DocBook, praktyczne wykorzystanie standardu XML

Tomasz Przechlewski
Uniwersytet Gdański, Wydział Zarządzania
tomasz@gnu.univ.gda.pl

Streszczenie

DocBook to deklaracja typu dokumentu (DTD) opracowana z myślą o tworzeniu dokumentacji technicznej. Dostępność gotowych szablonów DSSSL/XSL pozwala na publikowanie dokumentów XML zgodnych z DocDookiem w wielu formatach prezentacyjnych, bez konieczności tworzenia do tego celu własnych narzędzi czy też nauki kolejnego języka programowania. Zamierzeniem tego tekstu jest zapoznanie czytelnika z wybranymi narzędziami przetwarzania dokumentów typu DocBook, przy założeniu iż zaznajomiony jest On już z podstawami standardów XML/XSL.

Wprowadzenie

DocBook to aplikacja SGML/XML opracowana z myślą o tworzeniu dokumentacji technicznej, masowo wykorzystywana współcześnie do dokumentowania Wolnego Oprogramowania. Przykładami wykorzystania DocBooka są m.in. projekty Gnome (`http://developer.gnome.org/projects/gdp/handbook/gdp-handbook/`), KDE (`http://i18n.kde.org/doc/markup.html`) czy LinuxDoc (`http://linuxdoc.org/LDP/`). Zaletą i dużą zachętą do używania DocBooka jest dostępność gotowych szablonów przygotowanych zarówno w standardzie DSSSL jak i XSL (przez Normana Walsha), umożliwiających zamianę dokumentów XML do kilku popularnych formatów (HTML, XHTML, PDF, RTF).

Zamierzeniem tego tekstu jest zaznajomienie czytelnika z wybranymi narzędziami przetwarzania dokumentów typu DocBook, przy założeniu iż czytelnik jest już obeznany z podstawami standardów XML/XSL. Jeżeli tak nie jest, to dobrym wprowadzeniem w tematykę dokumentów strukturalnych jest [8], a z publikacji dostępnych *on-line* np. [10, 1, 4, 7].

Redagowanie dokumentu typu DocBook

Do redagowania dokumentów typu DocBook niezbędne jest zainstalowanie jednej z wersji DTD (`http://www.oasis-open.org/docbook/xml/`), parsera oraz edytora ułatwiającego pracę z tym standardem. Podobnie jak w przypadku innych dokumentów strukturalnych wygodnym środowiskiem pracy jest edytor Emacs z biblioteką psgml (`http://sourceforge.net/projects/psgml/`) wraz z dobrym parserem SGML/XML takim, jak: `onsgml` (`http://openjade.sourceforge.net/`) lub `xmllint` (`http://xmlsoft.org/`). W przypadku systemu Linuks wszystkie ww. narzędzia są bardzo łatwe do zainstalowania z pakietów dystrybucyjnych. W przypadku systemu Windows polecam Emacsa z płyty TeXLive 7, zaś pozostałe elementy należy skopiować z odpowiednich archiwów i zainstalować samodzielnie. Szczegółowy opis jak zainstalować i skonfigurować pakiet psgml w systemie MS Windows można znaleźć w [3] oraz [6].

Struktura dokumentu

Dokument DocBook winien zaczynać się od deklaracji XML oraz deklaracji typu dokumentu. Jeżeli używamy kodowania innego niż domyślne UTF-8, to w deklaracji XML należy dodać pseudoargument `encoding` z odpowiednią wartością. Przykład najprostszego dokumentu typu DocBook może zatem wyglądać następująco:

```
<?xml version="1.0" encoding="iso-8859-2" ?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
   "/usr/local/sgml/dtd/docbook/xml/4.1.2/docbookx.dtd">
<article lang='pl'>
  <title>Przykład</title>
  <sect1>
    <title>Punkt pierwszy</title>
    <para>Przykładowy akapit</para>
```

```
  </sect1>
</article>
```

Powyższe jest poprawnym dokumentem typu DocBook. Można się o tym przekonać wybierając z menu Emacsa polecenie `SGML->Validate` lub wpisując `xmllint --valid --noout plik.xml` z wiersza poleceń. Zalecane jest aby element główny dokumentu posiadał atrybut `lang` z wartością określającą język dokumentu (tak jak w przykładzie powyżej).

Hierarchia dokumentu DocBook jest typowa dla dokumentacji. Na pierwszym z poziomów jest zbiór dokumentów lub książek składających się z rozdziałów lub artykułów. Artykuły i rozdziały mogą być podzielone na punkty, podpunkty, akapity, wyliczenia, wypunktowania, spisy pojęć oraz inne typowe części książek takie jak: przedmowa, skorowidze, spisy treści, ilustracji, tabel itp. Kompletny opis DTD można znaleźć w [11].

Formatowanie: arkusze stylów XSL

Do zamiany dokumentów DocBook na postać prezentacyjną można wykorzystać gotowe arkusze stylów XSL (dalej określane w skrócie jako DBXSL, dostępne w `http://sourceforge.net/projects/docbook`) oraz dowolny procesor XSLT[1]. Dokumentacja DBXSL zaleca korzystanie z dwóch procesorów: `saxon` (`http://saxon.sourceforge.net`) oraz `xsltproc` (`http://xmlsoft.org/XSLT/`). Do zamiany plików XML do formatu PDF potrzebny będzie jeszcze program obsługujący standard XSL-FO, tak jak np. `fop` (`http://xml.apache.org/fop`). W przypadku systemu Linuks wszystkie ww. narzędzia są bardzo łatwe do zainstalowania z pakietów dystrybucyjnych. Użytkownicy systemu Windows muszą wykonać instalację samodzielnie kopiując odpowiednie archiwa. Programy `saxon` i `fop` wymagają uprzedniego zainstalowania interpretatora języka Java wraz z JDK. Ponadto `fop` – do prawidłowego działania w środowisku języka polskiego – wymaga dodatkowej konfiguracji (por. [7]).

Transformacja pliku XML do formatu prezentacyjnego sprowadza się do uruchomienia procesora XSLT i podania w wierszu poleceń odpowiedniego arkusza (`$DBXSL` oznacza korzeń dystrybucji):

$DBXSL/html/docbook.xsl utworzy pojedynczy plik HTML;

$DBXSL/chunk.xsl utworzy dokument HTML podzielony na wiele plików;

$DBXSL/html/docbook.xsl zamieni dokument DocBook na dokument w standardzie XSL-FO.

Przykładowo wykonanie polecenia:

```
xsltproc  --output plik.html  $DBXSL/html/docbook.xsl  plik.xml
```

uruchomi transformację `pliku.xml` do dokumentu HTML, umieszczonego w pojedynczym pliku.

Modyfikowanie sposobu formatowania

Wielką zaletą arkuszy DBXSL jest ich łatwa konfiguracja. Arkusze te w szczególności posiadają wiele parametrów określających różne aspekty formatowania dokumentów. Najprostszą metodą zmiany sposobu formatowania jest przypisanie określonemu parametrowi odpowiedniej wartości. Przykładowo w standardowym sposobie formatowania dokumentów HTML tytuły punktów, podpunktów i innych jednostek podziału hierarchicznego nie są numerowane. Aby to zmienić wystarczy przypisać parametrowi `section.autolabel` wartość niezerową. W podobnie prosty sposób wartość parametru `toc.section.depth` określa, które z nich mają być umieszczone w spisie treści.

W przypadku uruchamianie procesora XSLT „z wiersza poleceń”, konkretna składnia przypisania parametrowi wartości jest różna w zależności od wykorzystywanego narzędzia. W przypadku dwóch zalecanych w dokumentacji DBXSL procesorów uruchomienie transformacji z określeniem wartości obu ww. parametrów wyglądałoby następująco:

```
saxon -o plik.html plik.xml plik.xsl section.autolabel=1 toc.section.depth=1
```

lub

```
xsltproc  --output plik.html --stringparam section.autolabel=1 \
  --stringparam toc.section.depth=1 plik.xsl plik.xml
```

[1] Istnieją też arkusze opracowane w standardzie DSSSL, ale nie będziemy ich opisywać.

Pełne zestawienie dostępnych parametrów znajduje się w dokumentacji do DBXSL w katalogach `$DBXSL/doc/html` oraz `$DBXSL/doc/fo`, gdzie `$DBXSL` oznacza korzeń dystrybucji arkuszy stylów.

Specyfikowanie wartości parametrów z wiersza poleceń ma sens tylko wtedy, kiedy zmiany są niewielkie. W przypadku większych modyfikacji odpowiednie definicje należy umieścić w pliku „adaptującym" i ten plik podawać jako argument przy uruchamianiu procesora XSLT, przykładowo:

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:import href="/usr/share/docbook.xsl"/>

<xsl:param name="section.autolabel" select="1"/>
<xsl:param name="toc.section" select="1"/>
<xsl:param name="html.stylesheet" select="'plik.css'"/>
```

Powyższy przykład ilustruje w jaki sposób można nadawać wartości parametrom w języku XSLT: nazwę parametru określa atrybut `name`, a jego wartość atrybut `select`. Polecenie `<xsl:import>` importuje szablon XSLT i *musi* wystąpić *bezpośrednio* po `<xsl:stylesheet>`.

W składni języka XSLT znajduje się także polecenie umożliwiające definiowanie zbioru atrybutów (`<xsl:attribute-set>`). Taki zbiór przypomina wielowartościowy parameter, za pomocą którego można w wygodny sposób określać wartości wielu parametrów jednocześnie. W arkuszach DBXSL jest określonych wiele zbiorów atrybutów w pliku `$DBXSL/fo/param.xsl`. Dopasowanie wartości zbiorów atrybutów jest równie proste jak w przypadku pojedynczych parametrów. Przykładowo zbiór atrybutów `section.title.properties` określa sposób formatowania tytułów punktów:

```
<xsl:attribute-set name="section.title.properties">
  <xsl:attribute name="font-family"> <xsl:value-of select="$title.font.family"/>
  </xsl:attribute>
  <xsl:attribute name="font-weight">bold</xsl:attribute>
  <!-- font size is calculated dynamically by section.heading template -->
  <xsl:attribute name="keep-with-next.within-column">always</xsl:attribute>
  <xsl:attribute name="space-before.minimum">0.8em</xsl:attribute>
  <xsl:attribute name="space-before.optimum">1.0em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">1.2em</xsl:attribute>
</xsl:attribute-set>
```

Kolejnym sposobem dopasowanie arkuszy DBXSL do własnych potrzeb jest wypełnienie pustych szablonów wykonywanych w określonych miejscach. Standardowo szablony te nie zawierają żadnej treści, a ich wykonanie nie daje żadnego rezultatu. Przykładowo istnieje szablon `user.header.content` wykonywany na początku tworzenia każdego pliku HTML. Jakakolwiek treść generowana przez ten szablon pojawi się na początku każdego utworzonego dokumentu HTML.

Szczegółowy opis różnych sposobów dopasowywania DBXSL do własnych potrzeb zawiera dokumentacja arkuszy (umieszczona w katalogu `doc` dystrybucji) oraz [9].

Polonizacja

Funkcjonalność arkuszy DBXSL z punktu widzenia formatowania dokumentów przygotowanych w różnych językach sprowadza się w zasadzie wyłącznie do poprawnego wstawiania tekstów generowanych automatycznie. Aby osiągnąć ten efekt wystarczy prawidłowo określić wartość atrybutu `lang` głównego elementu dokumentu[2]. Sortowanie zgodnie ze zwyczajami lokalnymi nie jest zaimplementowane w bieżących wersjach arkuszy, ale można przypuszczać, że w przyszłych wersjach zostanie wykonane.

W chwili pisania tego tekstu arkusze DBXSL wspierają około 40 języków. Napisy wstawiane automatycznie znajdują się w plikach o nazwach określonych przez kod języka z rozszerzeniem `.xml` (np. `pl.xml`) znajdujących się w katalogu `$DBXSL/common`. Możliwe jest zdefiniowanie wstawiania różnych napisów w zależności od kontekstu, w jakim znajduje się element lub sposobu formatowania dokumentu.

Przykładowo, aby zmodyfikować tekst wstawiany przy formatowaniu odsyłacza należy zmienić zawartość elementu `<l:context>` oznaczoną atrybutem `name` o wartości `xref`:

[2] Oczywiście należy także odpowiednio zadeklarować sposób kodowania polskich znaków w deklaracji XML, por. punkt *Struktura dokumentu*.

```
<l:context name="xref">
   <l:template name="abstract" text="%t"/>
   <l:template name="answer" text="Odp: %n"/>
   ...
   <l:template name="figure" text="Rysunek %n. %t"/>
   ...
```

W powyższym kodzie symbol `%n` oznacza numer elementu, a `%t` jego tytuł. Zatem zapis:

```
dane na <xref linkend="rys.dyna.obr"/> potwierdzają, że
```

zostanie sformatowany, np. jako: „dane na Rysunek 11. Dynamika obrotów potwierdzają, że”, co przyznajmy nie do końca odpowiada regułom poprawnej polszczyzny.

Aby to zmienić należy skopiować odpowiedni element do szablonu „adaptacyjnego” i zmodyfikować go. Wszystkie elementy określające sposób generowania tekstu powinny zostać umieszczone wewnątrz parametru `local.l10n.xml`:

```
<xsl:param name="local.l10n.xml" select="document('')"/>
<l:i18n xmlns:l="http://docbook.sourceforge.net/xmlns/l10n/1.0">
  <l:l10n language="pl">
    <l:context name="xref">
      <l:template name="figure" text="rys. %n"/>
    </l:context>
  </l:l10n>
</l:i18n>
```

Spowoduje to zmianę sposobu formatowanie na: „dane na rys. 11. Dynamika obrotów potwierdzają, że”. W analogiczny sposób należy dopasować teksty wstawiane przy odesłaniach do elementów innych typów.

Podsumowanie

Najpoważniejszym ograniczeniem opisanego w tekście systemu produkcji dokumentów jest niska jakość typograficzna dokumentów PDF. Wynika to zarówno z ograniczeń standardu XSL-FO, jak też ciągle eksperymentalnego charakteru aplikacji wspierających rekomendację XSL-FO (przyjmniej tych rozpowszechnianych jako Oprogramowanie Wolne). Tym niemniej w wielu zastosowaniach DocBook może być użyteczną aplikacją.

Literatura

[1] Burnard L., Sperberg-McQueen C. M., *TEI Lite: An Introduction to Text Encoding for Interchange*, June 1995. `http://www.uic.edu/orgs/tei/intros/teiu5.html`

[2] DuCharme B., *Edytowanie dokumentów* SGML *z wykorzystaniem Emacsa i psgml*. Tłumaczenie drugiego rozdziału z książki *SGML CD*, dostępne w `http://panda.bg.univ.gda.pl/~tomasz`

[3] Hoenicka M., *SGML for Windows NT*, 2000. `http://ourworld.compuserve.com/homepages/hoenicka_markus/ntsgml.html`

[4] Olko M., XSL – czyli jak przeczytać XMLowego „Pana Tadeusza” Biuletyn GUST 16/2001, s. 25–30.

[5] Pawson D., *An Introduction to XSL Formatting Objects*, 2001

[6] Przechlewski T., *Definicja dokumentu typu PMLA*, 2003, `http://gnu.univ.gda.pl/~tomasz/sgml/pmla/`

[7] Przechlewski T., *Praktyczne wprowadzenie do standardu XSL-FO*, Biuletyn GUST 18/2002.

[8] Ray E. T., *Nauka języka XML*, Wydawnictwo RM, Warszawa 2001.

[9] Stayton R., *Using the DocBook XSL Stylesheets*, 2003, `http://www.sagehill.net/xml/docbookxsl/index.html`.

[10] Walsh N., *Technical introduction to XML*, 1998, `http://nwalsh.com/docs/articles/xml/`.

[11] Walsh N., Muellner L., *DocBook: The Definitive Guide*, wersja Version 2.0.7 (Jun/2002). Dokument dostępny w `http://www.docbook.org/tdg/en/html/`

[12] DocBook Wiki, `http://www.docbook.org/wiki/moin.cgi`