlepiej wbudowane w sam dokument

14

Specyfikacja XML Schema

- Najważniejsze dokumenty
  - 1999: dokument W3C opisujący wymagania stawiane przed nowym formatem: mechanizmy tworzenia struktury, typy proste, reguły przetwarzania
  - 2001: XML Schema staje się oficjalną rekomendacją W3C
    - XML Schema Part 0: Primer
    - XML Schema Part 1: Structures
    - XML Schema Part 2: Datatypes
  - 2004: drugie wydane Specyfikacji

15

Wprowadzenie

- Najważniejsze informacje
  - XML Schema rozszerza funkcjonalność DTD (możliwa jest automatyczna konwersja DTD do formatu XML Schema)
  - Do definicji typu dokumentu w formacie XML Schema wykorzystywana jest standardowa składnia XML-a
  - Składniki definicji należą do przestrzeni nazw XML Schema `http://www.w3.org/2001/XMLSchema` (najczęściej oznaczana `xsd`)
  - Cała definicja zawarta jest w elemencie głównym `<xsd:schema>`, zaś odpowiednikami deklaracji elementów i atrybutów z DTD są elementy `<xsd:element>` i `<xsd:attribute>`

16

Przestrzenie nazw w XML Schema

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

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

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

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

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

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

17

Elementy i atrybuty

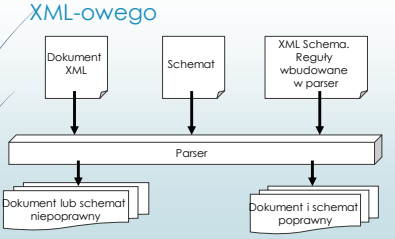
- W deklaracjach określa się
  - Typ przechowywanych danych
  - Wymagania związane z liczbą wystąpień
  - Wymagania związane z koniecznością wystąpień
  - Itp.

```
<?xml version="1.0" ?>
<imię wiek="20"> Zosia </imię>
```

```
<?xml version="1.0" ?>
<xsd:schema  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="imię" type="xsd:string" />
    <xsd:attribute name="wiek" type="xsd:integer" />
</xsd:schema>
```

18

Sprawdzanie poprawności dokumentu XML-owego



19

Typy proste i złożone

- XML Schema wprowadza do XML-a typy danych. Pod względem zawartości typy mogą być proste (ang. *simple type*) i złożone (ang. *complex type*)
- **Typy proste** odpowiadają zawartości bez wewnętrznej struktury (bez podelementów ani atrybutów). Mogą być przypisane elementom i atrybutom
- **Typy złożone** mogą mieć zawartość elementową i wprowadzać atrybuty
- Nadawanie typu jest realizowane poprzez atrybut `type`
  - Atrybut `type` może przybierać wartość
    - Typu wbudowanego, czyli zdefiniowanego bezpośrednio przez standard XML Schema
    - Typu prostego, czyli własnego, zdefiniowanego w dalszej części dokumentu `<xsd:simpleType>`
    - Typu złożonego, czyli własnego, zdefiniowanego w dalszej części dokumentu `<xsd:complexType>`

```
<xsd:element name="imie" type="xsd:string" />
<xsd:attribute name="semestrStudiów" type="xsd:integer" />
```

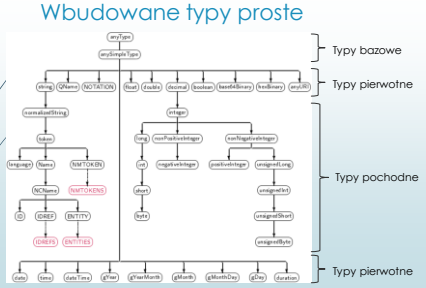
20

Typy proste

- Mogą być
  - **Wbudowane**, czyli zdefiniowane bezpośrednio przez standard
  - **Zdefiniowane przez użytkownika**, są zawsze typami pochodnymi
- Podział ze względu na pochodzenie
  - **Podstawowe**, czyli nie wynikające z innych typów
  - **Pochodne**, czyli wyprowadzone z innych typów
- Kategorie typów
  - Typy **atomowe** – takie, których wartości są niepodzielne
  - **Listy** – skończona lista wystąpień jednego typu atomowego
  - **Kombinacje** – złożenia dowolnych typów atomowych lub list jednocześnie

21

Wbudowane typy proste



22

Typy bazowe `xsd:anyType` i `xsd:anySimpleType`

- Typ `xsd:anyType` może być użyty jawnie do definiowania zawartości elementu

```
<xsd:element name="klient" type="xsd:anyType" />
```

- Taka definicja zezwala na dowolną zawartość znakową i elementową
- Jeśli w definicji elementu nie zostanie podana nazwa typu, typ `xsd:anyType` zostanie użyty jako domyślny
- Podobną funkcję dla atrybutów pełni typ `xsd:anySimpleType`, który nie dopuszcza zawartości elementowej

23

Wbudowane typy proste

Typy **czasowe**: `duration, dateTime, time, date, gYearMonth, gDay, GMonthDay, gMonth`

Typy **logiczne**: `boolean`

Typy **znakowe**: `string, normalizedString, token`

Typy XML: `anyURI, QName, NOTATION, language, Name, NCName, ID, IDREF, IDREFS, ENTITIES, NMTOKEN, NMTOKENS`

Typy **binarne**: `base64Binary, hexBinary`

Typy **wyliczeniowe**: `float, double, decimal, integer, nonPositiveInteger, negativeInteger, nonNegativeInteger, unsignedLong, unsignedInt, unsignedShort, unsignedByte, positiveInteger, long, int, short, byte`

24

Typ `duration`

```
P1Y3M5DT6H3M30S
P30D
PT5H
```

- Określa czas trwania (okres), bez ustalonego początku i końca
- Format: `PnYnMnDTnHnMnS`
  - `P` – znak wymagany określający okres
  - `nY` – wskazuje liczbę lat (5Y => 5 lat)
  - `nM` – wskazuje liczbę miesięcy (5M => 5 miesięcy)
  - `nD` – wskazuje liczbę dni
  - `T` – wskazuje początek opcjonalnej sekcji czasu. Każdorazowo, gdy chcemy określić liczbę godzin, minut i sekund należy użyć `T`
  - `nH` – wskazuje liczbę godzin
  - `nM` – wskazuje liczbę minut
  - `nS` – wskazuje liczbę sekund

```
<xsd:element name="okres" type="xsd:duration" />
<okres>P2Y6M</okres>
```

25

Typ dataTime

2010-10-10T14:30:50
2010-10-10T06:30:50Z
2010-10-10T14:30:50+08:00
(Seul – 8 godz. wcześniej, przesunięcie względem czasu CTU)

- Oznacza określony punkt w czasie rzeczywistym
- Format: CCYY-MM-DDThh:mm:ss
  - CC – reprezentuje setki lat określany dwoma lub większą liczbą cyfr (od 00 do ...)
  - YY – reprezentuje lata, określona dwoma cyframi (od 00 do 99)
  - MM – reprezentuje miesiąc określany dwoma cyframi (od 01 do 12)
  - DD – reprezentuje dzień, określany dwoma cyframi (od 01 do 31)
  - T – wskazuje początek obowiązkowej części określającej czas
  - hh – reprezentuje godziny określane dwoma cyframi (od 00 do 23)
  - mm – reprezentuje minuty określane dwoma cyframi (od 00 do 59)
  - ss – reprezentuje sekundy określane dwoma cyframi (od 00 do 59)
- Określenie stref czasowych
  - Format czasu UTC (czas Greenwich) → **Z**: CCYY-MM-DDThh:mm:ssZ
  - Przesunięcie względem UTC: CCYY-MM-DDThh:mm:ss+hh:mm lub CCYY-MM-DDThh:mm:ss-hh:mm

```
<xsd:attribute name="data" type="xsd:dateTime" />
<okres data="2006-10-16T14:30:50" />
```

26

Typ time

08:30:05
08:30:50**Z**
14:30:50+08:00

- time oznacza określony punkt czasowy w ciągu dnia
  - Format: hh:mm:ss
    - hh – reprezentuje godziny określane dwoma cyframi (od 00 do 23)
    - mm – reprezentuje minuty określane dwoma cyframi (od 00 do 59)
    - ss – reprezentuje sekundy określane dwoma cyframi (od 00 do 59)
  - Można określić strefę czasową
    - Format czasu UTC (czas Greenwich) → **Z**: hh:mm:ssZ
    - Przesunięcie względem UTC: hh:mm:ss+hh:mm lub hh:mm:ss-hh:mm

```
<xsd:element name="godzina" type="xsd:time" />
<godzina>14:30:50</godzina>
```

27

Typ date

- data oznacza datę wg kalendarza gregoriańskiego
  - Format: CCYY-MM-DD
    - CC – reprezentuje setki lat określany dwoma lub większą liczbą cyfr (od 00 do ...)
    - YY – reprezentuje lata, określona dwoma cyframi (od 00 do 99)
    - MM – reprezentuje miesiąc określany dwoma cyframi (od 01 do 12)
    - DD – reprezentuje dzień, określany dwoma cyframi (od 01 do 31)

```
<xsd:element name="dzień" type="xsd:date" />
<dzień>2006-10-16</dzień>
```

28

Typ gYear, gYearMonth, gMonthDay, gMonth, gDay

Typ	Opis	Format	Przykład
gYear	Oznacza rok	CCYY	2007
gYearMonth	Oznacza miesiąc w roku	CCYY-MM	2007-10
gMonthDay	Oznacza dzień konkretnego miesiąca, ale bez roku	--MM-DD	--12-10
gMonth	Numer dowolnego miesiąca, bez roku	--MM--	--10--
gDay	Oznacza dzień dowolnego miesiąca	---DD	---10

29

Typ boolean, hexBinary, base64Binary

- boolean używany do przedstawiania wartości logicznych prawda i fałsz
  - Format: true, false, 1, 0
- hexBinary oznacza dane binarne zakodowane w postaci tekstowej
  - Format: Literze N odpowiada kod: 4E
- base64Binary oznacza dane binarne zakodowane w postaci tekstowej za pomocą *Base64 Content-Transfer-Encoding*
  - Format: Ciągowi ABC odpowiada kod: GUJD

30

Typ string, normalizedString, token

- string oznacza ciąg bez znaków ze standardu Unicode, ciąg może zawierać spacje, znaki interpunkcyjne, itp.
- normalizedString oznacza ciąg bez znaków tabulacji, bez znaków nowego wiersza i bez znaków powrotu karetki
- token oznacza dowolny ciąg znaków bez tabulacji, nowego wiersza, powrotu karetki, wiodących i kończących spacji, ciągów spacji (dopuszczalne pojedyncze spacje); zwykle jest to pojedyncze słowo

```
<xsd:element name="tekst" type="xsd:string" />
<tekst>Student Jan Kowalski
Czy zaliczę kolejny semestr?
    @_#
</tekst>
```

31

Typ `string`, `normalizedString`, `token`

- `string` oznacza ciąg bez znaków ze standardu Unicode, ciąg może zawierać spacje, znaki interpunkcyjne, itp.
- `normalizedString` oznacza ciąg bez znaków tabulacji, bez znaków nowego wiersza i bez znaków powrotu karetki
- `token` oznacza dowolny ciąg znaków bez tabulacji, nowego wiersza, powrotu karetki, wiodących i kończących spacji, ciągów spacji (dopuszczalne pojedyncze spacje); zwykle jest to pojedyncze słowo

```
<xsd:element name="tekst" type="xsd:normalizedString" />
<tekst>Student Jan      Kowalski Czy zaliczę kolejny semestr?      @_#</tekst>
```

```
<xsd:element name="tekst" type="xsd:token" />
<tekst>Student Jan Kowalski Czy zaliczę kolejny semestr? @_#</tekst>
```

32

Dobre praktyki `string`, `normalizedString`, `token`

- `xsd:string`
  - gdy formatowanie białymi znakami ma znaczenie
  - dla długich napisów — warto też wtedy rozważyć użycie zawartości mieszanej
- `xsd:normalizedString`
  - gdy formatowanie białymi znakami nie ma znaczenia, ale ważne są pozycje znaków
- `xsd:token`
  - sprawdza się świetnie w przypadku krótkich napisów, zwłaszcza ograniczonych wyliczeniem lub wzorcem

33

Typ `float`, `double`, `decimal`, `integer`, `nonPositiveInteger`, `negativeInteger`, `nonNegativeInteger`, `positiveInteger`

- `float` – oznacza 32-bitową liczbę zmiennoprzecinkową, w typowej reprezentacji dziesiętnej
- `double` – oznacza 64-bitową liczbę zmiennoprzecinkową
- `decimal` – oznacza liczbę stałoprzecinkową
- `integer` – oznacza liczbę całkowitą (-∞, +∞); znak +/- nieskończoność jest niedozwolony
- `nonPositiveInteger` – oznacza liczbę całkowitą niedodatnią (-∞, 0); znak nieskończoności niedozwolony
- `negativeInteger` – oznacza całkowitą liczbę ujemną (-∞, -1); znak nieskończoności niedozwolony
- `nonNegativeInteger` – oznacza liczbę całkowitą nieujemną (0, +∞); znak nieskończoność niedozwolony
- `positiveInteger` – oznacza całkowitą liczbę dodatnią (1, +∞); znak nieskończoności niedozwolony

34

Typ `long`, `int`, `short`, `byte`, `unsignedLong`, `unsignedInt`, `unsignedShort`, `unsignedByte`

- `long` – oznacza liczby całkowite mieszczące się na ośmiu bajtach
- `int` – oznacza liczby całkowite mieszczące się na 4 bajtach
- `short` – oznacza liczbę całkowitą mieszczącą się na dwóch bajtach
- `byte` – oznacza liczbę całkowitą, która może być przedstawiona za pomocą jednego bajta
- `unsignedLong` – oznacza nieujemny typ `long`
- `unsignedInt` – oznacza nieujemny typ `int`
- `unsignedShort` – oznacza nieujemny typ `short`
- `unsignedByte` – oznacza nieujemny typ `byte`

35

Typ XML

`anyURI`	Oznacza dowolny identyfikator URI	`http://example.pl`
`QName`	Oznacza warstwę w jakiejś przestrzeni	`xsd:element`
`NOTATION`	Oznacza typ dla atrybutu NOTATION z XML 1.0; typ ten zaleca się używać tylko do atrybutów	`xsd:notation`
`language`	Oznacza dowolną prawidłową wartość `xml:lang`	`pl, en-US`
`Name`	Oznacza część (lub części) prawidłowej nazwy URI; także sam prefiks; typ ten musi zaczynać się od litery, podkreślenia lub dwukropka	`student:ftims`
`NCName`	Oznacza lokalną nazwę	`Wydział`
`ID`	Oznacz identyfikator; jest niepowtarzalny w ramach całego dokumentu XML – tylko do atrybutów	`Klucz15, p5`

36

Typ XML

`IDREF`	Oznacza typ dla atrybutu `IDREF` z XML 1.0 – stosowany tylko do atrybutu	`Klucz15`
`IDREFS`	Oznacza liczbę `IDREF` (elementy rozdzielone spacjami); należy stosować tylko do atrybutów	`Klucz15 klucz10`
`ENTITY`	Oznacza typ dla atrybutu `ENTITY` z XML 1.0; typ ten stosujemy do atrybutów	`obrazek`
`ENTITIES`	Oznacza listę wartości typu `ENTITY` (elementy listy są rozdzielone spacjami); należy go stosować tylko do atrybutów	`Rys1 rys2 rys3`
`NMTOKEN`	Oznacza typ dla atrybutów `NMTOKEN` z XML 1.0, typ ten należy stosować tylko do atrybutów	`Ala_Ma_Kota`
`NMTOKENS`	Oznacza listę wartości typu `NMTOKEN` (elementy listy rozdzielone są spacjami); należy go stosować tylko do atrybutów	`Ala_Ma_Kota Kot_Ma_Ale`

37

Stałe i domyślne wartości elementów

- W schemacie XML Schema można (w DTD nie można) podawać stałą i domyślną wartość elementów używając wykluczających się wzajemnie atrybutów `fixed` i `default`
  - Jeżeli wystąpił element o zawartości pustej, wynikowa zawartość elementu zostanie wypełniona wartością domyślną ze schematu
  - Jeżeli element w ogóle nie wystąpił, wartość domyślna nie zostanie użyta w ogóle (parser nie stworzy elementu)
  - Koncepcja wartości stałej jest identyczna, jeśli w dokumencie podano jakąś wartość, musi być ona równa wartości stałej, jeśli wartości nie podano, zostanie ona pobrana ze schematu

```
<xsd:element name="semestr" default="5" />
```

```
<xsd:element name="semestr" fixed="5" />
```

38

Definicje atrybutów

- `use`
  - Deklarowanie konieczności występowania (`required`), opcjonalności (`optional`) albo zabronienia wystąpienia (`prohibited`)
- `default`
  - Określenie wartości domyślnej
- `fixed`
  - Atrybut ustawia wartość stałą elementu lub atrybutu

```
<xsd:attribute name="wiek" type="xsd:positiveInteger" use="required" />
```

```
<xsd:attribute name="wersja" type="xsd:numer" use="optional" default="1.0" />
```

```
<xsd:attribute name="semestr" type="xsd:integer" use="required" fixed="7" />
```

39

Definiowane własnych typów

- Definiowane poprzez element `<xsd:simpleType>`
- Skorzystanie z aspektów wymaga zastosowania w definicji elementu `<xsd:restriction>` ograniczającego typ `<xsd:extension>` rozszerzającego typ wskazany atrybutem `base` lub podany w treści tego elementu

```
<xsd:element name="wiek">
  <xsd:simpleType>
      <xsd:restriction base="xsd:integer">
       <xsd:minInclusive value="1"/>
       <xsd:maxInclusive value="150"/>
      </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

40

Typy anonimowe i typy nazwane

- Typ, którego definicję podajemy w miejscu użycia, to **typ anonimowy**

```
<xsd:element name="wiek">
  <xsd:simpleType>
      ...
  </xsd:simpleType>
</xsd:element>
```

- Poprzez uniezależnienie definicji typu od miejsca jego wystąpienia możemy tworzyć **typy nazwane**

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

- I używać ich (wielokrotnie) w definicji schematu

```
<xsd:element name="cenaMasla" type="cena">
```

41

Tworzenie typów pochodnych

- Tworzenie typów pochodnych na podstawie innych typów może się odbywać z wykorzystaniem
  - Ograniczeń (`restriction`), czyli utworzenie typu, którego zbiór wartości jest podzbiorem wartości typu bazowego
  - Rozszerzeń (`extension`), np. poprzez dodanie do bazowego typu złożonego dodatkowych elementów
- Przestrzeń wartości typu wyprowadzonego musi być podzbiorem przestrzeni wartości typu bazowego

42

Własne typy danych

- Na podstawie typów predefiniowanych można łatwo tworzyć własne typy proste wykorzystując tzw. **aspekty** (ang. *facets*)
- Występujące aspekty
  - `length`, `minLength`, `maxLength` – określa dokładną minimalną i maksymalną długość
  - `maxInclusive`, `maxExclusive`, `minInclusive`, `minExlusive` – umożliwia ograniczenie zakresu wartości liczbowych
  - `totalDigits` – wyznacza maksymalną liczbę cyfr w liczbie (włącznie z ewentualnymi cyframi po przecinku, ale bez samego przecinka)
  - `fractionDigits` – określa liczbę miejsc po przecinku
  - `pattern` – umożliwia określenie dokładnego wzorca wartości elementu lub atrybutu
  - `enumaration` – określa wprost wartość dozwoloną w danym typie danych
  - `whiteSpace` – określa, jak mają być traktowane tzw. białe znaki (tabulator, nowy wiersz, spacja)
    - Dopuszczalne wartości: `preserve` – bez normalizacji ciągu – wartość domyślna, `replace` – zmiana wszystkich białych znaków na spacje, `collapse` – ciągi spacji zamieniane są na pojedynczą spację

43

Dozwolone apsekty

Typ	Dozwolone aspekty		
	pattern	enumaration	whiteSpace
Listy	+	+	+
Kombinacje	+	+	
Typy podstawowe, wbudowane			
string, base64Binary, hexBinary, anyURI, QName, NOTATION	+	+	+
boolean	+		+
float, double	+	+	+
decimal	+	+	+
duration, dateTime, time, date, gYearMonth, gYear, gDay, gMonthDay, gMonth	+	+	+
Wbudowane listy			
IDREF, NMTOKENS, ENTITIES		+	+

44

Typ	Dozwolone aspekty		
	length, minLength, maxLength	minInclusive, maxInclusive, minExclusive, maxExclusive	totalDigits, fractionDigits
Listy	+		
Kombinacje			
Typy podstawowe, wbudowane			
string, base64Binary, hexBinary, anyURI, QName, NOTATION	+		
boolean			
float, double		+	
decimal		+	+
duration, dateTime, time, date, gYearMonth, gYear, gDay, gMonthDay, gMonth		+	
Wbudowane listy			
IDREF, NMTOKENS, ENTITIES	+		

45

Aspekty – przykłady

```
<xsd:simpleType name="kodWaluty">
   <xsd:restriction base="xsd:string">
      <xsd:length value="3" />
   </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="nazwisko">
   <xsd:restriction base="xsd:string">
      <xsd:minLength value="2" />
      <xsd:maxLength value="30" />
   </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="maxCena">
   <xsd:restriction base="xsd:integer">
      <xsd:maxInclusive value="100" />
   </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="maxCena">
   <xsd:restriction base="xsd:integer">
      <xsd:maxExlusive value="101" />
   </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="wiek">
   <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="1" />
      <xsd:maxInclusive value="120" />
   </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="cena">
   <xsd:restriction base="decimal">
      <xsd:totalDigits value="5" />
      <xsd:fractionDigits value="2" />
   </xsd:restriction>
</xsd:simpleType>
```

46

Aspekt wzorca

- `pattern`
  - Określenie postaci wzorcowej przy użyciu wyrażeń regularnych

Metaznak	Opis	Przykład	
. (kropka)	pasuje do dowolnego znaku (bez znaku nowej linii)	`x.y`	wzorzec, w którym po x występuje dowolny znak ASCII, a po nim litera y
?	może być jeden raz, ale nie musi wcale	`xy?z`	wzorzec, który szuka łańcuchów xyz lub xz, znak przed ? może wystąpić 1 lub 0
*	może być więcej razy, ale nie musi być wcale	`xy*z`	wzorzec, który szuka łańcuchów xyz lub xz lub xyyyyz, znak przed * może wystąpić 0 lub kilka razy
+	musi wystąpić co najmniej raz	`xy+z`	wzorzec, który szuka łańcuchów xyz lub xyyyyz, znak przed + może wystąpić 1 lub kilka razy

47

Metaznak	Opis	Przykład	
{min, max}	Dokładne określenie zakresu wystąpień	`zima{2,5}`	zimaa, zimaaa, zimaaaa, zimaaaaa
		`kot{2,}`	kott, kottt, kotttt, ...
		`go{4}l`	gooool
Operator	**Opis**	**Przykład**	
|	lub	`ab|dc|ef`	

()	Do grupowania wyrażeń	`(S|s)tudent`
		`S|student`
		`pog(oda){3}`	

Klasa znaków	**Opis**	**Przykład**	**Właściwości**
[]	Stosowany do bardziej skomplikowanych wzorców	`[Ss]tudent`	
		`[*zosia]`	Metaznaki . * + tacą swoje własności
		`[xyz]?` x, y, x, pusty	można stosować operatory zakresu ?, *, +, {min, max)}
–	operator zakresu	`[a-z]`	

48

Aspekty – przykłady

- W DTD nie da się ograniczyć tekstowej zawartości elementów

```
<xsd:simpleType name="tabRejestracyjnaŁódź">
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="EL \d{5}" />
      <xsd:pattern value="EL \d{4}[A-Z]" />
      <xsd:pattern value="EL \d{3}[A-Z]{2}" />
   </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="NIP">
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="([0-9]{3}-[0-9]{2}-[0-9]{2}-[0-9]{3})
                | ([0-9]{3}-[0-9]{3}-[0-9]{2}-[0-9]{2})" />
   </xsd:restriction>
</xsd:simpleType>
```

49

Aspekty – przykłady

- W DTD nie da się ograniczyć tekstowej zawartości elementów
- Poszczególne dopuszczalne wartości podaje się w elementach typu `enumeration`

```
<xsd:simpleType name="TakNie">
    <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="tak" />
        <xsd:enumeration value="nie" />
    </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="gatunek">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="pop" />
        <xsd:enumeration value="rock" />
        <xsd:enumeration value="rap" />
    </xsd:restriction>
</xsd:simpleType>
```

50

Łączenie aspektów

- Aspekty mogą być łączone poprzez podanie kilku podelementów z poszczególnymi aspektami
  - Aspekty `pattern` oraz `enumeration` są łączone za pomocą logicznego „**albo**"
  - Pozostałe aspekty są łączone za pomocą logicznego „i"

```
<xsd:simpleType name="TakNie">
    <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="tak" />
        <xsd:enumeration value="nie" />
    </xsd:restriction>
</xsd:simpleType>
```

albo

```
<xsd:simpleType name="wiek">
    <xsd:restriction base="xsd:integer">
        <xsd:minInclusive value="1" />
        <xsd:maxInclusive value="120" />
    </xsd:restriction>
</xsd:simpleType>
```

i

51

Uwaga na ograniczenia

- Przestrzeń wartości wprowadzonego typu prostego musi być podzbiorem przestrzeni wartości typu bazowego
- Warto o tym pamiętać ograniczając typ wbudowany, dla którego określono niektóre aspekty. Typ `byte` może np. przyjmować wartości z przedziału -128 do 127

```
<xsd:simpleType name="typByteBłędny">
    <xsd:restriction base="xsd:byte">
        <xsd:minInclusive value="-256" />
        <xsd:maxInclusive value="255" />
    </xsd:restriction>
</xsd:simpleType>
```

52

Blokowanie wartości aspektów

- `fixed` – Blokowanie wartości wybranych aspektów

```
<xsd:simpleType name="nazwisko">
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="2" fixed="true" />
        <xsd:maxLength value="30" />
    </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="nazwiskoNowe">
    <xsd:restriction base="nazwisko">
        <xsd:minLength value="3" />
        <xsd:maxLength value="30" />
    </xsd:restriction>
</xsd:simpleType>
```

53

Listy wartości

- Listy to ciąg oddzielonych białymi znakami wartości wskazanego typu

```
<wynikDuzegoLotka>23 15 3 1 35 29</wynikDuzegoLotka>
```

```
<xsd:simpleType name="liczbyDuzyLotek">
    <xsd:restriction base="xsd:positiveInteger">
        <xsd:maxInclusive value="49"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="listaLiczb">
    <xsd:list itemType="liczbyDuzyLotek"/>
</xsd:simpleType>

<xsd:simpleType name="typWynikDuzegoLotka">
    <xsd:restriction base="listaLiczb">
        <xsd:length value="6"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:element name="wynikiDuzegoLotka" type="typWynikDuzegoLotka"/>
```

54

Unie, czyli kombinacje

- **Unia** to połączenie zakresów wartości kilku typów w jeden nowy zakres
- Instancja może mieć przypisaną wartość należącą do dowolnego z połączonych zakresów wartości

```
<xsd:simpleType name="stawkaVAT">
    <xsd:union>
        <xsd:simpleType>
            <xsd:restriction base="xsd:nonNegativeInteger">
                <xsd:maxInclusive value="100"/>
            </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="zw."/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:union>
</xsd:simpleType>

<xsd:element name="VAT" type="stawkaVAT"/>
```

55

Warianty

```
<xsd:simpleType name="RozmiarTyp">
  <xsd:union>
    <xsd:simpleType>
      <xsd:restriction base="xsd:integer">
        <xsd:minInclusive value="34"/>
        <xsd:maxInclusive value="62"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType>
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="S"/>
        <xsd:enumeration value="M"/>
        <xsd:enumeration value="L"/>
        <xsd:enumeration value="XL"/>
        <xsd:enumeration value="XXL"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:union>
</xsd:simpleType>

<xsd:element name="rozmiar"
             type="RozmiarTyp"/>
```

```
<rozmiar>48</rozmiar>
<rozmiar>XL</rozmiar>
```

```
<xsd:simpleType name="RozmiarLiczbowyTyp">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="34"/>
    <xsd:maxInclusive value="62"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="RozmiarTekstowyTyp">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="S"/>
    <xsd:enumeration value="M"/>
    <xsd:enumeration value="L"/>
    <xsd:enumeration value="XL"/>
    <xsd:enumeration value="XXL"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="RozmiarTyp">
  <xsd:union memberTypes="RozmiarLiczbowyTyp
                          RozmiarTekstowyTyp"/>
</xsd:simpleType>

<xsd:element name="rozmiar" type="RozmiarTyp"/>
```

56

Typy złożone

- Typy złożone umożliwiają definiowanie modeli zawartości oraz dołączenie atrybutów
- Definicję typu złożonego tworzy element `<xsd:complexType>`, w którego treści zawarte są podelementy odpowiadające modelowi zawartości
  - `<xsd:sequence>` – sekwencja wystąpień elementów
  - `<xsd:choice>` – alternatywa (| z DTD)
  - `<xsd:all>` – nieuporządkowany zbiór podelementów

57

sequence

DTD

```
<!ELEMENT uczeń(imię, nazwisko, klas)>
```

XML Schema

```
<xsd:element name="uczeń" >
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="imię" type="xsd:string" />
         <xsd:element name="nazwisko" type="xsd:string" />
         <xsd:element name="klasa" type="xsd:string" />
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

```
<uczeń>
  <imię>Jan</imię>
  <nazwisko>Kowalski</nazwisko>
  <klasa>3a</klasa>
</uczeń>
```

58

choice

DTD

```
<!ELEMENT osoba (imię | nazwisko | klas)>
```

XML Schema

```
<xsd:element name="uczeń" >
   <xsd:complexType>
      <xsd:choice>
         <xsd:element name="imię" type="xsd:string" />
         <xsd:element name="nazwisko" type="xsd:string" />
         <xsd:element name="klasa" type="xsd:string" />
      </xsd:choice>
   </xsd:complexType>
</xsd:element>
```

```
<uczeń>
  <imię>Jan</imię>
</uczeń>
```

```
<uczeń>
  <nazwisko>Kowalski</nazwisko>
</uczeń>
```

```
<uczeń>
  <klasa>3a</klasa>
</uczeń>
```

59

all

DTD

```
Nie da się określić dowolnej kolejności!!!
```

XML Schema

```
<xsd:element name="uczeń" >
   <xsd:complexType>
      <xsd:all>
         <xsd:element name="imię" type="xsd:string" />
         <xsd:element name="nazwisko" type="xsd:string" />
         <xsd:element name="klasa" type="xsd:string" />
      </xsd:all>
   </xsd:complexType>
</xsd:element>
```

```
<uczeń>
  <klasa>3a</klasa>
  <nazwisko>Kowalski</nazwisko>
  <imię>Jan</imię>
</uczeń>
```

```
<uczeń>
  <imię>Jan</imię>
  <nazwisko>Kowalski</nazwisko>
  <klasa>3a</klasa>
</uczeń>
```

60

Ograniczenia dla `all`

- W skład `all` mogą wchodzić jedynie pojedyncze podelementy
- Żaden podelement nie może się powtarzać
- Podelementy zawarte w `all` mają zawsze `maxOccures="1"`, natomiast `minOccurs="0"` lub `"1"`

```
<xsd:element name="student">
  <xsd:complexType>
     <xsd:all>
        <xsd:element name="imie" type="xsd:string"/>
        <xsd:element name="nazwisko" type="xsd:string"/>
        <xsd:element name="nrDowodu" type="xsd:string"
minOccurs="0"/>
     </xsd:all>
   </xsd:complexType>
</xsd:element>
```

61

Sekwencja i kontrola liczby wystąpień

- Określenie liczby wystąpień elementów odbywa się za pomocą atrybutów `minOccurs` i `maxOccurs`, które mogą przyjmować wartości naturalne rozszerzone o specjalną wartość `unbounded` oznaczającą nieograniczoną liczbę wystąpień
- **Uwaga!** Wartość domyślna obu atrybutów wynosi 1

```
<xsd:element name="imię" type="xsd:string" minOccurs="1" maxOccurs="3" />
<xsd:element name="nazwisko" type="xsd:string" minOccurs="0" maxOccurs="unbounded" />
```

62

Sekwencja i kontrola liczby wystąpień elementów

DTD

```
<!ELEMENT osoba (tytuł?, imię+, nazwisko, adres*)>
```

XML Schema

```
<xsd:element name="osoba">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="tytuł" minOccurs="0"/>
      <xsd:element name="imię" maxOccurs="unbounded"/>
      <xsd:element name="nazwisko"/>
      <xsd:element name="adres"
            minOccurs="0" maxOccurs="unbounded"/>
  </xsd:complexType>
</xsd:element>
```

63

Dowolna kombinacja elementów

DTD

```
<!ELEMENT osoba (imię | nazwisko | klas)*>
```

XML Schema

```
<xsd:element name="uczeń" >
   <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="imię" type="xsd:string" />
         <xsd:element name="nazwisko" type="xsd:string" />
         <xsd:element name="klasa" type="xsd:string" />
      </xsd:choice>
   </xsd:complexType>
</xsd:element>
```

64

Łączenie składania podelementów

DTD

```
<!ELEMENT dostawca (firma | (imie, nazwisko))>
```

XML Schema

```
<xsd:element name="dostawca">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="firma" type="xsd:string" />
      <xsd:sequence>
        <xsd:element name="imie" type="xsd:string" />
        <xsd:element name="nazwisko" type="xsd:string" />
      </xsd:sequence>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

```
<dostawca>
  <imie>Jan</imie>
  <nazwisko>Kowalski</nazwisko>
</dostawca>
```

```
<dostawca>
  <firma>Polmos</firma>
</dostawca>
```

65

Łączenie składania podelementów

DTD

```
<!ELEMENT dostawca (nazwisko, (NIP | Pesel))>
```

XML Schema

```
<xsd:element name="dostawca">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="nazwisko" type="xsd:string" />
      <xsd:choice>
        <xsd:element name="NIP" type="xsd:string" />
        <xsd:element name="Pesel" type="xsd:string" />
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<dostawca>
  <nazwisko>Nowak</nazwisko>
  <NIP>720-822-29-29</NIP>
</dostawca>
```

```
<dostawca>
  <nazwisko>Kowalski</nazwisko>
  <Pesel>64101004063</Pesel>
</dostawca>
```

66

Element pusty

DTD

```
<!ELEMENT hr EMPTY>
```

XML Schema

```
<xsd:element name="hr">
  <xsd:complexType />
</xsd:element>
```

67

Dowolna zawartość

- Dopuszczenie zawartości dowolnej ma zwykle na celu włączenie do dokumentu elementów z innej przestrzeni nazw (np. zagnieżdżenie fragmentu XHTML-a, MathML-a czy SVG)
- XML Schema udostępnia do tego celu element `<xsd:any>` z dwoma „specjalizowanymi" atrybutami
  - `namespace`, określającym przestrzeń nazw włączanej zawartości
  - `processContents`, określającym poziom walidacji włączanej zawartości i przyjmującym wartości
    - `skip` — nie waliduj
    - `lax` — waliduj wyłącznie elementy, dla których określono przestrzenie nazw
    - `strict` — wykonaj pełną walidację (wartość domyślna)

68

Dowolna zawartość

DTD

```
<!ELEMENT kodXHTML ANY>
```

XML Schema

```
<xsd:element name="kodXHTML">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any
              namespace="http://www.w3.org/1999/xhtml"
              maxOccurs="unbounded"
              processContents="skip"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

69

Dowolne atrybuty

- Istnieje także możliwość dopuszczenia wystąpienia w elemencie dowolnych atrybutów (dzięki czemu można np. skorzystać z zaawansowanych mechanizmów linkowania oferowanych przez standard XLink)

```
<xsd:element name="stronaWWW">
  <xsd:complexType>
    <xsd:anyAttribute namespace="http://www.w3.org/1999/xlink"/>
  </xsd:complexType>
</xsd:sequence>
```

70

Definicja atrybutów

DTD

```
<!ELEMENT osoba EMPTY>
<!ATTLIST osoba pesel CDATA #REQUIRED
                nip CDATA #IMPLIED
                gatunek CDATA #FIXED "Homo sapiens">
```

XML Schema

```
<xsd:element name="osoba">
  <xsd:complexType>
    <xsd:attribute name="pesel" type="xsd:string"
           use="required"/>
    <xsd:attribute name="nip" type="xsd:string" />
    <xsd:attribute name="gatunek"  type="xsd:string"
           fixed="Homo sapiens"/>
  </xsd:complexType>
</xsd:element>
```

71

Model mieszany

DTD

```
<!ELEMENT tekst (#PCDATA | wyróżnienie)*>
```

XML Schema

```
<xsd:element name="tekst">
  <xsd:complexType mixed="true">
    <xsd:choice>
      <xsd:element name="wyróżnienie" minOccurs="0"
            maxOccurs="unbounded"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

72

Model mieszany – co wolno w XML Schema?

- Trochę więcej niż w DTD — można określać
  - Porządek wystąpienia elementów (nie tylko `<xsd:choice>`, ale także `<xsd:sequence>`)
  - Liczbę wystąpień elementów w treści mieszanej (używając `minOccurs` i `maxOccurs`)

73

Element pusty z atrybutem

```
<galeriaZdjec>
    <zdjecie href="http://wp.pl/zdjecie1.jpg" />
    <zdjecie href="http://wp.pl/zdjecie2.jpg" />
    <zdjecie href="http://wp.pl/zdjecie3.jpg" />
</galeriaZdjec>
```

```
<xsd:element name="galeriaZdjec">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="zdjecie" type="linkObrazek"
                                      maxOccurs="unbounded" />
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>

<xsd:complexType name="linkObrazek">
    <xsd:attribute name="href" type="xsd:anyURI"
                                          use="required" />
</xsd:complexType>
```

74

Dozwolone operacji dla `restriction`

- Ograniczenie zawartości prostej – za pomocą aspektów
- Ograniczenie atrybutu
  - Ograniczenie typu atrybutu
  - Zmiana atrybutu opcjonalnego na wymagany (`required`) lub zabroniony (`prohibited`)
  - Dodanie, zmiana lub usunięcie wartości domyślnej
  - Dodanie wartości ustalonej, jeśli jej nie było,
- Ograniczenie modelu zawartości
  - Ściślejsze ograniczenia liczebności (`minOccurs`, `maxOccurs`)
  - Usunięcie elementów opcjonalnych w grupach `sequence` i `all`
  - Wybranie podzbioru elementów w grupie `choice`, ograniczenie typu poszczególnych podelementów

75

Dozwolone operacji dla `restriction`

```
<xsd:complexType name="osoba">
   <xsd:sequence>
      <xsd:element name="imie" type="xsd:string" maxOccurs="unbounded" />
      <xsd:element name="nazwisko" type="xsd:string" maxOccurs="1" />
   </xsd:sequence>
</xsd:comlexType>
```

```
<xsd:complexType name="uczen">
   <xsd:complexContent>
      <xsd:restriction base="osoba">
         <xsd:secuence>
            <xsd:element name="imie" type="xsd:string" />
            <xsd:element name="nazwisko" type="xsd:string" />
         </xsd:sequence>
      </xsd:restriction>
   </xsd:complexContent>
</xsd:comlexType>
```

76

Dozwolone operacji dla `extension`

- Rozszerzanie zawartości prostej – dodawanie atrybutów do: typu prostego, typu złożonego o zawartości prostej.
- Rozszerzanie zawartości złożonej: dodawanie do typu bazowego dodatkowych elementów i/lub atrybutów, elementy dodawane w rozszerzeniu występują zawsze na końcu, po elementach zadeklarowanych w typie bazowym.

77

Dozwolone operacji dla `extension`

```
<xsd:complexType name="osoba">
   <xsd:sequence>
      <xsd:element name="imie" type="xsd:string" maxOccurs="unbounded" />
      <xsd:element name="nazwisko" type="xsd:string" />
   </xsd:sequence>
</xsd:comlexType>
```

```
<xsd:complexType name="uczen">
   <xsd:complexContent>
      <xsd:extension base="osoba">
         <xsd:secuence>
            <xsd:element name="nrLeg" type="xsd:integer"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:comlexType>
```

78

Blokowanie rozszerzeń, ograniczeń

- `final`
  - Umieszczenie tego atrybutu uniemożliwia wprowadzenie ograniczenia (`final="restriction"`), rozszerzenia (`final="extension"`) lub każdego rodzaju wyprowadzenia (`final="#all"`)

```
<xsd:complexType name="osoba" final="#all" > ...
<xsd:complexType name="osoba" final="restriction" > ...
<xsd:complexType name="osoba" final="extension" > ...
```

79

Element z prostą zawartością i z atrybutem

<dostawca formaDzialanosci="IT">
Firma informatyczna SuperXML Sp. z o.o.
</dostawca>

- simpleContent
  - Definiowanie elementu nie posiadającego podelementów (atrybuty i treść)
  - Występuje tylko z <xsd:extension>

```
<xsd:simpleType name="nazwaFirm">
   <xsd:restriction base="xsd:string">
     <xsd:maxLength value="100" />
   </xsd:restriction>
</xsd:simpleType>
<xsd:element name="dostawca">
  <xsd:complexType>
    <xsd:simpleContent name="rodzajDostawcy" >
      <xsd:extension base="nazwaFirmy">
        <xsd:attribute name="formaDzialnosci" type="xsd:token" use="required" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

80

Deklaracje globalne i lokalne

- Elementy i atrybuty zadeklarowane na najwyższym poziomie hierarchii (będące bezpośrednimi podwęzłami elementu głównego <xsd:schema>) są deklarowane jako **globalne** i mogą być użyte w innych obszarach schematu
  - Atrybuty nie posiadają atrybutu use
  - Odwołanie do deklaracji globalnych poprzez atrybut ref
- Deklaracje na innych poziomach hierarchii są deklarowane jako **lokalne** i nie mogą być użyte ponownie w innych obszarach

81

```
<?xml version="1.0">
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="nazwisko" type="xsd:string" />
   <xsd:attribute name="plec">
      <xsd:simpleType>
          <xsd:restriction base="xsd:string">
              <xsd:enumeration value="kobieta" />
              <xsd:enumeration value="mężczyzna" />
          </xsd:restriction>
      </xsd:simpleType>
   </xsd:attribute>
   <xsd:simpleType name="kodPocztowy">
      <xsd:restriction base="xsd:string">
         <xsd:pattern value="[0-9]{2}-[0-9]{3}" />
      </xsd:restriction>
   </xsd:simpleType>

   <xsd:element name="osoba">
     <xsd:complexType>
       ...
     </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Atrybut nie ma use

Deklaracje globalne

Poprzez poniższy complexType zdefiniowano lokalny, anonimowy typ (definicja – następny slajd)

82

```
<xsd:element name="osoba">
   <xsd:complexType>
      <xsd:element ref="nazwisko" />
      <xsd:attribute ref="plec" use="required" />
      <xsd:element name="PESEL" type="xsd:string">
      <xsd:attribute name="stanCywilny">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
             <xsd:enumeration value="kawaler / panna"/>
             <xsd:enumeration value="zonaty / mezatka"/>
             <xsd:enumeration value="wdowiec / wdowa"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
   </xsd:complexType>
</xsd:element>
```

Poprzez ref odwołujemy się do deklaracji globalnych

Tutaj już pojawia się atrybut use

Kolejna deklaracja lokalna

83

Grupy elementów

Uwaga: Grupy muszą być nazwane i globalne

DTD

```
<!ENTITY % osoba 'imię, hobby'>
```

XML Schema

```
<xsd:group name="osoba">
 <xsd:sequence>
   <xsd:element name="imie" type="xsd:string" />
   <xsd:element name="hobby" type="xsd:string" />
 </xsd:sequence>
</xsd:group>
```

```
<xsd:element name="uczeń">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:group ref="osoba" />
            <xsd:element name="szkoła" type="xsd:string" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

84

Grupy atrybutów

DTD

```
<!ENTITY % osobaAtrybut 'PESEL CDATA #IMPLIED,
                         NIP CDATA #IMPLIED'>
```

XML Schema

```
<xsd:attributeGroup name="osobaAtrybut">
   <xsd:attribute name="PESEL" type="xsd:string" />
   <xsd:attribute name="NIP" type="xsd:string" />
</xsd:attributeGroup>
```

```
<xsd:element name="uczeń">
    <xsd:complexType>
      <xsd:attributeGroup ref="osobaAtrybut" />
    </xsd:complexType>
</xsd:element>
```

85

Definiowanie wartości niepowtarzalnych

- Za pomocą atrybutów typu `xsd:ID` i `xsd:IDREF` — jak w DTD
- **Za pomocą ograniczeń integralności (ang. *identity constraints*)**
  - **kluczy `<xsd:key>`**
  - **odwołań do kluczy `<xsd:keyref>`**
  - **wartości unikatowych `<xsd:unique>`**

86

Więzy integralności – definicja

- Definicja ograniczenia integralności składa sie z trzech części
  - **Zasięg** (ang. *scope*) jest określony przez element, w którego deklaracji zdefiniowano ograniczenie
  - **Selektor** (ang. *selector*) pozwala wskazać węzły, których dotyczy ograniczenie
  - Definicja jednego lub więcej **pól** (ang. *fields*) wskazuje elementy i atrybuty, których wartości muszą być unikatowe wśród węzłów wybranych selektorem. **Uwaga:** w każdym węźle wskazanym przez selektor może być co najwyżej jeden egzemplarz każdego pola
- Selektory i pola określa się przy pomocy ograniczonego podzbioru wyrażeń **XPath**

87

XPath w 30 sekund

- XPath to język do zapisu ścieżek do elementów i atrybutów
- Składnia — prawie jak w systemie plików:
  - `książka/rozdział` – wybierz podelementy typu `rozdział` z elementu `książka` będącego podelementem bieżącego elementu
  - `*/rozdział` – podelementy typu `rozdział` z dowolnych podelementów bieżącego elementu
  - `@nr` — wartość atrybutu `nr` bieżącego elementu

88

Ograniczenia integralności – unikalny identyfikator

```
<księgowość>
  <zamówienia>
    <zamówienie>
      <numer>125</numer>
        ...
    </zamówienie>
    <zamówienie>
      <numer>665</numer>
        ...
    </zamówienie>
  </zamówienia>
  <faktury>
    ...
  </faktury>
</księgowość>
```

Unikalność numerów zamówień w całym dokumencie;
Numery zamówień będą **przywoływane** na fakturach:

```
<xsd:key name="unikNrZam">
  <xsd:selector
    xpath="zamówienia/zamówienie"/>
  <xsd:field xpath="numer"/>
</xsd:key>
```

89

Ograniczenia integralności: unikalne numery i odwołania

```
<księgowość>
  ...
  <faktury>
    <faktura nr="123">
      <rok>2007</rok>
      <do-zamówienia
            nr="125"/>
        ...
    </faktura>
    <faktura nr="123">
      <rok>2006</rok>
      <do-zamówienia
            nr="665"/>
        ...
    </faktura>
  </faktury>
</księgowość>
```

Unikalność numerów faktur w danym roku w całym dokumencie

```
<xsd:unique name="fakturaId">
  <xsd:selector
      xpath="faktury/faktura"/>
  <xsd:field xpath="@nr"/>
  <xsd:field xpath="rok"/>
</xsd:unique>
```

90

Ograniczenia integralności: unikalne numery i odwołania

```
<księgowość>
  ...
  <faktury>
    <faktura nr="123">
      <rok>2007</rok>
      <do-zamówienia
            nr="125"/>
        ...
    </faktura>
    <faktura nr="123">
      <rok>2006</rok>
      <do-zamówienia
            nr="665"/>
        ...
    </faktura>
  </faktury>
</księgowość>
```

Odwołanie do numeru zamówienia

```
<xsd:keyref
      name="zamówienieRef"
      refer="zamówienieId">
  <xsd:selector
      xpath="faktury/faktura
          /do-zamówienia"/>
  <xsd:field xpath="@nr"/>
</xsd:keyref>
```

91

Ograniczenia integralności: cała definicja

```
<xsd:element name="ksiegowosc" type="typKsiegowosc">
  <xsd:unique name="unikalnyNrFaktury">
    <xsd:selector xpath="faktury/faktura"/>
    <xsd:field xpath="@nr"/>
    <xsd:field xpath="rok"/>
  </xsd:unique>
  <xsd:keyref name="odwołanieDoZamówienia" refer="nrZamówienia">
    <xsd:selector xpath="faktury/faktura/do-zamówienia"/>
    <xsd:field xpath="@nr"/>
  </xsd:keyref>
  <xsd:key name="nrZamówienia">
    <xsd:selector xpath="zamówienia/zamówienie"/>
    <xsd:field xpath="numer"/>
  </xsd:key>
</xsd:element>
```

92

Atrybuty przestrzeni nazw

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

Określenie przestrzeni nazw

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

Domyślna przestrzeń, do której przydzielane są wszystkie deklaracje elementów i atrybutów o nazwach bez prefiksu

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

Wskazuje na przestrzeń nazw identyfikowaną przez URI, której nadano prefiks **moje:**

93

Docelowa przestrzeń nazw dokumentów schematu

- Jeśli chcemy, by nazwy elementów, atrybutów i typów zdefiniowanych w dokumencie schematu należały do określonej przestrzeni nazw, musimy ją określić w atrybucie `targetNamespace` elementu głównego `<xsd:schema>`
- Brak tego atrybutu oznacza, że nazwy komponentów wynikowych nie będą należeć do żadnej przestrzeni nazw

94

Docelowa przestrzeń nazw dokumentu schematu

Dobrze:

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

>
  <xsd:complexType name="typBazowy">
    ...
  </xsd:complexType>
  <xsd:complexType name="typPochodny">
    <xsd:complexContent>
      <xsd:restriction base="typBazowy">
        ...
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="element" type="typPochodny">
</xsd:schema>
```

95

Docelowa przestrzeń nazw dokumentu schematu

Za mało:

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

>
  <xsd:complexType name="typBazowy">
    ...
  </xsd:complexType>
  <xsd:complexType name="typPochodny">
    <xsd:complexContent>
      <xsd:restriction base="typBazowy">
        ...
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="element" type="typPochodny">
</xsd:schema>
```

`targetNamespace="http://.../"`

Określenie docelowej przestrzeni nazw dla wszystkich składników

96

Docelowa przestrzeń nazw dokumentu schematu

Dobrze:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.example.org/test"
            xmlns="http://example.org/test"
>
  <xsd:complexType name="typBazowy">
    ...
  </xsd:complexType>
  <xsd:complexType name="typPochodny">
    <xsd:complexContent>
      <xsd:restriction base="typBazowy">
        ...
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="element" type="typPochodny">
</xsd:schema>
```

97

Docelowa przestrzeń nazw dokumentu schematu

Dobrze:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.example.org/test"
            xmlns:typy="http://example.org/test"
>
  <xsd:complexType name="typBazowy">
    ...
  </xsd:complexType>
  <xsd:complexType name="typPochodny">
    <xsd:complexContent>
       <xsd:restriction base="typy:typBazowy">
          ...
       </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="element" type="typy:typPochodny">
</xsd:schema>
```

98

Nazwy kwalifikowane i niekwalifikowane

- Nazwy **kwalifikowane** (ang. *qualified*) należą do pewnej przestrzeni nazw. Mogą być poprzedzone prefiksem lub należeć do domyślnej przestrzeni nazw
- Nazwy **niekwalifikowane** (ang. *unqualified*) nie należą do żadnej przestrzeni
- Autor schematu może zdecydować, czy w dokumentach elementy i atrybuty zdefiniowane lokalnie muszą być kwalifikowane czy nie:
  - na poziomie schematu — z użyciem atrybutów `elementFormDefault` i `attributeFormDefault` o wartościach `qualified` lub `unqualified` (**domyślnie**)
  - na poziomie lokalnej definicji — atrybutem `form` o takiej samej zawartości

99

Lokalne nazwy niekwalifikowane – przykład

Schemat

```
<xsd:complexType name="typOsoba">
  <xsd:sequence>
    <xsd:element name="imię" type="xsd:token" minOccurs="0" maxOccurs="2"/>
    <xsd:element name="nazwisko" type="xsd:token"/>
  </xsd:sequence>
  <xsd:attribute name="pesel"/>
</xsd:complexType>

<xsd:element name="osoba" type="typOsoba"/>
```

Dokument

```
<os:osoba xmlns:os="http://www.example.org/osoby" pesel="04250101234">
  <imię>Jan</imię>
  <nazwisko>Kowalski</nazwisko>
</os:osoba>
```

100

Lokalne nazwy kwalifikowane

Schemat

```
<xsd:schema ... elementFormDefault="qualified" attributeFormDefault"qualified">
```

Dokument 1

```
<os:osoba xmlns:os="http://www.example.org/osoby" os:pesel="04250101234">
  <os:nazwisko>Kowalski</os:nazwisko>
</os:osoba>
```

Dokument 2

```
<osoba xmlns="http://www.example.org/osoby"
       xmlns:osoby="http://www.example.org/osoby"
       osoby:pesel="63511756789"
>
  <nazwisko>Kowalski</nazwisko>
</osoba>
```

`elementFormDefault="qualified"` — Dodawanie elementów lokalnych do przestrzeni nazw (`unqualified`)

`attributeFormDefault="qualified"` — Dodawanie atrybutów do przestrzeni nazw (`unqualified`)

101

Powiązanie dokumentu XML z XMLSchema

- Składa się z trzech elementów
  - Deklaracji przestrzeni nazw dla egzemplarza dokumentu zgodnego z XML Schema `xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"`
  - Powiązania schematu dla elementów nie należących do żadnej przestrzeni nazw – poprzez podanie URL-a schematu w atrybucie `xsd:noNamespaceSchemaLocation`
  - Ewentualnych powiązań listy używanych przestrzeni nazw z URL-ami schematów mających posłużyć do walidacji elementów, których nazwy należą do używanych w dokumencie przestrzeni nazw – w atrybucie `xsd:schemaLocation`

102

Powiązanie dokumentu XML z XMLSchema

```
<?xml version="1.0" ?>
<moje:elementKorzeń
    xmlns:moje="http://www.example.org"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schamaLocation="http://www.example.org plik.xsd">
  ...
</moje:elementKorzeń>
```

`xmlns:xsi="http://wwww.w3.org/2001/XMLSchema-instace"` — xsi – standardowy prefiks dla przestrzeni nazw instancji schematu XML

```
<?xml version="1.0" ?>
<elementKorzeń
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamepaceSchemaLocation="plik.xsd">
  ...
</elementKorzeń>
```

`xsi:noNamespaceSchemaLocation` — Odniesienie do lokalizacji, gdy brak definicji docelowej przestrzeni nazw dokumentu

`xsi:schemaLocation` — Odniesienie do lokalizacji, gdy nie została zdefiniowana docelowa przestrzeń nazw dokumentu

103

Strategie budowania schematów

- Metoda zagnieżdżenia (Faktura1.xsd)
  - Nie stosuje globalnych definicji typów ani globalnych deklaracji elementów i atrybutów
  - Wiele deklaracji jest powielanych, przez co utrudnione jest wprowadzanie zmian do schematu, a sam schemat jest bardzo obszerny
- Metoda płaskiego katalogu (Faktura2.xsd)
  - Stosuje się globalne deklaracje elementów i atrybutów, ale nie używa się globalnych definicji typów
- Metoda definiowania typów (Faktura3.xsd, Faktura4.xsd)
  - Jest najmniej powieleń

104

Łączenie schematów

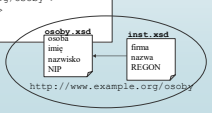
- Schemat (struktura logiczna) może być zapisany w wielu dokumentach schematów (plikach .xsd)
- Specyfikacja XML Schema określa trzy metody łączenia dokumentów schematów
  - `include`
  - `import`
  - `redefine`
- Lokalizacje dokumentów opisujących schemat są określone w egzemplarzu, a ponadto
  - procesor może używać dokumentów schematów z predefiniowanych lokalizacji
  - lokalizacje dokumentów schematów mogą być przekazywane jako parametry wiersza poleceń

105

Modularyzacja schematów metodą `<xsd:include>`

- Dołączanie dokumentu schematu do docelowej przestrzeni nazw głównego dokumentu schematu
- Dołączany dokument musi mieć taką samą docelową przestrzeń nazw jak dokument główny, lub nie mieć docelowej przestrzeni nazw

```
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://www.example.prg/osoby"
    targetNamespace="http://www.example.org/osoby">
  <xsd:include schemaLocation="inst.xsd"/>
  ...
</xsd:schema>
```



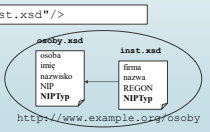
Uwaga! Dołączane schematy nie muszą być kompletne.
Źle, bo musimy pilnować zależności między schematami
Dobrze, bo możemy parametryzować schematy (np. definiować różnie wersje typów dla elementów o danych nazwach)

106

Modularyzacja schematów metodą `<xsd:redefine>`

- Dołączanie dokumentu schematu do docelowej przestrzeni nazw głównego dokumentu schematu z możliwością przedefiniowania
  - typów prostych i złożonych
  - nazwanych grup modeli
  - grup atrybutów

```
<xsd:redefine schemaLocation="inst.xsd"/>
```

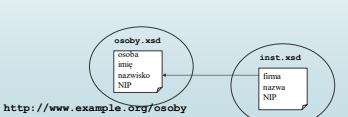


107

Modularyzacja schematów metodą `<xsd:import>`

- Dołącza dokument schematu z innej przestrzeni nazw

```
<xsd:import schemaLocation="inst.xsd"
    namespace="http://www.example.org/instytucje"/>
```



108

Wartości nieokreślone

- Koncepcja wartości niezdefiniowanych (ang. *nil values*) umożliwia zapis informacji o nieokreśloności konstrukcji wyrażonej danym elementem XML-owym
- Użycie
  - Możliwość wystąpienia wartości nieokreślonej zapisuje się w schemacie oznaczając element atrybutem `nillable="true"`
  - Tak oznaczony element będzie mógł być w dokumencie opatrywany specjalnym atrybutem `xsi:nil` (z przestrzeni nazw dla egzemplarzy dokumentu — `http://www.w3.org/2001/XMLSchema-instance`) o wartości `true`, co będzie odpowiadać wartości nieokreślonej

109

Wartości nieokreślone – przykład

Schemat

```
<xsd:element name="książka">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="autor" nillable="true">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="imię-i-nazwisko"/>
            <xsd:element name="data-urodzenia"/>
            <xsd:element name="data-śmierci"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="tytuł"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

110

Wartości nieokreślone – przykład

Użycie w dokumencie

```
<książka xsi:noNamespaceSchemaLocation="book.xsd"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <autor xsi:nil="true"/>
    <tytuł>Biblia</tytuł>
</książka>
```

Uwagi

- Element o wartości nieokreślonej musi mieć zawartość pustą
- Nieokreśloność jest **ważniejsza niż zdefiniowany model zawartości**
- Atrybuty elementu o wartości nieokreślonej muszą być w każdym wypadku zgodne z modelem

111

O warunkach nieokreślonych

- Użycie wartości nieokreślonych (ang. *nil values*)
  - Nie osłabia definicji typu poprzez dopuszczenie zawartości pustej
  - Pozwala na jednoznaczne określenie, że informacja nie istnieje
  - Umożliwia przekazanie informacji o nieokreśloności bez usuwania elementu z zawartości (obecność elementu może być wykorzystywana w aplikacji)
  - Umożliwia wyłączenie dodawania wartości domyślnych
- Nie da się ich użyć dla wartości typów nienapisowych, chyba że ...
  - wykonamy pewną sztuczkę

```
<xsd:simpleType>
  <xsd:union memberTypes="xsd:integer">
    <xsd:simpleType>
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value=""/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:union>
</xsd:simpleType>
```

112

Sposoby prezentacji wartości pustej

- Dla atrybutów wartość pustą można reprezentować na jeden sposób: `use="optional"`
- Dla elementów mamy więcej sposobów reprezentacji wartości pustej, np.
  - brak elementu
  - element pusty
  - element nieokreślony
- Dany element może więc mieć następujący model

```
<xsd:element name="pojawiam-się-i-znikam"
                                  minOccurs="0"
                                  nillable="true">
    <xsd:complexType/>
</xsd:element>
```

113

Sposoby prezentacji wartości pustej

- **Napis pusty** (o zerowej długości) można reprezentować elementem pustym lub brakiem elementu
- **Napis o nieokreślonej wartości** można reprezentować wartością `nil` lub brakiem elementu
- Jeśli jednak definiujemy strukturę, w której wystąpienie elementu ma znaczenie (np. dla jej budowy, rozmiaru), brak elementu okazuje się złym reprezentantem czegokolwiek

```
<tablica>
<napis>Ala ma żółwia</napis>
<napis xsi:nil="true"/>
<napis/>
<napis xsi:nil="true"/>
</tablica>
```

114

Dobre praktyki

115

Nazwy elementów, atrybutów, typów

- Nazwy powinny być
  - znaczące i proste do zapamiętania
  - jak najmniej magiczne i skrótowe (PTNM)
  - poręczne (adresZamawiającegoProjektBudowlany)
  - zapisane w spójny sposób (KodPocztowy, kod-pocztowy, kod_pocztowy, kod.pocztowy, kodPocztowy), z użyciem ustandaryzowanego słownictwa
  - dla typów i grup opatrzone odpowiednim prefiksem/sufiksem

`<produkt>/<numerProduktu>` czy jednak `<produkt>/<numer>`?

\+ Czytelniejsze znaczenie elementu
\+ Prostsze przetwarzanie (niezależność od rodzica, możliwość pobrania elementu wg nazwy)
− Zapis mimo wszystko nadmiarowy
− Ukrywa fakt reprezentacji podobnych własności w różnych elementach

116

Własności: nazwy konkretne czy ogólne?

Nazwy konkretne

```
<długość>60</długość>
<szerokość>50</szerokość>
<wysokość>52</wysokość>
<ciężar>25</ciężar>
```

\+ możliwość przypisywania typów danych
\+ możliwość określania wymagalności i liczby wystąpień
− konieczność zmiany schematu w przypadku dodania nowej własności

Nazwy ogólne

```
<cecha nazwa="długość">60</cecha>
<cecha nazwa="szerokość">50</cecha>
<cecha nazwa="wysokość">52</cecha>
<cecha nazwa="ciężar">25</cecha>
```

\+ niezmienność schematu przy dodaniu nowej własności
\+ łatwiejsze przetwarzanie (np. wyświetlenie listy wszystkich własności)
− brak możliwości definiowania typów, określania wymagalności i liczby wystąpień

117

Globalne czy lokalne deklaracje elementów?

Deklaracje globalne

```
<xsd:element name="osoba"/>
<xsd:element name="zwycięzcy">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="osoba" maxOccurs="10"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

\+ mogą być wielokrotnie wykorzystywane
− muszą mieć unikalne nazwy w całym schemacie (uwaga na części dołączane)
− jeśli określono docelową przestrzeń nazw, używamy nazw kwalifikowanych

Deklaracje lokalne

```
<xsd:element name="zwycięzcy">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="osoba" maxOccurs="10"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

\+ nie muszą być unikalne w całym schemacie
\+ mogą być kwalifikowane lub nie (co upraszcza dokument)
− nie mogą być walidowane niezależnie od elementu nadrzędnego

118

Typy nazwane czy anonimowe?

Typy nazwane

```
<xsd:complexType name="odldo10osób">
  <xsd:sequence>
    <xsd:element name="osoba" maxOccurs="10"/>
  </xsd:sequence>
</xsd:complexType>
```

\+ mogą być wielokrotnie wykorzystywane
\+ mogą być podstawą innych typów; można ich użyć w definicjach list i unii
\+ uczytelniają schemat zawierający złożone definicje

Typy anonimowe

```
<xsd:element name="zwycięzcy">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="osoba" maxOccurs="10"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

\+ jeśli definicja nie będzie powtórnie wykorzystywana, mogą uprościć utrzymanie schematu (bo nie trzeba martwic sie o ich wpływ na pozostałe komponenty)
\+ w prostych przypadkach mogą być czytelniejsze

119

Elementy grupujące

Bez elementu grupującego

```
<klient>
  <nazwa>MIMUW</nazwa>
  <ulica>Banacha 2</ulica>
  <kod>02-097</kod>
  <miasto>Warszawa</miasto>
</klient>
```

Z elementem grupującym

```
<klient>
  <nazwa>MIMUW</nazwa>
  <adres>
    <ulica>Banacha 2</ulica>
    <kod>02-097</kod>
    <miasto>Warszawa</miasto>
  </adres>
</klient>
```

\+ bardziej intuicyjne, łatwiejsze do wypełnienia przez człowieka
\+ łatwiejsze do przetwarzania, np. przez XSLT
− trochę nadmiarowe

Stachelska-Miemiejewicka

120

Konflikty nazw

- Zasady ogólne
  - Elementy i atrybuty mogą nazywać się tak samo
  - Typy mogą nazywać się tak samo jak elementy lub atrybuty
  - Typy nie mogą nazywać się tak samo jak inne typy
- Czy taka definicja jest poprawna?

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="tekst"/>
  <xsd:element name="tekst"/>
  <xsd:attribute name="tekst"/>
  <xsd:simpleType name="tekst">
    <xsd:list itemType="xsd:token"/>
  </xsd:simpleType>
</xsd:schema>
```

121

Listy wartości – wyzwania

- Potencjalnie częste zmiany, często poza kontrolą projektanta schematu (kody języków, walut, państw)
  - warto utrzymywać je w osobnych dokumentach schematu, co pozwala na ich wersjonowanie
- Długość list spowalnia walidacje, zaśmieca schemat, utrudnia zarządzanie
  - warto ograniczać listy do maksymalnie 15-20 wartości, dokumentować listy, używać wzorców
- Brak rozszerzalności typów prostych
  - warto użyć typu `xsd:token` lub ewentualnie jego unii z listą wyliczeniową, by inne dokumenty mogły zawęzić tę listę

```
<xsd:simpleType name="województwo">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="łódzkie"/>
    <xsd:enumeration value="wielkopolskie"/>
      ...
  </xsd:restriction>
</xsd:simpleType>
```

122

Rozszerzalne listy wartości – przykład

```
<xsd:simpleType name="językiKanady">
  <xsd:union memberTypes="xsd:token">
    <xsd:simpleType>
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="angielski"/>
        <xsd:enumeration value="francuski"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:union>
</xsd:simpleType>
```

```
<xsd:simpleType name="językiKanadyPoImigracji">
  <xsd:restriction base="językiKanady">
    <xsd:enumeration value="angielski"/>
    <xsd:enumeration value="polski"/>
  </xsd:restriction>
</xsd:simpleType>
```

123

Przestrzenie nazw

- Zalecenia ogólne
  - Warto umieszczać deklaracje przestrzeni nazw w elemencie głównym
  - Domyślne przestrzenie nazw sprawdzają się tylko wtedy, gdy jedna z nich dominuje w dokumencie
  - Prefiksy nie mają znaczenia, ale warto trzymać się konwencji (`xsd` lub `xs` dla schematów...)
- Trzy sposoby użycia przestrzeni nazw w schemacie
  - Przestrzeń docelowa jako domyślna (rozwiązanie najczęstsze)
  - Przestrzeń XML Schema jako domyślna
  - Bez domyślnej przestrzeni nazw

124

Przestrzeń docelowa jako domyślna

- Komponenty XML Schema zawsze prefiksowane
- Bezprefiksowy zapis odwołań do komponentów zdefiniowanych w schemacie (**oprócz ścieżek XPath** w składnikach elementów `<xsd:key>` i `<xsd:keyref>` – tam domyślna przestrzeń nazw nie obowiązuje)

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.pl/usos"
        xmlns="http://www.example.pl/usos">
  <xsd:complexType name="typOsoba">
    ...
  </xsd:complexType>
  <xsd:element name="student" type="typOsoba"/>
  <xsd:element name="data-rozp-studiów" type="xsd:date"/>
</xsd:schema>
```

125

Przestrzeń XML Schema jako domyślna

- Komponenty XML Schema nie są prefiksowane
- Prefiksowy zapis odwołań do komponentów zdefiniowanych w schemacie
- **UWAGA:** nie da się używać bez docelowej przestrzeni nazw – nie ma jak odwołać się do zdefiniowanych komponentów

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.pl/usos"
        xmlns:usos="http://www.example.pl/usos">
  <complexType name="typOsoba">
    ...
  </complexType>
  <element name="student" type="usos:typOsoba"/>
  <element name="data-rozp-studiów" type="date"/>
</schema>
```

126

Brak domyślnej przestrzeni nazw

- Wszystko z prefiksem (oczywiście oprócz wprowadzanych nazw)
- Najlepsze rozwiązanie dla przypadku współistnienia wielu przestrzeni nazw

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.pl/usos"
        xmlns:usos="http://www.example.pl/usos">
  <xsd:complexType name="typOsoba">
    ...
  </xsd:complexType>
  <xsd:element name="student" type="usos:typOsoba"/>
  <xsd:element name="data-rozp-studiów" type="xsd:date"/>
</xsd:schema>
```

127

Trzy sposoby dokumentacji schematu

- Za pomocą
  - Specjalnego elementu `<xsd:annotation>`
  - Komentarzy XML-owych
  - Atrybutów dołączanych (ang. *foreign attributes*)
- Oraz oczywiście wykorzystując sposoby nieschematocentryczne
  - Opisując schemat poza nim samym
  - Przechowując łącznie opisy danego komponentu schematu
  - Dostępne przekształcenia i style
  - ...

128

`<xsd:annotation>` - dokumentacja schematu

- Element `<xsd:annotation>` może wystąpić w dowolnym miejscu na poziomie globalnym oraz na początku wszystkich konstrukcji XML Schema
- Jego zawartość to mieszanka elementów `<xsd:documentation>` i `<xsd:appinfo>` zawierających dodatkowe informacje (tekst i znaczniki) odpowiednio dla ludzi i maszyn

```
<xsd:element name="nazwisko" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation xml:lang="pl">
       Nazwisko adresata.
    </xsd:documentation>
    <xsd:appinfo>
      <form:label>Podaj nazwisko adresata:</form:label>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>
```

129

Własne atrybuty dołączane

- Każdy komponent schematu może zawierać atrybuty pochodzące z dowolnej przestrzeni nazw
- Atrybuty te nie są walidowane i nie trzeba ich deklarować

```
<xsd:schema
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:doc="http://www.example.pl/xml">
  <xsd:element name="wykład" type="typPrezentacja"
      doc:opis="Element główny prezentacji
            na potrzeby wykładu."/>
</xsd:schema>
```