



16 października 2003

Definiowanie typów dokumentów

Część 2: XML Schema





### Dlaczego DTD nie wystarcza?

**Zastosowania w integracji aplikacji – struktury danych:**

- przeniesienie zadania sprawdzania poprawności z tworzonej aplikacji na narzędzie walidujące daje spore oszczędności.
- 60% tworzonoego kodu dotyczy weryfikacji poprawności danych.  
Roger L. Costello, XML Schema Tutorial


**Cechy DTD:**

- jedynie podstawowa kontrola nad strukturą dokumentów,
- bardzo ogólne metody definiowania częstości wystąpień,
- mało "obiektywne", nierozszerzalne modele struktury.



### DTD – XML Schema

|   |   |
|---|---|
| <ul style="list-style-type: none"> <li>• Wywodzi się z SGML-a</li> <li>• Specyficzna składnia</li> <li>• 10 typów danych</li> <li>• Brak kontroli tekstowej zawartości elementów</li> <li>• Typowy mieszany model zawartości</li> </ul> | <ul style="list-style-type: none"> <li>• Zaprojektowany na potrzeby XML-a</li> <li>• Składnia XML</li> <li>• 41+ typów danych</li> <li>• Zaawansowana kontrola tekstowej zawartości elementów</li> <li>• Możliwość definiowania własnych typów danych.</li> </ul> |
|---|---|



### 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 wymaganiami do wersji 1.1 XML Schema.**  
**Przestrzeń nazw XML Schema: <http://www.w3.org/2001/XMLSchema>**




### Definiowanie 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="xsd:string"/>
  </xsd:complexType>
</xsd:element>

```



### Określanie typu elementu/attributu

**Atrybut type:**

```

<xsd:element name="imie" type="xsd:string"/>


```

**Podelement complexType lub simpleType:**

```

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

```



## Typy proste

### Wbudowane typy proste:

- string,
- boolean,
- integer,
- float,
- dateTime,
- ID, IDREF, CDATA,
- ...



## Typy proste

### Tworzenie własnych typów prostych przy pomocy aspektów (facets):

- minInclusive, maxInclusive, minExclusive, maxExclusive,
- pattern,
- enumeration,
- list,
- union,
- length, minLength, maxLength.

```
<xsd:element name="wiek">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="120"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```



## Przykłady

### Lista wartości:

```
<xsd:simpleType name="LotteryNumbers">
  <xsd:restriction>
    <xsd:simpleType>
      <xsd:list itemType="OneToNinetyNine">
      </xsd:simpleType>
    <xsd:length value="6"/>
  </xsd:restriction>
</xsd:simpleType>
```

### Wyrażenia regularne:

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



## Typy złożone

### Możliwość definiowania typów złożonych:

- sequence,
- choice,
- group,
- all.

### Kontrola użycia podelementów:

- minOccurs,
- maxOccurs.

### Kontrola użycia atrybutów:

- atrybut use o dopuszczalnych wartościach: required, optional lub prohibited.



## 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="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```



## Typ złożony, ale prosty

```
<xsd:element name="roślina">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="gatunek"
          type="xsd:string"
          use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```



## Model mieszany w XML Schema

### Możliwość kontroli ilości i kolejności podelementów:

```
<xsd:element name="zeznanie">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="poszkodowany"
        type="xsd:string"/>
      <xsd:element name="data"
        type="xsd:string"/>
      <xsd:element name="godzina"
        type="xsd:string"
        maxOccurs="2"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```



## Typ zawartości jako samodzielny byt

### Oddzielenie deklaracji elementu od typu zawartości:

```
■ typy anonimowe:
<xsd:element name="osoby">
  <xsd:complexType>...</xsd:complexType>
</xsd:element>

■ typy nazwane:
<xsd:element name="osoba" type="osoba"/>
...
<xsd:complexType name="osoba">
  ...
</xsd:complexType>
```



## Inne możliwości XML Schema

### Dziedziczenie typów:

- rozszerzenie typu bazowego,
- zawężenie typu bazowego,
- typy abstrakcyjne.

### Alternatywne nazwy elementów:

- możliwość zamiennego użycia w dokumentach.

### Zaawansowana kontrola referencji: key – keyref:

- dowolna ilość kluczy:
  - definiowane przy pomocy ścieżek XPath,
  - wartości w ramach klucza muszą być unikatowe;
- referencja odwołuje się do konkretnego klucza:
  - wartości referencji musi odpowiadać wartość klucza.



## Rozszerzanie typów

```
<xsd:complexType name="Publikacja">
  <xsd:sequence>
    <xsd:element name="Tytuł" type="xsd:string"/>
    <xsd:element name="Autor" type="xsd:string"/>
    <xsd:element name="RokPubl" type="xsd:year"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Książka">
  <xsd:complexContent>
    <xsd:extension base="Publikacja">
      <xsd:sequence>
        <xsd:element name="ISBN" type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```



## Czego nie można zamodelować w XML Schema?

### Brak walidacji kontekstowej, np.:

- zawartość elementu `cena-netto` jest mniejsza lub równa od zawartości elementu `cena-brutto`,
- jeżeli wartość atrybutu `sposób-transportu` jest `powietrze`, to element `środek-transportu` ma zawartość `samolot` lub `balon`.

### Metody walidacji kontekstowej:

- zaprogramowana w kodzie aplikacji,
- transformacja XSLT,
- Schematron:
  - definiowanie własności, jakie ma spełniać dokument,
  - komunikaty o błędach poprawności.



## RELAX NG

### REGular LAnguage description for XML – New Generation:

- składnia XML-owa, "bliższa opisowi struktury w języku naturalnym",
- wspiera typy danych (np. XML Schema Datatypes),
- atrybuty opisywane (prawie) tak samo, jak elementy,
- prosta technika modularyzacji: `define / ref`,
- model przetwarzania oparty na wyrażeniach regularnych.

### RELAX NG a inne języki:

- dostępne konstrukcje z SGML DTD, usunięte w XML DTD:
  - elementy wymagane, ale bez określonego porządku,
  - model mieszany – więcej możliwości;
- pozwala opisać niedostępne w XML Schema:
  - niedeterministyczne modele zawartości,
  - modele zawartości wrażliwe na kontekst.



## Przykład

### DTD:

```
<!DOCTYPE addressBook [
  <!ELEMENT addressBook (card*)>
  <!ELEMENT card (name, email)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT email (#PCDATA)>
]>
```

### RELAX NG:

```
<element name="addressBook"
  xmlns="http://relaxng.org/ns/structure/1.0">
  <zeroOrMore>
    <element name="card">
      <element name="name"> <text/> </element>
      <element name="email"> <text/> </element>
    </element>
  </zeroOrMore>
</element>
```

Źródło: RELAX NG Tutorial, <http://www.relaxng.org/tutorial-20011203.html>



## Przykład – niedeterminizm

### Konstrukcja zabroniona w XML Schema:

```
<element name="name">
  <choice>
    <text/>
    <group>
      <element name="firstName"><data type="token"/></element>
      <optional>
        <element name="middleName"><data type="token"/></element>
      </optional>
      <element name="lastName"><data type="token"/></element>
    </group>
  </choice>
</name>
```

Źródło: RELAX NG Tutorial, <http://www.relaxng.org/tutorial-20011203.html>



## Examplotron

### Definiowanie schematu "przez przykład":

- instancja dokumentu definiuje schemat,
- konwencje, np.:
  - powtórzenie elementu oznacza dowolną krotność,
  - przykładowa zawartość elementu definiuje typ.

### Ograniczenia:

- "przez przykład" nie można wyrazić konstrukcji abstrakcyjnych,
- dodatkowa, specyficzna składnia pozwala na dokładniejszą kontrolę struktury.

### Status:

- projekt na wczesnym etapie rozwoju (wersja 0.7),
- dostępne narzędzie: compile.xsl – przekształca schematy Examplotronowe do RELAX NG.



## Examplotron – przykład

```
<?xml version="1.0" encoding="UTF-8"?>
<order
  date="2003-02-01"
  eg:content="eg:interleave"
  xmlns:eg="http://examplotron.org/0/">
  <eg:attribute name="no">1234</eg:attribute>
  <quantity>1</quantity>
  <ref>AZERTY</ref>
  <ref>ZXCVBN</ref>
  <item>Tee shirt</item>
  <price unit="USD">10.</price>
</order>
```

atrybut opcjonalny  
typu data

element powtarzalny

atrybut wymagany  
typu liczba

dowolna kolejność  
podelementów



## Gdzie szukać dalej

### W3C Architecture Domain: XML Schema

[www.w3.org/XML/Schema](http://www.w3.org/XML/Schema)

### Roger Costello, XML Schema Tutorial

[www.xfront.com/xml-schema.html](http://www.xfront.com/xml-schema.html)

### RELAX NG Home Page

[www.relaxng.org](http://www.relaxng.org)

### Examplotron

[examplotron.org](http://examplotron.org)

### Eric van der Vlist, Comparing XML Schema Languages

[www.xml.com/pub/a/2001/12/12/schemacompare.html](http://www.xml.com/pub/a/2001/12/12/schemacompare.html)

### Eric van der Vlist, Relax NG Compared

[www.xml.com/pub/a/2002/01/23/relaxng.html](http://www.xml.com/pub/a/2002/01/23/relaxng.html)

### Patryk Czarnik, DTD, XML Schema – i co dalej?

Software 2.0, nr 6/2003, Wydawnictwo Software

