Introducing LaTeX users to XSL-FO*

Jean-Michel Hufflen
LIFC (EA CNRS 4157)
University of Franche-Comté
16, route de Gray
25030 BESANÇON CEDEX
France
hufflen (at) lifc dot univ-fcomte dot fr
http://lifc.univ-fcomte.fr/~hufflen

Abstract

This talk aims to introduce LaTeX users to XSL-FO. It does not attempt to give an exhaustive view of XSL-FO, but allows a LaTeX user to get started. We show the common and different points between these two approaches of word processing.
Keywords LaTeX, XSL-FO, XML, XSLT.

Streszczenie

Prezentacja jest wprowadzeniem użytkowników LaTeX-a do XSL-FO. Nie próbujemy omówić XSL-FO w sposób wyczerpujący, ale umożliwimy użytkownikom LaTeX-a rozpoczęcie pracy w tej technologii. Pokażemy punkty wspólne i różnice obu podejść do formatowania tekstów.
Słowa kluczowe LaTeX, XSL-FO, XML, XSLT.

0 Introduction

This talk aims to introduce LaTeX users to XSL-FO.[1] Both have common points, in the sense that they are not WYSIWYG.[2] In both cases, users prepare a source file that is processed and the result is a file that can be send to a laser printer. [11, § 18] lists some implementations of processors of XSL-FO texts. Among them, we personally have experienced:

- PassiveTeX [10, p. 180]: this is an (incomplete) adaptation of TeX in order to process XSL-FO texts; the result may be a DVI[3] or PDF[4] file;
- Apache FOP[5] [3], written in Java, is more complete; the result may be a PDF or PostScript file, with other formats also being possible.

XSL-FO is an XML[6] format that aims to describe high-quality print outputs. As we will see, this format is very verbose, but it is not intended for direct use. Usually, XSL-FO texts result from applying an XSLT[7] stylesheet to an XML text, as we will see. Thus this approach clearly separates *presentation* and *contents*. An XML text specifies contents, an XSL-FO text specifies presentation. However, we begin with a text directly typed in XSL-FO to give the broad outlines of this language, then we show an example of an XSLT program that generates such a text. We end with some words about internationalisation. Reading this article requires only basic knowledge of XML and XSLT.

1 Getting started

1.1 Basic notions

The notion equivalent to a document class of LaTeX consists of a *page model*, an example being given in Figure 1. The page model here is very simple: only one page, specified by the `fo:simple-page-master` element. It specifies a paper format and its margins, where anything cannot be printed. It also defines *regions*, as shown in Figure 2. You can define *several* single page models, and another element, `fo:page-sequence-master`, allows the combination of single or repeatable pages. Repeatable pages may

* Title in Polish: *Wprowadzenie do XSL-FO dla użytkowników LaTeX-a.*

[1] **EX**tensible **S**tylesheet **L**anguage-**F**ormatting **O**bjects.

[2] **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et. This expression identifies typical interactive word processors.

[3] **De**V**ice-**I**ndependent **F**ile.

[4] **P**ortable **D**ocument **F**ormat.

[5] **F**ormatting **O**bjects **P**rocessor.

[6] **EX**tensible **M**arkup **L**anguage. Readers interested in an introductory book to this formalism can consult [12].

[7] **EX**tensible **S**tylesheet **L**anguage **T**ransformations. Several introductory talks to this language have already been given at BachoTeX conferences [4, 5]; the reference is [14].

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<fo:layout-master-set xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <!--  xmlns:fo declares a prefix for the namespace associated with XSL-FO texts.  -->
  <fo:simple-page-master master-name="page-simple" page-height="297mm" page-width="210mm"
                         margin-top="10mm" margin-bottom="20mm" margin-left="25mm"
                         margin-right="25mm">
    <fo:region-before extent="30mm"/>     <!--  Declaration of the header, footer, left and right  -->
    <fo:region-after extent="30mm"/>      <!--  margin. These usual terms have been viewed as too  -->
    <fo:region-start extent="30mm"/>      <!--  specific to left-to-right writing, thus a  -->
    <fo:region-end extent="30mm"/>        <!--  terminology based on ‘before’, ‘after’, ‘start’, ‘end’  -->
    <fo:region-body/>                     <!--  is preferred. The body is defined as the page’s  -->
                                          <!--  remainder. See Figure 2.  -->
  </fo:simple-page-master>
</fo:layout-master-set>
```

Figure 1: Example of a page model in XSL-FO.

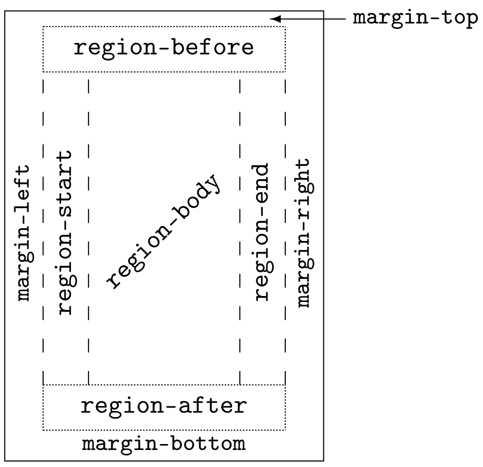
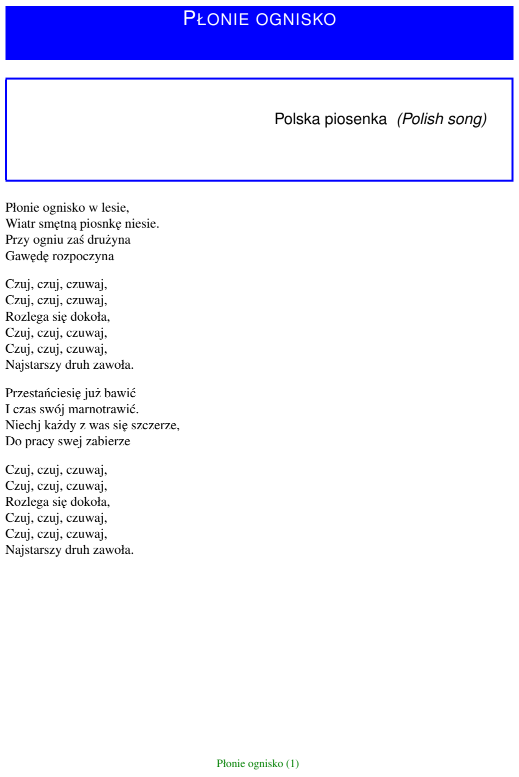


Figure 2: Regions defined by XSL-FO.



PŁONIE OGNISKO

Polska piosenka *(Polish song)*

Płonie ognisko w lesie,
Wiatr smętną piosnkę niesie.
Przy ogniu zaś drużyna
Gawędę rozpoczyna

Czuj, czuj, czuwaj,
Czuj, czuj, czuwaj,
Rozlega się dokoła,
Czuj, czuj, czuwaj,
Czuj, czuj, czuwaj,
Najstarszy druh zawoła.

Przestańciesię już bawić
I czas swój marnotrawić.
Niechj każdy z was się szczerze,
Do pracy swej zabierze

Czuj, czuj, czuwaj,
Czuj, czuj, czuwaj,
Rozlega się dokoła,
Czuj, czuj, czuwaj,
Czuj, czuj, czuwaj,
Najstarszy druh zawoła.

Płonie ognisko (1)

Figure 3: The formatted output of Figure 4.

vary w.r.t. the position, that is, you can alternate two models for even and odd pages, or define a separate model for initial and final pages, ….

Figure 3 shows how an XSL-FO text may be formatted, the source text being given in Figure 4. We will see that page models are not specified by including a file as in LaTeX. If you wish a page model to be shared among several XSL-FO texts, an external entity is to be used [12, pp. 50–52]. This implies the introduction of a ‘dummy’ `DOCTYPE` tag.[8] We see that an XSL-FO text is rooted by an `fo:root` element, whose children are a page model and a page sequence. A page sequence defines what is written and where. In Figure 4, a *static content* — a song’s title, followed by the number of the current page — is related to any page foot, whereas a *flow* allows the specification of a text possibly printed on regions belonging to several successive pages. A flow is bound to a region by means of the `flow-name` attribute, referring to the `region-name` attribute’s value of an element for a region. There are default conventions; for example, the definition of the ‘body’ region given in Figure 1 is equivalent to:

```
<fo:region-body
  region-name="xsl-region-body"/>
```

[8] … which is a *trick*. A better method consists of using tags belonging to XInclude [15], but make sure that they are recognised by the tools you are using.

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<!DOCTYPE root [<!ENTITY layout SYSTEM "layout.fo">
                <!ENTITY refren-1 "Czuj, czuj, czuwaj,">]>

<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  &layout;

  <fo:page-sequence master-reference="page-simple" font-family="serif" font-size="12pt"
                    text-align="left">

    <fo:static-content flow-name="xsl-region-after">
      <fo:block text-align="center" line-height="14pt" color="green" font-size="10pt"
        font-family="serif">
        Płonie ognisko (<fo:page-number/>)
      </fo:block>
    </fo:static-content>

    <fo:flow flow-name="xsl-region-body" xml:lang="po">

      <fo:block font-family="sans-serif" font-size="18pt" font-variant="small-caps"
                padding-top="3pt" text-align="center" color="white" background-color="blue"
                space-after="15pt" line-height="24pt">
        Płonie ognisko
      </fo:block>
      <fo:block font-family="sans-serif" font-size="14pt" space-after="18pt" border-style="solid"
                border-width="0.5mm" border-color="blue" padding="4mm" start-indent="80mm"
                end-indent="4mm">
        <fo:block text-align="right">
          Polska piosenka
          <fo:inline font-style="italic" xml:lang="en">(Polish song)</fo:inline>
        </fo:block>
      </fo:block>

      <fo:block space-before.minimum="10pt" space-before.optimum="11pt"
                space-before.maximum="12pt">
        Płonie ognisko w lesie,
      </fo:block>
      <fo:block>Wiatr smętną piosnkę niesie.</fo:block>
      <fo:block>Przy ogniu zaś drużyna</fo:block>
      <fo:block>Gawędę rozpoczyna</fo:block>

      <fo:block ...>   <!--  As above for the stanza's first line.  -->
        &refren-1;
      </fo:block>
      <fo:block>&refren-1;</fo:block>
      <fo:block>Rozlega się dokoła,</fo:block>
      <fo:block>&refren-1;</fo:block>
      <fo:block>&refren-1;</fo:block>
      <fo:block>Najstarszy druh zawoła.</fo:block>

      <fo:block ...>Przestańciesię już bawić</fo:block>
      <fo:block>I czas swój marnotrawić.</fo:block>
      <fo:block>Niechj każdy z was się szczerze,</fo:block>
      <fo:block>Do pracy swej zabierze</fo:block>

      ...  <!--  The refrain again.  -->

    </fo:flow>

  </fo:page-sequence>

</fo:root>
```

Figure 4: Complete source of the text of Figure 3.

Attribute	Default value	Other values
`font-family`	`serif`	`sans-serif`
`font-size`		Absolute sizes: `xx-small`, `x-small`, `medium`, `large`, `x-large`, `xx-large`, Relative sizes: `smaller`, `larger` Lengths: e.g., `10pt`
`font-stretch`	`normal`	`wider`, `narrower`, `ultra-condensed`, `extra-condensed`, `condensed`, `semi-condensed`, `semi-expanded`, `expanded`, `extra-expanded`, `ultra-expanded`
`font-weight`	`normal`	`bold`, `bolder`, `lighter`
`font-style`	`normal`	`italic`, `reverse-normal`, `reverse-oblique`
`font-variant`	`normal`	`small-caps`

Table 1: Possible values for most of font attributes.

1.2 Formatting texts

At first glance, `fo:block` elements are analogous to paragraphs in LaTeX. The text inside a block may be justified, left or right aligned, according to the value of `text-align`. The attributes `color` and `background-color` specify colours for the text and background. Other attributes:

`border-style` `border-width` `border-color`

allows us to draw a box around this block. Of course, `border-width` is set to a null value by default, so no border is drawn. The '`padding-...`' attributes characterise the padding between the text and border [10, pp. 96–100].

Blocks may be nested and most attributes are inherited. As an example, let us consider the second block of the flow. It defines some attributes related to fonts — `font-family` and `font-size` — these attributes being inherited in the nested block containing the Polish and English words for 'Polish song' ('*Polska piosenka*'). The `fo:inline` element allows some attributes to be redefined without opening a new block: it corresponds to changing some parameters — font style or size, etc. — inside the same paragraph in LaTeX. In fact, we can consider that `fo:block` elements, due to this recursive nature, are equivalent to both the `\par` command and the `minipage` environment of LaTeX. The possible values associated with most of the font attributes are given in Table 1. In comparison with LaTeX where the family, weight, style, and variant of a font are expressed by combinations of commands being the same syntax, '`\text...{...}`', the attributes of XSL-FO are more 'typed'. That may be seem quite artificial to a LaTeX user, but emphasises all the possible combinations.

The `start-indent` attribute specifies the distance from the start-edge of the box surrounding the contents to the start-edge of the contents itself. The `end-indent` attribute is analogous, but end-edges are considered. The vertical spacing between successive blocks is controlled by the two attributes `space-before` and `space-after`. The specification of *stretcheable lengths* in LaTeX [7, § A.1.15] is implemented in XSL-FO by means of *components*. Let us look at the first stanza given in Figure 4: the vertical spacing before this block is ideally 11 pt long, at least 10 pt long, and at most 12 pt long, according to the values of the components `optimum`, `minimum`, and `maximum` of the `space-before` attribute. Just specifying `space-before="11pt"` sets the three components of the `space-before` attribute to this length. Putting:

```
space-before="11pt"
space-before.minimum="10pt"
```

only redefines the `minimum` component, the two others being 11 pt long.

Going thoroughly into this notion, XSL-FO provides two other components for the specification of spacing. The `conditionality` component controls whether a space-specifier has effect at the beginning or end of a reference area — e.g., the beginning (resp. end) of a page for the `space-before` (resp. `space-after`) attribute of the `fo:block` element, or the beginning (resp. end) of a line for the `space-start` (resp. `start-end`) attribute of the `fo:inline` element. The possible values for this `conditionality` component are `discard` (by default) and `retain`. The `precedence` component can either be an integer or the keyword `force`. It determines what happens when the end of a reference area conflicts with the next one. If the `precedence` component is set to `force`, this will override any other space specifiers that conflict with it.

Let us briefly mention two attributes for blocks or inline texts: `text-decoration` is used to draw a line above, below, or through a text [16, § 7.17.4], `baseline-shift` is used for subscripts and superscripts. Since XSL-FO only aims to give nice layout of a text, there is no practical way to do computations on this text. For example, the fragment:

```
\iflanguage{polish}{Polska piosenka}{%
 Polish song}
```

(cf. [7, § 9.2.1] about the `\iflanguage` command) cannot be transcribed into an XSL-FO text. However, some typical transformations can be put into action by means of the `text-transform` attribute, whose values may be `none` (by default), `capitalize`, `uppercase`, `lowercase`. Let us notice that using this attribute is somewhat deprecated because these operations do not make sense given internationalisation issues.

Other attributes prevent the breaking of a text into lines, columns, and pages when blocks are typeset: `keep-with-next`, `keep-with-previous`, and `keep-together`. Each of these three attributes has three components: `within-line`, `within-column`, and `within-page`. The associated values are `auto` (by default), that is, no constraint, `always`, or an integer expressing the strength of this property. This integer can be compared to the optional argument of the LaTeX commands `\pagebreak` and `\linebreak`. For example, if there is a `fo:block` element with a `keep-with-next` attribute set to `always`, there cannot be a page break between this block and the preceding one. If you want to force breaking in such situations, use the attributes `break-before` and `break-after`, whose values are `auto` (by default), `column`, `page`, `even-page`, and `odd-page`. See [10, pp. 70–72] for more details.

1.3 Additional elements

Now we mention some additional functionalities of XSL-FO, in order to give an idea of its expressive power. It provides elements to express lists, analogous to LaTeX's, rooted by the `fo:list-block` element [10, pp. 102]. The way to specify tabulars is analogous to HTML's,[9] the most commonly used element for this being `fo:table` [10, pp. 104–110]. Footnotes are specified via the `fo:footnote` element [10, pp. 154–155], analogous to the `\footnote` command. Cross references as in LaTeX are supported by means of the `fo:basic-link` element [10, pp. 146–148]; hyper-link references to external documents are also possible. The notion of floating objects is known within XSL-FO: see [16, § 6.12.2] about the `fo:float` element. The language provides elements and attributes for building indexes [16, § 7.24], analogous to what is used within LaTeX's `theindex` environment (cf. [7, § 11.1]). Last, let us notice that there is no mathematical mode in XSL-FO.

[9] **H**yper**T**ext **M**arkup **L**anguage. Readers interested in an introduction to this language can refer to [9].

2 XSLT and XSL-FO together

The Polish song given in Figure 4 has already been specified as a 'pure' XML text in [6, Fig. 1]. We reproduce it as Figure 5. Then we give an XSLT stylesheet (Figures 6 and 7) that yields the text of Figure 4 when it is applied to the XML text of Figure 5. That shows how XSL-FO texts should be built: by derivation from XML texts by applying a stylesheet.

The use of two namespaces [12, pp. 41–45] given by prefixes, `xmlns:xsl` and `xmlns:fo`,[10] clearly separates what is evaluated ('`<xsl:.../>`') when the XSLT program is running, and what results from this operation ('`<fo:.../>`'). Finally, let us notice that XSL-FO does not provide a way to build a table of contents automatically, but doing this task is easy when an XSLT program is used [10, pp. 149–150].

3 Some words on internationalisation

XSL-FO provides properties — that is, attributes — for specifying hyphenation properties [16, § 7.10]. These attributes includes the specification of a country, a language, a hyphenation character, etc. In practice, the predefined attribute `xml:lang` — see the two occurrences of this attributes in Figure 4 — is treated as a shorthand and used to set the country and language properties [16, § 7.31.24]. This attribute characterizes the language of a content by a two-letter language, optionally followed by a two-letter country code, as specified in [1].

XSL-FO is not limited to languages using the Latin alphabet and can deal with any writing mode. The `writing-mode` attribute can be set to:

- `lr-tb`, for 'left-to-right, top-to-bottom' (by default),
- `rl-tb`, for 'right-to-left, top-to-bottom',
- `tb-rl`, for 'top-to-bottom, right-to-left',
- or others [16, § 7.29.7].

This specifies two directions: the first is the inline-progression-direction which determines the direction in which words will be placed and the second is the block-progression-direction which determines the direction in which blocks and lines are placed one after another. The inline-progression-direction for a sequence of characters may be implicitly determined using bidirectional character types for those characters from the Unicode Character Database [13] and the Unicode bidirectional algorithm [13, Annex 9].

[10] In fact, the information identifying a precise namespace is not the prefix itself, but the value associated with it, e.g., '`http://www.w3.org/1999/XSL/Transform`' for an XSLT program. XInclude (see Footnote 8, p. 110) introduces another namespace to model file inclusions [15].

```
<?xml version="1.0" encoding="ISO-8859-2"?>

<!DOCTYPE poem0 SYSTEM "poem0.dtd" [<!ENTITY refren-1 "<verse>Czuj, czuj, czuwaj,</verse>">]>

<poem0>
  <preamble><title>Płonie ognisko</title></preamble>
  <body>
    <stanza>
      <verse>Płonie ognisko w lesie,</verse>
      <verse>Wiatr smętną piosnkę niesie.</verse>
      <verse>Przy ogniu zaś drużyna</verse>
      <verse>Gawędę rozpoczyna</verse>
    </stanza>
    <stanza label="refren">   <!--  label is an optional attribute being type ID.    -->
      &refren-1;&refren-1;   <!--  Syntactical replacement.     -->
      <verse>Rozlega się dokoła,</verse>
      &refren-1;&refren-1;
      <verse>Najstarszy druh zawoła.</verse>
    </stanza>
    <stanza>
      <verse>Przestańciesię już bawić</verse>
      <verse>I czas swój marnotrawić.</verse>
      <verse>Niechj każdy z was się szczerze,</verse>
      <verse>Do pracy swej zabierze</verse>
    </stanza>
    <stanza>
      <!--  A stanza is a non-empty list of verses, but can be a repetition of a previous stanza, in which case we
            use the resume element with a required attribute, ref. The value associated with this IDREF
            attribute unambiguously identifies a subtree.
        -->
      <resume ref="refren"/>
    </stanza>
  </body>
</poem0>
```

Figure 5: Example of a Polish song as an XML text.

4 Going further

Of course, we have not shown all the features of XSL-FO; our goal was merely to show that the basic features are analogous to LaTeX's, even if methods for advanced features diverge. We hope you are now able to write simple texts in XSL-FO. If you wish to go thoroughly into learning it, the reference is the W3C[11] recommendation of the latest version (1.1) [16]. [10] is more didactic, but is based on XSL-FO's Version 1.0, although the differences are very slight for simple examples. [2, ch. 8] and [8] are very didactic, too, and may be of interest for French- or German-speaking people, but have the same drawback.

5 Acknowledgements

Many thanks to Jerzy B. Ludwichowski, who wrote the Polish translation of the abstract. I also thank Karl Berry and Barbara Beeton, who proofread the definitive version.

[11] **W**orld **W**ide **W**eb **C**onsortium.

References

[1] Harald Tveit ALVESTRAND: *Request for Comments: 3066. Tags for the Identification of Languages.* UNINETT, Network Working Group. March 1995. `http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc3066.html`.

[2] Bernd AMMAN et Philippe RIGAUX : *Comprendre XSLT.* Éditions O'Reilly France. Février 2002.

[3] *Apache FOP*. January 2007. `http://xmlgraphics.apache.org/fop/`.

[4] Jean-Michel HUFFLEN: "Introduction to XSLT". *Biuletyn GUST*, Vol. 22, pp. 64. In *BachoTEX 2005 conference.* April 2005.

[5] Jean-Michel HUFFLEN: "Advanced Techniques in XSLT". *Biuletyn GUST*, Vol. 23, pp. 69–75.

```
<?xml version="1.0"?>
<!DOCTYPE stylesheet [<!ENTITY layout SYSTEM "layout.fo">]>

<xsl:stylesheet version="1.0" id="poemfr0-2-fo" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:fo="http://www.w3.org/1999/XSL/Format">

  <xsl:output method="xml" indent="yes"/>

  <xsl:param name="polish-song-en" select="'Polish song'"/>
  <xsl:param name="polish-song-po" select="'Polska piosenka'"/>

  <xsl:strip-space elements="*"/>   <!--  Rule blank nodes out.  -->

  <xsl:template match="poem0">
    <fo:root>
      &layout;   <!--  The contents of this file is inserted ‘verbatim’ into the result of the XSLT program.  -->
      <xsl:variable name="the-title" select="preamble/title"/>
      <xsl:call-template name="put-footer">
        <xsl:with-param name="the-string" select="the-title"/>
      </xsl:call-template>
      <fo:page-sequence master-reference="page-simple" font-family="serif" font-size="12pt"
                        text-align="left">
        <fo:flow flow-name="xsl-region-body" xml:lang="po">
          <xsl:call-template name="put-title">
            <xsl:with-param name="the-title" select="$the-title"/>
          </xsl:call-template>
          <xsl:apply-templates select="body"/>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>

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

  <xsl:template match="stanza">
    <xsl:choose>
      <xsl:when test="resume"><xsl:apply-templates select="id(resume/@ref)"/></xsl:when>
      <xsl:otherwise>
        <xsl:apply-templates select="verse[1]">
          <xsl:with-param name="first-line-p" select="true()"/>
        </xsl:apply-templates>
        <xsl:apply-templates select="verse[position() &gt; 1]"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>

  <xsl:template match="verse">
    <xsl:param name="first-line-p" select="false()"/>   <!--  ‘false’ is the default value.  -->
    <fo:block>
      <xsl:if test="$first-line-p">
        <xsl:attribute name="space-before.minimum">10pt</xsl:attribute>
        <xsl:attribute name="space-before.optimum">11pt</xsl:attribute>
        <xsl:attribute name="space-before.maximum">12pt</xsl:attribute>
      </xsl:if>
      <xsl:value-of select="."/>
    </fo:block>
  </xsl:template>
```

Figure 6: An XSLT stylesheet that transforms the source given in Figure 5 to Figure 4..

```
  <xsl:template name="put-footer">
    <xsl:param name="the-string"/>
    <fo:static-content flow-name="xsl-region-after">
      <fo:block text-align="center" line-height="14pt" color="green" font-size="10pt"
                font-family="serif">
        <xsl:value-of select="concat($the-string,' (')"/><fo:page-number/><xsl:text>)</xsl:text>
      </fo:block>
    </fo:static-content>
  </xsl:template>

  <xsl:template name="put-title">
    <xsl:param name="the-title"/>
    <fo:block font-family="sans-serif" font-size="18pt" font-variant="small-caps"
              padding-top="3pt" text-align="center" color="white" background-color="blue"
              space-after="15pt" line-height="24pt">
      <xsl:value-of select="$the-title"/>
    </fo:block>
    <fo:block font-family="sans-serif" font-size="14pt" space-after="18pt" border-style="solid"
              border-width="0.5mm" border-color="blue" padding="4mm" start-indent="80mm"
              end-indent="4mm">
      <fo:block text-align="right">
        <xsl:value-of select="concat($polish-song-po,' ')"/>
        <fo:inline font-style="italic" xml:lang="en">
          <xsl:value-of select="concat('(',$polish-song-en,')')"/>
        </fo:inline>
      </fo:block>
    </fo:block>
  </xsl:template>

</xsl:stylesheet>
```

Figure 7: XSLT program of Figure 6, continued.

In *BachoTEX 2006 conference*. April 2006.

[6] Jean-Michel Hufflen: "Writing Structured and Semantics-Oriented Documents: TEX vs. XML". *Biuletyn GUST*, Vol. 23, pp. 104–108. In *BachoTEX 2006 conference*. April 2006.

[7] Frank Mittelbach, Michel Goossens and Johannes Braams, with David Carlisle, Chris A. Rowley, Christine Detig and Joachim Schrod: *The LATEX Companion*. 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.

[8] Manuel Montero Pineda und Manfred Krüger: *XSL-FO in der Praxis. XML-Verarbeitung für PDF und Druck*. 2004.

[9] Chuck Musciano and Bill Kennedy: *HTML & XHTML: The Definitive Guide*. 5th edition. O'Reilly & Associates, Inc. August 2002.

[10] Dave Pawson: *XSL-FO*. O'Reilly & Associates, Inc. August 2002.

[11] Sebastian Rahtz: "The TEI/TEX Interface". In: *Proc. EuroTEX 2005*, pp. 38–49. Pont-à-Mousson, France. March 2005.

[12] Erik T. Ray: *Learning XML*. O'Reilly & Associates, Inc. January 2001.

[13] The Unicode Consortium: *The Unicode Standard Version 5.0*. Addison-Wesley. November 2006.

[14] W3C: *XSL Transformations (XSLT). Version 1.0*. W3C Recommendation. Edited by James Clark. November 1999. `http://www.w3.org/TR/1999/REC-xslt-19991116`.

[15] W3C: *XML Inclusions (XInclude) Version 1.0*, 2nd edition. W3C Recommendation. Edited by Jonathan Marsh, David Orchard, and Daniel Veillard. November 2006. `http://www.w3.org/TR/2006/REC-xinclude-20061115/`.

[16] W3C: *Extensible Stylesheet Language (XSL). Version 1.1*. W3C Recommendation. Edited by Anders Berglund. December 2006. `http://www.w3.org/TR/2006/REC-xsl11-20061205/`.