

# XML Schema

Typy proste, wyprowadzanie typów, modularyzacja schematu

Patryk Czarnik

Instytut Informatyki UW

XML i nowoczesne technologie zarządzania treścią – 2007/08

# Plan

- 1 Typy proste
  - Typy wbudowane
  - Zawężanie typów prostych
  - Listy i unie
- 2 Wyprowadzanie typów złożonych
  - Rozszerzanie zawartości prostej
  - Rozszerzanie zawartości złożonej
  - Ograniczanie typów złożonych
- 3 Zarządzanie schematami
  - Podział schematu na moduły
  - Odwołania do schematów w egzemplarzu

# Typy proste i złożone

## Typy złożone

- struktura (podelementy, atrybuty),
- odpowiednie dla elementów.

## Typy proste

- zawartość tekstowa,
- odpowiednie dla atrybutów i elementów,
- wiele typów predefiniowanych,
- możliwość definiowania własnych typów.

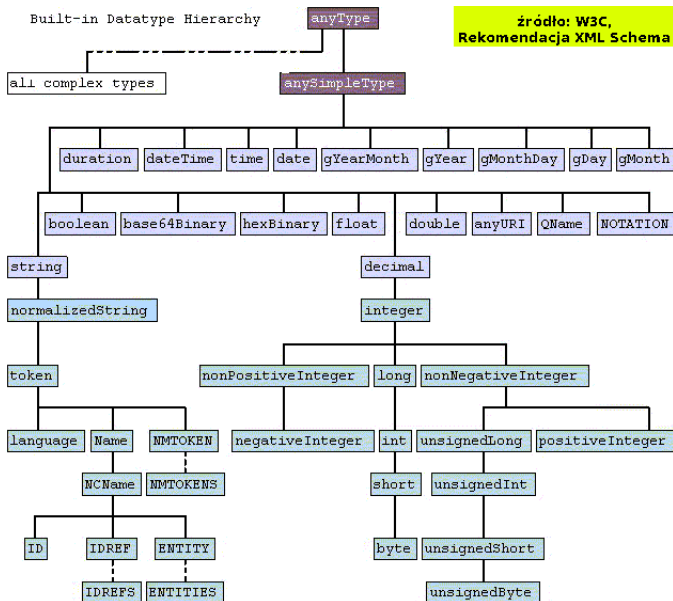
# Typy proste – teoria

- Typ prosty posiada:
  - **przestrzeń wartości** (*value space*) – zbiór wartości w warstwie pojęciowej (liczby, daty itp.),
  - **przestrzeń leksykalną** (*lexical space*) – zbiór napisów reprezentujących wartości w dokumencie,
  - **aspekty** (*facets*) – cechy opisujące typ:
    - *fundamental facets* – podstawowe własności semantyczne jak relacje równości, porządku itp.,
    - *constraining facets* – pozwalają na ograniczanie typów.

# Typy proste – sposoby definiowania

- Wbudowane:
  - prymitywne,
  - pochodne (*derived*) – zdefiniowane z typów prymitywnych.
- Definiowane przez użytkownika jako:
  - zawężenie,
  - lista,
  - unia.

# Typy wbudowane XML Schema



# Wybrane typy proste

typ	przykładowe wartości	
decimal	12.3	+000012.300
float, double	+24.3e-3 NaN	12 -INF
QName	os:osoba	osoba
date	1968-04-02 1968-04-02Z	1968-04-02-05:00 -0045-02-02
time	13:20:00.887	13:30:00-05:00
dateTime	1968-04-02T13:20:00.887	
gYearMonth	1968-04	
gMonthDay	--04-02	
duration	P2Y6M5DT12H35M30S	

# Aspekty

- Aspekt zawężający (*constraining facet*) – ze względu na niego można zawężyć typ prosty:
  - minExclusive, minInclusive,
  - maxExclusive, maxInclusive,
  - length, minLength, maxLength,
  - totalDigits, fractionDigits,
  - enumeration,
  - pattern,
  - whiteSpace.
- Różne (ale nie rozłączne) zbiory aspektów dla różnych typów prymitywnych.
- Tylko pattern i enumeration można użyć wielokrotnie w jednej definicji typu.

# Przykłady zawężania typów prostych

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

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

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

## O zawężaniu typów prostych

- Przestrzeń wartości typu wyprowadzonego musi być podzbiorem przestrzeni wartości typu bazowego.
- Typy wbudowane mają określone niektóre aspekty, np.: typ `byte`:
  - `minInclusive`: -128,
  - `maxInclusive`: 127.

### Niepoprawne wyprowadzenie

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

# Aspekt pattern

- Wyrażenie regularne, z którym zgodne muszą być wartości:
  - typowa składnia,
  - kategorie i podkategorie znaków, np. litery (małe, duże...), cyfry (dziesiętne, inne), znaki interpunkcyjne, białe znaki itd.
- Wiele wyrażień – alternatywa.

## Przykład

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

## Aspekt whiteSpace

- Może wpływać na przetwarzanie.
- `preserve` – wszystkie białe znaki są pozostawiane bez zmian,
- `replace` – każdy biały znak jest podczas przetwarzania zastępowany przez spację,
- `collapse` – każdy biały znak jest podczas przetwarzania zastępowany przez spację, a następnie każdy ciąg spacji jest zastępowany przez jedną spację, zaś spacje na początku i na końcu są usuwane.

# Listy i unie

- **Typ atomowy** – jego wartości są „niepodzielne”, w praktyce typ prymitywny lub jego ograniczenie.
- **Lista** – lista wartości atomowych (w dokumencie rozdzielonych spacjami).
- **Unia** – suma zbiorów wartości poszczególnych typów.

# Tworzenie list

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

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

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

# Tworzenie list

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

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

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

# Tworzenie list – inaczej

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

# Tworzenie unii

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

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

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

# Tworzenie unii – inaczej

```
<xs:simpleType name="RozmiarTyp">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:integer">
        <xs:minInclusive value="2"/>
        <xs:maxInclusive value="18"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="S"/>
        <xs:enumeration value="M"/>
        <xs:enumeration value="L"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```

# Plan

- 1 Typy proste
  - Typy wbudowane
  - Zawężanie typów prostych
  - Listy i unie
- 2 Wyprowadzanie typów złożonych
  - Rozszerzanie zawartości prostej
  - Rozszerzanie zawartości złożonej
  - Ograniczanie typów złożonych
- 3 Zarządzanie schematami
  - Podział schematu na moduły
  - Odwołania do schematów w egzemplarzu

# Wyprowadzanie typów złożonych

- **Zawężanie** – ograniczanie zbioru wartości;  
wartość typu bazowego poprawną wartością typu pochodnego:
  - ograniczanie zawartości prostej,
  - ograniczanie zawartości złożonej,
  - ograniczanie opcjonalności atrybutów.
- **Rozszerzanie** – rozszerzanie struktury; wartość typu bazowego niekoniecznie poprawną wartością typu pochodnego:
  - rozszerzanie zawartości prostej (dodawanie atrybutów do typu prostego lub typu złożonego o zawartości prostej),
  - rozszerzanie zawartości złożonej (dodawanie elementów i/lub atrybutów, dodatkowe elementy zawsze na końcu, po elementach zadeklarowanych w typie bazowym).

# Wyprowadzanie typów złożonych

- **Zawężanie** – ograniczanie zbioru wartości;  
wartość typu bazowego poprawną wartością typu pochodnego:
  - ograniczanie zawartości prostej,
  - ograniczanie zawartości złożonej,
  - ograniczanie opcjonalności atrybutów.
- **Rozszerzanie** – rozszerzanie struktury; wartość typu bazowego niekoniecznie poprawną wartością typu pochodnego:
  - rozszerzanie zawartości prostej (dodawanie atrybutów do typu prostego lub typu złożonego o zawartości prostej),
  - rozszerzanie zawartości złożonej (dodawanie elementów i/lub atrybutów, dodatkowe elementy zawsze na końcu, po elementach zadeklarowanych w typie bazowym).

# Rozszerzanie zawartości prostej

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

```
<xs:complexType name="MasaNiedokładnaTyp">
  <xs:simpleContent>
    <xs:extension base=" MasaTyp">
      <xs:attribute name="dokładność"
        type="xs:decimal"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

# Rozszerzanie zawartości złożonej

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

```
<xs:complexType name="OsobaZDokumentemTyp">
  <xs:complexContent>
    <xs:extension base=" OsobaTyp">
      <xs:choice>
        <xs:element name="dowód-os"
          type="DowódTyp"/>
        <xs:element name="paszport"
          type="PaszportTyp"/>
      </xs:choice>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

# Ograniczanie typów złożonych

- Ograniczenie zawartości prostej (tak jak typu prostego).
- 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, np.:
  - ściślejsze ograniczenia liczebności (*minOccurs*, *maxOccurs*),
  - usunięcie elementów opcjonalnych w grupach *sequence* i *all*,
  - wybranie podzbioru elementów w grupie *choice*,
  - ograniczenie typu poszczególnych podelementów.

# Ograniczanie zawartości prostej i atrybutów

```
<xs:complexType name=" MasaNiedokladnaTyp">
  <xs:simpleContent>
    <xs:extension base="xs:decimal">
      <xs:attribute name="jm" type="xs:string"/>
      <xs:attribute name="dokładność"
        type="xs:decimal"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="MasaDoUniesieniaTyp">
  <xs:simpleContent>
    <xs:restriction base=" MasaNiedokladnaTyp">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="50"/>
      <xs:attribute name="dokładność" use="prohibited"/>
      <xs:attribute name="jm" fixed="kg"/>
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>
```

# Ograniczanie zawartości złożonej

```
<xs:complexType name=" BazowyTyp">
  <xs:sequence>
    <xs:element name="a" type="xs:string"
      minOccurs="3" maxOccurs="7"/>
    <xs:element name="b" type="xs:string"
      minOccurs="0"/>
    <xs:element name="c" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="OgraniczonyTyp">
  <xs:complexContent>
    <xs:restriction base=" BazowyTyp">
      <xs:sequence>
        <xs:element name="a" type="xs:string"
          minOccurs="4" maxOccurs="6"/>
        <xs:element name="c" type="xs:string"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

# Plan

- 1 Typy proste
  - Typy wbudowane
  - Zawężanie typów prostych
  - Listy i unie
- 2 Wyprowadzanie typów złożonych
  - Rozszerzanie zawartości prostej
  - Rozszerzanie zawartości złożonej
  - Ograniczanie typów złożonych
- 3 Zarządzanie schematami
  - Podział schematu na moduły
  - Odwołania do schematów w egzemplarzu

# Przestrzenie nazw i schematy

- **Schemat** zapisuje się przy pomocy jednego lub więcej **dokumentów schematów**.
- Schemat deklaruje nazwy należące do zero lub więcej przestrzeni nazw.
- Dokument schematu deklaruje nazwy należące do zero lub jednej przestrzeni nazw.
- Przestrzeń nazw zawiera nazwy zadeklarowane w zero lub więcej schematach.

# Dokument schematu

```
< xs:schema
  xmlns:xsd=" http://www.w3.org/2001/XMLSchema"
  xmlns=" http://szz.mimuw.edu.pl/osoby"
  targetNamespace=" http://szz.mimuw.edu.pl/osoby">

  < xs:complexType name=" OsobaTyp">
    ...
  </ xs:complexType>

  < xs:element name=" osoba" type=" OsobaTyp"/>
  < xs:element name=" numer" type=" xs:integer"/>
  ...
</ xs:schema>
```

# Dokument schematu – inaczej

```
< schema
  xmlns=" http://www.w3.org/2001/XMLSchema"
  xmlns:os=" http://szz.mimuw.edu.pl/osoby"
  targetNamespace=" http://szz.mimuw.edu.pl/osoby">

  < complexType name=" OsobaTyp">
    ...
  </ complexType>

  < element name=" osoba" type=" os:OsobaTyp"/>
  < element name=" numer" type=" integer"/>
  ...
</schema>
```

# Łączenie dokumentów schematów

- Schemat zapisuje się przy pomocy jednego lub więcej dokumentów schematów.
- Metody budowania schematu z dokumentów schematów:
  - instrukcje `include`, `import` i `redefine` w schemacie,
  - lokalizacje (wielu) dokumentów schematów określone w egzemplarzu,
  - predefiniowane (skonfigurowane) w procesorze lokalizacje,
  - lokalizacje dokumentów schematów przekazywane jako parametry aplikacji.

# 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.

## Przykład

```
<xs:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://szz.mimuw.edu.pl/osoby"
  targetNamespace="http://szz.mimuw.edu.pl/osoby">
  <xs:include schemaLocation="inst.xsd"/>
  ...
</xs:schema>
```

# redefine

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

## Przykład

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

# import

- Odwołanie do komponentów w innej przestrzeni nazw, zadeklarowanych w innym dokumencie schematu.

## Przykład

```
<xs:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://szz.mimuw.edu.pl/osoby"
  xmlns:inst="http://szz.mimuw.edu.pl/instytucje"
  targetNamespace="http://szz.mimuw.edu.pl/osoby">

  <xs:import schemaLocation="inst.xsd"
    namespace="http://szz.mimuw.edu.pl/instytucje"/>
  ...
</xs:schema>
```

# Egzemplarz

- **Przestrzeń nazw egzemplarzy XML Schema:**  
`http://www.w3.org/2001/XMLSchema-instance`
- **Atrybuty:**
  - `nil`,
  - `type`,
  - `schemaLocation`,
  - `noNamespaceSchemaLocation`.

# schemaLocation

```
<osoba
  xmlns="http://szz.mimuw.edu.pl/osoby"
  xmlns:inst="http://szz.mimuw.edu.pl/instytucje"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://szz.mimuw.edu.pl/osoby      osoby.xsd
                     http://szz.mimuw.edu.pl/instytucje inst.xsd">
  <imie>Jan</imie><nazwisko>Kowalski</nazwisko>
  <pracuje-w>
    <inst:firma>
      <inst:nazwa>Business Consulting</inst:nazwa>
      <inst:NIP>987-654-32-10</inst:NIP>
    </inst:firma>
  </pracuje-w>
</osoba>
```

# noNamespaceSchemaLocation

```
<osoba
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="osoby.xsd">
  <imie>Jan</imie>
  <nazwisko>Kowalski</nazwisko>
</osoba>
```

# Wartości puste w egzemplarzach

- Sposoby oznaczenia braku wartości:
  - brak elementu/atributu,
  - element/atribut występuje, ale jest pusty,
  - element/atribut występuje i zawiera specjalną wartość, np. N/A,
  - element ma wartość `nil`.

```
<osoba
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <imie>Jan</imie>
  <drugie-imie xsi:nil="true"/>
  <nazwisko>Kowalski</nazwisko>
</osoba>
```

- Dopuszczenie wartości `nil` w deklaracji elementu:

```
<xs:element name="drugie-imie" type="xs:string"
  nillable="true"/>
```