

## XSLT – część 2. Inne standardy związane z XML-em

## Rodzaje przetwarzania XSLT (1)

- Przetwarzanie sterowane strukturą dokumentu źródłowego:
  - przechodzimy po strukturze dokumentu źródłowego,
  - generujemy fragmenty struktury dokumentu wyjściowego.

```
<xsl:template match="...">
  ...
  <xsl:apply-templates/>
  ...
</xsl:template>
```

## Rodzaje przetwarzania XSLT (2)

- Przetwarzanie sterowane strukturą dokumentu wyjściowego:
  - jedna duża reguła dla węzła *root*,
  - generujemy strukturę dokumentu docelowego,
  - wyciągamy odpowiednie wartości z dokumentu źródłowego.

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns="http://www.w3.org/TR/xhtml1/strict">
  <xsl:template match="/">
    <html><head><title>Expense Report Summary</title></head>
    <body>
      <h1>Company: <xsl:value-of select="company/name"/></h1>
      <p>Total Amount:
        <xsl:value-of select="expense-report/total"/></p>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

## Zmienne

- Jak w funkcyjnych językach programowania:
  - brak instrukcji przypisania,
  - brak efektów ubocznych.
- Deklaracja:
  - `<xsl:variable name=.../>`
  - wartość:
    - atrybut `select` → wyrażenie odpowiedniego typu,
    - zawartość elementu → fragment drzewa wynikowego,
- Użycie:
  - w wyrażeniach: `$name`,
  - `<xsl:copy-of select=expression/>`

## Zaawansowane możliwości XSLT

- Sortowanie węzłów.
- Wzorce nazwane:
  - wywoływane jak podprogramy (procedury),
  - przekazywanie parametrów,
  - rekursja.
- Tryby przetwarzania (*modes*):
  - przełączanie między trybami,
  - niezależnie definiowane wzorce dla każdego trybu.

## Wykorzystanie rekursji w XSLT (1)

- Sposób na brak „prawdziwych” zmiennych i pętli iteracyjnych.
- Przykład:

```
<xsl:template name="Books">
  <xsl:param name="cnt" select="1"/>
  <xsl:if test="$cnt > 0">
    <book>
      <tr><xsl:apply-templates/></tr>
    </book>
    <xsl:call-template name="Books">
      <xsl:with-param name="cnt" select="$cnt - 1"/>
    </xsl:call-template>
  </xsl:if>
</xsl:template>
```

## Wykorzystanie rekursji w XSLT (2)

- Przykład – c.d.:

```
<xsl:template match="book">
  <xsl:choose>
    <xsl:when test="@count">
      <xsl:call-template name="Books">
        <xsl:with-param name="cnt" select="@count"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:call-template name="Books"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

## Generowanie przekształceń XSL (1)

- Problem:
  - źródło przekształcenia nie zawiera metainformacji o strukturze dokumentu,
  - metainformacje pojawiają się na wyjściu.

```
<wniosek-urlopowy>
<wniosek>
  <pracownik>Szymon Ziolo</pracownik>
  <rodzaj>wypoczynkowy</rodzaj>
  <od>2003-06-20</od>
  <do>2003-06-27</do>
  <dni-roboczych>6</dni-roboczych>
</wniosek>
<decyzja>
  <zgoda>1</zgoda>
  <zastepca>Jan Kowalski</zastepca>
</decyzja>
</wniosek-urlopowy>
```

### Wniosek urlopowy

Wniosek

Pracownik:	Szymon Ziolo
Rodzaj urlopu:	wypoczynkowy
Od dnia:	2003-06-20
Do dnia:	2003-06-27
Łość dni roboczych:	6

Decyzja przełożonego

Zgoda przełożonego:	tak
Zastępca:	Jan Kowalski

Źródło: Ziolo, Sz., XSLT do kwadratu, Software 2.0, nr 6/2003

## Generowanie przekształceń XSL (2)

- Rozwiązanie:
  - zapisanie metainformacji w szablonie,
  - generowanie przekształcenia z szablonu.

```
<dokument nazwa="wniosek-urlopowy"
  etykieta="Wniosek urlopowy">
  <sekcja nazwa="wniosek" etykieta="Wniosek">
    <pole nazwa="pracownik" etykieta="Pracownik:"/>
    <pole nazwa="rodzaj" etykieta="Rodzaj urlopu:"/>
    <pole nazwa="od" etykieta="Od dnia:"/>
    <pole nazwa="do" etykieta="Do dnia:"/>
    <pole nazwa="dni-roboczych"
      etykieta="Łość dni roboczych:"/>
  </sekcja>
  <sekcja nazwa="decyzja" etykieta="Decyzja przełożonego">
    <pole nazwa="zgoda" etykieta="Zgoda przełożonego:"
      typ="boolean"/>
    <pole nazwa="zastepca" etykieta="Zastępca:"/>
  </sekcja>
</dokument>
```

## Generator – przykład (1)

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:res="http://www.w3.org/1999/XSL/TransformAlias">
  <xsl:namespace-alias stylesheet-prefix="res"
    result-prefix="xsl"/>

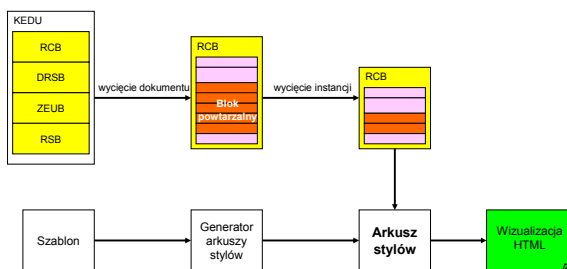
  <xsl:template match="/">
    <res:stylesheet version="1.0">
      <res:output method="html"/>
      <xsl:apply-templates/>
    </res:stylesheet>
  </xsl:template>

  <xsl:template match="sekcja">
    <res:template match="{@nazwa}">
      <p><b><xsl:value-of select="@etykieta"/></b></p>
      <table><res:apply-templates/></table>
    </res:template>
  <xsl:apply-templates/>
</xsl:template>
```

## Generator – przykład (2)

```
<xsl:template match="pole">
  <res:template match="{@nazwa}">
    <td><td><xsl:value-of select="@etykieta"/></td>
  </td></td>
  <xsl:choose>
    <xsl:when test="@typ='boolean'">
      <xsl:choose>
        <res:when test="text()='1'">tak</res:when>
        <res:otherwise>nie</res:otherwise>
      </res:choose>
    </xsl:when>
    <xsl:otherwise>
      <res:value-of select="text()"/>
    </xsl:otherwise>
  </xsl:choose>
</td></td></td>
</res:template>
<xsl:apply-templates/>
</xsl:template>
</xsl:stylesheet>
```

## Zastosowanie w projekcie: KEDU ZUS



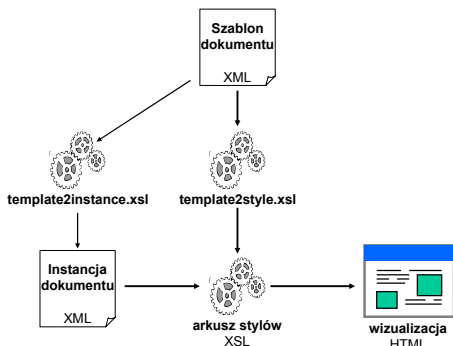
## KEDU ZUS – przykład wizualizacji

Legnica	EMERYTALNE	REZERWY	CHOROBLIWE	WYPIŁIOWE
15. Liczba ubezpieczonych	111,11	12,22	1333,33	1,44
16. Kwota składek	155,55	16666,66	1,77	18,88
17. Liczba ubezpieczonych	1999,99	100,00		
18. Kwota składek	1010,10		146,80	135,79
19. Liczba ubezpieczonych	187,65	16543,21		1,01
20. Kwota składek	19999,99		13,44	1,01
21. Liczba ubezpieczonych	189,01			134,56
22. Kwota składek				

## Zastosowanie w projekcie: Era DCO

- Document Collection Office:
  - system obiegu dokumentów strukturalnych,
  - edycja przy pomocy formularzy HTML,
  - zastosowanie: obieg protokołów z przeglądów stacji bazowych.
- Szablon dokumentu:
  - struktura (nazwy pól, typy, kontroli edycyjne),
  - pola automatycznie obliczane,
  - reguły walidacji,
  - role i uprawnienia,
  - proces obiegu dokumentu.
- Generatory:
  - schematu XML Schema,
  - pustej instancji dokumentu,
  - arkuszy stylów,
  - ewaluatorów pól automatycznie obliczanych,
  - walidatorów.

## Era DCO – schemat przetwarzania



## Formatting Objects – przykład drzewa wynikowego

```

<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
...
  <fo:page-sequence>
    <fo:flow>
      <fo:block font-size="18pt" font-weight="bold"
        text-align="center">Preface</fo:block>
      <fo:block font-size="12pt" space-before="1pc"
        text-align="justified"> This is a simple
        test document. It shows a
        <fo:inline font-style="italic">partial</fo:inline>
        fo-result tree (page layout missing).</fo:block>
    </fo:flow>
  </fo:page-sequence>
...
</fo:root>
    
```

## Formatting Objects – przykłady reguł

```

<xsl:template match="chapter">
  <fo:flow><xsl:apply-templates/></fo:flow>
</xsl:template>

<xsl:template match="chapter/title">
  <fo:block font-size="18pt" font-weight="bold"
    text-align="center">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>

<xsl:template match="para">
  <fo:block font-size="12pt" space-before="1pc"
    text-align="justified">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>

<xsl:template match="emphasis">
  <fo:inline font-style="italic"><xsl:apply-templates/>
</fo:inline>
</xsl:template>
    
```

## Rozwój XSL-a i okolic

- XQuery 1.0 and XPath 2.0 Data Model Working Draft.
- XQuery 1.0 Working Draft:
  - język zapytań o kolekcje dokumentów XML.
- XSLT 2.0 Working Draft:
  - wsparcie dla XML Schema,
  - grupowanie węzłów (for-each-group),
  - definiowanie i wywoływanie własnych funkcji,
  - generowanie wielu dokumentów wyjściowych,
  - ...
- XPath 2.0 Working Draft
  - wyrażenia warunkowe (if i inne),
  - kwantyfikator,
  - typ „sekwencja”, pętla,
  - ...

## Narzędzia

- Procesory XSLT:
  - XT, James Clark (Java),
  - Oracle XML Parser for Java / C / PL-SQL,
  - Xalan, Apache (Java, C++),
  - SAXON, Michael H. Kay (Java; implementuje XSLT 2.0 i XQuery),
  - Sablotron (C++, open source),
  - Microsoft XML Core Services (MSXML 4.0).
- Procesor XSL:FO:
  - FOP, Apache (Java; generuje dokumenty w formacie PDF).
- Edytory XSLT:
  - XMLSPY 2004, Altova
  - Xselerator XSL Editor/Debugger, MarrowSoft,
  - xslide – Emacs Major Mode for XSL Stylesheets.

## XQuery – zapytania o dokumenty XML

- Język zapytań o dokumenty XML:
  - podobny do SQL-a,
  - pozwala pytać o kolekcje dokumentów,
  - rozszerza funkcjonalność XPath m. in. o:
    - zmienne,
    - wyrażenia FLWOR (for, let, where, order by, return),
    - złączenia,
    - wyrażenia warunkowe,
    - definiowanie funkcji,
    - typ sekwencji,
    - konstruktory, umożliwiające tworzenie struktur XML w zapytaniu,
    - wyrażenia walidujące względem schematu.
- Dostępne składnie:
  - SQL-opodobna,
  - oparta na XML-u.

## XQuery – przykład

- Zwróć nazwę każdego wydawcy i średnią cenę jego książek:

```
FOR $p IN distinct(document("bib.xml"))//publisher
LET $a :=
avg(document("bib.xml")//book[publisher = $p]/price)
RETURN
<publisher>
  <name>{ $p/text() }</name>
  <avgprice>{ $a }</avgprice>
</publisher>
```

Źródło: XML Syntax for XQuery 1.0 (XQueryX), <http://www.w3.org/TR/xqueryx>

## XPointer – adresowanie fragmentów dokumentu XML

- XPath „opakowany” w składnię URI, np.:  
[http://www.sejm.gov.pl/ustawa.xml#xpointer\(/art\[5\]/par\[2\]\)](http://www.sejm.gov.pl/ustawa.xml#xpointer(/art[5]/par[2]))
- Dodatkowe możliwości:
  - proste adresowanie elementów opatrzonych identyfikatorami, np.:  
urlopy
  - adresowanie elementów bez znajomości struktury, z wykorzystaniem identyfikatorów jako kotwic, np.:  
element(/4/2/3)  
element(urlopy/2/3)
  - wskazanie punktu w dokumencie,
  - wskazanie zakresu pomiędzy dwoma punktami,
  - wskazanie punktów w tekście i fragmentów tekstu.

## XPointer – status

- Rekomendacje W3C z 25 marca 2003:
  - XPointer Framework,
  - XPointer element() scheme,
  - XPointer xmlns() scheme.
- W3C Working Draft:
  - XPointer xpointer() scheme.

## XInclude – łączenie zawartości dokumentów

- Załączanie zawartości jednego dokumentu XML do innego:
  - pozwala załączyć fragment dokumentu (wskazany XPointer-em),
  - pozwala określić zawartość używaną w razie błędu.
- Przykład:

```
<file name="salatka.xml">
  <xi include
    href="salatka.xml#xpointer(/przepis/tytul)"/>
</file>
– po przetworzeniu procesorem XInclude:
<file name="salatka.xml">
  <tytul>Sałatka z ogórków</tytul>
</file>
```
- Status: W3C Proposed Recommendation

## XLink – dowiązania w XML-u

- Linki jakie znamy (HTML):
  - łączą dwa dokumenty: źródło i cel linku,
  - źródłem linku jest zawsze element opisujący link (A, IMG).
- XLink – rozszerzona koncepcja dowiązań:
  - dowolne elementy przechowują informacje o linkach,
  - informacja o linkach poza połączonymi dokumentami,
  - więcej niż dwa końce linku.
- Status:
  - rekomendacja W3C z 27 czerwca 2001,
  - korzenie historyczne: HyTime.

## Terminologia

- **Zasób (resource)** – dowolna adresowalna jednostka informacji lub usługa.
- **Dowiązanie (link)** – jawnie wyrażona (przy pomocy **elementu wiążącego (linking element)**) relacja pomiędzy zasobami.
  - te zasoby uczestniczą (*participate*) w dowiązaniu.
- **Przejsięcie (traversal)** – użycie pary zasobów połączonej dowiązaniem.
- **Łuk (arc)** – informacja o przejściu między dwoma zasobami (kierunek, zachowanie aplikacji, itp.):
  - wychodzący (*outbound*),
  - wchodzący (*inbound*),
  - niezależny (*third party*).

## Dowiązania XLink

- **Extended link:**
  - wiąże dowolną liczbę zasobów:
    - zasoby zewnętrzne (np. odwołania do innych dokumentów),
    - zasoby lokalne (zawarte w elemencie wiążącym).
  - łuki opisujące sposoby przechodzenia pomiędzy zasobami,
  - role zasobów uczestniczących w linku,
  - role łuków.
- **Simple link:**
  - link wychodzący,
  - wiąże dokładnie 2 zasoby: 1 lokalny i 1 zewnętrzny,
  - jeden łuk z zasobu lokalnego do zewnętrznego.

## Simple link – przykład

```
<osoba xmlns:xlink="http://www.w3.org/1999/xlink">
  <nazwisko>Kopernik, Mikołaj</nazwisko>
  <biogram>Wybitny polski astronom, matematyk, lekarz,
  prawnik, tłumacz poezji włoskiej i ekonomista,
  pochodził z rodziny wywodzącej się z mieszczan
  krakowskich. Urodzony w <geogr xlink:type="simple"
  xlink:href="Torun.xml">Toruniu</geogr>.</biogram>
</osoba>
```

## Extended link - przykład

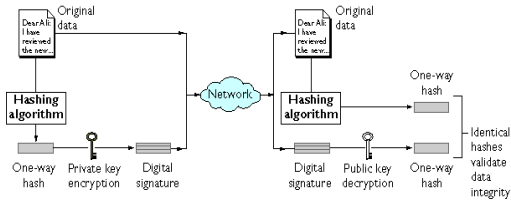
```
<fikcja xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:type="extended">
  <wypowiedz xlink:type="resource">Kopernik była
  kobietą!</wypowiedz>
  <film xlink:type="locator" xlink:href="seksmisja.xml"
  xlink:title="Seksmisja"/>
  <osoba xlink:type="locator" xlink:href="kopernik.xml"
  xlink:title="Kopernik, Mikołaj"/>
  <pojecie xlink:type="locator"
  xlink:href="kobieta.xml"
  xlink:title="kobieta"/>
</fikcja>
```

## Przyszłość XLink

- **Zastosowania:**
  - organizowanie, kojarzenie zasobów, nawet gdy nie mamy prawa zapisu,
  - dostarczanie wartości dodanej – zbiorów linków.
- **Zasięg:**
  - lokalny: serwery linków operujące na bazie linków,
  - Internet?
- **Problemy:**
  - wizualizacja extended links,
  - synchronizacja zasobów i linków (Internet).

## XML Signature – podpis elektroniczny

- Zasada działania podpisu elektronicznego:



- Kluczowa rola:
  - jakości algorytmu haszującego (funkcji skrótu),
  - jakości asymetrycznego algorytmu szyfrowania/deszyfrowania,
  - zaufania do wystawcy certyfikatu.

## XML Signature – podpis elektroniczny

- Podpis dokumentu XML-owego:
  - zapisany w postaci struktury XML-owej,
  - umieszczony w elemencie Signature:
    - w osobnym dokumencie,
    - dołączonym do podpisywanego dokumentu,
    - zawierającym podpisywane dane.
- Możliwości XML Signature:
  - podpisywanie fragmentów dokumentu XML,
  - podpisy wielokrotne.

## XML Signature – przykład

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm=
      "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod Algorithm=
      "http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
    <!-- w URI znajduje się wskazanie na podpisywane dane -->
    <Reference URI="http://przyklad.pl/pliki/do-podpisu.xml">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#base64"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>60NvZvtdTB+7UnLp/H24p7h4bs=</DigestValue>
    </Reference>
  </SignedInfo>
  <!-- zaszyfrowany skrót z SignedInfo - podpis -->
  <SignatureValue>OsH9AljTNL...</SignatureValue>
  <KeyInfo><KeyValue><DSAKeyValue>
    <P>imup6lm...</P><Q>xDve3j7...</Q><G>NlugA...</G>
    <Y>W7dOmH/v...</Y>
  </DSAKeyValue></KeyValue></KeyInfo>
</Signature>
```

Źródło: Kazienko, P., Co tam panie w XML-u?, Software 2.0, 6/2003

## XML Encryption – szyfrowanie XML-a

- Cel: zagwarantowanie poufności danych XML-owych.
- Najczęstszy scenariusz:
  - wygenerowanie losowego klucza symetrycznego (sesyjnego),
  - zaszyfrowanie nim danych źródłowych,
  - zaszyfrowanie klucza sesyjnego kluczem publicznym odbiorcy.

```
<InfoPlatnicza xmlns="http://przyklad.pl/platnosci">
  <Nazwa>Józef Nowak</Nazwa>
  <KartaKredytowa Limit="2,000"
    Waluta="PLN" System="Visa">
    <NrKarty>4019244502775567</NrKarty>
    <Wystawca>Nasz Bank S.A.</Wystawca>
    <DataWaznosci>10/03</DataWaznosci>
  </KartaKredytowa>
</InfoPlatnicza>
```

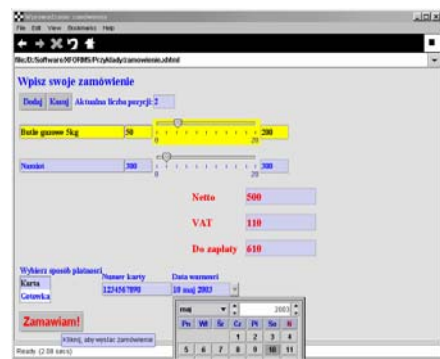
Źródło: Kazienko, P., Co tam panie w XML-u?, Software 2.0, 6/2003

```
<InfoPlatnicza xmlns="http://przyklad.pl/platnosci">
  <Nazwa>Józef Nowak</Nazwa>
  <EncryptedData
    Type="http://www.w3.org/2001/04/
      xmlelement"
    xmlns="http://www.w3.org/2001/04/
      xmlelement">
    <CipherData>
      <CipherValue>A2s3B4f5gCbDyBReHwTWC
        5cx6wQ3g5teV=</CipherValue>
    </CipherData>
  </EncryptedData>
</InfoPlatnicza>
```

## XForms – zaawansowane formularze

- Odpowiedź na ograniczenia formularzy w HTML-u:
  - kontrola dziedziny wprowadzanych danych po stronie klienta,
  - specyfikowanie pól obowiązkowych i opcjonalnych,
  - zależności między polami, np.:
    - wartość pola jest funkcją wartości innych pól,
    - pole jest widoczne tylko przy określonej zawartości innych pól;
  - wyzwalacze aktywowane zdarzeniami interfejsu użytkownika,
  - bogaty zasób kontrolki, np.:
    - powtórzenia, grupy, wielopoziomowe wybory,
    - suwaki do wyboru wartości z zakresu,
    - kontrolki wyboru plików.
- Status:
  - rekomendacja W3C z 14 października 2003 r.,
  - możliwość zanurzania w XHTML-u.

## XForms – przykład (program X-Smiles)



Źródło: Kazienko, P., Co tam panie w XML-u?, Software 2.0, 6/2003

## Gdzie szukać dalej

- Tyszko, S., *Rekurencyjne szablony w XSLT*
  - Software 2.0, nr 6/2002, Wydawnictwo Software
- Ziolo, Sz., *XSLT do kwadratu*
  - Software 2.0, nr 6/2003, Wydawnictwo Software
- Kazienko, P., *Co tam panie w XML-u?*
  - Software 2.0, nr 6/2003, Wydawnictwo Software



## Gdzie szukać dalej

- Arciniegas, A. F., *What is XLink?*
  - 🔗 [www.xml.com/pub/a/2000/09/xlink](http://www.xml.com/pub/a/2000/09/xlink)
- Carr, L., *Initial Experiences of an XLink Implementation*
  - 🔗 [journals.ecs.soton.ac.uk/xml4j/xlinkexperience.html](http://journals.ecs.soton.ac.uk/xml4j/xlinkexperience.html)
- *XPointer tutorial*
  - 🔗 [www.zvon.org/xxl/xpointer/tutorial/OutputExamples/xpointer\\_tut.html](http://www.zvon.org/xxl/xpointer/tutorial/OutputExamples/xpointer_tut.html)
- *Tamino XQuery Demo*
  - 🔗 [tamino.demozone.softwareag.com/demoXQuery](http://tamino.demozone.softwareag.com/demoXQuery)
- Dubinko, M., *Ten Favorite XForms Engines*
  - 🔗 [www.xml.com/pub/a/2003/09/10/xforms.html](http://www.xml.com/pub/a/2003/09/10/xforms.html)

