Настройка оформления выходных документов в DocBook/XML

Рекомендации по освоению

Михаил Острогорский

Введение

Настройка оформления выходных документов не на шутку пугает многих технических писателей, использующих DocBook. Вместе с тем, освоить решение этой задачи необходимо, поскольку обычно фирму (работодателя или заказчика) не устраивает типовой макет, реализованный в комплекте XSLT-стилей DocBook XSL. Как правило, фирма требует, чтобы внешний вид технической документации отвечал ее официальному стилю, который заключается в определенной верстке титульного листа, колонтитулов,

врезок и других элементов. Еще одна распространенная потребность — соблюдение заданного внешнего стандарта оформления, например ГОСТ 2.105-95.

О настройке оформления документов в DocBook можно было бы написать большую книгу, в чем, к счастью, нет необходимости, потому что такая книга уже существует. Ее автор Боб Стейтон, а называется она «DocBook XSL: The Complete Guide» [1]. Частично эта книга повторяет документацию, которая входит в поставку DocBook XSL, но в ней больше примеров и разъяснений. Значительный объем книги делает ее отличным справочником, но приводит к тому, что читатель перестает видеть лес за деревьями. Описание каждого отдельного параметра понятно, но непонятно, как все это работает в целом и что делать, чтобы добиться нужного эффекта. Кроме того, некоторые элементы оформления невозможно или неудобно настраивать с помощью параметров, управление ими требует вмешательства (грамотного) в структуру DocBook XSL. Впрочем, об этом у Боба Стейтона тоже есть глава, так что можно было бы ограничиться ссылкой на всем известный, в общем-то, источник и закончить разговор, однако постоянные вопросы, как сделать то или иное изменение типового макета, наводят на мысль о потребности в каком-нибудь бодрящем документе на эту тему.

Предлагаемая статья написана в качестве краткого путеводителя для начинающих, чтобы избавить технических писателей от ужаса перед настройкой оформления выходных документов в DocBook и показать, что на самом деле там все просто. После нее читателю будет легче взяться за Боба Стейтона и оригинальную документацию, по крайней мере, автор испытывает такую надежду.

Необходимая начальная подготовка

Знания, умения и средства труда

В этой статье предполагается, что вы уже освоили технологию DocBook в целом, т. е. умеете самостоятельно набирать входные документы формата DocBook/XML и конвертировать их в выходные документы форматов HTML, CHM и PDF. Это означает, что у вас есть какой-то XSLT-процессор и вы умеете его запускать. Кроме того, вы обзавелись какой-то версией комплекта XSLT-стилей DocBook XSL и понимаете, для чего он нужен и как им пользоваться.

Правда, в жизни часто бывает иначе: кто-нибудь другой установил техническому писателю на компьютер все необходимые программы, сам их настроил и сказал: «Вот

командный файл, просто запускай его, когда надо будет сформировать выходной документ». Если у вас так и получилось, то XSLT-процессор и DocBook XSL у вас, наверное, все равно есть, но вы можете не знать, в каких каталогах файловой системы компьютера они находятся и как с ними обращаться. Попробуйте расспросить о них вашего благодетеля.

Так или иначе, использованию комплекта DocBook XSL в его первозданном состоянии недолго научиться, но это тема для отдельной статьи или небольшого учебного курса[1].

Чтобы в полной мере управлять внешним видом выходных документов, вообще говоря, требуется следующая подготовка.

- Язык разметки XSL-FO. Умение описывать на этом языке макеты документов.
- Язык преобразований XSLT. Умение писать и отлаживать *XSLT-стили*.

Если вы еще не овладели языками XSL-FO и XSLT, все равно имеет смысл дочитать эту статью до конца. По крайней мере, вы сможете свободно пользоваться предусмотренными в DocBook XSL *параметрами* типового макета выходного документа. Кроме того, вы посмотрите на примерах, как писать несложные *подменные шаблоны*. Это позволит вам настраивать оформление выходного документа в довольно широких пределах.

Язык разметки XSL-FO

XSL-FO — это язык разметки макетов. Для «печатных» документов он играет примерно ту же роль, что HTML и CSS для веб-страниц, но между ними есть два важных различия.

Во-первых, на практике документ в разметке XSL-FO, как правило, формируют из некоторого источника с помощью той или иной программы. Авторы не набирают такие документы непосредственно в «Блокноте» или в специализированных редакторах вроде DreamViewer, как это нередко случается с веб-страницами.

Во-вторых, документ формата HTML можно использовать непосредственно, например опубликовать в Интернете, а документы формата XSL-FO, как правило, нуждаются в дополнительной обработке. Чаще всего их преобразуют в формат PDF, хотя программы, которые переводят XSL-FO в PostScript, RTF и некоторые другие форматы, ориентированные на печать, тоже существуют. Программы, преобразующие документы XSL-FO в «читабельные» форматы, называются *FO-процессорами*. Существуют разные

[1] http://www.philosoft.ru/TO001.zhtml

FO-процессоры; основным критерием выбора FO-процессора, очевидно, является поддержка нужных выходных форматов, но есть и другие качества, которые мы не готовы рассматривать в данной статье.

Владение языком XSL-FO требуется для настройки оформления «печатных» документов. Если вы планируете выпускать только документы на основе формата HTML (включая контекстную справку), можно о нем спокойно забыть.

Основы XSL-FO хорошо изложены в [2], полное официальное описание приведено в [3].

Язык преобразований XSLT

XSLT — это (не пугайтесь!) язык программирования, на котором описывают правила преобразования XML-документов в другие XML-документы, HTML-документы или простой текст.

Язык XSLT представляет собой т. н. *приложение XML*. Он имеет характерный для всех языков этого типа теговый синтаксис, который должен быть знаком читателю по работе с языками DocBook/XML и XHTML. Формально XSLT можно было бы назвать языком разметки, однако по существу это было бы неверно, так как в отличие от DocBook/XML или XHTML он не описывает никаких данных.

Сфера применения языка XSLT обширна. Нам он полезен в основном тем, что позволяет сравнительно небольшими усилиями описать любое[2] преобразование документа из формата DocBook/XML в HTML и XSL-FO. Без него всякий раз приходилось бы тратить много времени на программирование синтаксического разбора разметки, что явно выходит за пределы квалификации и задач технического писателя. И хотя для описания достаточно сложных преобразований навыки программирования все-таки нужны, многие простые задачи вполне доступны обычному пользователю.

Говоря о настройке оформления документов, можно выделить два уровня владения языком XSLT. На минимальном уровне достаточно:

- знать основы XML, понимать, что такое XML-документ, элемент, атрибут;
- знать, что такое XSLT-стиль, и уметь создавать его в текстовом редакторе;
- ссылаться из XSLT-стиля на XSLT-стили DocBook XSL;
- задавать в XSLT-стиле значения параметров.

[2] Разумеется, речь идет о формальных преобразованиях документов из одного формата в другой. Для перевода на другой язык или написания романа по заданному синопсису придется искать другие средства.

Это элементарные вещи, вы научитесь им, дочитав статью примерно до середины, и сможете задавать такие свойства выходного документа, как формат листа бумаги, используемые шрифты и т. п.

Для решения более сложных задач вроде реализации собственного дизайна информационных врезок или какого-нибудь странного способа нумерации рисунков понадобится освоить написание *шаблонов*. Это ближе к программированию, хотя тоже, как говорится, не бином Ньютона.

Книга [4] представляет собой хороший (лучший из знакомых автору) учебник по XSLT. Кроме того, по XSLT выпущено много разных справочников и сборников рецептов.

Технология

Создание XSLT-стиля

XSLT-стиль — это набор правил преобразования XML-документа в другой XML-документ, HTML-документ или текст.

Технически XSLT-стиль представляет собой обычный текстовый файл. Обычно файлу XSLT-стиля присваивают расширение **xsl**, например **docbook2fo.xsl**. Файл XSLT-стиля допускается располагать в любом каталоге файловой системы. Синтаксически XSLT-стиль представляет собой XML-документ, набранный на языке XSLT.

Для создания и редактирования XSLT-стиля можно использовать любой текстовый редактор, например «Блокнот», или текстовый редактор с поддержкой редактирования кода, например Notapad++, что несколько удобнее, поскольку в нем реализованы подсветка синтаксиса и работа с отступами при наборе вложенных конструкций. Кроме того, существуют XML-редакторы с расширенной поддержкой набора элементов XSLT, например oXygen.

Текст XSLT-стиля должен иметь следующую структуру.

```
<?xml version="1.0" encoding="windows-1251"?>

<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

   <!—содержание XSLT-стиля -->

</xsl:stylesheet>
```

В самом начале располагается *инструкция* **xml**. Она указывает обрабатывающим программам, что данный файл представляет собой XML-документ и какая кодировка символов в нем используется.

Вы можете использовать (и указать в инструкции **xml**) любую нужную кодировку.

Кодировка windows-1251 включает в себя цифры, знаки препинания, латинские буквы, символы кириллицы и ряд других распространенных символов.

Кодировка UTF-8 охватывает алфавиты большинства распространенных естественных языков, математические и прочие символы.

Некоторые редакторы сами «видят», какую кодировку вы указали в директиве **xml**, и автоматически используют ее при записи файла на диск. В общем же случае следует не только указать кодировку в инструкции **xml**, но и выбрать ее при сохранении файла в текстовом редакторе.

Кодировка UTF-8 универсальна, а кодировка windows-1251 специфична для русского языка. Существует много других кодировок, в основном они специфичны для определенных языков или групп языков.

Элемент **stylesheet** заключает в себе XSLT-стиль как таковой. В нем находятся все остальные элементы XSLT-стиля, о которых будет рассказано ниже, в том числе параметры, шаблоны и ссылки на другие XSLT-стили.

Обратите внимание на выражение, которое начинается символами **xmlns**. Это так называемое определение *пространства имен*. В данном случае оно означает, что в элементе **stylesheet** (и внутри него) имена элементов, которые начинаются префиксом **xsl**, относятся к языку XSLT.

Определения пространств имен нужны для того, чтобы при использовании нескольких языков разметки в одном XML-документе не возникала путаница. Вдруг окажется, что вы используете два разных языка разметки и в них есть элементы-«тезки»? Как обрабатывающие программы будут их различать?

Дальше мы будем использовать внутри XSLT-стиля элементы языка разметки XSL-FO, указывая их имена с префиксом **fo**. Тогда мы добавим в элемент **stylesheet** определение этого пространства имен.

Способы настройки оформления

Технология DocBook предусматривает два основных способа настройки оформления выходных документов: один попроще (параметры) и один посложнее (подмена шаблонов). В большинстве реальных проектов приходится пользоваться обоими способами, поэтому лучше сразу настроиться на их совместное освоение, а не надеяться, что дело обойдется одними параметрами.

Параметры и подменные шаблоны представляют собой определенные конструкции на языке XSLT. Как именно они работают, мы поговорим дальше и покажем на примерах. Начнем с того, как в целом организовать формирование выходных документов, оформленных по вашему собственному макету.

Чтобы создать собственный макет выходного документа, вы должны сделать три вещи.

- Создать собственный XSLT-стиль для формирования выходного документа и пользоваться дальше им, а не оригинальным XSLT-стилем из комплекта DocBook XSL. Для краткости будем называть этот XSLT-стиль *авторским*.
- Из авторского XSLT-стиля сослаться (с помощью элемента **`import`**) на тот оригинальный XSLT-стиль, которым вы раньше пользовались напрямую. Например, в случае формата HTML таковым будет стиль **html\docbook.xsl** из каталога комплекта DocBook XSL.
- В авторском XSLT-стиле, во-первых, указать значения параметров и, во-вторых, если возможностей параметризации не хватает для решения всех поставленных задач, написать подменные шаблоны.

Указание параметров макета

Параметром называется именованное значение, которое интерпретируется XSLT-стилями комплекта DocBook XSL определенным образом и оказывает влияние на определенное свойство выходного документа.

Например, параметр **`chapter.autolabel`** позволяет включать и отключать автоматическую нумерацию глав в выходном документе. Если вы присвоите этому параметру значение `1`, то в выходном документе главы будут пронумерованы, а если `0`, то нет. Если вы ничего не укажете, то будут использоваться значение по умолчанию, которое равно `1` (если я ошибаюсь и на самом деле оно равно `0`, принципиально ничего не меняется).

В комплекте DocBook XSL предусмотрено много параметров, которыми можно регулировать всевозможные свойства выходного документа: шрифты, способы нумерации глав и разделов, способы нумерации рисунков, таблиц и других формальных объектов, глубину оглавления и т. п.

Перечень поддерживаемых параметров уже задан в комплекте DocBook XSL. Вы не можете добавлять свои параметры или переименовывать имеющиеся, не производя более глубоких изменений. Документация, которая поставляется в составе комплекта DocBook XSL, представляет собой, главным образом, именно справочник по параметрам.

Для разных выходных форматов наборы параметров, вообще говоря, разные. Например, для HTML и CHM бессмысленно указывать формат листа бумаги, а для PDF этот параметр крайне важен. Но в том, что касается отражения структуры документа (той же нумерации, скажем), наборы параметров для разных выходных форматов совпадают.

Синтаксис указания параметра таков:

```
<xsl:param name="chapter.autolabel" select="1"/>
```

либо

```
<xsl:param name="chapter.autolabel">1</xsl:param>.
```

Значения параметров можно задавать не только в авторском XSLT-стиле, но и непосредственно в командной строке при запуске XSLT-процессора. В некоторых случаях это удобно, но если надо задать много параметров, командная строка становится чересчур громоздкой. Ниже приведен пример указания параметров при запуске XSLT-процессора. В примере используется XSLT-процессор xsltproc.exe из библиотеки Libxml2[3]. Учтите, что у каждого XSLT-процессора может быть свой порядок указания аргументов.

```
xsltproc.exe –o manual.html --param chapter.autolabel 0
➥docbook-xsl/html/docbook.xsl manual.xml
```

Подмена шаблонов

Как правило, XSLT-стиль состоит из одного или нескольких *шаблонов*.

Напомним, что в языке XSLT под шаблоном понимается не заготовка документа, как в текстовых процессорах. В языке XSLT шаблон — это правило преобразования выбранного по заданному критерию элемента входного XML-документа в соответствующий ему элемент или фрагмент выходного документа.

[3] http://xmlsoft.org

Например, легко можно представить себе шаблон, который преобразует элемент **ulink** языка DocBook в элемент **a** языка HTML. Он «объясняет» XSLT-процессору, что конструкция вида

```
<ulink url="www.bugaga.com">Прикольные ржаки</ulink>
```

должна превратиться в конструкцию

```
<a href="www.bugaga.com">Прикольные ржаки</a>.
```

На языке XSLT это правило можно выразить следующим образом[4]:

```
<!-- Начало шаблона -->
<!-- атрибут match указывает, какие элементы
     обрабатывает этот шаблон -->
<xsl:template match="ulink">
   <!-- формируем элемент языка HTML a -->
   <a>
      <!-- формируем атрибут href элемента a -->
      <xsl:attribute name="href">
         <!-- переносим в атрибут href значение атрибута url -->
         <xsl:value-of select="@url"/>
      </xsl:attribute>
      <xsl:apply-templates/>
   </a>
</xsl:template>
```

Такой шаблон действительно существует внутри комплекта DocBook XSL, правда, он несколько сложнее, чем мы здесь описали. Его текст находится в файле **html\xref.xsl**. Этот файл включен в файл **html\docbook.xsl** посредством элемента **include** и поэтому, с точки зрения XSLT-процессора, является его частью.

Что произойдет, если мы напишем в авторском XSLT-стиле другой шаблон с таким же значением атрибута **match**? Транслятор какого-нибудь «добропорядочного» языка программирования вывел бы сообщение об ошибке, мол, у вас, дорогие товарищи, получилась путаница: одно и то же описано два раза по-разному. Но XSLT-процессор ведет себя иначе. Из двух вариантов шаблона он выберет тот, который описан выше по иерархии импорта, т. е. в данном случае он применит к входному документу шаблон из авторского XSLT-стиля, а не из оригинального XSLT-стиля, который входит в комплект DocBook XSL.

Что это нам дает? Возможность вносить изменения в работу оригинального XSLT-стиля, не исправляя (а во многих случаях даже не читая предварительно) текст последнего. Если

[4] Читатель, знакомый с программированием, наверно согласится, что написание такого преобразования на любом алгоритмическом языке, включая самые передовые, потребовало бы значительных усилий по разбору входной строки и обработке возможных в ней синтаксических ошибок.

нас не устраивает, как работает тот или иной шаблон из комплекта DocBook XSL, мы можем подменить его, написав в авторском XSLT-стиле собственный шаблон с таким же значением атрибута **match**. Так, если нам не нравится, как обрабатываются гипертекстовые ссылки, оформленные во входном документе с помощью элемента **ulink**, мы просто пишем в авторском XSLT-стиле шаблон с атрибутом **match="ulink"**. Подчеркнем еще раз, мы можем сделать это, ничего не зная о файле **html\xref.xsl** и его содержимом.

Как мы увидим в примере «Фирменный титульный лист», описанный метод работает и в том случае, когда шаблон идентифицируется значением атрибута **name**, а не **match**.

Примеры

Конфигурация для работы примерами

Договоримся, что конфигурация, в которой мы работаем с примерами, устроена самым примитивным образом. У нас есть один рабочий каталог, в котором находятся:

- исходный файл документа, допустим, **manual.xml**;
- подкаталог **images**, в котором находятся все рисунки;
- подкаталог **docbook-xsl**, в котором находится комплект DocBook XSL.

В этом же рабочем каталоге мы будем создавать файлы авторских XSLT-стилей. Далее они будут фигурировать здесь под именем **my-docbook2fo.xsl**.

В системе имеется XSLT-процессор **xsltproc.exe** из библиотеки Libxml2. Он хранится в каком-то другом каталоге, но путь к нему прописан в переменной среды **PATH**, т. е. мы можем запустить его командой **xsltproc** со всеми положенными аргументами, не указывая путь к нему явно.

В системе имеется FO-процессор, например RenderX XEP[5]. Его тоже можно запускать непосредственно, без указания пути. Также FO-процессор настроен для работы с кириллицей.

При работе с примерами рабочий каталог является текущим каталогом.

[5] http://www.renderx.com. На момент завершения работы над данной статьей этот FO-процессор поддерживает выходные форматы PDF и PostScript. По адресу http://www.xmlmind.com доступен FO-процессор, который поддерживает выходной формат RTF и ряд других форматов текстовых процессоров.

В примерах будем рассматривать настройку оформления для выходных документов в формате PDF (или других, формируемых на основе FO), поскольку их внешнему виду обычно уделяют особое внимание. Выходной документ в формате PDF мы формируем в два этапа:

I. Преобразуем входной документ из DocBook/XML в разметку XSL-FO. Получаем промежуточный выходной документ, который будет называться, например, **manual.fo**. Для этого используем XSLT-процессор, комплект DocBook XSL и авторский XSLT-стиль.

```
xsltproc.exe –o manual.fo my-docbook2fo.xsl manual.xml
```

II. Преобразуем полученный документ XSL-FO в формат PDF с помощью FO-процессора.

```
xep.bat –valid manual.fo
```

(Выходной формат в данном случае можно не указывать явно, так как по умолчанию FO-процессор RenderX XEP формирует PDF.)

Простейшие параметры оформления

Создадим файл XSLT-стиля my-docbook2fo.xsl и зададим в нем несколько параметров оформления выходного документа.

Файл my-docbook2fo.xsl

```
<?xml version="1.0" encoding="windows-1251"?>

<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

   <!-- Импорт оригинального стиля -->
   <xsl:import href="docbook-xsl/fo/docbook.xsl"/>

   <!-- Указываем формат листа бумаги -->
   <xsl:param name="paper.type" select="'A4'"/>

   <!—Для основного текста использовать шрифт с засечками -->
   <xsl:param name="body.font.family" select="'serif'"/>

   <!-- Для заголовков использовать шрифт без засечек -->
   <xsl:param name="title.font.family" select="'sans-serif'"/>

   <!-- Сколько уровней section включать в оглавление -->
   <xsl:param name="toc.section.depth" select="2"/>

</xsl:stylesheet>
```

Обратите внимание на апострофы внутри кавычек. Текстовые значения параметров в атрибуте **`select`** должны быть указаны именно таким образом.

Оформление выделенного текста

Предположим, нам требуется, чтобы обозначения кнопок были выделены в тексте полужирным шрифтом и темно-зеленым цветом. Для этого напишем в авторском XSLT-стиле подменный шаблон, обрабатывающий элемент **guibutton**.

В этом примере мы используем элементы языка разметки XSL-FO. Образующие их открывающие и закрывающие теги имеют префикс **fo**.

Обратите внимание, что в элементе **xsl:stylesheet** указано пространство имен fo. В предыдущем примере мы этого не делали, потому что не пользовались элементами FO непосредственно.

Файл my-docbook2fo.xsl

```
<?xml version="1.0" encoding="windows-1251"?>

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

   <!-- Импорт оригинального стиля -->
   <xsl:import href="docbook-xsl/fo/docbook.xsl"/>

   <!-- Шаблон для обработки элемента guibutton -->
   <xsl:template match="guibutton">

      <!-- Создаем в выходном документе строковый элемент,
           задаем в нем оформление текста с помощью атрибутов -->
      <fo:inline color="#008800" font-weight="bold">

         <!-- Переносим содержимое элемента guibutton
              внутрь элемента fo:inline. При этом обрабатываем
              его всеми шаблонами, которые предусмотрены для
              него в DocBook XSL -->
         <xsl:apply-templates/>

      </fo:inline>

   </xsl:template>

</xsl:stylesheet>
```

Фирменный титульный лист

В типовом макете, который реализован в комплекте DocBook XSL, на титульном листе (точнее, на его лицевой стороне) печатаются заголовок и подзаголовок документа. При этом в качестве заголовка можно указать, например, название продукта, а в качестве подзаголовка тип документа. Получается примерно так, как показано на рис. 1.

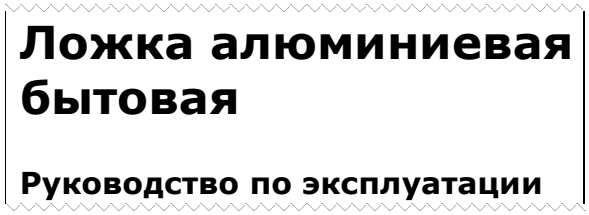

Рисунок 1. Типовой титульный лист, лицевая сторона

Во многих случаях этого недостаточно. Кроме заголовка и подзаголовка на титульный лист обычно помещают:

- название фирмы-разработчика и ее логотип;
- номер версии, модели или исполнения продукта;
- изображение или скриншоты продукта;
- место и год выпуска документа (например, Москва, 2011);
- всевозможные декоративные элементы: плашки, рамки и т. п.

Титульный лист состоит из двух страниц: лицевой и оборотной. Лицевая страница называется “recto”, а оборотная — “verso”. Соответственно, шаблон, который формирует лицевую страницу, носит имя **`book.titlepage.recto`**, а оборотную — **`book.titlepage.verso`**. Написав в авторском XSLT-стиле подменные шаблоны с такими именами, вы сможете заменить типовой титульный лист собственным. Простейший пример (в стиле “Hello, World!”) приведен ниже.

Файл my-docbook2fo.xsl

```
<?xml version="1.0" encoding="windows-1251"?>

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

   <!-- Импорт оригинального стиля -->
   <xsl:import href="docbook-xsl/fo/docbook.xsl"/>

   <!-- Шаблон для формирования лицевой стороны титульного листа -->
   <xsl:template name="book.titlepage.recto">

      <!-- Блок с надписью -->
      <fo:block font-size="24pt" font-weight="bold">

         <xsl:text>Мой титульный лист! Гоп-ца-ца!</xsl:text>
      </fo:block>

   </xsl:template>

</xsl:stylesheet>
```

Для оборотной стороны титульного листа необходимо сделать то же самое, только имя шаблона должно быть **`book.titlepage.verso`**.

Вы можете написать шаблоны для титульного листа самостоятельно. Дальше покажем, как это сделать. Кроме того, можно воспользоваться инструментарием для описания макета титульного листа, который входит в поставку комплекта DocBook XSL. Этот инструментарий состоит из XML-файла **`fo\titlepage.templates.xml`** (для других форматов имеются аналогичные файлы в соответствующих подкаталогах) и XSLT-стиля **`template\titlepage.xsl`**. Первый представляет собой описание макета титульного листа. Вы можете его модифицировать, приспособив к собственным нуждам. Дальше вы применяете к нему второй и получаете шаблоны для вставки в авторский XSLT-стиль. Вставка полученных шаблонов в авторский XSLT-стиль делается вручную в текстовом редакторе. Язык описания макета титульного листа и технология в целом подробно описаны у Боба Стейтона. Но вернемся к примеру. Предположим, нам требуется формировать титульный лист по образцу на рис. 2.

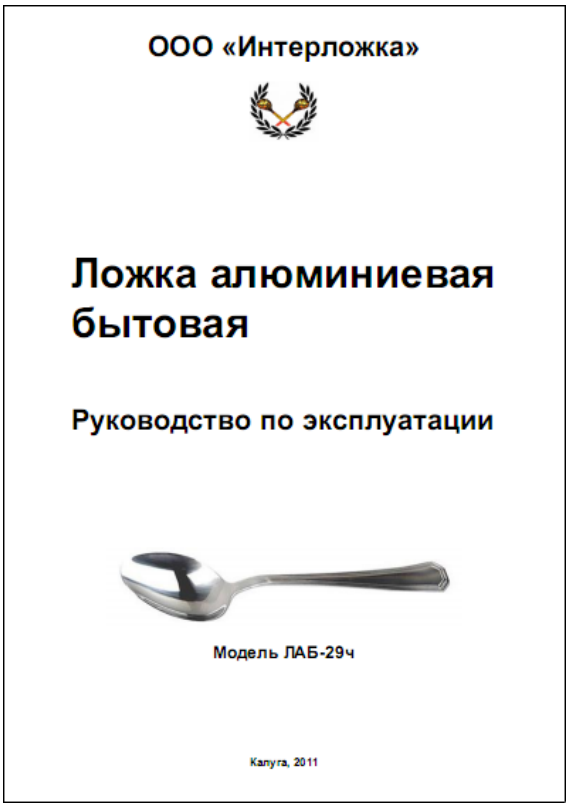

Рисунок 2. Пример макета лицевой стороны титульного листа

Название фирмы, название модели, город и год вводятся техническим писателем во входном документе. Для этого используются соответствующие дочерние элементы

элемента **bookinfo** (см. листинг). Изображения постоянны, они хранятся в подкаталоге **images** рабочего каталога и имеют заведомо известные имена файлов. Пример авторского XSLT-стиля, формирующего документ с титульным листом заданного вида, приведен ниже.

Файл my-docbook2fo.xsl

```
<?xml version="1.0" encoding="windows-1251"?>

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

   <!-- Импорт оригинального XSLT-стиля -->
   <xsl:import href="docbook-xsl/fo/docbook.xsl"/>

   <!-- Указываем шрифт, который будет использоваться в качестве
        шрифта без засечек -->
   <xsl:param name="sans.font.family">Verdana</xsl:param>

   <!-- Указываем формат листа бумаги  -->
   <xsl:param name="paper.type" select="'A4'"/>

   <!-- Шаблон для формирования лицевой стороны титульного листа -->
   <xsl:template name="book.titlepage.recto">

      <!-- Название компании (из bookinfo/corpname) -->
      <fo:block-container absolute-position="fixed"
                          top="1cm"
                          left="2cm"
                          right="2cm">
         <fo:block text-align="center"
                   font-family="sans-serif"
                   font-size="30pt"
                   font-weight="bold">
            <xsl:value-of select="//bookinfo/corpname/node()"/>
         </fo:block>
      </fo:block-container>

      <!-- Логотип -->
      <fo:block margin-top="0.3cm" text-align="center">
         <fo:external-graphic src="url('sources/images/logo.png')"/>
      </fo:block>

      <!-- Заголовок -->
      <fo:block margin-top="4cm"
                text-align="left"
                hyphenate="false"
                font-family="sans-serif"
                font-size="44pt"
                font-weight="bold">
         <xsl:apply-templates select="//book/title/node()"/>
      </fo:block>

      <!-- Подзаголовок -->
      <fo:block margin-top="2cm"
                font-family="sans-serif"
```

```
            font-size="30pt"
            font-weight="bold">
         <xsl:apply-templates select="//book/subtitle/node()"/>
      </fo:block>

      <!-- Изображение продукта -->
      <fo:block margin-top="4cm" text-align="center">
         <fo:external-graphic src="url('sources/images/spoon.jpg')"/>
      </fo:block>

      <!-- Название модели (из bookinfo/productname) -->
      <fo:block margin-top="0.5cm"
            text-align="center"
            font-family="sans-serif"
            font-size="18pt"
            font-weight="bold">
         <xsl:value-of select="'Модель '"/>
         <xsl:value-of select="//bookinfo/productname"/>
      </fo:block>

      <!-- Город, год (из bookinfo/address/city
           и bookinfo/date) -->
      <fo:block-container absolute-position="fixed"
                          top="28cm" left="2cm"
                          right="2cm">
         <fo:block text-align="center"
                   font-family="sans-serif"
                   font-size="12pt"
                   font-weight="bold">
            <!-- город -->
            <xsl:value-of select="//bookinfo/address/city"/>
            <!-- запятая между городом и годом -->
            <xsl:value-of select="', '"/>
            <!-- год -->
            <xsl:value-of select="substring(//bookinfo/date, 7)"/>
         </fo:block>
      </fo:block-container>

   </xsl:template>

</xsl:stylesheet>
```

Обладая развитым эстетическим чувством и достаточными знаниями языков XSLT и FO, вы сможете сделать сколь угодно изысканный титульный лист.

Литература

[1] Bob Stayton. DocBook XSL: The Complete Guide. http://www.sagehill.net/docbookxsl/

[2] И. Ш. Хабибуллин. Самоучитель XML. — СПб.: БХВ-Петербург, 2003. — 336 с.: ил.

[3] Extensible Stylesheet Language (XSL). Version 1.0. W3C Recommendation. http://www.w3.org/TR/2001/REC-xsl-20011015/

[4] А. Н. Валиков. Технология XSLT. — СПб.: БХВ-Петербург, 2002. — 544 с.: ил.