

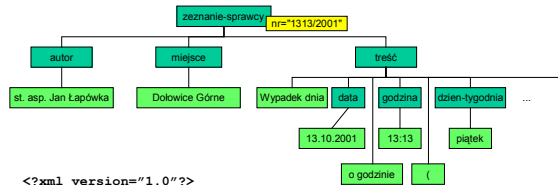
Definiowanie typów dokumentów Część 1. DTD, XML Schema

Jak wygląda XML?

```
<?xml version="1.0"?>
<zeznanie-sprawcy nr="1313/2001">
  <autor>st. asp. Jan Łapówka</autor>
  <miejsce>Dołowice Górne</miejsce>
  <treść>Wypadek dnia
  <data>13.10.2001r</data>
  o godzinie <godzina>13:13</godzina>
  (<dzien-tygodnia>piątek
  </dzien-tygodnia>) miał miejsce nie
  z mojej winy. <poszkodowany>Łojzy
  M.</poszkodowany> nie miał żadnego
  pomysłu w którą stronę uciekać, więc
  go przejechałem.</treść>
</zeznanie-sprawcy>
```

Deklaracja XML
Element główny
Atrybut
Element
Znacznik początkowy
Znacznik końcowy
Zawartość tekstowa

Struktura logiczna dokumentu XML



```
<?xml version="1.0"?>
<zeznanie-sprawcy nr="1313/2001">
  <autor>st. asp. Jan Łapówka</autor>
  <miejsce>Dołowice Górne</miejsce>
  <treść>Wypadek dnia <data>13.10.2001r</data>
  o godzinie <godzina>13:13</godzina> (<dzien-tygodnia>piątek
  </dzien-tygodnia>) miał miejsce nie z mojej winy.
  <poszkodowany>Łojzy M.</poszkodowany> nie miał żadnego
  pomysłu w którą stronę uciekać, więc go przejechałem.</treść>
</zeznanie-sprawcy>
```

Składnia XML

- Deklaracja XML:


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```
- Znaczniki:


```
<tag atrybutname="attribute value">
</tag>
```
- Znaczniki elementu pustego:


```
<br></br>
<br/>
```
- Komentarz:


```
<!-- komentarz -->
```
- Instrukcja przetwarzania:


```
<?target processing-instruction-body?>
```
- Sekcja CDATA:


```
<![CDATA[dowolny <tekst " nieprzetwarzany & przez [
parser]]>
```

Encje predefiniowane

&	&
<	<
>	>
'	'
"	"

Unicode

- Światowy standard kodowania znaków.
- Odmiany:
 - UTF-7 (korzysta tylko z 7-bitowego zestawu znaków ASCII),
 - UTF-8 (pierwsze 128 – ASCII, znak zajmuje od 1 do 6 bajtów),
 - UTF-16 (każdy znak to jedno lub dwa słowa 16-bitowe),
 - UTF-32 (każdy znak to jedno słowo 32-bitowe),
 - UCS-4 (znak zajmuje zawsze 4 bajty).
- Obowiązkowy standard dla dokumentów XML:
 - każde narzędzie XML-owe musi wspierać przynajmniej UTF-8.
- Sposób użycia w dokumentach XML:
 - jako odpowiedni ciąg bajtów,
 - &#kod; - kod dziesiętny znaku,
 - &#xkod; - kod szesnastkowy znaku.

DTD – XML Schema

- Wywodzi się z SGML-a
- Specyficzna składnia
- 10 typów danych
- Brak kontroli tekstowej zawartości elementów
- Typowy mieszany model zawartości
- Zaprojektowany na potrzeby XML-a
- Składnia XML
- 44 wbudowane typy proste
- Zaawansowana kontrola tekstowej zawartości elementów
- Możliwość definiowania własnych typów danych

Status XML Schema

- 15 lutego 1999: Dokument W3C opisujący wymagania stawiane przed nowym formatem:
 - mechanizmy tworzenia struktury,
 - typy proste,
 - reguły przetwarzania.
- 2 maja 2001: XML Schema staje się oficjalną rekomendacją W3C:
 - XML Schema Part 0: Primer,
 - XML Schema Part 1: Structures,
 - XML Schema Part 2: Datatypes.
- Obecnie trwają prace nad wersją 1.1 XML Schema (status: *working draft*).
- Przestrzeń nazw XML Schema:
<http://www.w3.org/2001/XMLSchema>

Deklarowanie elementów i atrybutów

```
<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>
```

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>
```

Typy

- Typy wg zasięgu definicji:
 - typy nazwane,
 - typy anonimowe.
- Typy wg zawartości:
 - typy proste,
 - typy złożone o zawartości:
 - prostej,
 - elementowej,
 - mieszanej,
 - pustej.
- Typy wg pochodzenia:
 - typy wbudowane,
 - typy zdefiniowane w schemacie:
 - rozszerzenia innych typów,
 - ograniczenia innych typów,
 - listy i unie.

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.
- Atrybut może mieć wyłącznie typ prosty.

Typy złożone – typy zawartości

- Zawartość elementowa:

```
<osoba PESEL="12345678901">  
  <imie>Jan</imie>  
  <nazwisko>Kowalski</nazwisko>  
</osoba>
```
- Zawartość prosta:

```
<masa jm="kg">10.55</masa>
```
- Zawartość mieszana:

```
<tresc>Wypadek dnia <data>13.10.2001 r.</data>  
o godzinie <godzina>13:13</godzina>  
(<dzien-tygodnia>piatek</dzien-tygodnia>) miał  
miejsce nie z mojej winy. <poszkodowany>Alojzy  
M.</poszkodowany> nie miał żadnego pomysłu w którą  
stronę uciekać, więc go przejechałem.</tresc>
```
- Zawartość pusta:

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

Definiowanie zawartości elementowej

- Grupy deklaracji elementów:
 - sequence,
 - choice,
 - all.
- 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>
```

Grupa all – ograniczenia

- Nie może zawierać innych grup (tylko deklaracje elementów i odwołania do elementów).
- Każdy element może wystąpić co najwyżej raz.
- Grupa all nie może być zagnieżdżona w innej grupie.
- Zagnieżdżanie grup:

```
<xsd:complexType name="OsobaTyp">  
  <xsd:all>  
    <xsd:element name="imie" type="xsd:string"/>  
    <xsd:element name="drugie-imie" type="xsd:string"  
      minOccurs="0"/>  
    <xsd:element ref="nazwisko"/>  
  </xsd:all>  
</xsd:complexType>
```

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>
```

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>
```

Definiowanie zawartości pustej

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

Gdzie szukać dalej

- *DTD Tutorial*
📄 www.xmlfiles.com/dtd
- Megginson, D., *Structuring XML Documents*, Prentice Hall, 1998
- *W3C Architecture Domain: XML Schema*
📄 www.w3.org/XML/Schema
- Costello, R., *XML Schema Tutorial*
📄 www.xfront.com/xml-schema.html
- Costello, R., *XML Schemas: Best Practices*
📄 www.xfront.com/BestPracticesHomepage.html
- Walmsley, P., *Definitive XML Schema*, Prentice Hall PTR, 2002

