Wstępniak do XML (oraz co nieco o DTD i XML Schema)

Dawid Weiss

Institute of Computing Science
Poznań University of Technology

June 7, 2004

Presentation outline

1. XML — znaczenie języka, elementy
2. Walidacja plików XML przy pomocy DTD
3. Walidacja plików XML przy pomocy XML Schema

Part I

Język XML

Definicja

XML is a simple, very flexible text format derived from SGML.

Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

Źródło: `www.w3.org/XML`

Definicja

XML is a simple, very flexible text format derived from SGML.

Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

Źródło: `www.w3.org/XML`

- XML definiuje strukturę dokumentu, pozostawiając interpretację znaczenia elementów tej struktury programiście
- XML jest sposobem na utworzenie meta-opisu, dowolnej innej struktury danych

Pojęcie struktury (*structure*) i znaczenia (*meaning*)

Struktura

```
<?xml version="1.0" encoding="UTF-8" ?>
<xxxxxx xxxxxxxx-xx="xxxxxxxx">
    <xxxxx-xxxx xxxx="xxxx" xxxxx="x" xxx="xx" />
    <xxxxx-xxxx>xxxxx</xxxxx-xxxx>
    <xxxxxxx>xxxxx</xxxxxxx>
</xxxxxx>
```

Znaczenie

```
<?xml version="1.0" encoding="UTF-8" ?>
<person identity-nr="ab385639">
    <birth-date year="1997" month="2" day="12" />
    <first-name>Dawid</first-name>
    <surname>Weiss</surname>
</person>
```

Krótka historia języka XML

- XML powstał jako „uproszczenie” SGML
- Standard Generalized Markup Language (1960/70), standard ISO
- SGML jest bardzo rozbudowany, kosztowny w implementacji i długotrwały w opanowaniu
- XML — podzbiór SGML, 1998 (specyfikacja tylko ok. 20 stronicowa)
- XML szybko się rozprzestrzenił, powstają parsery dla wielu języków
- Technologie „follow-up”: Xpath, Xpointer, XSLT, XSL:FO, Xlink...

Dlaczego XML jest użyteczny?

- XML jest zrozumiały dla ludzi (jest plikiem tekstowym)

Dlaczego XML jest użyteczny?

- XML jest zrozumiały dla ludzi (jest plikiem tekstowym)
  - chyba, że struktura dokumentu nie ma sensu!
- XML ułatwia pracę programisty
  - parsery
  - walidacja
  - łatwo rozszerzalny o nowe elementy
  - XML jest „trendy” (?)

Przykład użyteczności XML

O przykładzie

Fragment dokumentu z napisem „hello world” zapisanego w programie OpenOffice (w XML) i MS Word (w formacie własnym).

```
<!DOCTYPE office:document-content PUBLIC "-//OpenOffice.org//DTD OfficeDocument 1.0//EN" "office.dtd">

<office:document-content xmlns:office="http://openoffice.org/2000/office" xmlns:style="http://openoffice.org/2

<office:script/>

<office:font-decls>
    <style:font-decl style:name="Arial Unicode MS" fo:font-family="&apos;Arial Unicode MS&apos;" style:font-pi
    <style:font-decl style:name="HG Mincho Light J" fo:font-family="&apos;HG Mincho Light J&apos;" style:font-
    <style:font-decl style:name="Thorndale" fo:font-family="Thorndale" style:font-family-generic="roman" style
</office:font-decls>

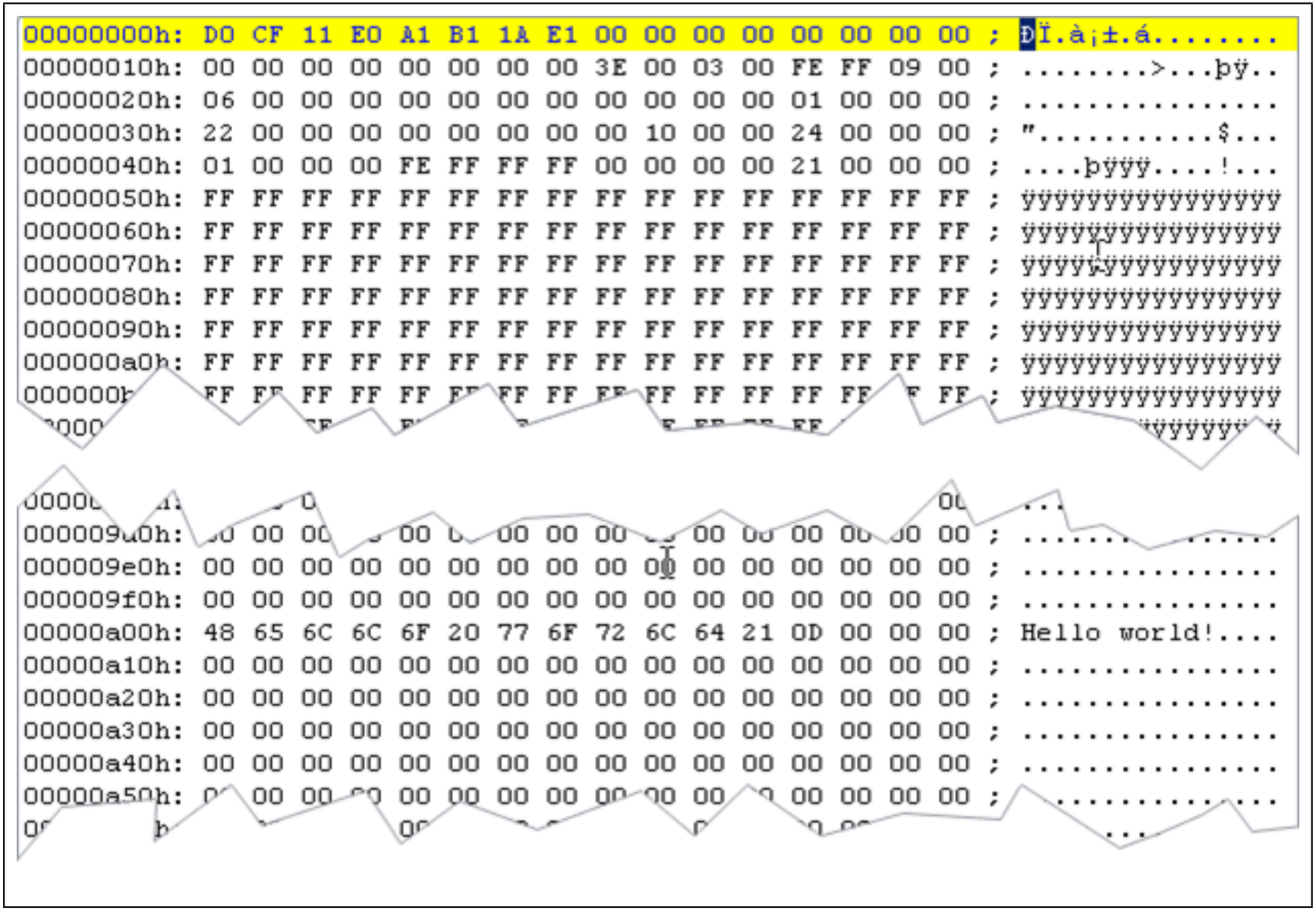
<office:automatic-styles/>

<office:body>
    <text:sequence-decls>
        <text:sequence-decl text:display-outline-level="0" text:name="Illustration"/>
        <text:sequence-decl text:display-outline-level="0" text:name="Table"/>
        <text:sequence-decl text:display-outline-level="0" text:name="Text"/>
        <text:sequence-decl text:display-outline-level="0" text:name="Drawing"/>
    </text:sequence-decls>

    <text:p text:style-name="Standard">Hello World!</text:p>

</office:body>

</office:document-content>
```


00000000h: D0 CF 11 E0 A1 B1 1A E1 00 00 00 00 00 00 00 00 ; ÐÏ.à¡±.á........
00000010h: 00 00 00 00 00 00 00 00 3E 00 03 00 FE FF 09 00 ; ....>...þÿ..
00000020h: 06 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 ; ....
00000030h: 22 00 00 00 00 00 00 00 00 10 00 00 24 00 00 00 ; "...........$...
00000040h: 01 00 00 00 FE FF FF FF 00 00 00 00 21 00 00 00 ; ....þÿÿÿ....!...
00000050h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000060h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000070h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000080h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000090h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000a0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000009e0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ....
000009f0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ....
00000a00h: 48 65 6C 6C 6F 20 77 6F 72 6C 64 21 0D 00 00 00 ; Hello world!....
00000a10h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ....
00000a20h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ....
00000a30h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ....
00000a40h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ....

- XML jest plikiem tekstowym zwykle rozpoczynającym się od prologu
- Prolog określa kodowanie znaków, oraz wersję języka XML
- Prolog jest opcjonalny, ale powinien być wyspecyfikowany
- Wszystkie parsery muszą wspierać przynajmniej UTF-8 i UTF-16

prolog — przykład

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Znaczniki (tagi)

- Struktura jest zdefiniowana przez znaczniki, zwane również tagami (ang. *markup*, *tags*)
- Tag to sekwencja alfanumerycznych znaków zawartych między „<” i „>” (dozwolone są również „-” oraz „_”)
- Tag otwierający: <nazwa>
- Tag zamykający: </nazwa>
- Tag pusty (bez zawartości): <nazwa/>
- Tagi można zagnieżdżać w sobie
- Tagi mogą zawierać tekst jako zawartość

Elementy (tagi — przykłady)

tagi — przykłady

- `<fee>129</fee>`
- ```
  <przodownicy-pracy>
          <osoba>
                  <imie>Krzysztof</imie>
                  <nazwisko>Kowalczykiewicz</nazwisko>
          </osoba>
          <osoba>
                  <imie>Tadeusz</imie>
                  <nazwisko>Morzy</nazwisko>
          </osoba>
  </przodownicy-pracy>
  ```
- `<jestem-pusty />`
  ```

Atrybuty

- Atrybuty pozwalają na związanie z tagami par nazwa-wartość
- Jedynie tagi otwierające mogą posiadać atrybuty
- Nazwa atrybutu jest unikalna w obrębie jednego tagu
  - niepoprawny przykład: `<person age="79" age="50" />`

atrybuty — przykłady

```
<fee currency="usd">129</fee>
<person marital-status="married" age="79" />
```

Pierwszy XML właściwie można napisać. . .

```
<?xml version="1.0" encoding="UTF-8" ?>
<tutorial id="ZWAI">
    <name>Zaawansowane wytwarzanie aplikacji internetowych</name>

    <duration>
        <semester>I</semester>
        <academic-year>2003</academic-year>
    </duration>

    <tutors>
        <tutor surname="Weiss" first-name="Dawid">
            <e-mail>dawid.weiss@cs.put.poznan.pl</e-mail>
        </tutor>
        <turor surname="Kowalczykiewicz" first-name="Krzysztof" />
    </tutors>

    <description>An awesome tutorial everyone has to take.</description>
</tutorial>
```

Instrukcje sterujące

- Instrukcje sterujące właściwie nie są częścią struktury, ale są wykrywane i sygnalizowane przez parsery XML
- Instrukcje sterujące mają bardzo luźną składnię (wiadomo tylko kiedy się zaczanają a kiedy kończą)
- Instrukcje sterujące są zwykle wykorzystywane aby dawać „wskazówki” procesorom przetwarzającym plik XML
- Prolog XML jest przykładem instrukcji sterującej

Instrukcje sterujące — przykłady

```
<?my processing instruction is here ?>
<?my fake-attribute="is here" ?>
```

Komentarze

- Komentarze to bloki tekstowe występujące między sekwencjami znaków: „<!--” i „-->”
- Komentarze nie mogą być deklarowane w definicji tagów
- W komentarzach nie może pojawić się sekwencja „--”

Instrukcje sterujące — przykłady

```
<person>
    <!-- here goes definition of a person -->
    <surname>Weiss<!-- who is this guy? --></surname>
</person>
```

Tekst, czyli Piąty Element

- Tekst może wystąpić wszędzie pomiędzy tagami

```
<poem>
  <author>
    M. Białoszewski
  </author>
  <title>Być to...</title>
  <contents>
  być
  to źle
  a nie?
  </contents>
</poem>
```

```
<poem>
␣␣<author>
␣␣␣␣M.␣Białoszewski
␣␣</author>
␣␣<title>Być␣to...</title>
␣␣<contents>
␣␣być
␣␣to␣źle
␣␣a␣nie?
␣␣</contents>
</poem>
```

Więcej o tekście

- Spacje między elementami mogą być, na życzenie programisty, zignorowane przez parsery XML (*ignorable whitespace*)
- Jeśli zachodzi potrzeba umieszczenia znaków specjalnych — „<>&”, należy posłużyć się albo encjami, albo blokiem CDATA

Więcej o tekście

- Spacje między elementami mogą być, na życzenie programisty, zignorowane przez parsery XML (*ignorable whitespace*)
- Jeśli zachodzi potrzeba umieszczenia znaków specjalnych — „<>&”, należy posłużyć się albo encjami, albo blokiem CDATA
  - Encje to nazwy, z którymi zostały skojarzone kody znaków: `&nazwa-encji;`
  - Encje mogą być również wyrażone numerycznie, jako kody Unicode znaków do wstawienia: dziesiętnie (`&#169;`) lub szesnastkowo (`&#x2105;`)
  - Blok CDATA ma postać: `<![CDATA[tekst]]>`

Przykład użycia bloku CDATA oraz encji

```
<fragment-xmla>
    <contents><![CDATA[
    <xml>
        <przyklad-xmla>
        </przyklad-xmla>
    </xml>
]]></contents>
</fragment-xmla>
```

```
<fragment-xmla>
    <contents>
    &lt;xml&gt;
        &lt;przyklad-xmla&gt;
        &lt;/przyklad-xmla&gt;
    &lt;/xml&gt;
</contents>
</fragment-xmla>
```

Więcej o tekście — kodowanie znaków

- Prolog mówi parserowi jakiego kodowania należy użyć w interpretacji dokumentu
- Nazwy stron kodowych: IETF RFC 1766
- Domyślna strona kodowa: UTF-8

Przykład — plik XML z literką „ą”

Plik XML w kodowaniu UTF-16

```
<?xml version="1.0" encoding="UTF-16"?>
<tag>ą</tag>
```

```
00000000h: FF FE 3C 00 3F 00 78 00 6D 00 6C 00 20 00 76 00 ; ÿþ<.?.x.m.l. .v.
00000010h: 65 00 72 00 73 00 69 00 6F 00 6E 00 3D 00 22 00 ; e.r.s.i.o.n.=.".
00000020h: 31 00 2E 00 30 00 22 00 20 00 65 00 6E 00 63 00 ; 1...0.". .e.n.c.
00000030h: 6F 00 64 00 69 00 6E 00 67 00 3D 00 22 00 55 00 ; o.d.i.n.g.=.".U.
00000040h: 54 00 46 00 2D 00 38 00 22 00 3F 00 3E 00 0D 00 ; T.F.-.8.".?.>...
00000050h: 0A 00 3C 00 74 00 61 00 67 00 3E 00 05 01 3C 00 ; ..<.t.a.g.>...<.
00000060h: 2F 00 74 00 61 00 67 00 3E 00 0D 00 0A 00       ; /.t.a.g.>.....
```

Plik XML w kodowaniu ISO8859-2

```
<?xml version="1.0" encoding="iso8859-2"?>
<tag>ą</tag>
```

```
00000000h: 3C 3F 78 6D 6C 20 76 65 72 73 69 6F 6E 3D 22 31 ; <?xml version="1
00000010h: 2E 30 22 20 65 6E 63 6F 64 69 6E 67 3D 22 69 73 ; .0" encoding="is
00000020h: 6F 38 38 35 39 2D 32 22 3F 3E 0D 0A 3C 74 61 67 ; o8859-2"?>..<tag
00000030h: 3E B1 3C 2F 74 61 67 3E 0D 0A                   ; >±</tag>..
```

Za chwilkę dalszy ciąg programu...