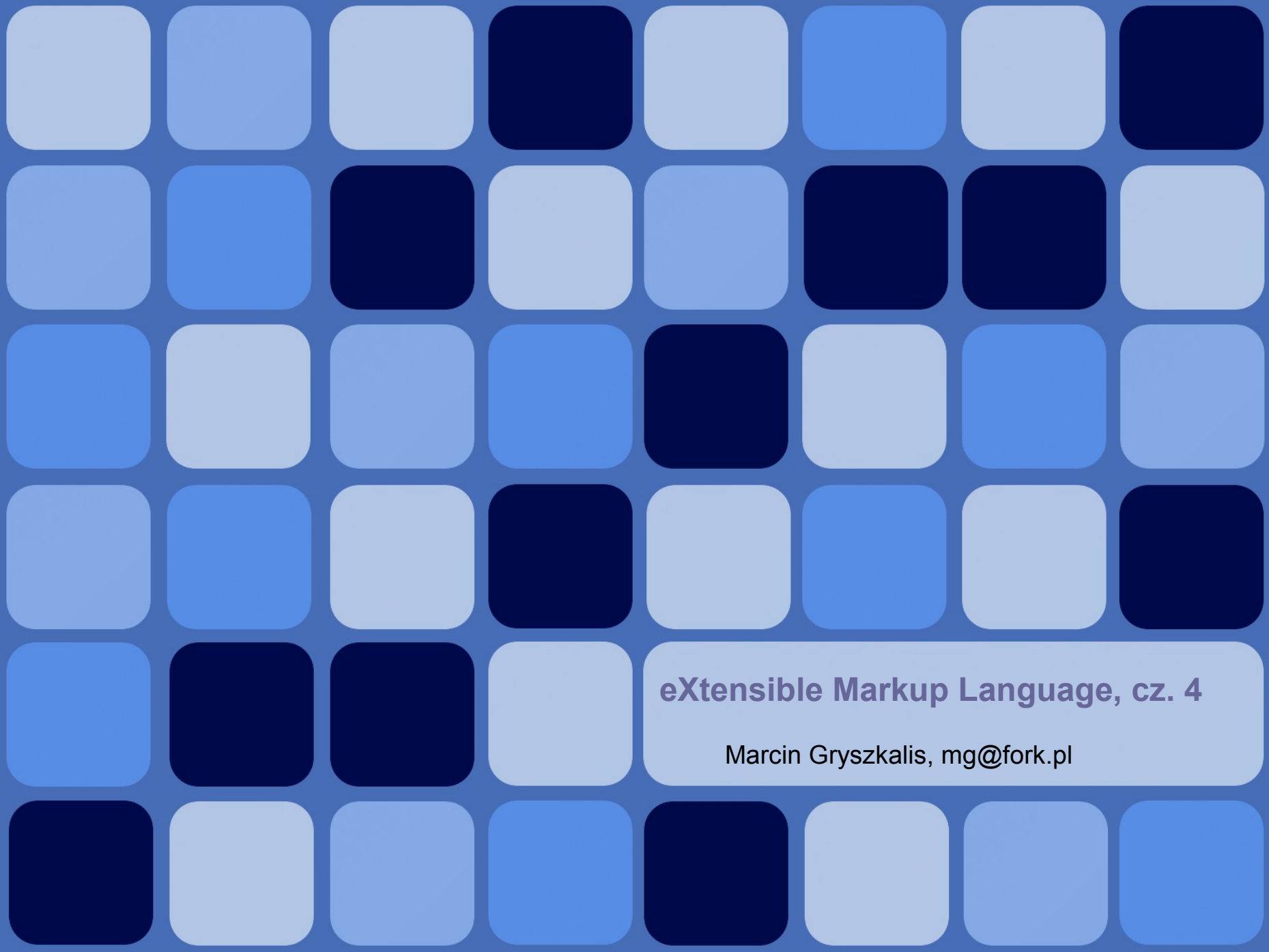


eXtensible Markup Language, cz. 4

Marcin Gryszkalis, mg@fork.pl

XML 5 – XML Schema

- Projektowany jako rozszerzenie i uzupełnienie DTD
- Jest w stanie odwzorować dowolną definicję DTD (ale nie odwrotnie!)
- Bardzo rozbudowana specyfikacja
- System typów (liczby, daty)
- Możliwość nakładania ograniczeń na wartości
- Możliwość tworzenia nowych typów
- Obsługa przestrzeni nazw
- i jeszcze więcej...
- od 2001 oficjalna rekomendacja W3C (XML Schema 1.0)
- standardowe rozszerzenie: .xsd (jak i prefix przestrzeni nazw)

XML 5 – XML Schema a DTD

- DTD
  - wywodzi się z SGML-a
  - specyficzna składnia
  - 10 typów danych
  - brak kontroli wartości
- XML Schema
  - zaprojektowany dla XML-a
  - składnia XML
  - 44 typy proste
  - zaawansowana kontrola zawartości elementów

XML 5 – Deklaracja schematu w pliku XML

- Bez przestrzeni nazw

```
<elGłówny
        xmlns:xsi=”http://www.w3c.org/2001/XMLSchema-instance”
        xsi:noNamespaceSchemaLocation=”schemat.xsd”
>
```

- Z przestrzenią nazw

```
<abc:elGłówny
        xmlns:abc=”http://abc.abc/abc” <!-- nasza przestrzeń nazw -->
        xmlns:xsi=”http://www.w3c.org/2001/XMLSchema-instance”
        xsi:schemaLocation=”schemat.xsd”
>
```

XML 5 – Korzeń pliku schematu

- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:abc="http://test/abc"
  targetNamespace="http://test/abc"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  >
- targetNamespace – docelowa przestrzeń nazw (dla której definiujemy schemat)
- element/attribute FormDefault – czy elementy/atrybuty mają być prefiksowane odpowiednią przestrzenią nazw

XML 5 – XML Schema – podstawowy przykład

```
<xsd:element name="osoba">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="imie" type="xsd:string"/>
      <xsd:element name="nazwisko"
                   type="xsd:string"/>
      <xsd:element name="plec" type="xsd:string"/>
      <xsd:element name="wiek" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID"/>
    <xsd:attribute name="NIP" type="NIPTyp"/>
  </xsd:complexType>
</xsd:element>
```

XML 5 – Kompozycja

- Obok deklaracji elementów i atrybutów, istnieje możliwość globalnego zdefiniowania
  - własnego typu prostego (zazwyczaj przez ograniczenia innego typu prostego), *simpleType*
  - własnego typu złożonego (składającego się z podelementów i atrybutów), *complexType*

XML 5 – Kompozycja

- Deklaracje atrybutów mogą występować w ramach definicji elementu lub definicji typu złożonego
- Deklaracje elementów mogą występować globalnie, w ramach definicji typu złożonego lub w ramach definicji innego elementu
- Definicje typów mogą wystąpić globalnie (z atrybutem *name*) albo wewnątrz deklaracji opisywanego elementu (anonimowo, celem jednorazowego wykorzystania)

XML 5 – Typy – podział

- w/g zasięgu definicji
  - nazwane
  - anonimowe
- w/g zawartości
  - proste
  - złożone o zawartości
    - prostej
    - elementowej
    - mieszanej
    - pustej
- w/g pochodzenia
  - wbudowane
  - zdefiniowane
    - rozszerzenia
    - ograniczenia
    - listy
    - unie

XML 5 – Typy nazwane i anonimowe

- Typy nazwane:

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

- Typy anonimowe:

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

Typy proste i złożone

- Typy proste:

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

- Typy złożone:

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

- Element może mieć typ prosty lub złożony.

XML 5 – Typy złożone – rodzaje zawartości

- Zawartość elementowa:

```
<osoba PESEL="12345678901">
  <imie>Jan</imie>
  <nazwisko>Kowalski</nazwisko>
</osoba>
```

- Zawartość prosta:

```
<masa jm="kg">10.55</masa>
```

- Zawartość mieszana:

```
<treść>Wypadek dnia <data>13.10.2001 r.</data>
o godzinie <godzina>13:13</godzina>
(<dzien-tygodnia>piątek</dzien-tygodnia>) miał  miejsce
nie z mojej winy. <poszkodowany>Alojzy
M.</poszkodowany></treść>
```

- Zawartość pusta:

```
<osoba PESEL="12345678901 />
```

XML 5 – Definiowanie zawartości elementów

- Grupy deklaracji elementów
  - *sequence* – sekwencja
  - *choice* – wybór
  - *all* - wszystko
- Zagnieżdżanie grup:

```
<xsd:complexType name="OsobaTyp">
  <xsd:sequence>
    <xsd:element name="imie" type="xsd:string"/>
    <xsd:element ref="nazwisko"/>
    <xsd:choice>
      <xsd:element name="nr-dowodu" type="DowódTyp"/>
      <xsd:element name="nr-paszportu"
                   type="PaszportTyp"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
```

XML 5 – Definiowanie zawartości elementów

- Grupa *all*
  - nie może zawierać innych grup (tylko deklaracje elementów i odwołania do elementów)
  - Każdy element może wystąpić co najwyżej raz
  - nie może zawierać się w innej grupie

XML 5 – definiowanie zawartości prostej

```
<xsd:complexType name="MasaTyp">
     <xsd:simpleContent>
       <xsd:extension base="xsd:decimal">
         <xsd:attribute name="jm" type="xsd:string"/>
       </xsd:extension>
     </xsd:simpleContent>
   </xsd:complexType>
```

XML 5 – definiowanie zawartości mieszanej

```
<xsd:complexType name="ZeznanieTyp" mixed="true">
   <xsd:sequence>
     <xsd:element name="data"
                  type="xsd:string"/>
     <xsd:element name="godzina"
                  type="xsd:string"/>
     <xsd:element name="dzien-tygodnia"
                  type="xsd:string"/>
     <xsd:element name="poszkodowany"
                  type="xsd:string"/>
   </xsd:sequence>
 </xsd:complexType>
```

XML 5 – definiowanie zawartości pustej

```
<xsd:complexType name="OsobaTyp">
  <xsd:attribute name="PESEL" type="PESELTyp"/>
</xsd:complexType>
```

XML 5 – Deklaracje globalne i lokalne

- Deklaracje lokalne:

```
<xsd:complexType name="OsobaTyp">
  <xsd:sequence>
    <xsd:element name="imie" type="xsd:string" />
    <xsd:element name="nazwisko" type="xsd:string" />
  </xsd:sequence>
  <xsd:attribute name="NIP" type="NIPTyp" />
</xsd:complexType>
```

- Deklaracje globalne:

```
<xsd:element name="imie" type="xsd:string"/>
<xsd:element name="nazwisko" type="xsd:string"/>
<xsd:attribute name="NIP" type="NIPTyp"/>
<xsd:complexType name="OsobaTyp">
  <xsd:sequence>
    <xsd:element ref="imie" />
    <xsd:element ref="nazwisko"/>
  </xsd:sequence>
  <xsd:attribute ref="NIP" />
</xsd:complexType>
```

XML 5 – kontrola użycia elementów i atrybutów

```
<xsd:element name="osoba">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="imie" type="xsd:string"
                   minOccurs="1" maxOccurs="2"/>
      <xsd:element name="nazwisko"
                   type="xsd:string"/>
      <xsd:element name="plec" type="xsd:string"/>
      <xsd:element name="wiek" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID"
                   use="required"/>
    <xsd:attribute name="NIP" type="NIPTyp"/>
  </xsd:complexType>
</xsd:element>
```

XML 5 – kontrola użycia elementów

- Ograniczenia na liczbę wystąpień podelementów
  - domyślnie 1
  - minOccurs – minimalna ilość wystąpień (>=0)
  - maxOccurs – maksymalna ilość wystąpień (>=1)
  - maxOccurs=”unbounded” - nieskończoność
  - nie można określić dla elementów globalnych (zawsze 1)
- Wartość domyślna
  - default=”qwerty” (użyta zostanie tylko dla elementu pustego, nie zostanie podstawiona gdy element w ogóle nie wystąpił)

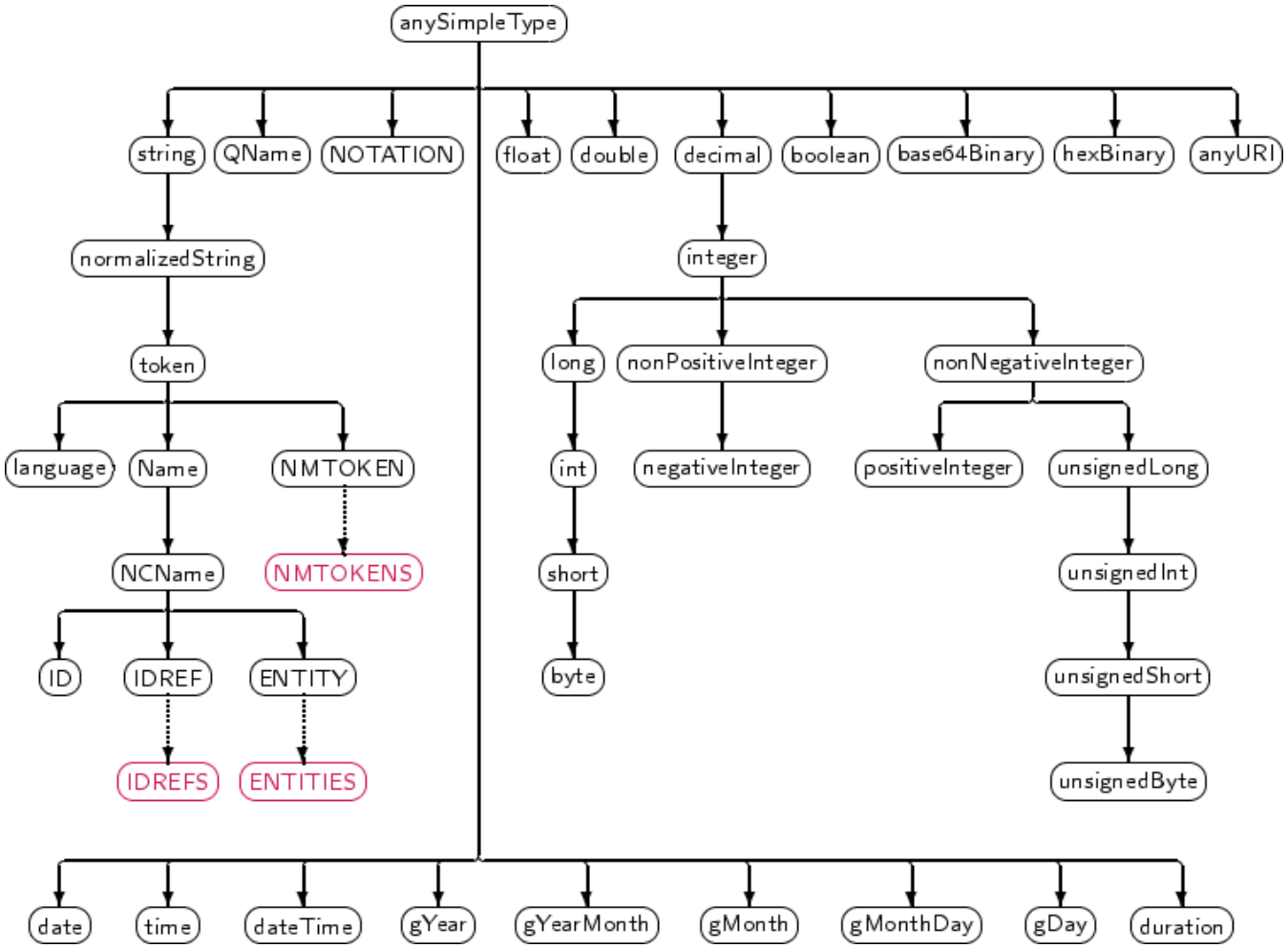
XML 5 – kontrola użycia atrybutów

- Atrybuty mogą być tylko typami prostymi
- Ograniczenia na występowanie atrybutu
  - use=”optional” - atrybut opcjonalny (wartość domyślna)
  - use=”required” - atrybut obowiązkowy
  - use=”prohibited” - atrybut niedopuszczalny
- Wartość domyślna
  - default=”xyz” (wymusza use=”optional”)
- Wartość stała
  - fixed=”qwerty”

XML 5 - Więzy integralności

- Więzy integralności (ang. *identity contraints*)
  - unique – wartość niepowtarzalna
  - key – wartość niepowtarzalna i niepusta (podobnie do ID)
  - keyref – odwołanie do wstkazanego klucza (podobnie do IDREF)
- Zalety w porównaniu do ID/IDREF
  - można tworzyć wiele niezależnych od siebie ograniczeń
  - odwołanie wskazuje konkretny klucz
  - więzy mogą obejmować kombinację wartości
  - więzy mogą być nałożone na zawartość elementów

XML 5 – Wbudowane typy proste



XML 5 – Wbudowane typy proste – podstawowe

- string – napis
- boolean – wartość logiczna (0, 1, false, true)
- decimal – liczba dziesiętna (20, -5, +12.3, -0001.300)
- float, double – liczba zmiennoprzecinkowa (1, -24.3e-3, 1E4, INF, NaN)
- date – data i (opcjonalnie) czas (2007-01-23, 2007-01-24Z)
- time – czas (13:20:01.123, 15:33:00+02:00)
- dateTime – data i czas (2007-01-23T10:31:22.123)
- duration – okres (P2Y6M1D12H5M59S)

XML 5 – Wbudowane typy proste - podstawowe

- base64Binary – dane binarne zapisane jako Base64 (dGVzdHRlc3RoZXNoCg==)
- hexBinary – dane binarne zapisane szesnastkowo (07F4DBA4)
- anyURI – URI (http://x.y.z)

XML 5 – Wbudowane typy proste - pochodne

- normalizedString – białe znaki zamieniane na spacje
- token – białe znaki zamieniane na 1 spację, usuwane z początku i końca
- QName – nazwa kwalifikowana, z dwukropkiem (abc:osoba)
- NCName – nazwa niekwalifikowana
- integer – liczba dziesiętna całkowita (1, -3)
- negativeInteger, positiveInteger, ... - ograniczone dziesiętne liczby całkowite
- long – liczba całkowita możliwa do zapisania na 64 bitach (odpowiednio – int – 32 bity, short – 16 bitów, byte – 8 bitów)

XML 5 – Wyprowadzanie typów

- Wyprowadzanie typów prostych
  - ograniczanie
  - tworzenie list
  - tworzenie unii
- Wyprowadzanie typów złożonych
  - ograniczanie
  - rozszerzanie
    - typów prostych
    - typów złożonych

XML 5 – Ograniczanie typów prostych

- Aspekty (ang. *facets*)
  - minExclusive, minInclusive – wartość minimalna (bez/z ogr.)
  - maxExclusive, maxInclusive – wartość maksymalna
  - length, minLength, maxLength – długość
  - totalDigits, fractionDigits – ilość cyfr (całości, cz. ułamkowej)
  - enumeration – wyliczenie dopuszczalnych wartości
  - pattern – wzorzec (wyrażenie regularne)
  - whiteSpace – białe znaki
- tylko pattern i enumeration mogą być użyte wielokrotnie w jednej definicji

XML 5 – Ograniczanie typów prostych – przykłady

```
<xsd:simpleType name="NumerLottoTyp">
   <xsd:restriction base="xsd:integer">
     <xsd:minInclusive value="1"/>
     <xsd:maxInclusive value="49"/>
   </xsd:restriction>
 </xsd:simpleType>

 <xsd:simpleType name="DokumentTyp">
   <xsd:restriction base="xsd:token">
     <xsd:enumeration value="dowód osobisty"/>
     <xsd:enumeration value="paszport"/>
   </xsd:restriction>
 </xsd:simpleType>

 <xsd:simpleType name="TemperaturaTyp">
   <xsd:restriction base="xsd:decimal">
     <xsd:fractionDigits value="1"/>
   </xsd:restriction>
 </xsd:simpleType>
```

XML 5 – Ograniczanie typów prostych

- Nie wolno naruszyć ograniczeń typu bazowego w typie pochodnym,
- np. byte ma wbudowane ograniczenia:
  - minInclusive = -128
  - maxInclusive = 127
- Nie można zatem napisać:
- ```
  <xsd:simpleType name="ExtendedByte">
    <xsd:restriction base="xsd:byte">
      <xsd:minInclusive value="-256"/>
      <xsd:maxInclusive value="255"/>
    </xsd:restriction>
  </xsd:simpleType>
  ```
  ```

XML 5 – Aspekt whiteSpace

- Dopuszczalne są trzy wartości
  - preserve – białe znaki pozostawiane bez zmian
  - replace – wszystkie białe znaki zastępowane spacją
  - collapse – po kolei:
    - każdy biały znak jest zastępowany spacją
    - każdy ciąg spacji zastępowany pojedynczą spacją
    - spacje na początku i na końcu są usuwane

XML 5 – Aspekt pattern

- Wyrażenia regularne zgodne z perlem (*Perl Compatible Regular Expressions, PCRE*)
- http://pl.wikipedia.org/wiki/Wyrażenia_regularne
- http://www.regular-expressions.info/
- http://perldoc.perl.org/perlre.html
- przykłady:
  - kod pocztowy: [0-9][0-9]-[0-9][0-9][0-9] lub \d\d-\d\d\d
  - pesel: \d{11}
  - NIP: (d{3}-\d{3}-\d{2}-\d{2}|d{3}-\d{2}-\d{2}-\d{3})
  - Rejestracja samochodowa: [A-Z]+[0-9]+[A-Z]*
  - ISBN: \d{9}(\d|[Xx])

XML 5 - Listy

- Listy wartości (w jednym elemencie podajemi kilka wartości)

```
<xsd:simpleType name="NumerLottoTyp">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="1"/>
    <xsd:maxInclusive value="49"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="NumeryLottoTyp">
  <xsd:list itemType="NumerLottoTyp"/>
</xsd:simpleType>

<xsd:simpleType name="KuponLottoTyp">
  <xsd:restriction base="NumeryLottoTyp">
    <xsd:length value="6"/>
  </xsd:restriction>
</xsd:simpleType>
```

XML 5 – Listy – wersja anonimowa

```
<xsd:simpleType name="KuponLottoTyp">
   <xsd:restriction>
     <xsd:simpleType>
       <xsd:list>
         <xsd:simpleType>
           <xsd:restriction base="xsd:integer">
             <xsd:minInclusive value="1"/>
             <xsd:maxInclusive value="49"/>
           </xsd:restriction>
         </xsd:simpleType>
       </xsd:list>
     </xsd:simpleType>
     <xsd:length value="6"/>
   </xsd:restriction>
 </xsd:simpleType>
```

XML 5 - Unie

```
<xsd:simpleType name="RozmiarLiczbowyTyp">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="2"/>
    <xsd:maxInclusive value="18"/>
  </xsd:restriction>
</xsd:simpleType>

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

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

XML 5 – Czego nie da się zamodelować w XSD?

- Brak kontekstowego sprawdzania poprawności, np.:
  - `cena-netto ≤ cena-brutto`
  - jeśli
    `sposób-transportu = powietrze`
    to
    `środek-transporu = samolot`
    lub
    `środek transportu = balon`
- Kontekstowe sprawdzanie
  - w aplikacji
  - w transformacjach XSLT
  - inny język schematów (np. Schematron)