XML

Praktyczne wprowadzenie do standardu XSL

Tomasz Przechlewski

Streszczenie

Język formatowania XSL składa się z trzech rekomendacji: XSLT, XPath oraz FO opisujących w jaki sposób zamienić dokument XML z postaci strukturalnej do postaci prezentacyjnej. Dwie pierwsze są związane z przekształcaniem dokumentów XML, rekomendacja FO jest specyfikacją standardowych obiektów formatujących, tj. obiektów definiujących semantykę samego formatowania.

Artykuł zawiera wprowadzenie do standardu XSL-FO oraz przedstawia rozwijaną w ramach projektu Apache aplikację FOP umożliwiającą zamianę pliku XML na dokument w formacie PDF. Pozostałe części języka XSL nie są opisane.

1. Wprowadzenie do XSL-FO

XSL-FO jest skomplikowanym językiem o dużych możliwościach, zawierającym ponad 50 różnych „obiektów formatujących”, począwszy od najprostszych, takich jak prostokątne bloki tekstu, poprzez wyliczenia, tabele i odsyłacze. Obiekty te można formatować wykorzystując przeszło 200 różnych właściwości (*properties*), takich jak: kroje, odmiany i wielkości pisma, odstępy, kolory itp. W tym dokumencie przedstawione jest absolutne minimum informacji na temat standardu XSL-FO.

Dokument XSL-FO to *specyficzny* dokument XML, którego elementy i atrybuty nie opisują struktury dokumenty ale wyłącznie jego postać graficzną. Takie podejście różni XSL-FO od arkuszy stylów typu CSS, w których polecenia formatujące są (zwykle) oddzielone od dokumentu, którego dotyczą. XSL-FO pod tym względem przypomina raczej tradycyjne systemy przetwarzania tekstu takie jak TEX, czy troff. Dokument XSL-FO winien być generowany automatycznie na podstawie danych zapisanych w innym formacie (np. „strukturalnego” dokumentu XML lub informacji z relacyjnej bazy danych), a po wykorzystaniu usunięty. Format XSL-FO nie jest przeznaczony ani do przechowywania informacji, ani do jej wymiany.

W typowym scenariuszu dokument XSL-FO jest tworzony za pomocą odpowiedniego arkusza stylu XSLT (por. rys. 1). W poniższym przykładowym fragmencie arkusza XSLT element `<para>` wyjściowego dokumentu jest transformowany do elementu `<fo:block>`:

```
<template match="para">
 <fo:block space-after.optimum="0pt" hyphenate="true" >
<xsl:apply-templates/> </fo:block>
</xsl:template>
```

W tym artykule pominięto zagadnienie konwersji dokumentów XML za pomocą arkusza XSLT koncentrując się wyłącznie na standardzie XSL-FO. Doskonały wstęp do standardów XSLT i XPath można znaleźć w [6].

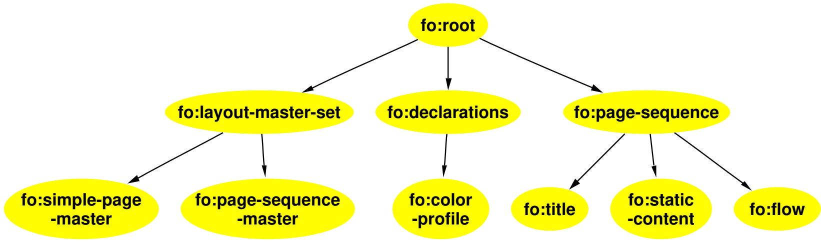
Rysunek 1: Typowy scenariusz przetwarzania dokumentu XSL-FO

Cały dokument XSL-FO zawarty jest wewnątrz elementu `<fo:root>`. Element ten zawiera (w podanej niżej kolejności):

- dokładnie jeden element `<fo:layout-master-set>` zawierający szablony określające wygląd poszczególnych stron oraz sekwencji stron (te ostatnie są opcjonalne, ale typowo są definiowane);
- zero lub więcej elementów `<fo:declarations>`;

- jeden lub więcej elementów <fo:page-sequance> zawierających treść formatowanego dokumentu wraz z opisem jego sformatowania i podziału na strony.

Treść formatowanego dokumentu znajduje się wyłącznie wewnątrz elementów <fo:page-sequance>, podczas gdy element <fo:layout-master-set> zawiera definicje szablonów. Strukturę dokumentu przedstawiono schematycznie na rysunku 2.



Rysunek 2: Ogólna struktura dokumentu XSL-FO

Ponieważ element <root> *musi* być w przypadku dokumentu XSL-FO związany z przestrzenią nazw http://www.w3.org/1999/XSL/Format, oznacza to, że przykładowy początek szablonu może wyglądać następująco:

```
<?xml version="1.0" encoding="iso-8859-2" ?>
 <fo:root xmlns:fo='http://www.w3.org/1999/XSL/Format'>
```

2. Określenie ogólnego układu graficznego dokumentu

Element <fo:layout-master-set> zawiera *co najmniej* jeden element <fo:simple-page-master> lub <fo:page-sequence-master>.

Element <fo:simple-page-master> (dalej określany skrótowo jako SPM) definiuje układ graficzny pojedynczej strony. Strona może posiadać do pięciu różnych obszarów definiowanych za pomocą elementów: <fo:region-body>, <fo:region-before>, <fo:region-after>, <fo:region-start> oraz <fo:region-end>. W przypadku konwencji używanych w drukarstwie europejskim obszary te odpowiadają: kolumnie głównej tekstu, paginie górnej, paginie dolnej, marginaliom wewnętrznym i marginaliom zewnętrznym. Szczegóły układu graficznego każdego obszaru (tj. wymiary, marginesy, ramki, kolory) są określone za pomocą wartości odpowiednich atrybutów elementu SPM (więcej na ten temat dalej w tekście).

Specyfikacja obszaru <fo:region-body> jest obowiązkowa. W wersji 1.0 standardu element <fo:simple-page-master> stanowi *jedyny* sposób definiowania układu graficznego stron. Przyszłe wersje standardu mogą wprowadzić bardziej skomplikowane układy graficzne stron.

Element <fo:page-sequence-master> (PSM) określa porządek w jakim poszczególne strony, zdefiniowane za pomocą SPM, będą zapełniane. Porządek ten określają elementy-dzieci elementu <fo:page-sequence-master>:

<fo:single-page-master-reference> powoduje sformatowanie dokładnie jednej strony o układzie graficznym określonym wartością atrybutu master-reference. Element ten jest typowo wykorzystywany do definiowania stron tytułowych.

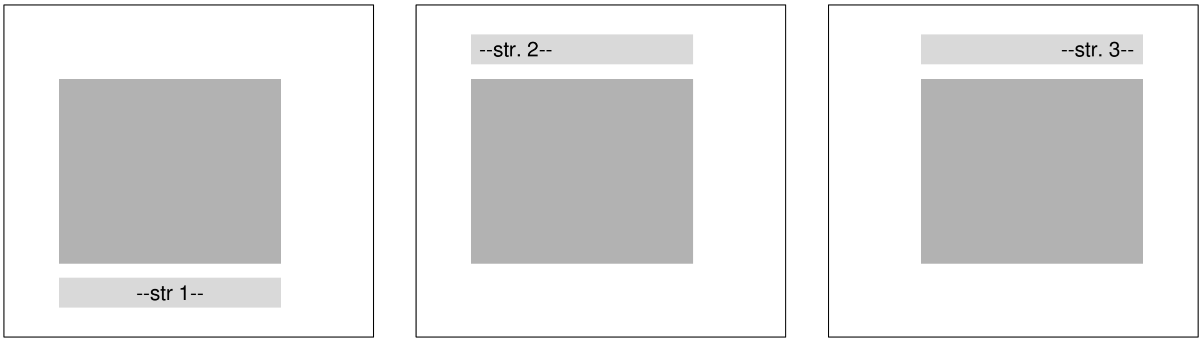
<fo:repeatable-page-master-reference> deklaracja formatowania sekwencji stron przy wykorzystaniu identycznego układu graficznego (tj. tego samego SPM). Liczba stron może być określona za pomocą wartości atrybutu maximum-repeats. Brak atrybutu lub wartość domyślna równa no-limit spowoduje wygenerowanie tylu stron ile będzie niezbędne do wydrukowania całej zawartości bieżącego elementu <fo:flow>.

<fo:repeatable-page-master-alternatives> Element <fo:repeatable-page-master-alternatives> jest deklaracją formatowania sekwencji stron o *różnych* układach graficznych. Element ten nie posiada atrybutu master-reference, ponieważ nie wybiera odpowiedniej strony bezpośrednio, ale za pomocą elementów-dzieci <conditional-page-master-reference> (w skrócie CPMR).

Każdy CPMR określa warunek, który jeżeli jest spełniony, powoduje wybranie odpowiedniego SPM. Warunek może określać kolejność strony w sekwencji stron, numer strony, parzystość-nieparzystość numeru strony czy też sprawdzenie czy strona jest pusta, czy też nie. Warunki są określane za pomocą odpowiednich wartości atrybutów `page-position`, `odd-or-even`, `blank-or-not-blank`:

- Atrybut `page-position` o wartościach `first last`, `rest`, `any` pozwala na wybranie odpowiednio: pierwszej, ostatniej, wszystkich, za wyjątkiem pierwszej i ostatniej oraz każdej strony.
- Atrybut `odd-or-even` o wartościach `odd`, `even`, `any` pozwala na wybranie odpowiednio: nieparzystej, parzystej oraz każdej strony.
- Atrybut `blank-or-not-blank` o wartościach `blank`, `not-blank`, `any` pozwala na wybranie odpowiednio: pustej, niepustej oraz każdej strony.

Najczęściej spotykany układ graficzny dokumentu składa się z kolumn trzech rodzajów: tytułowej, lewej i prawej (rys 3). Oto prosty przykład szkieletu szablonu określającego taki układ (symbol `@@@...` oznacza pominięte – na tym etapie opisu – definicje atrybutów określających wymiary, marginesy, ramki, kolory itp. elementy układu graficznego):



Rysunek 3: Szkielet układu graficznego stronic: tytułowej, parzystej i nieparzystej

```
<fo:layout-master-set>
<!-- definiujemy trzy makiety: tytułowa, lewa i prawa -->
  <fo:simple-page-master master-name="tytulowa" @@@...>
    <fo:region-body @@@... />
    <fo:region-after region-name="tytulowa-p.dolna" @@@... />
    <fo:region-before region-name="tytulowa-p.gorna" @@@... />
  </fo:simple-page-master>

<!-- definiujemy poszczególne fragmenty na stronach -->
  <fo:simple-page-master master-name="lewa" @@@... >
    <fo:region-body @@@... />
    <fo:region-after region-name="lewa-p.dolna" @@@... />
    <fo:region-before region-name="lewa-p.gorna" @@@... />
  </fo:simple-page-master>

  <fo:simple-page-master master-name="prawa" @@@... >
    <fo:region-body @@@... />
    <fo:region-after region-name="prawa-p.dolna" @@@... />
    <fo:region-before region-name="prawa-p.gorna" @@@... />
  </fo:simple-page-master>

<!-- określamy układ dwustronny: -->
  <fo:page-sequence-master master-name="d-stronny">
    <fo:repeatable-page-master-alternatives>
      <fo:conditional-page-master-reference master-reference="tytulowa" page-position="first" />
```

```
      <fo:conditional-page-master-reference master-reference="prawa" odd-or-even="odd" />
      <fo:conditional-page-master-reference master-reference="lewa" odd-or-even="even" />
    </fo:repeatable-page-master-alternatives>
  </fo:page-sequence-master>

<!-- określamy układ jednostronny (z wyróżnioną str. tyt.): -->
  <fo:page-sequence-master master-name="j-stronny">
    <fo:repeatable-page-master-alternatives>
      <fo:conditional-page-master-reference master-reference="tytulowa" page-position="first" />
      <fo:conditional-page-master-reference master-reference="prawa" />
    </fo:repeatable-page-master-alternatives>
  </fo:page-sequence-master>
</fo:layout-master-set>
```

Treść formatowanego dokumentu znajduje się wewnątrz elementu `<fo:page-sequence>`. Układ graficzny kolumny tekstu jest określony poprzez skojarzenie tegoż elementu z odpowiednim SPM lub PSM za pomocą identycznej wartości atrybutów `master-reference`/`master-name`.

Zawartością `<fo:page-sequence>` jest opcjonalny element `<fo:title>`, zero lub więcej elementów `<fo:static-content>` oraz dokładnie jeden element `<fo:flow>`.

Elementy `<fo:static-content>` służą typowo do określania zawartości pagin i marginaliów. Wartość atrybutu `flow-name`[1] identyfikuje obszar, w którym zostanie umieszczony dany element `<fo:static-content>`.

Ilustracją tego, co powiedziano w poprzednich trzech akapitach, może być następujący fragment szablonu (symbol `###...` oznacza pominiętą – w imię zwięzłości opisu – zawartość elementów `<static-content>` oraz `<flow>`):

```
 <fo:page-sequence master-reference="d-stronny">
  <fo:static-content flow-name="prawa-p.dolna"> ###...  </fo:static-content>
  <fo:static-content flow-name="lewa-p.dolna"> ###... </fo:static-content>
  <fo:static-content flow-name="prawa-p.gorna"> ###... </fo:static-content>
  <fo:static-content flow-name="lewa-p.gorna"> ###...  </fo:static-content>
  <fo:static-content flow-name="tytulowa-p.dolna" > ###... </fo:static-content>
  <fo:static-content flow-name="tytulowa-p.gorna"> ###... </fo:static-content>

  <fo:flow flow-name="xsl-region-body"> ###... <fo:flow>
 </fo:page-sequence>
</fo:root>
```

3. Określenie szczegółów układu graficznego kolumn

W tym punkcie omówimy w jaki sposób określić wygląd poszczególnych obszarów stronicy zdefiniowanych w punkcie poprzednim. Wielkość strony jest określona za pomocą atrybutów `page-height` oraz `page-width` elementu `<fo:simple-page-master>`. Wymiary kolumny tekstu są określone pośrednio jako różnica pomiędzy wymiarami strony a wielkością odpowiednich marginesów (określonych za pomocą atrybutów: `margin-left`, `margin-right`, `margin-top`, `margin-bottom`). Przykładowo:

```
<fo:simple-page-master master-name="tytulowa" page-width='210mm' page-height='297mm' />
<fo:region-body margin-left='30mm' margin-right='20mm' margin-top='20mm' margin-bottom='30mm'/>
```

W przypadku atrybutu `margin` (i w wielu innych miejscach) możliwe jest przypisanie wszystkim marginesom, tj. `margin-top`, `margin-bottom`, `margin-left` i `margin-right` tej samej wartości za pomocą *skrótu* `margin='1in'`. Zapis `margin='1in 1.5in 1in 0.5in'` jest przykładem innego rodzaju skrótu, pozwalającego przypisać różne wartości marginesów.

Uwaga: nie wszystkie procesory XSL-FO poprawnie interpretują skróty, lepiej z nich nie korzystać.

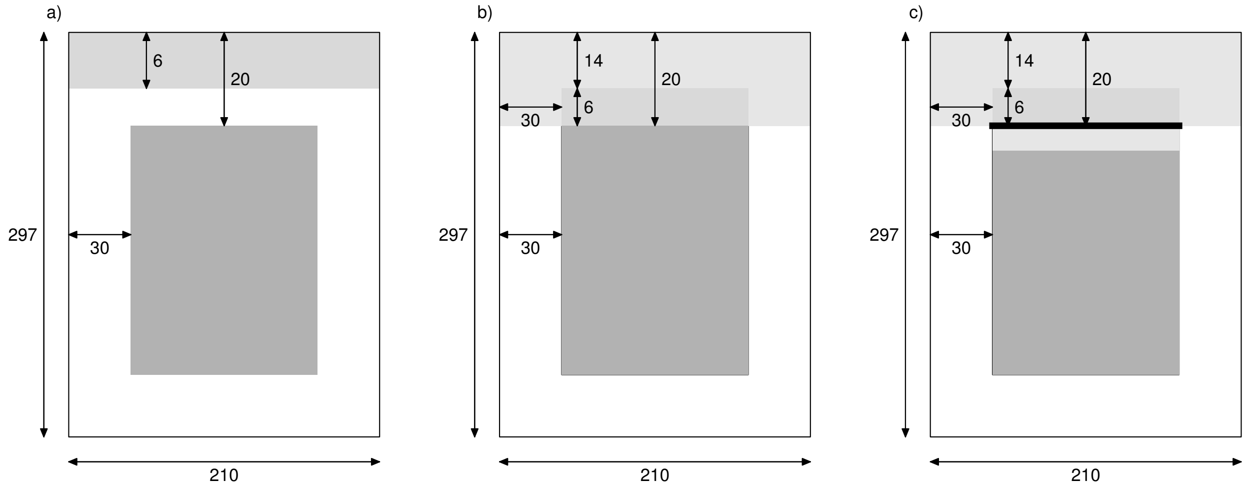
Pozostałe obszary przylegają do odpowiednich krawędzi *strony* a ich wielkość (wysokość dla pagin i szerokość dla marginaliów) określa atrybut `extent`. Wielkość pagin/marginaliów nie ma wpływu na

[1] Odpowiadająca wartości atrybutu `region-name` elementów `<region-...>`.

wielkość kolumny głównej. Innymi słowy margines określony dla obszaru <region-body> musi być na tyle duży żeby to co ma być umieszczone w paginach lub na marginesach nie znalazło się poza kartką (nie mniej niż wielkość extent). Przykład:

```
<fo:region-before region-name="tytulowa-p.gorna" extent="6mm" />
<fo:region-after region-name="tytulowa-p.dolna" extent="6mm" />
```

Zdefiniowaną w ww. sposób stronicę przedstawia schematycznie rys 4a). Jak widać, położenie paginy nie jest prawidłowe: jest ona umieszczona zbyt wysoko nad kolumną tekstu, a jej szerokość jest równa szerokości stronicy.



Rysunek 4: Wymiary kolumny głównej i paginy górnej

Dla wszystkich obszarów można określić *obramowanie* (*border*), *marginesy wewnętrzne* (*padding*) oraz *kolor tła*. Możliwe jest określenie następujących właściwości obramowania: *kolor* (16 kolorów predefiniowanych w specyfikacji HTML lub wartość RGB koloru), *rodzaj ramki* (linia ciągła, kreski, kropki itp...), *szerokość* (predefiniowane wartości thin, medium, thick lub jawnie podany wymiar, np. 3pt).

Każda właściwość obramowania może być zdefiniowana dla każdej krawędzi obszaru odzielnie za pomocą atrybutu o postaci: border-*krawędź*-*właściwość*. Przykład:

```
<fo:region-before
     border-top-color="C0C0C0" border-top-style="solid" border-top-width="1.5pt" />
```

Używając skrótu border-*właściwość* można zdefiniować każdą właściwość dla czterech krawędzi łącznie. Podobnie, odpowiednio określając wartość atrybutu postaci border-*krawędź*, można określić łącznie wiele właściwości dla pojedynczej krawędzi. Wreszcie pojedynczy atrybut border pozwala na łączne określenie wielu właściwości dla wszystkich krawędzi. Przykład:

```
<fo:region-before border-color="C0C0C0" border-style="solid" border-width="1.5pt" />
<fo:region-after border-bottom="C0C0C0 solid 1.5pt" />
<fo:region-end border=""
```

Uwaga: kolejność poszczególnych właściwości w definicji atrybutu border-*krawędź* i border jest dowolna. [Nie wszystkie procesory XSL-FO poprawnie interpretują skróty.]

Margines wewnętrzny jest definiowany za pomocą atrybutu padding-*krawędź*, który może zostać skrócony do postaci padding[2]. Przykład:

```
<fo:region-before padding-befor="24pt" />
```

[2] Określa wtedy margines wewnętrzny dla wielu wartości. [Nie jest interpretowany przez wszystkie procesory:-)].

```
<fo:region-after padding="24pt" />
<fo:region-end padding="1cm 3cm"/>     <!-- góra/dół=1cm, lewy/prawy=3cm      -->
<fo:region-end padding="1cm 2cm 3cm"/> <!-- góra=1cm, lewy/prawy=2cm, dół=3cm -->
```

Jeżeli wartość padding zawiera dwa wymiary to pierwszy określa wartość marginesu wewnętrznego dla krawędzi górnej i dolnej, a druga dla prawej i lewej. Jeżeli zawiera trzy wymiary, to pierwszy określa margines górny, druga marginesy prawy i lewy a trzecia margines dolny. Cztery wartości określają kolejno wielkość marginesu: górnego, prawego, dolnego i lewego. Jedna wartość określa wspólną wielkość marginesu dla wszystkich krawędzi łącznie.

Dla każdego obszaru można także zdefiniować kolor tła używając atrybutu: background-color. Specyfikacja wartości jest identyczna jak w przypadku koloru obramowania.

Z powyższych rozważań wynika, że odpowiednio ustawiając margines wewnętrzny dla obszaru odpowiadającego paginie górnej można osiągnąć efekt przedstawiony schematycznie na rys. 4b):

```
<fo:region-before region-name="tytulowa-p.gorna" extent="20mm"
     padding-before="14mm" padding-start="30mm" />
```

Brakuje jeszcze tylko kreski oddzielającej zwykle paginę od tekstu głównego. Dodając obramowanie i niezbędne marginesy wewnętrzne:

```
<fo:region-body margin="20mm 20mm 30mm 30mm" padding-top="4mm" />
<fo:region-before region-name="tytulowa-p.gorna" extent="20mm"
    border-width=".4pt" border-color="black" border-style="solid"
    padding-before="14mm" padding-start="30mm" />
```

osiągniemy wreszcie zadowalający efekt przedstawiony na rys. 4 c).

4. Formatowanie tekstu

Przypomnijmy (por. rys. 2), że treść dokumentu jest umieszczana wewnątrz elementów <fo:flow> lub <fo:static-content> (elementy te są dziećmi <fo:page-sequence>). Z tym, że treść *nie może* być umieszczona bezpośrednio wewnątrz nich ale powinna być „opakowana” za pomocą takich elementów, jak: <fo:block>, <fo:list-block> oraz <fo:float> i kilku innych. Poszczególne akapity, śródtytuły i podobne elementy są umieszczane wewnątrz <fo:block>, wyliczenia i wypunktowania wewnątrz <fo:list-block>, tabele wewnątrz <fo:table> lub <fo:table-and-caption>. Konkretny wygląd graficzny określają odpowiednie atrybuty w sposób podobny, jak w przypadku elementów definiujących obszary.

4.1. Bloki tekstu: odstępy i marginesy

Elementy <fo:block> to zwykle akapity i śródtytuły. Tego typu elementy umieszczane jeden pod drugim składają się na zawartość kolejnych stron dokumentu. Poszczególne bloki tekstu mogą zostać oddzielone odstępem przy wykorzystaniu opisanych wyżej atrybutów typu *margin*. Ustalanie zwłaszcza odstępu pomiędzy blokami za pomocą tych atrybutów nie jest jednak wskazane. Do tego celu lepiej wykorzystać atrybuty space-before i space-after[3].

Różne traktowanie obszarów i bloków, które na pierwszy rzut oka są tym samym obiektem (prostokątem), wynika z tego, że sposób wstawiania odstępów pomiędzy bloki tekstu powinien uwzględniać dodatkowo takie detale, jak np. domyślne usuwanie odstępu na dole/górze strony, oraz „inteligentne” scalanie sąsiadujących odstępów w jeden.

Przykładowo: odstęp pomiędzy akapitami winien zwykle znikać jeżeli akapit jest pierwszym na stronie aby wysokości stron sąsiadujących były jednakowe. Także odstępy pomiędzy różnymi blokami tekstu nie powinny się zwykle sumować. Przykładem jest umieszczanie większych niż normalne odstępów przed i po śródtytułach[4].

[3] Uwaga: atrybuty określające odstępy „poziome”: space-start oraz space-end nie dotyczą bloków ale obiektów wewnątrz bloków (*inline objects*).

[4] Załóżmy, że zakładamy odstęp wielkości np. 24 punkty przed oraz 12 punktów po śródtytule pierwszego stopnia, 12 punktów przed, oraz 6 punktów po śródtytule drugiego stopnia i 1,5 punktów przed i po akapicie. Wówczas odstęp pomiędzy śródtytułami będzie wynosił $12 + 12 = 24$ punkty; a odstęp

Przypisanie atrybutom space-before.*przyrostek* oraz space-after.*przyrostek* odpowiednich wartości pozwala na rozwiązanie ww. problemów. Atrybut space-before.optimum określa naturalny odstęp pomiędzy blokami, space-before.minimum minimalny a space-before.maximum – maksymalny. Zatem odpowiednikiem LaTeXowego polecenia \vspace{6mm plus 3mm minus 1.5mm} będzie:

```
<fo:block space-before.optimum='6mm' space-before.minimum='4.5mm'
    space-before.maximum='7.5mm'>
```

Atrybut space-before.*przyrostek* jest przykładem tzw. *atrybutu złożonego*, którego nazwa składa się z części zasadniczej, takiej jak np. space-before oraz oddzielonego kropką przyrostka, np. optimum. W standardzie XSL-FO zdefiniowanych jest więcej tego typu atrybutów.

Domyślnie sąsiadujące odstępy nie są sumowane ale zastępowane jednym, zwykle większym, choć reguły określone w specyfikacji [9] są bardziej skomplikowane. Do bardziej szczegółowego określenie sposobu ustalania odstępów należy użyć odpowiedniego atrybutu z przyrostkiem precedence, którego wartością może być liczba lub specjalna wartość force, określająca *priorytet*. Odstępy pomiędzy sąsiadującymi z sobą blokami tekstu są łączone w oparciu o wartość nadanego im priorytetu. Odstępy z przydzieloną najwyższą wartością są pozostawiane, a pozostałe usuwane. Wartość atrybutu force powoduje, że odstęp jest zachowywany bezwzględnie. Przykład:

```
<fo:block space-after='18pt' space-after.precedence='1'>Tytuł</fo:block>
<fo:block space-before='3pt'>Pierwszy akapit...</fo:block>
```

W powyższym przykładzie zostanie wstawiony odstęp o większym priorytecie, tj. 18pt po pierwszym bloku tekstu, zaś odstęp 3pt przed drugim blokiem zniknie. Atrybut postaci space-after jest skrótem; jeżeli jego wartością jest pojedynczy odstęp, to oznacza to przypisanie odstępowi (naturalnemu, minimalnemu i maksymalnemu) tej samej wartości.

Przyrostek .conditionality określa czy odstęp ma zostać usunięty (domyślnie odpowiada wartości atrybutu discard) czy pozostawiony (retain), jeżeli blok będzie pierwszym lub ostatnim blokiem wewnątrz innego bloku lub na stronie.

Bloki tekstu mogą zostać wcięte w stosunku do otaczającego je bloku lub strony poprzez wykorzystanie atrybutu start-indent oraz end-indent. Ten sam efekt można osiągnąć korzystając z atrybutów left-margin oraz rigth-margin.

W czasie podziału dokumentu na strony blok może zostać podzielony jeżeli nie może być zmieszczony na bieżącej stronicy (łamie). Także w tym przypadku istnieją różne ograniczenia wynikające z tradycji, zwyczajów drukarskich i wydawniczych. Przykładowo powszechnie nieakceptowane jest łamanie stron, w którym śródtytuły są ostatnimi elementami na stronie; wielu wydawców nie akceptuje także sytuacji, w której ostatni wiersz akapitu zaczyna stronę lub pierwszy wiersz akapitu stronę kończy. Oczywiste jest także, że np.: śródtytuły nie mogą znajdować się bezpośrednio na dole strony, ani też wielowierszowe śródtytuły nie mogą być podzielone pomiędzy stronice.

Atrybut keep-together.*przyrostek* określa czy blok *może* zostać podzielony w czasie stronicowania dokumentu. Wartością atrybutu może być: auto (może być podzielony), always (nie może być nigdy podzielony) lub liczba określająca stan pośredni. Dodanie przyrostka .within-page zakazuje podziału bloku pomiędzy stronice, podczas gdy .within-columns dotyczy kolumn w składzie wielołamowym.

Atrybut keep-with-previous.*przyrostek* określa czy bieżący blok musi znajdować się na tej samej stronie (z przyrostkiem .within-page) lub łamie (.within-column) co blok poprzedni. Zbiór możliwych wartości atrybutu to: auto (nie musi), always (musi) lub liczba (określa stan pośredni pomiędzy auto i always). Odwrotnością keep-with-previous jest atrybut break-before, którego zbiorem wartości jest: auto, column, page odd-page even-page. Wartości te oznaczają odpowiednio: rozpoczęcie nowego łamu, nowej stronicy, stronicy nieparzystej i stronicy parzystej.

Można także określić minimalną liczbę wierszy na końcu lub na początku kolumny za pomocą przypisania atrybutom orphans oraz widows odpowiedniej liczby. Domyślną wartością obu atrybutów jest 2.

---

pomiędzy śródtytułem pierwszego stopnia a tekstem tylko 1,5 + 12 = 13,5 punkty. W tej sytuacji tylko jeden z odstępów powinien być zachowany.

Bloki mogą posiadać *obramowanie*, *marginesy wewnętrzne* i *kolor tła*, specyfikowane identycznie jak w przypadku obszarów. W przeciwieństwie do obszaru blok może zostać podzielony pomiędzy sąsiadujące stronice lub łamy. W tej sytuacji możliwe jest określenie sposobu obramowania brzegu obszaru przylegającego do krawędzi podziału bloku. Wartość `discard` atrybutu `border-`*`krawędź`*[5]`-width.conditionality` zawiesza drukowanie ramki. Domyślnie ramka jest drukowana.

4.2. Interlinia, justowanie tekstu, wcięcia akapitowe

Atrybut `line-height` określa wielkość interlinii, która może być określona w wartościach bezwzględnych jako liczba bądź wartość procentowa. Liczby i procenty są interpretowane jako krotności bieżącego stopnia pisma. Atrybut `text-align` określa sposób justowania tekstu; dopuszczalne wartości to: `left`, `right`, `center` i `justify`. Wielkość wcięcia akapitowego określa atrybut `text-indent`. Sposób justowania ostatniego wiersza akapitu określa atrybut `text-align-last` opisany w punkcie 6.4. Wcięcie ostatniego wiersza (od prawej strony) można określić za pomocą odpowiedniej wartości atrybutu `last-line-end-indent`.

Atrybut `hyphenate` określa, czy wyrazy mogą (wartość `true`) czy nie mogą (wartość `false`) być przenoszone w procesie podziału akapitu na wiersze. Standard umożliwia dokładniejsze określenie sposobu przenoszenia. Przykładowo, atrybut `hyphenation-character` określa znak przeniesienia (wartość początkowa to znak o numerze `U+2010` (Unicode), czyli „-”).

Za pomocą określenia odpowiedniej wartości atrybutu `hyphenation-keep` elementu `<fo:block>` można kontrolować przenoszenie wyrazów pomiędzy szpaltami i/lub stronami. W szczególności wartość `column` tego atrybutu określa, że wyraz nie może być podzielony pomiędzy kolejnymi szpalatami tekstu, podczas gdy wartość `page` jest nieco mniej restrykcyjna i zabrania jedynie dzielenia wyrazów pomiędzy kolejne stronice. Wreszcie atrybut `hyphenation-ladder-count` określa maksymalną dopuszczalną liczbę kolejnych przeniesień.

Atrybuty `hyphenation-remain-character-count` oraz `hyphenation-push-character-count` umożliwiają określenie minimalnej liczby znaków w obu częściach podzielonego słowa.

4.3. Określenie kroju pisma i koloru

Atrybuty `font-family`, `font-size`, `font-style`, `font-variant`, `font-weight` pozwalają na określenie kroju, stopnia i odmiany pisma. Wartością atrybutu `font-family` jest lista oddzielonych przecinkami nazw krojów lub typów krojów. Typ kroju to `serif`, `sans-serif`, `cursive`, `fantasy` oraz `monospace`. Do formatowania dokumentu wybrany zostanie pierwszy krój z listy dostępnych podczas formatowania, np:

```
<fo:block font-family='Bembo, Times, serif' >
```

spowoduje wybranie kroju `Bembo` lub `Times` lub domyślnego kroju szeryfowego w zależności od tego, czym dysponuje program formatujący dokument.

Wartością atrybutu `font-size` może być wymiar, np. `12pt`. Wielkość kroju można także określić względnie w stosunku do stopnia pisma w elemencie nadrzędnym albo poprzez podanie jednej z wartości `larger` lub `smaller`, albo poprzez podanie procentu. Możliwe jest wreszcie określenie stopnia za pomocą wartości `xx-small`, `x-small small`, `medium large`, `x-large xx-large`.

Atrybuty `font-style`, `font-variant`, `font-weight` służą do określenia odmiany pisma. Wartościami atrybutu `font-style` mogą być: `normal` – odmiana prosta, `oblique` – odmiana pochyła, `italic` – kursywa. Atrybut `font-variant` ma następujące wartości: `normal` – odmiana zwykła oraz `small-caps` – kapitaliki. Wreszcie atrybut `font-weight` określa grubość pisma i może przyjąć następujące wartości: `normal` – pismo zwykłe, `bold` i `bolder` – grube, `lighter` – cienkie lub liczbę od 100, 200, 300 itd. aż do wartości maksymalnej 900.

Uwaga: atrybuty określające krój pisma mogą zostać wyspecyfikowane dla każdego elementu w arkuszu XSL-FO, nie tylko `<fo:block>`. Przykładowo: `<fo:root font-family='Univers'>` określa, że dokument ma zostać złożony za pomocą kroju Univers.

Możliwe jest także wykorzystanie notacji skrótowej, a wtedy składnia atrybutu `font` jest następująca:

[5] Dopuszczalne wartości to wyłącznie: `before`, `after`, `start`, `end`.

[*font-style* [*font-weight*][*font-variant*]] *font-size*[/*line-height*] *font-family*

Fragmenty zawarte pomiędzy [] są opcjonalne. Atrybuty pominięte nie są dziedziczone lecz przyjmują wartości domyślne, co może być pewnym zaskoczeniem jeżeli się o tym nie wie.

Atrybut `color` określa koloru tekstu. Wartość tego atrybutu można określić w sposób identyczny jak w przypadku koloru obramowania (por. punkt 3).

Przykładem opisanych w punktach 4.1–4.3 właściwości niech będzie następujący fragment szablonu:

```
<!-- Blok tekstu zawierający śródtytuł  -->
<fo:block font-size="14pt" color="#0060A0" font-weight="bold"
  keep-together.within-column="always"  <!-- nie dzielić bloku między łamy -->
  keep-with-next.within-column=always"  <!-- blok nie może kończyć łamu -->
  text-align="left" hyphenate="false"   <!-- skład w chorągiewkę, bez przenoszenia wyrazów -->
  space-before.conditionality="discard" <!-- usuń poniższy odstęp jeżeli zaczyna stronicę -->
  space-before.optimum="18pt" space-after.optimum="6pt" >
  ###...
</fo:block>
<!-- Blok tekstu składany literalnie (tj. verbatim): -->
  <fo:block font-family="monospace" font-size="9.5pt"
     text-align="left" space-before="6pt" space-after="6pt"
     white-space-collapse="false" linefeed-treatment="preserve"
     border="0.5pt gray solid" padding="3pt" <!-- ramka wokół tekstu -->
     border-before-width.conditionality="discard" <!-- usuń ramkę na dole strony -->
     border-after-width.conditionality="discard"  <!-- oraz na górze -->
       orphans="3" widows="3" >
  ###...
</fo:block>
```

W specyfikacji bloku tekstu składanego literalnie określono m.in., że ma być on otoczona obramowaniem w kolorze szarym, które ma zniknąć na brzegach stronic, jeżeli blok zostanie pomiędzy nie podzielony.

Element `<fo:wrapper>` jest opakowaniem na fragment tekstu, dla którego można zadeklarować odrębne atrybuty, takie jak: krój lub odmianę pisma, itp. Podobny do `<fo:wrapper>` jest element `<fo:inline>`, dla którego dodatkowo można określić obramowanie, margines wewnętrzny i tło. Przykład:

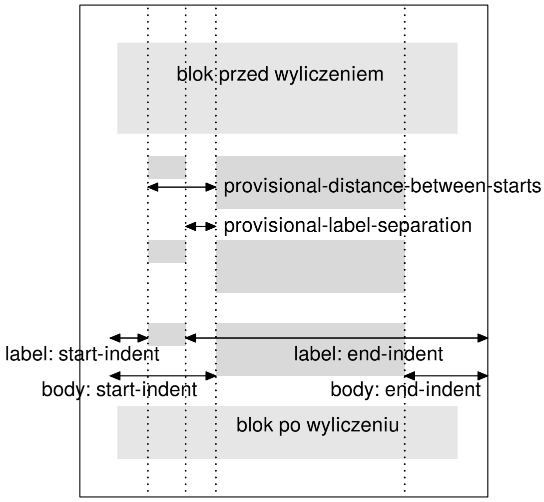
```
<fo:wrapper font-style="italic">Ten fragment jest wyróżniony... </fo:wrapper>
<fo:inline font-weight="bold" background-color="red">Na czerwonym tle... </fo:inline>
```

5. Wyliczenia

Wyliczenie jest formatowane za pomocą elementu `<fo:list-block>`, zawierającego jeden lub więcej elementów `<fo:list-item>`. Każdy `<fo:list-item>` składa się z *etykiety* (zawartej wewnątrz `<fo:list-item-label>`) oraz *hasła* (`<fo:list-item-body>`). Oba elementy zawierają bloki, które są wyrównywane w pionie i umieszczane obok siebie. Często etykieta to pojedynczy znak lub kolejny numer, ale może być to także większy fragment tekstu. W tym ostatnim przypadku tekst ten będzie łamany w obszarze o zadanej szerokości.

Atrybut `provisional-distance-between-starts` określa odległość pomiędzy lewą krawędzią elementu `item-label` a lewą krawędzią elementu `item-body`. Atrybut `provisional-label-separation` określa odległość pomiędzy prawą krawędzią etykiety a lewą krawędzią hasła. Atrybut `end-indent` elementu `<fo:list-item-label>` określa prawą krawędź obszaru, w którym zostanie umieszczona etykieta, zaś atrybut `start-indent` elementu `<fo:list-item-body>` lewą krawędź obszaru, w którym zostanie umieszczone hasło (rys. 5). Aby oba obszary nie zachodziły na siebie, należy ww. atrybutom przypisać sensowne wartości. Można je policzyć na kartce i wstawić „ręcznie", a można skorzystać ze specjalnych wartości: `label-end()`/`body-start()` a wtedy obliczenia wykona procesor XSL-FO. Przykład:

```
<fo:list-block provisional-distance-between-starts="20mm"
    provisional-label-separation="4mm">
  <fo:list-item>
    <fo:list-item-label end-indent="label-end()">
      <fo:block background-color="yellow">
       Pierwszy
      </fo:block>
    </fo:list-item-label>
    <fo:list-item-body start-indent="body-start()">
       <fo:block>Opis pierwszego punktu...</fo:block>
    </fo:list-item-body>
  </fo:list-item>
  <fo:list-item>
    <fo:list-item-label end-indent="label-end()">
      <fo:block background-color="#EFEFEF">Drugi
       trzeci czwarty piąty...
      </fo:block>
    </fo:list-item-label>
    <fo:list-item-body start-indent="body-start()>
       <fo:block>Opis drugiego punktu, opis
            trzeciego punktu ...
       </fo:block>
    </fo:list-item-body>
  </fo:list-item>
</fo:list-block>
```



Rysunek 5: Określenie wymiarów elementów wyliczeń

6. Pozostałe wybrane elementy

6.1. Hiperłącza

Istnieją dwa typu łączy hipertekstowych: do innych części dokumentu oraz do obiektów zewnętrznych. W obu wypadkach są one tworzone za pomocą elementu `<fo:basic-link>`. Łącza do innej części dokumentu są definiowane za pomocą nadania atrybutowi `internal-destination` wartości identycznej z tą, którą ma atrybut `id` elementu, do którego tworzymy odesłanie. Przykład:

```
Owocnik czubajki kani przedstawia
<fo:basic-link internal-destination='czubajka'>rys. 3</fo:basic-link>
```

Łącze zewnętrzne jest definiowane poprzez określenie wartości atrybutu `external-destination`. Wartością tego atrybutu jest URL. Przykład:

```
<fo:basic-link
external-destination="url('http://www.grzyby.pl/')">www.grzyby.pl</fo:basic-link>
```

Zwróćmy uwagę na sposób określenia identyfikatora URL.

6.2. Rysunki

Ilustracje nie są zwykle częścią pliku XSL-FO ale są przechowywane w specyficznych, binarnych formatach i znajdują się w oddzielnych plikach. Są dołączane do dokumentu za pomocą elementu `<fo:external-graphic>`. Atrybut `src`, którego wartością jest identyfikator URI identyfikuje plik z rysunkiem. Rysunek jest wstawiany w miejscu, w którym znajduje się element `<fo:external-graphic>`. Przykład:

```
<fo:block>Zdjęcie czubajki kani:
 <fo:external-graphic src='url(czubajka.jpg)'></fo:block>
```

6.3. Kropkowania

Do tworzenia kropkowania służy element `<fo:leader>`. Atrybut `leader-pattern` tego elementu określa sposób kropkowania. Wartość `space` jest domyślna i oznacza wypełnienie odstępem. Wartość `rule` oznacza wypełnienie kreską zaś `dots` oznacza wypełnienie kropkami. Jeżeli wartością atrybutu jest `use-content`, do wypełnienia kropkowania zostanie użyta zawartość elementu `<fo:leader>`. Atrybut `leader-pattern-width` określa wielkość odstępu między kropkami zaś atrybuty `rule-style` i `rule-thickness` odpowiednio rodzaj i grubość kreski (więcej szczegółów zawiera [9]).

Atrybuty `leader-length.optimum`, `leader-length.minimum` oraz `leader-length.maximum` określają odpowiednio naturalna, minimalną oraz maksymalną długość kropkowania.

6.4. Numery stron

Do umieszczenia w składzie numeru bieżącej strony należy użyć elementu `<fo:page-number>`. Zwykle element ten jest wykorzystywany w paginach i marginaliach.

Element `<page-number-citation>` wstawia numer strony, na której znajduje się inny obiekt. Obiekt ten powinien posiadać atrybut `id` zaś `<page-number-citation>` atrybut `ref-id` o identycznych wartościach.

Jako przykład wykorzystania elementów opisanych w punktach 6.3–6.4 zdefiniujmy paginę górną strony parzystej, tak aby na zewnętrznym marginesie pojawił się numer strony, a na wewnętrznym tytuł dokumentu, np. *Praktyczne wprowadzenie...*:

```
<fo:static-content flow-name="prawa-p.gorna">
 <fo:block font-size="9pt" text-align-last="justify" > Praktyczne wprowadzenie...
   <fo:leader leader-pattern="use-content" /> <fo:page-number /> </fo:block>
 </fo:static-content>
```

6.5. Przypisy i wstawki

Element `<fo:float>` jest odpowiednikiem LaTeXowego środowiska `float`, tj. zawiera niepodzielne elementy, które, jeżeli nie mogą być umieszczone w „normalnym” ciągu dokumentu, mogą być przesunięte na stronę następną.

Elementy `<fo:inline>` oraz `<fo:footnote-body>` służy do umieszczenia przypisu na dole stronicy. Typowo do sformatowania przypisu, wewnątrz elementu `<fo:footnote-body>` tworzy się jednoelementowe wyliczenie. Przykład (numer przypisu typowo jest wyliczany przez aplikację generującą dokument `.fo`, np. procesor XSLT):

```
 <fo:footnote> <fo:inline>1</fo:inline>
  <fo:footnote-body> <fo:list-block> <fo:list-item>
   <fo:list-item-label end-indent="label-end()">
     <fo:block>1</fo:block>
   </fo:list-item-label>
   <fo:list-item-body start-indent="body-start()">
              ###...
   </fo:list-item-body>
 </fo:list-item> </fo:list-block> </fo:footnote-body> </fo:footnote>
```

7. Procesor FOP

Procesor FOP przekształca dokumenty FO na pliki PDF. Program jest napisany w języku Java i rozwijany w ramach *Apache Software Foundation*. Udostępnionej w chwili pisania tego tekstu wersji 0.20.4. daleko jeszcze do miana oprogramowania produkcyjnej jakości; może co najwyżej być wykorzystana do eksperymentów lub tworzenia prostych dokumentów.

Po pobraniu np. ze strony `xml.apache.org` i rozpakowaniu archiwum `.jar`, program jest gotowy do użycia[6]. Po zainstalowaniu odpowiednich fontów i dodatkowej nieskomplikowanej konfiguracji umożliwia nawet formatowanie tekstów w języku polskim.

Do eksperymentów z FOPem wykorzystałem zestaw fontów firmy Microsoft w formacie TrueType (dostępnych np. ze strony `http://www.sourceforge.net/corefonts`). FOP wymaga wygenerowania z pliku `.ttf` pliku metrycznego w specyficznym formacie. Do wygenerowania tego pliku należy wykorzystać aplikację `TTFReader` znajdującą się w dystrybucji, uruchamiając ją w następujący sposób (`$FOPLIBS` zawiera listę ścieżek do niezbędnych archiwów `.jar`):

```
java -cp $FOPLIBS org.apache.fop.fonts.apps.TTFReader plik.ttf plik.xml
```

[6] Wymaga oczywiście uprzedniego zainstalowania interpretatora języka Java.

Po wygenerowaniu plików metrycznych fonty należy zarejestrować w pliku konfiguracyjnym znajdującym się w `~/conf/userconfig.xml` (`~` oznacza korzeń instalacji FOPa). Dodanie odpowiedniej informacji powinno być proste – można się wzorować na wpisach zawartych w pliku z dystrybucji. Aby FOP mógł skorzystać z fontów, musi być uruchomiony z opcją `-c` *ścieżka-do-pliku-konfiguracyjnego*, np:

```
fop -c /opt/Jlib/fop/config/userconfig.xml -fo plik.fo -pdf plik.pdf
```

Należy także „włączyć” nasz font dodając atrybut `font-family`, np. do elementu `<fo:root>`:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
   font-family="Times Roman">
```

Jak wspomniano na początku artykułu, typowym sposobem tworzenia plików PDF jest zamiana pliku XML na plik w formacie FO za pomocą arkusza stylu XSLT. W przypadku FOPa wymaga to uruchomienia aplikacji z trzema następującymi parametrami:

```
fop -xml plik.xml -xsl arkusz.xslt -pdf plik.pdf
```

Można także przetworzyć plik `.fo` bezpośrednio uruchamiając program z parametrami `-fo plik.fo -pdf plik.pdf`.

8. Zakończenie

Z uwagi na ciągle eksperymentalny charakter aplikacji wspierających rekomendację FO w tym krótkim artykule ograniczyliśmy się jedynie do podstaw standardu. Osoby zainteresowane (albo raczej: nie zniechęcone) mogą znaleźć przykładowe szablony, pliki konfiguracyjne do FOPa i dodatkowe informacje pod adresem `http://www.gust.org.pl/BIUL/biul18.html`.

Literatura

[1] Bradley Neil: *The XSL Companion*. Addison-Wesley 2000.

[2] Dokumentacja procesora FOP, patrz `http://www.apache.org/fop`.

[3] Druździel Mieczysław, Fijałkowski Tadeusz: *Zecerstwo*, *Inwentarium wiedzy o poligrafii*, pod red. Henryka Jankowskiego. Ossolineum 1988.

[4] Kay Michael: *XSLT Programmer's Reference, 2nd Edition*. Wrox Press Ltd., 2001.

[5] Nowacki Janusz M.: *TEXnologia a typografia*, Biuletyn GUST 6/1995, s. 3–11. Dokument dostępny także w `http://www.gust.org.pl/`.

[6] Olko Mariusz: *XSL – czyli jak przeczytać XMLowego „Pana Tadeusza”*, Biuletyn GUST 16/2001, s. 25–30. Dokument dostępny także w `http://www.gust.org.pl/`.

[7] Pawson Dave: *An introduction to XSL Formatting Objects*, 2001. Patrz `http://www.dpawson.co.uk/xsl/sect3/bk/index.html`.

[8] World Wide Web Consortium: *XSL Transformations (XSLT)*, patrz `http://www.w3.org/TR/xslt.html`.

[9] World Wide Web Consortium: *Extensible Stylesheet Language (XSL)*, patrz `http://www.w3.org/TR/xsl/`.

[10] World Wide Web Consortium, *XML Path Language (XPath)*, patrz `http://www.w3.org/TR/xpath.html`.

◇ Tomasz Przechlewski
tomasz@gnu.univ.gda.pl