

# Obiekty formatujące w języku XSL

Tomasz Traczyk

*Politechnika Warszawska*

*e-mail: ttraczyk@ia.pw.edu.pl*

## Abstrakt

XSL został stworzony jako język formatowania dokumentów XML. Składa się on z części służącej do transformacji dokumentów, zwanej XSLT (XSL Transformations) oraz ze słownika obiektów formatujących (FO - Formatting Objects), które służą do określania sposobu prezentacji zawartości i powinny być interpretowane przez przeglądarki. Ze względu na brak implementacji FO w przeglądarkach, popularna stała się metoda formatowania dokumentów XML przez transformację do HTML, ewentualnie z CSS. Ma ona jednak liczne ograniczenia, wynikające z niewielkich możliwości HTML co do formatowania. Obecnie pojawiać się zaczęły pierwsze implementacje FO, warto więc zacząć rozważać wykorzystanie obiektów formatujących do tworzenia bardziej złożonych prezentacji dokumentów XML. Artykuł przypomina podstawowe fakty związane z językiem XSL, przedstawia możliwości formatowania za pomocą obiektów formatujących oraz podsumowuje obecne osiągnięcia w dziedzinie implementacji tego standardu, w tym możliwości jego wykorzystania z użyciem narzędzi firmy Oracle.

## 1. Wprowadzenie

Znaczniki języka XML nie mają przypisanego żadnego predefiniowanego wyglądu. Do prezentacji dokumentów w XML potrzebne jest zatem ich uprzednie sformatowanie.

Język XSL (*Extensible Stylesheet Language* [1, 3]) stworzono właśnie jako narzędzie do formatowania dokumentów w XML. Podstawowym zastosowaniem XSL miało zatem być pisanie arkuszy stylistycznych. Pełna wersja standardu XSL kształtowała się jednak długo i brakowało popularnych implementacji. Na razie rozpowszechniło się więc stosowanie wcześniej ustabilizowanej części języka, zwanej XSLT (*XSL Transformations*), jako uniwersalnego narzędzia do przekształcania dokumentów XML (niekoniecznie związanego z formatowaniem). Często używana jest „kombinacja” XSLT z językiem HTML. Zastosowanie wbudowanych w typowe przeglądarki WWW procesorów XSLT, w połączeniu z możliwościami HTML i CSS, umożliwia publikowanie dokumentów XML w sieci Web w formie przyjaznej dla człowieka.

Ten sposób wykorzystania XSL, choć w wielu zastosowaniach skuteczny, nie jest jak się wydaje w pełni zgodny z intencjami twórców języka. W dodatku zastosowanie HTML jako języka pośredniego implikuje pewne ograniczenia, wynikające przede wszystkim z tego, że HTML jest z założenia przeznaczony do prezentacji interaktywnej na ekranie. Brak w nim więc elementów charakterystycznych dla języków opisu publikacji drukowanych, związanych np. ze stronicowaniem.<sup>1</sup>

Tymczasem pierwotny, obecnie wreszcie zrealizowany pomysł na formatowanie za pomocą XSL polega na przekształceniu dokumentu wejściowego w XML, reprezentującego pewien specyficzny schemat<sup>2</sup> odwzorowujący logiczną strukturę informacji, w inny dokument w XML, o standardowym schemacie, odwzorowującym wizualną strukturę prezentacji czy wydruku. Przekształcenia tego dokonuje się za pomocą XSLT, zaś wynikowy dokument tworzony jest w języku XSL-FO. Ten wynikowy dokument jest bezpośrednio interpretowany przez tzw. serializer, który wyświetla dokument, tworzy postać drukowalną itp. W przyszłości można spodziewać się wbudowania takich serializerów w przeglądarki, na razie jednak dostępne są jedynie jako osobne produkty, nastawione zwykle na tworzenie publikacji drukowanych.

Obiekty formatujące XSL-FO (*XSL Formatting Objects*) to drugi obok XSLT podzbiór XSL. Standard ten definiuje obszerny zbiór znaczników opisujących prezentację dokumentu w postaci drukowanej, wyświetlanej interaktywnie, a nawet automatycznie odczytywanej głosowo. Duży nacisk położono na cechy związane z tworzeniem profesjonalnych publikacji drukowanych; w tym kierunku idą także pierwsze implementacje standardu.

Obecnie, po ukazaniu się rekomendacji całości XSL [1] i pierwszych ważnych implementacji standardu XSL-FO, można już do formatowania dokumentów XML – zwłaszcza złożonych i przeznaczonych do druku – użyć niemal pełnych możliwości języka; jest to zatem odpowiedni moment by przyjrzeć się owym możliwościom.

---

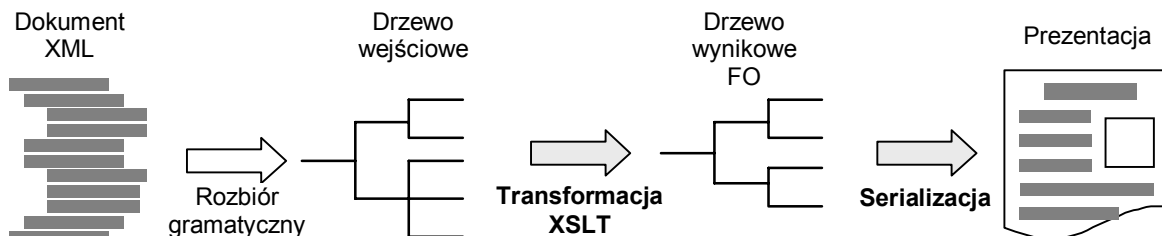
<sup>1</sup> Daje się to częściowo rozwiązać dzięki językowi CSS 2.

<sup>2</sup> Słowo „schemat” jest tu użyte w znaczeniu właściwym dla XML: chodzi o definicję składni dokumentu, opisującą dopuszczalne znaczniki, ich strukturę, atrybuty, typy danych itp.

## 2. Podstawy XSL-FO

### 2.1.1. Formatowanie za pomocą XSL

Jak już powiedziano, formatowanie za pomocą obiektów formatujących odbywa się w dwóch fazach, jak to przedstawia **Rysunek 1**.



Rysunek 1. Formatowanie dokumentu XML za pomocą XSL

Wynikiem transformacji XSLT jest dokument XML o schemacie określonym w standardzie XSL-FO, reprezentowany jako drzewo obiektów formatujących. Może ono być zapisane w postaci pliku (.fo) albo bezpośrednio przekształcone w wydruk czy prezentację.

### 2.1.2. Budowa dokumentu FO

Dokument XSL-FO zawiera elementy opisujące wynikowy wygląd (wydruk, prezentację). Dokument ten oczywiście spełniać musi wszystkie wymogi poprawności XML (w sensie *valid*).

Znaczniki opisujące obiekty formatujące należą do specjalnej przestrzeni nazw, oznaczanej zwykle prefiksem `fo`. Cały dokument zawarty jest w elemencie-korzeniu o nazwie `root`. Dokument składa się z dwóch zasadniczych części:

- opisu sposobu stronicowania;
- opisu właściwego tekstu.

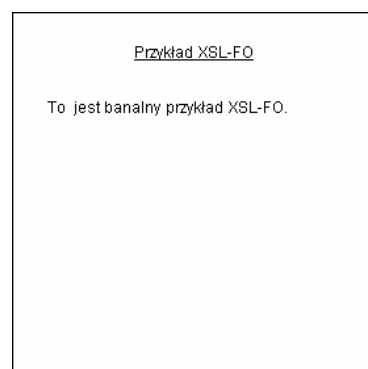
Formatowany tekst traktowany jest jako tzw. strumień tekstu: w kolejne strony „wlewa” się kolejno napływający tekst, łamiąc go na linie i strony tak, by pasowały one do określonych na początku wzorców stron. Podział na linie i strony jest dokonywany automatycznie, natomiast podział tekstu na inne części, np. akapity, należy już do twórcy dokumentu (czy raczej arkusza stylistycznego).

Choć składnia języka XSL-FO jest całkiem XML-owa, to wiele jego składników (np. nazwy atrybutów) bardzo przypomina CSS 2. Nie jest to oczywiście przypadek – oba języki są przecież dziełem W3C (*World Wide Web Consortium*).

#### Przykład 1

Oto bardzo prosty przykładowy dokument XSL-FO:

```
<?xml version="1.0" encoding="windows-1250"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="prosta-strona"
      page-height="10cm" page-width="10cm"
      margin-top="1cm" margin-bottom="1cm"
      margin-left="1cm" margin-right="1cm"
    >
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>
```



```

<fo:page-sequence master-reference="prosta-strona">
  <fo:flow flow-name="xsl-region-body">
    <fo:block text-align="center"
      text-decoration="underline" space-after="1cm"
    >
      Przykład XSL-FO
    </fo:block>
    <fo:block>
      To jest banalny przykład XSL-FO.
    </fo:block>
  </fo:flow>
</fo:page-sequence>
</fo:root>

```

Pierwsza część dokumentu definiuje stronicowanie – w tym wypadku tylko jeden format strony. Druga część zawiera właściwy tekst, a odwołuje się do zdefiniowanego wcześniej stronicowania. Tekst podzielony jest na bloki odpowiadające akapitom.

### 2.1.3. Dziedziczenie

Wiele cech obiektów formatujących podlega dziedziczeniu w hierarchii tych obiektów: obiekty podrzędne dziedziczą cechy od obiektów nadrzędnych (oznacza to, że znaczniki zagnieżdżone dziedziczą od znaczników obejmujących). Dziedziczenie dotyczy większości cech, w tym cech wizualnych, wcięć itp. Na przykład

```
<fo:root font-size="12pt" font-family="serif" language="pl" hyphenate="true" ...>
```

oznacza, że w całym dokumencie – o ile tego na niższych poziomach hierarchii znaczników jawnie nie przedefiniowano – użyta będzie czcionka bezszeryfowa 12-punktowa i aktywne będzie przenoszenie wyrazów według reguł języka polskiego.

Istnieje specjalny element wrapper, który służy jedynie jako „przezroczysty” dodatkowy poziom w hierarchii, na którym można zdefiniować cechy przeznaczone do dziedziczenia.

## 3. Elementy XSL-FO

### 3.1. Stronicowanie

W XSL-FO zdefiniowanie sposobu stronicowania jest obowiązkowe. W najprostszej wersji można określić tylko jeden format strony, ale złożone dokumenty mogą mieć zdefiniowane wiele wzorców stron, używanych do formatowania poszczególnych części tekstu.

#### 3.1.1. Kierunki

W XSL-FO stosuje się dwa różne rodzaje określania kierunków:

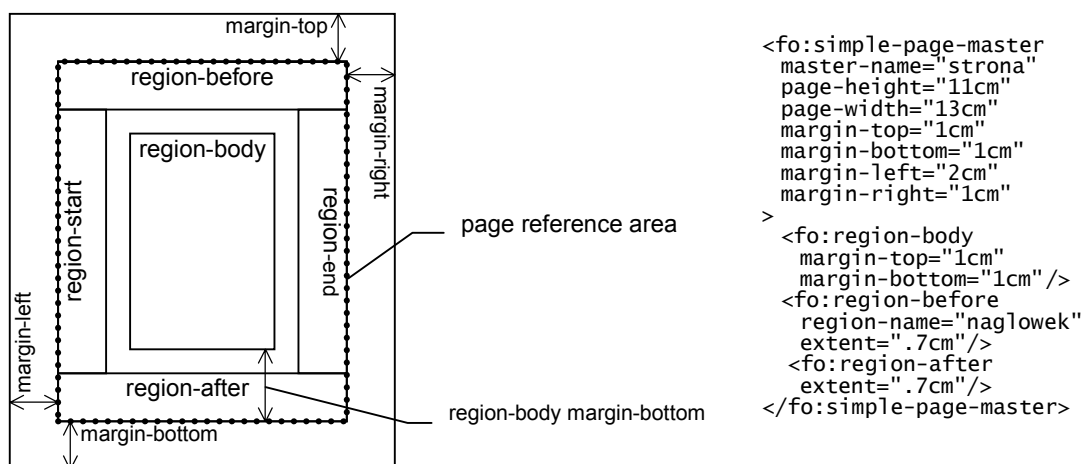
- bezwzględny, związany z nośnikiem: w pionie *top* – *bottom*, w poziomie *left* – *right*;
- względny, związany z kierunkiem pisania: w kierunku pisania kolejnych liter *start* – *end*, w kierunku przyrastania wierszy/bloków tekstu *before* – *after*.

Takie rozróżnienie jest niezbędne, ponieważ wszystkie standardy związane z XML z założenia przewidują obsługę pełnego Unicode, a więc także języków, w których sposób pisania odbiega od europejskiego. Kierunek pisania, można zdefiniować dla całego dokumentu lub dla jego poszczególnych fragmentów. Rozróżnienie to przydaje się także, gdy zawartość strony ma być obrócona w stosunku do orientacji nośnika (standard przewiduje możliwość obracania stron, a nawet fragmentów, o wielokrotność 90°).

### 3.1.2. Model strony

Strona w XSL-FO składa się z pustych marginesów oraz z regionów, w których umieszczona będzie zawartość dokumentu, co przedstawia **Rysunek 2**.

„Czynne” pole strony (*page reference area*) wyznaczone jest przez jej wielkość oraz marginesy. W polu tym mieści się obszar *region-body*, przeznaczony na właściwy tekst. Jego marginesy określają wielkość obszaru i jego położenie względem pola strony. W „czynnym” polu strony mogą się także znaleźć dodatkowe obszary<sup>3</sup>, przeznaczone np. na nagłówki lub stopki, jak to pokazano w przykładowej specyfikacji strony obok rysunku.



Rysunek 2. Model strony XSL-FO (typowa orientacja kierunków dla języków europejskich)

Do opisu wzorca pojedynczej strony lub ciągu jednakowo zbudowanych stron służy element (obiekt) `simple-page`. Wzorcowi nadaje się nazwę (`master-name`), do której odwoływać się będą elementy zawierające formatowany strumień tekstu.

### 3.1.3. Sekwencje stron

Proste dokumenty mogą mieć tylko jeden wzorec strony, ale w przypadku profesjonalnych publikacji zapewne niezbędne będzie stworzenie większej liczby wzorców, np. w celu


- wyróżnienia strony tytułowej;
- odróżnienia stron parzystych od nieparzystych (niesymetryczne marginesy, różne nagłówki itp.);
- wydzielenia części o innym sposobie łamania (wieloszpaltowych, obróconych itp.).

Z wzorców stron buduje się wzorce sekwencji (`page-sequence-master`), które sterują wykorzystaniem wzorców stron w czasie formatowania strumienia tekstu. Wzorce sekwencji mogą zawierać bezwzględne lub warunkowe odwołania do wzorców stron, wywołane jednokrotnie lub w sposób powtarzalny. Warunki pozwalają odróżniać strony parzyste od nieparzystych, puste od niepustych oraz pierwszą (w danej sekwencji), ostatnią i pozostałe.

### Przykład 2

Przykładowy dokument, który ukazuje **Rysunek 3**, wymaga aż sześciu wzorców stron: dla stron tytułowych (pierwszych w sekcji tekstu), parzystych i nieparzystych – odpowiednio jedno- i dwuszpaltowych.

<sup>3</sup> Obszary te powinny mieścić się w marginesach obszaru *body*.

<p style="text-align: center;"><b>Oferta programowa</b></p> 	<p><b>1. Wstęp</b></p> <p>W niniejszym dokumencie prezentuje się ofertę programową Kursu Podyplomowego Wiedzy Ogólnej w Wyższej Szkole Zarządzania i Informatyki.</p> <p style="text-align: center;">- 3 -</p>	<p style="text-align: right;"><i>Wstęp</i></p> <hr/> <p style="text-align: center;">- 4 -</p>								
<p><b>2. Oferta</b></p> <p><b>2.1. Aspekty praktyczne ogólnych teorii</b></p> <p>Wykładowca: dr Bogdan Babacki          Zakres przedmiotu: Przedstawione zostaną zajęcia obejmują elementy zastosowań ogólnej teorii w praktyce gospodarczej i administracyjnej, ze szczególnym uwzględnieniem problemów integracji europejskiej.</p> <p style="text-align: center;">- 5 -</p>	<p style="text-align: right;"><i>Oferta</i></p> <hr/> <p>szczególnej teorii niektórych rzeczy w marketingu i zarządzaniu relacjami z klientem;</p> <ul style="list-style-type: none"> <li>- implikacje teorii rzeczy nieobojętnych uniosem dla praktyki administracji państwowej i samorządowej.</li> </ul> <p style="text-align: center;">- 6 -</p>	<p style="text-align: right;"><i>Oferta</i></p> <hr/> <p><b>2.2. Ogólna teoria wszystkiego</b></p> <p>Wykładowca: prof. Adam Abacki          Zakres przedmiotu: Przedmiot traktuje o absolutnych podstawach wiedzy o świecie jako takim, w jego wielkiej złożoności.</p> <p>Teoria ta zdaje się rozstrzygnąć większość problemów pozawspółczesnego świata dobieg globalizacji. Uzupełnienie tej</p> <p style="text-align: center;">- 7 -</p>								
<p><b>3. Zestawienie</b></p> <table border="1" style="width: 100%;"> <thead> <tr> <th>Przedmiot</th> <th>Liczba godzin</th> </tr> </thead> <tbody> <tr> <td>Aspekty praktyczne ogólnych teorii</td> <td style="text-align: center;">3</td> </tr> <tr> <td>Ogólna teoria wszystkiego</td> <td style="text-align: center;">10</td> </tr> <tr> <td>Szczególna teoria niektórych rzeczy</td> <td style="text-align: center;">5</td> </tr> </tbody> </table> <p style="text-align: center;">- 11 -</p>	Przedmiot	Liczba godzin	Aspekty praktyczne ogólnych teorii	3	Ogólna teoria wszystkiego	10	Szczególna teoria niektórych rzeczy	5	<p><b>4. Zakończenie</b></p> <p>Zaprezentowano imponującą merytorycznie ofertę programową Kursu.</p> <p>Zainteresowani udziałem w Kursie proszeni są o skontaktowanie się z Kasą<sup>*</sup> celem wniesienia opłaty.</p> <p style="text-align: center;"><small>*Kasa czynna codziennie od 8<sup>00</sup> do 16<sup>00</sup>.</small></p> <p style="text-align: center;">- 13 -</p>	<p><b>Spis treści</b></p> <p>1. Wstęp ..... 3          2. Oferta ..... 5          3. Zestawienie ..... 11          4. Zakończenie ..... 13</p> <p style="text-align: center;">- 15 -</p>
Przedmiot	Liczba godzin									
Aspekty praktyczne ogólnych teorii	3									
Ogólna teoria wszystkiego	10									
Szczególna teoria niektórych rzeczy	5									

Rysunek 3. Złożony dokument będący wynikiem przetwarzania XSQL – FOP (fragmenty)

Dla tego dokumentu specyfikacja stronicowania jest następująca:

```
<fo:layout-master-set>
<fo:simple-page-master master-name="pierwsza-prosta" margin-left="2cm" ...>
<fo:region-body margin-top="1cm" margin-bottom="1cm" column-count="1"/>
<fo:region-before region-name="naglowek-pusty" extent="0cm"/>
<fo:region-after extent=".7cm"/>
</fo:simple-page-master>

<fo:simple-page-master master-name="nieparzysta-prosta" margin-left="2cm" ...>
<fo:region-body margin-top="1cm" margin-bottom="1cm" column-count="1"/>
<fo:region-before region-name="naglowek-prawy" extent=".7cm"/>
<fo:region-after extent=".7cm"/>
</fo:simple-page-master>

<fo:simple-page-master master-name="parzysta-prosta" margin-left="1cm" ...>
<fo:region-body margin-top="1cm" margin-bottom="1cm" column-count="1"/>
<fo:region-before region-name="naglowek-lewy" extent=".7cm"/>
<fo:region-after extent=".7cm"/>
</fo:simple-page-master>
... analogicznie dla stron dwuszpaltowych

<fo:page-sequence-master master-name="prosta-sekwencja">
<fo:repeatable-page-master-alternatives>
<fo:conditional-page-master-reference master-reference="pierwsza-prosta"
page-position="first"/>
<fo:conditional-page-master-reference master-reference="nieparzysta-prosta"
page-position="rest" odd-or-even="odd"/>
<fo:conditional-page-master-reference master-reference="parzysta-prosta"
page-position="rest" odd-or-even="even"/>
</fo:repeatable-page-master-alternatives>
</fo:page-sequence-master>
```

```
<fo:page-sequence-master master-name="dwuszpaltowa-sekwencja">
  ... analogicznie dla stron dwuszpaltowych
</fo:page-sequence-master>
</fo:layout-master-set>
```

Poszczególne wzorce stron różnią się marginesami, układem nagłówków oraz liczbą szpalt sekcji *body*. Wzorce te są następnie zebrane w dwa wzorce sekwencji – dla stron jedno- i dwuszpaltowych.

## 3.2. Budowanie tekstu podstawowego

### 3.2.1. Strumień tekstu

Podstawową zawartość sekwencji stron stanowi strumień tekstu. Strumień ten jest umieszczony w obszarze *region-body* strony. W czasie formatowania następuje automatyczne łamanie tekstu na linie i strony; możliwe jest automatyczne przenoszenie wyrazów według wzorców właściwych dla danego języka narodowego; uwzględniane są też ograniczenia dotyczące tzw. szewców i bękartów.

W dokumencie XSL-FO strumień tekstu umieszcza się w elemencie *flow*, zawartym w elemencie *page-sequence*, odnoszącym się do jednego z określonych wcześniej wzorców sekwencji, np.

```
<fo:page-sequence force-page-count="even" master-reference="prosta-sekwencja">
  <fo:flow flow-name="xsl-region-body">
    ... strumień tekstu
  </fo:flow>
</fo:page-sequence>
```

### 3.2.2. Bloki

Tekst zorganizowany jest w bloki, które zawierają fragmenty rozpoczynające się od nowej linii<sup>4</sup>; najprostszym zastosowaniem bloków jest zatem wyróżnianie ustępów (akapitów).

Oprócz określania „złamania linii” bloki służą także do określania licznych cech wizualnych tekstu, jak sposób wyrównywania tekstu (*text-align*), czcionka itp. Same bloki mają także interesujące cechy wizualne: wcięcie całego bloku (*start-indent* i *end-indent*), wcięcie pierwszej linii tekstu (*text-indent*), obramowanie (*border*), odstęp zawartości od obramowania (*padding*), odstęp od poprzedniego/następnego obszaru (*space-before*, *space-after*) itd., np.<sup>5</sup>

```
<fo:block text-align="justify" text-indent="1em" space-after="6pt">
  W niniejszym dokumencie prezentuje się ...
</fo:block>
```

Bloki mogą być zagnieżdżane, co pozwala w złożony sposób sterować odstępami między częściami tekstu oraz daje możliwość wykorzystania dziedziczenia niektórych cech (patrz punkt 2.1.3).

W czasie formatowania blok może być podzielony na tzw. obszary (*areas*), jeśli zawartość bloku musi być rozmieszczona na więcej niż jednej stronie lub szpalcie.

Na stronach wieloszpaltowych poszczególne bloki mogą rozciągać się na szerokość kilku szpalt, czym steruje atrybut *span*, np.

```
<fo:block hyphenate="false" font-weight="bold" font-family="sans-serif" space-after="6pt"
  span="all"
>
  2.1. Aspekty praktyczne ogólnych teorii
</fo:block>
```

opisuje tytuł podrozdziału w części dwuszpaltowej przykładowego dokumentu.

<sup>4</sup> Słowo „linia” dla języków nieeuropejskich może oznaczać linię pionową.

<sup>5</sup> Przykłady tu przytoczone są fragmentami dokumentu, który prezentuje **Rysunek 3**.

### 3.2.3. Odstępy i łamanie

W najprostszym przypadku odstęp między blokami jest określony po prostu długością<sup>6</sup> – obowiązuje ona wtedy w sposób bezwzględny. Jednak często warto zdefiniować odstęp jako wartość złożoną, określając osobno odległość minimalną, maksymalną i optymalną (np. `space-after.optimum`). Podanie takiej „elastycznej” odległości umożliwi poprawne formatowanie, gdy pogodzone być muszą sprzeczne wymagania, np. jeśli po bloku poprzednim żądano innego odstępu, niż przed blokiem następnym. Można też określić, czy odstęp przed/po obszarze ma być zachowywany, jeśli obszar ten znajdzie się na początku/końcu szpalty.

Dla bloków można określić obowiązkowe łamanie przed lub po bloku; może ono dotyczyć wymuszenia nowej strony (ew. parzystej lub nieparzystej) lub szpalty; służą do tego atrybuty `break-before` i `break-after`. Można także zakazać lub ograniczyć łamanie zawartości bloku lub między blokami za pomocą atrybutów `keep-together`, `keep-with-previous`, `keep-with-next`.

Wszystko to, wraz z możliwością zagnieżdżania bloków, daje duże możliwości panowania nad sposobem łamania tekstu.

### 3.2.4. Elementy *inline*

Zawartość bloku stanowi strumień tekstu łamany automatycznie na linie. Jeśli w strumieniu tym trzeba jednak wyróżnić pewien fragment np. w celu zmiany jego cech wizualnych lub wymuszenia utrzymywania w jednej linii, to fragment ten umieszcza się w elemencie `inline`, np.

```
kasa czynna codziennie od 8<fo:inline vertical-align="super">00</fo:inline>
do 16<fo:inline vertical-align="super">00</fo:inline>.
```

Specyficznym elementem stosowanym w liniach jest `leader`, służący do rysowania poziomych linii o wybranym wzorze (np. ciągłych lub kropkowanych) i długości podanej albo automatycznie wypełniającej dostępne miejsce w linii (przykłady podano w punktach 3.4.1 i 3.4.2).

### 3.2.5. Listy

Często potrzebną formą jest lista, dlatego przewidziano do jej realizacji specjalny typ bloku. Pozwala on osobno określić etykietę, a osobno zawartość pozycji listy, rozmieszczając je obok siebie (z możliwością określenia wcięć), np.

```
<fo:list-block text-align="start" start-indent="1em" space-before="6pt" space-after="6pt">
  <fo:list-item space-before="3pt" space-after="3pt">
    <fo:list-item-label><fo:block>-</fo:block></fo:list-item-label>
    <fo:list-item-body start-indent="2em">
      <fo:block>zastosowania ogólnej teorii wszystkiego w zarządzaniu;</fo:block>
    </fo:list-item-body>
  </fo:list-item>
</fo:list-block>
```

### 3.2.6. Tabele

Do tworzenia tabel przewidziano specjalny zestaw elementów, o działaniu podobnym do elementów tabel w HTML.<sup>7</sup> Tabele składają się z wierszy, a te zawierają komórki.

Kolumny tabeli mogą mieć szerokość wyznaczaną automatycznie, podaną w sposób bezwzględny, w procentach szerokości całej tabeli lub przez podanie proporcji między szerokościami poszczególnych kolumn, np.

<sup>6</sup> Długości (*lengths*) podaje się w jednostkach długości, którymi mogą być np. punkty, cale, centymetry, jednostki em (długość litery „m” w bieżącej czcionce).

<sup>7</sup> Jednak inaczej niż w HTML, tabele XSL-FO stosuje się przede wszystkim do rysowania „prawdziwych” tabelek, a nie do różnych sztuczek.

```

<fo:table width="80%">
  <fo:table-column column-width="proportional-column-width(10)"/>
  <fo:table-column column-width="proportional-column-width(3)"/>

  <fo:table-body>
    <fo:table-row>
      <fo:table-cell padding="3pt" border="solid 1px black" display-align="center">
        <fo:block text-align="center">Przedmiot</fo:block>
      </fo:table-cell>
      <fo:table-cell padding="3pt" border="solid 1px black" display-align="center">
        <fo:block hyphenate="false" text-align="center">Liczba godzin</fo:block>
      </fo:table-cell>
    </fo:table-row>
  </fo:table-body>
</fo:table>

```

Komórki – podobnie jak w HTML – mogą rozciągać się na kilka wierszy lub kolumn. Możliwe jest precyzyjne sterowanie wyglądem obramowań. Tabela może być zaopatrzona w nagłówki oraz stopkę; jeśli tabela nie zmieści się na jednej stronie, komórki nagłówka i stopki będą powtórzone na kolejnych stronach.

### 3.3. Cechy wizualne tekstu

Jak można się domyślać, język XSL-FO zawiera środki do bardzo rozbudowanego sterowania cechami wizualnymi tekstu, takimi jak czcionka, wielkość, kolor itp. Wybór ważniejszych cech wizualnych i atrybutów nimi sterujących przedstawia tabela.

Cechy	Atrybuty FO
Czcionki	font-family, font-size, font-weight, font-stretch, font-style (np. <i>italic</i> ), font-variant (np. <i>small-caps</i> )
Inne cechy pisma	text-decoration (np. <u>underline</u> ), baseline-shift, text-transform (np. <i>uppercase</i> ), text-shadow
Kolory	color, background-color
Tło	background-image, background-repeat

Na przykład tytuł na pierwszej stronie dokumentu (**Rysunek 3**) utworzono tak:

```

<fo:block
  space-before="1cm" space-after="1cm"
  padding="6pt" text-align="center"
  color="white" background-color="blue"
  font-family="sans-serif" font-size="24pt" font-weight="bold"
>
  oferta programowa
</fo:block>

```

### 3.4. Pomocnicze elementy tekstu

#### 3.4.1. Nagłówki i stopki

W okalających regionach strony (*region-before*, *region-after*) umieścić można elementy tzw. żywej paginy: nagłówki i stopki. W przykładowym dokumencie zdefiniowano je tak:

```

<fo:page-sequence force-page-count="even" master-reference="prosta-sekwencja">
  <fo:static-content flow-name="naglowek-lewy">
    <fo:block font-size="9pt" font-style="italic" line-height="10pt" text-align="left">
      wstęp
    </fo:block>
    <fo:block line-height="3pt"><fo:leader leader-pattern="rule"/></fo:block>
  </fo:static-content>

  <fo:static-content flow-name="naglowek-prawy">
    <fo:block font-size="9pt" font-style="italic" line-height="10pt" text-align="right">
      wstęp
    </fo:block>
    <fo:block line-height="3pt"><fo:leader leader-pattern="rule"/></fo:block>
  </fo:static-content>

```

```

</fo:static-content>
<fo:static-content flow-name="xsl-region-after">8
  <fo:block font-size="9pt" text-align="center">
    - <fo:page-number/> -
  </fo:block>
</fo:static-content>
</fo:page-sequence>

```

W przykładzie tym użyto automatycznego numerowania stron za pomocą elementu `page-number`.

Jeśli tekst umieszczony w nagłówku lub stopce ma się zmieniać w ramach jednej sekwencji stron, można skorzystać z tzw. markerów. Element `marker`, umieszczony wewnątrz bloku, pozwala zdefiniować pewien zmienny tekst do wykorzystania w żywej paginie. Element `retrieve-marker`, umieszczony wewnątrz `static-content`, generuje tekst markera na wszystkich stronach, na których znajdzie się zawartość bloku w którym ów marker ustawiono.

### 3.4.2. Odnośniki

Ważną zaletą formatowania za pomocą XSL-FO jest możliwość tworzenia odnośników do numerów stron. Dzięki temu można tworzyć spisy treści, indeksy itp.

Miejsce, na które wskazywać ma odnośnik, trzeba oznaczyć identyfikatorem `id`, np.

```
<fo:block id="wstęp"...>1. wstęp</fo:block>
```

i wówczas w spisie treści odpowiedni numer strony można uzyskać tak:

```

<fo:block wrap-option="no-wrap" end-indent="2em">
  1. wstęp
  <fo:leader leader-pattern="dots"/>
  <fo:page-number-citation ref-id="wstęp"/>
</fo:block>

```

### 3.4.3. Obiekty ruchome

Obiekty ruchome (*out-of-lines*) w XSL-FO to `footnote` (przypis) oraz `float` – element przeznaczony do automatycznego rozmieszczania dużych obiektów, jak rysunki czy tabele. Obiekt ruchomy nie jest umieszczany w miejscu wynikającym z kolejności w strumieniu tekstu, ale w miejscu odpowiednim ze względu na jego rolę i/lub rozmiar. Dla przypisów jest to koniec strony, dla obiektów `float` zwykle początek strony lub osobna strona.

W przykładowym dokumencie użyto przypisu:

```

<fo:footnote>
  <fo:inline font-size="9pt" vertical-align="super">*</fo:inline>
  <fo:footnote-body>
    <fo:block space-after="3pt" end-indent="7cm"><fo:leader leader-
      pattern="rule"/></fo:block>
    <fo:block font-size="10pt">
      <fo:inline font-size="9pt" vertical-align="super">*</fo:inline>
      Kasa czynna codziennie ...
    </fo:block>
  </fo:footnote-body>
</fo:footnote>

```

Zawartość tekstowa elementu `footnote` określa oznaczenie przypisu w głównym tekście, zaś zawartość `footnote-body` – właściwy obiekt ruchomy.

Obiekty ruchome dziedziczą cechy z kontekstu, w którym umieszczone są znaczniki je definiujące.

---

<sup>8</sup> `xsl-region-after` jest domyślną nazwą dolnego obszaru strony.

## 3.5. Inne elementy

### 3.5.1. Grafika

XSL-FO pozwala umieszczać w składanym dokumencie obrazy graficzne. Akceptowane formaty zależą od implementacji; zwykle są to typowe formaty map bitowych stosowane w Internecie, niekiedy także XML-owy format wektorowy SVG.

Elementy graficzne mogą stanowić zawartość linii, osobne bloki lub obiekty ruchome (float). Obrazy można skalować do żądanego rozmiaru, np. znak graficzny na stronie tytułowej przykładowego dokumentu uzyskano tak:

```
<fo:block text-align="center">  
  <fo:external-graphic src="url(logo.gif)" height="2.5cm" scaling="uniform"/>  
</fo:block>
```

### 3.5.2. Profesjonalna typografia

Standard przewiduje także szereg elementów przeznaczonych do użycia w profesjonalnej typografii, np. profile kolorów, precyzyjne sterowanie wymiarami i położeniem czcionek, wysokością linii i jej części itp.

Choć obecne implementacje nie mają jeszcze wszystkich możliwości wymaganych od profesjonalnego pakietu DTP, sam standard FO zawiera niezbędne dla tego typu narzędzi elementy.

### 3.5.3. Cechy interaktywne, dźwiękowe itp.

Budując standard XSL-FO przewidywano różne możliwe media wyjściowe: nie tylko wydruk, ale także prezentację interaktywną, a nawet dźwiękową. Standard zawiera zatem także elementy służące do sterowania dźwiękiem (np. barwą lub natężeniem głosu, pauzami) oraz typowo interaktywne, np. do definiowania łączników.

Co prawda na razie istniejące implementacje skupiają się na tych częściach standardu, które są związane z formatowaniem dokumentów drukowanych (zapewne dlatego, że tu najbardziej odczuwa się niedostatki HTML), ale należy się spodziewać, że w przyszłości powstaną implementacje służące do wykorzystania interaktywnego, np. w przeglądarkach.

## 4. Narzędzia

Ostatnio pojawiło się kilka znaczących narzędzi służących do formatowania dokumentów z użyciem XSL-FO. Narzędzia te służą do tworzenia publikacji przeznaczonych do druku, a działają w sposób wsadowy. Jak się wydaje droga do formaterów FO wbudowanych w przeglądarki internetowe jest jeszcze dość daleka, przede wszystkim ze względu na niedostateczną szybkość działania obecnych narzędzi.

Żadne z istniejących narzędzi nie implementuje całości standardu, ale zwykle implementowane są wszystkie elementy niezbędne do osiągnięcia podstawowego poziomu zgodności określonego w specyfikacji oraz pewien podzbiór elementów poziomu rozszerzonego.

Wśród produktów komercyjnych najbardziej znanym narzędziem jest jak się wydaje XEP firmy RenderX, zaś wśród produktów dostępnych za darmo największe znaczenie ma projekt Apache FOP – on też omówiony zostanie dokładniej, gdyż może być wygodnie wykorzystywany z narzędziami firmy Oracle.

## 4.1. FOP – XSL-FO w Apache

FOP (*Formatting Objects Processor*) jest produktem tworzonym w ramach projektu typu *open source* Apache XML Project. Ze względu na dostępność za darmo i na renomę, jaką cieszą się inne produkty Apache, produkt ten ma szansę wejść do powszechnego użycia.

Obecna, ciągle jeszcze dość wczesna wersja FOP ma już dość znaczące możliwości. FOP pozwala na podstawie dokumentów XSL-FO generować pliki wyjściowe w formacie PDF, w PostScriptcie, w SVG oraz w formacie drukarkowym PCL, planowane jest generowanie plików RTF; można też wyświetlać wynik formatowania wprost na ekranie.

FOP jest w całości napisany w Javie, co powoduje, iż jest całkowicie przenośny. Klasy realizujące FOP mogą być włączane do innych aplikacji. Istnieje także wersja w postaci serwletu, przeznaczona do działania na serwerach aplikacyjnych.

FOP służy do formatowania publikacji przeznaczonych do druku. Wprawdzie nie zrealizowano całości standardu, a niektóre braki są istotne (np. brak obiektów ruchomych typu `float`), inne jednak można łatwo obejść (np. nie ma możliwości użycia `text-align="outside"`, trzeba więc osobno definiować prawe i lewe nagłówki). FOP zintegrowano z procesorem SVG Apache Batik, co umożliwia włączanie grafiki wektorowej SVG do formatowanych dokumentów.

Z punktu widzenia rodzimego użytkownika istotne są możliwości związane z obsługą języka polskiego:

- przenoszenie wyrazów: w typowej dystrybucji FOP umieszczono reguły przenoszenia dla języka polskiego (stworzone na podstawie cieszących się świetną opinią reguł dla `TEX-a`);
- możliwość wykorzystania polskich czcionek w plikach PDF: do generowanych plików włączać można czcionki postscriptowe oraz TTF, wymaga to jedynie wygenerowania plików z metrykami (odpowiednie narzędzie dołączono) oraz prostej modyfikacji pliku konfiguracyjnego.

W niniejszym opracowaniu opisano przede wszystkim te elementy standardu XSL-FO, które zostały zaimplementowane w FOP; także do wykonania przykładu użyto właśnie FOP.

## 4.2. Użycie FOP z Oracle XDK

Najnowsze wersje Oracle XDK umożliwiają integrację FOP z przetwarzaniem *XSQL Server Pages* (patrz [4]). Strony XML generowane przez *XSQL Servlet* mogą być na bieżąco formatowane do postaci PDF za pomocą FOP. W tym celu należy:

- dołączyć biblioteki FOP do ścieżki `CLASSPATH`;
- dołączyć definicję FOP do sekcji `<serializerdefs>` w pliku `XSQLConfig.xml`;
- w instrukcji sterującej `xml-stylesheet` strony XSQL dodać odpowiedni atrybut:

```
<?xml-stylesheet type="text/xsl" href="..." serializer="FOP"?>
```

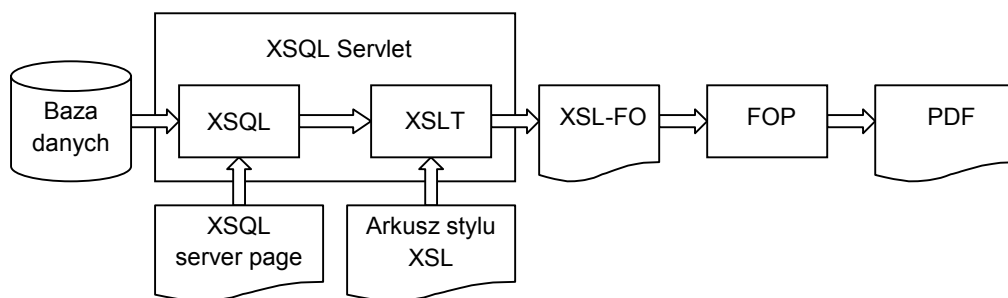
## 5. Wykorzystanie XSL-FO w systemach z bazami danych

Możliwości tworzenia złożonych publikacji za pomocą obiektów formatujących XSL-FO są bardzo zaawansowane, powstaje jednak pytanie w jakich okolicznościach można i warto je stosować w systemach z bazami danych.

Nie ma wątpliwości, że do tworzenia typowych zestawień i raportów z danych wygodniej jest używać standardowych i sprawdzonych narzędzi do raportowania, np. Oracle Reports<sup>9</sup>. Możliwości tych standardowych narzędzi okazują się jednak niewystarczające w przypadkach, gdy wymagane jest stworzenie profesjonalnie złożonej publikacji o charakterze książkowym, z rozdziałami, żywą paginą, spisem treści, indeksami itp. W takich sytuacjach dotychczas radzono sobie wyprawdzając dane z bazy do plików, które następnie obrabiano narzędziami typu DTP (np. T<sub>E</sub>X-em). Jest to jednak kłopotliwe i nie umożliwia publikowania danych *on-line*, np. w sieci Web. Zastosowanie obiektów formatujących pozwoli uzyskać *on-line* publikacje o jakości zbliżonej do efektów DTP.

Typowe narzędzia raportujące nie mogą także być użyte, gdy tworzona publikacja ma zawierać mieszaninę danych relacyjnych pobieranych z bazy oraz już sformatowanych tekstów, także zapisanych w bazie danych. Takie teksty przechowywać możemy w bazie danych w postaci XML (lub XHTML), elementy formatowania (akapity, nagłówki, listy itp.) oznaczając znacznikami. Ale oczywiście włączenie takich tekstów w typowe raporty, np. Oracle Reports, wymaga usunięcia znaczników, co powoduje utratę formatowania. Zastosowanie XSL i obiektów formatujących pozwala tymczasem tworzyć złożone publikacje na podstawie dowolnej kombinacji danych relacyjnych, relacyjno-obiektowych i tekstów w XML.

**Rysunek 4** przedstawia jeden z możliwych sposobów tworzenia tego typu publikacji za pomocą rozwiązania firmy Oracle o nazwie *XSQL Server Pages* [4, 7]. Dane oraz teksty w XML są pobierane z bazy danych i umieszczane w przygotowanym szablonie strony XML (*server page*), który następnie jest transformowany za pomocą arkusza stylistycznego w XSL na drzewo obiektów formatujących XSL-FO, ono zaś jest przekształcane przez FOP na plik PDF. Ponieważ zarówno XSQL jak i FOP są dostępne w postaci serwetów i zintegrowane ze sobą (jak to opisano w punkcie 4.2), całe przetwarzanie może być wykonywane przez serwer aplikacyjny jako skutek żądania HTTP w sieci Web, a wynikowy plik PDF może być zwrócony *on-line* jako odpowiedź na owo żądanie.



Rysunek 4. Przetwarzanie za pomocą XSQL i FOP

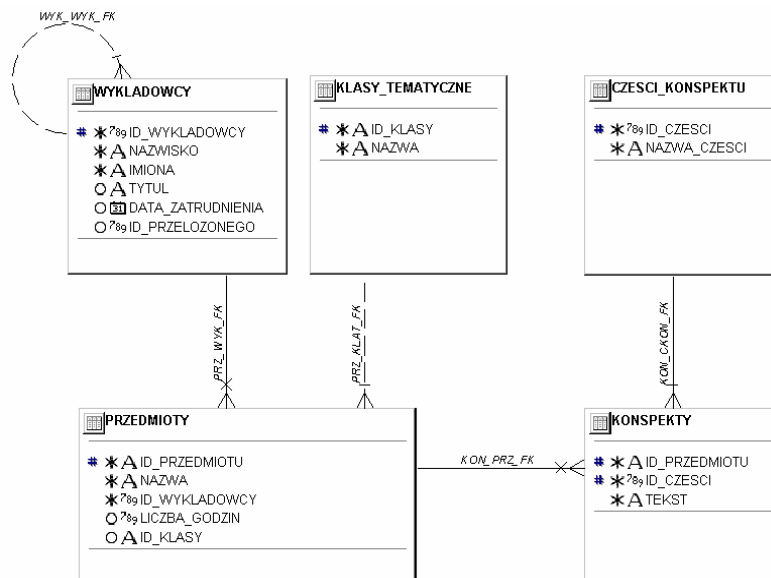
### Przykład 3

W systemie informacyjnym opisującym zajęcia prowadzone na wyższej uczelni zapisuje się pewne proste atrybuty poszczególnych przedmiotów (symbol, nazwę, liczbę godzin itp.) oraz konspekty przedmiotów, będące zredagowanym tekstem podzielonym na stałe części. Teksty części konspektów przechowuje się w bazie danych w XML, ze znacznikami reprezentującymi elementy redakcji tekstu: podtytuły, listy, wyróżnienia itp. (por. [2]).

<sup>9</sup> Chyba że raporty mają być na bieżąco publikowane w Internecie, a nie chcemy ponosić kosztów nieograniczonej licencji na Reports Server.

Ofertę dydaktyczną uczelni publikuje się w postaci książkowej. Części stałe tej publikacji są zapisane w postaci plików XML, zaś opisy i konspekty przedmiotów powinny być na bieżąco pobrane z bazy danych.

Schemat przykładowej struktury danych przedstawia **Rysunek 5**.



Rysunek 5. Schemat przykładowej struktury danych

Przykładowy dokument (**Rysunek 3**) został stworzony na podstawie zawartości bazy danych o takiej strukturze za pomocą *XSQL Server Pages* oraz FOP. Zastosowano dwie współpracujące strony XSQL:

- W pierwszej wykonywane jest zapytanie z operatorem `CURSOR`, które zwraca złożoną strukturę XML, bardziej dostosowaną do potrzeb od struktury kanonicznej XSU (patrz [4]):

```
<xsql:query xmlns:xsql="urn:oracle-xsql"
connection="kowalski"
rowset-element="PRZEDMIOTY"
row-element="PRZEDMIOTY_ROW"
>
SELECT
  p.ID_PRZEDMIOTU AS "@ID_PRZEDMIOTU", p.NAZWA, p.LICZBA_GODZIN,
  CURSOR(
    SELECT w.ID_WYKLADOWCY as "@ID_WYKLADOWCY", w.NAZWISKO, w.IMIONA, w.TYTUL
    FROM WYKLADOWCY w WHERE w.ID_WYKLADOWCY = p.ID_WYKLADOWCY
  ) AS wyklawowca,
  CURSOR(
    SELECT k.ID_CZESCI as "@ID_CZESCI", c.NAZWA_CZESCI, k.TEKST
    FROM CZESCI_KONSPEKTU c, KONSPEKTY k
    WHERE k.ID_PRZEDMIOTU = p.ID_PRZEDMIOTU and c.ID_CZESCI = k.ID_CZESCI
    ORDER BY k.ID_CZESCI
  ) AS konspekt
FROM PRZEDMIOTY p
ORDER BY p.ID_PRZEDMIOTU
</xsql:query>
```

Arkuszy stylistyczny tej strony powoduje jedynie „przepuszczenie” znaków `< i >` okalających pobrane z kolumny `TEKST` znaczniki XML bez ich konwersji na encje:

```
<xsl:template match="@*|node()">
  <xsl:copy>
    <xsl:apply-templates select="@*|node()"/>
  </xsl:copy>
</xsl:template>

<xsl:template match="text()">
  <xsl:value-of select="." disable-output-escaping="yes"/>
</xsl:template>
```

- Druga strona XSQL łączy stałe części publikacji:

```
<xsql:include-xml href="wstep.xml"/>
```

oraz wyniki działania pierwszej strony XSQL:

```
<xsql:include-xsql href="przedmioty.xsql" reparse="yes"/>
```

Arkuszy stylistyczny drugiej strony XSQL dokonuje właściwego formatowania przez zamianę danych na obiekty XSL-FO.

## 6. Podsumowanie

Obiekty formatujące XSL-FO, w połączeniu z transformacjami XSLT, umożliwiają tworzenie złożonych publikacji spełniających wymogi profesjonalnej typografii na podstawie dokumentów XML.

Mogą być one wykorzystane nie tylko do publikowania typowych tekstów zapisanych w XML lub XHTML, ale także do prezentowania danych przechowywanych w bazach danych. Dzięki dużym możliwościom XSL-FO, związanym ze stronicowaniem, tworzeniem odnośników itp., możliwe jest generowanie na podstawie zawartości baz danych kompletnych profesjonalnie przygotowanych publikacji drukowanych lub prezentowanych *on-line* w sieci Web. Pokazano przykład tego typu zastosowania, z wykorzystaniem technologii Oracle i implementacji obiektów formatujących Apache FOP.

Po ukazaniu się całości standardu XSL i pojawieniu się pierwszych ważnych implementacji XSL-FO, obiekty formatujące stały się dostępnym i wartym uwagi narzędziem. Należy mieć nadzieję, że implementacje obiektów formatujących będą się rozwijać, a w przyszłości staną się także częścią typowych przeglądarek internetowych.

## Bibliografia

1. Extensible Stylesheet Language (XSL), Version 1.0. W3C Recommendation. <http://www.w3.org/TR/2001/REC-xsl-20011015/>
2. Traczyk T.: Język XML w aplikacjach z bazami danych – po roku, Materiały V Konferencji Deweloperów i Użytkowników Oracle „Integracja danych i systemów informatycznych”, Zakopane 1999.
3. Traczyk T.: Język XSL, Materiały VI Konferencji Deweloperów i Użytkowników Oracle „Systemy informatyczne w dobie Internetu”, Zakopane 2000.
4. Traczyk T.: Czy już warto używać XML? Tutorial, Materiały I Seminarium PLOUG, Warszawa 2001.
5. Pawson D.: An introduction to XSL Formatting Objects. <http://www.dpawson.co.uk/xsl/sect3/bk/>
6. FOP (Formatting Objects Processor). <http://xml.apache.org/fop/>
7. Oracle XML Developer's Kits. <http://technet.oracle.com/tech/xml/>