Zpravodaj Československého sdružení uživatelů TeXu

Jiří Kosek

DocBook a generování rejstříků

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 14 (2004), No. 3-4, 125–135

Persistent URL: http://dml.cz/dmlcz/149966

DocBook a generování rejstříků

Jiří Kosek

DocBook je dnes již považován za zcela standardní formát dokumentace. Dobrá dokumentace se ovšem neobejde bez dobrého rejstříku. Příspěvek posluchače seznámí s možnostmi DocBooku a XSL stylů pro generování rejstříku. Pozornost bude věnována i dodržování českých specifik při řazení a seskupování rejstříkových hesel. Budou ukázány techniky, jak s využitím standardních nástrojů generovat několik rejstříků v dokumentu a jak rejstříky automaticky vytvářet na základě sémantického značkování.

DocBook se stává stále populárnějším formátem pro vytváření dokumentace k mnoha softwarovým projektům, psaní článků, knih, učebních textů, skript apod. Obliba roste zejména díky aplikacím podporujícím DocBook. Editory XML dokumentů jsou stále pohodlnější a dokonce již existují WYSIWYG editory, které jsou zdarma[1]. Nástroje pro zpracování DocBooku a zejména nejpoužívanější XSL styly[2] umožňují dokument převádět do mnoha výstupních formátů a podobu výstupu přitom ovlivnit řadou parametrů.

Užitnou hodnotu dokumentů, zvláště tištěných, výrazně zvyšuje kvalitní rejstřík. Vytvoření dobrého rejstříku je velmi pracné a zodpovědné, a proto bývá často svěřeno do rukou specialisty. Bohužel v podmínkách dokumentace k open-source projektům, samizdatových publikací a dokonce i knih určených pro malý český trh se finanční a časové zdroje na práci rejstříkového specialisty většinou nedostávají. Proto se v tomto příspěvku podíváme na možnosti tvorby rejstříků v DocBooku tak, aby si každý autor dokumentu mohl sestavit rejstřík sám. Ukážeme si i pokročilejší možnosti, jako více rejstříků k jednomu dokumentu. Nakonec si ukážeme, jak snadno a automaticky doplnit rejstřík do dokumentů, které používají sémantické značkování k vyznačení objektů jako jsou názvy funkcí, jména souborů apod.

Vyznačování rejstříkových hesel

Rejstříková hesla se zapisují přímo do dokumentu pomocí elementu `indexterm`. Jeho obsah se v dokumentu nezobrazuje, ale slouží jako podklad pro generování rejstříku.

```
<para>Bohatství moderních společností je založeno na
informacích<indexterm><primary>informace</primary></indexterm>.</para>
```

[1] `http://xmlmind.com/xmleditor/`

[2] `http://docbook.sf.net`

Do elementu `indexterm` se zapisují hesla a to i víceúrovňová:

```
<indexterm>
<primary>informace</primary>
</indexterm>

<indexterm>
<primary>informace</primary>
<secondary>získání</secondary>
</indexterm>

<indexterm>
<primary>informace</primary>
<secondary>šíření</secondary>
</indexterm>

<indexterm>
<primary>informace</primary>
<secondary>šíření</secondary>
<tertiary>ústní</tertiary>
</indexterm>
```

V rejstříku se pak takto definovaná hesla objeví například jako:

```
informace, 13
  šíření, 17
    ústní, 25
  získání, 15
```

Pokud nějakému termínu odpovídá větší úsek dokumentu, můžeme ho celý pokrýt jako rozsah. Použijí se dva elementy `indexterm`, které označují začátek a konec platnosti určitého hesla.

```
<indexterm class="startofrange" id="ix.xml.historie">
<primary>XML</primary>
<secondary>historie</secondary>
</indexterm>
  ...
<indexterm class="endofrange" startref="ix.xml.historie"/>
```

Ve vygenerovaném rejstříku pak dostaneme interval:

```
XML
  historie, 27--42
```

Pokud chceme, aby se položka řadila nestandardním způsobem, použijeme atribut `sortas`. Třídění se pak provede podle jeho obsahu, ne podle skutečného textu hesla. To je výhodné v případech, kdy heslo obsahuje nějaké speciální znaky apod. Následující příklad v rejstříku zobrazí písmeno Ω, ale bude se řadit jako text „Omega“.

```
<indexterm>
<primary sortas="Omega">&Omega;</primary>
</indexterm>
```

Chceme-li některé výskyty hesla v rejstříku zvýraznit (například mít stránku s primární definicí hesla tučně), můžeme u hesla nastavit jeho důležitost.

```
<indexterm significance="preferred">
<primary>informace</primary>
</indexterm>
```

Nechceme-li, aby rejstříkové heslo obsahovalo odkaz na konkrétní číslo strany, ale odkaz na jiné heslo, můžeme k tomu využít elementy `see` a `seealso`.

```
<indexterm>
<primary>DTD</primary>
</indexterm>

<indexterm>
<primary>definice typu dokumentu</primary>
<see>DTD</see>
</indexterm>

<indexterm>
<primary>XML schéma</primary>
<seealso>DTD</seealso>
</indexterm>
```

Ve výsledném rejstříku bychom dostali:

```
- D -
definice typu dokumentu, viz DTD
DTD, 42

- X -
XML schéma, 81, viz též DTD
```

Tímto jsme se seznámili skoro se všemi možnostmi zápisu rejstříkových hesel v DocBooku. Nezmínili jsme pouze možnost uložit definici rejstříkových hesel zcela mimo místo jejich výskytu s využitím atributu `zone`. Tato metoda není dle mého názoru příliš praktická, více se o ní můžete dočíst v [4].

Generování rejstříku

XSL styly pro DocBook generují rejstřík zcela automaticky. Na místě, kde chceme mít rejstřík, stačí uvést element `index`. Ten je při zpracování automaticky nahrazen rejstříkem. Pro generování rejstříku tak není potřeba dokument zpracovávat opakovaně, jak to známe např. ze systému TEX a makeindex, nebo z DSSSL stylů.

Při generování rejstříku se přitom samozřejmě respektují možnosti použitého výstupního formátu. Rejstřík v HTML stránce tak neobsahuje čísla stran, ale názvy sekcí či kapitol, ve kterých se heslo vyskytuje. Názvy zároveň slouží jako hypertextové odkazy, které dokáží skočit na místo výskytu rejstříkového hesla.

Při výstupu do HTML Helpu se z rejstříkových hesel vytvoří přímo rejstřík na úrovni HTML Helpu.

Lehce problematický je však tištěný výstup prováděný přes formátovací objekty do PDF. Princip generování rejstříku dokumentu XML v XSL je dvoufázový proces [2]. V první fázi se pomocí XSLT dokument přetransformuje na formátovací objekty, které abstraktním způsobem popisují vzhled tištěného dokumentu. Čísla stran v rejstříku v tuto chvíli nejsou a nemohou být vyhodnocena. K samotnému zalomení textu do stránek a vyhodnocení čísel stránek dojde až během následující fáze formátování. Stene-li se však, že jedno rejstříkové heslo se vyskytuje na jedné stránce dvakrát, objeví se číslo této stránky ve výstupu duplicitně. Tento velmi nepříjemný nedostatek lze naštěstí řešit několika způsoby, o kterých se zmíníme v další části článku.

Dalším problémem je generování rejstříků pro jiné jazyky, než je angličtina. Generování rejstříku odpovídá seskupení všech rejstříkových hesel do skupin podle jejich prvního písmena, seřazení skupin podle abecedy a seřazení hesel v jedné skupině podle abecedy. Přesně tento algoritmus implementují i XSL styly. Pro češtinu je však nedostatečný. Písmeno „ch“ je složeno ze dvou znaků, ale přitom tvoří samostatnou skupinu. Slova začínající na „e“ a „é“ patří do jedné skupiny, kdežto slova začínající na „c“ a „č“ patří do dvou různých skupin.

Implementace korektního českého rejstříku není jednoduchá. Nástroje nabízené standardem XSLT pro seskupování jsou velmi slabé, a i řazení pro jednoduché jazyky jako angličtina je výzva. Navíc jsou styly pro DocBook psány tak, aby podporovaly dokumenty v různých jazycích. Každý jazyk má odlišná pravidla pro řazení a seskupování rejstříkových hesel. Jazyk XSLT však nenabízí dostatečné možnosti pro parametrizaci seskupovacího kódu na základě jazyka. Lze však využít rozšíření, které některé implementace XSLT nabízejí a dosáhnout tak požadovaného výsledku. K problematice generování rejstříku podle českých zvyklostí se také ještě vrátíme.

Odstranění duplicitních čísel stran v rejstříku

Jak jsme již zmínili, současné verze jazyků XSLT a XSL-FO nenabízejí podporu pro eliminování duplicitních čísel stran v rejstříku. Problém jde obejít dvěma způsoby. Prvním z nich je využití procesoru FO, které implementuje rozšíření XSL-FO pro generování rejstříků. Druhá možnost spočívá ve víceprůchodovém zpracování dokumentu, během kterého se detekují a následně odstraní duplicity.

Rozšíření pro generování rejstříků obsahují dva nejpoužívanější a nejlepší komerční procesory FO – XEP[3] a XSL Formatter[4]. Styly pro DocBook tato rozšíření podporují, stačí pomocí parametru říci, že se mají použít. Např. XEP by se pak spouštěl pomocí parametrů:

```
xep -xml dokument.xml -xsl .../fo/docbook.xsl -param xep.extensions=1
```

V praxi však většinou parametrů nastavujeme více, a proto je praktické vytvořit si styl s úpravami (podrobněji viz např. [3] a [1]). Tento styl nejprve naimportuje standardní styly a pak provede potřebné změny v nastavení parametrů.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">

<xsl:import
  href="http://docbook.sourceforge.net/release/xsl/current/fo/docbook.xsl"/>

<xsl:param name="paper.type" select="'A4'"/>
<xsl:param name="xep.extensions" select="1"/>

</xsl:stylesheet>
```

Pro XSL Formatter se odpovídající parametr jmenuje `axf.extensions`. Nastavení parametrů způsobí odstranění duplicitních čísel stran a vytvoření intervalů ze sekvencí po sobě jdoucích stránek. Vyskytuje-li se jedno rejstříkové heslo na stranách

```
5, 5, 8, 9, 10, 37
```

dostaneme na výstupu mnohem lepší podobu

```
5, 8--10, 37
```

V budoucnu nebude nutné přizpůsobovat výstup stylů použitému procesoru, protože příští verze XSL-FO 1.1 bude již přímo obsahovat podporu pro generování rejstříků[5].

Používáme-li jiný procesor než XEP nebo XSL Formatter, musíme pro odstranění duplicit podstoupit mnohem složitější proces. Tento postup je nutné využít například společně s procesorem FOP[6], který je jednou z mála alespoň částečně použitelných open-source implementací XSL-FO. Nejprve dokument zpracujeme se zapnutým parametrem `make.index.markup`. To způsobí, že rejstřík bude

[3] `http://www.renderx.com/`

[4] `http://www.antennahouse.com/`

[5] `http://www.w3.org/TR/2003/WD-xsl11-20031217/#d0e12534`

[6] `http://xml.apache.org/fop/`

v PDF dokumentu obsahovat okolo hesel a čísel stran značkování. PDF dokument pak můžeme převést na čistý text, ze značek extrahovat čísla stran, odstranit duplicity a nově vzniklý rejstřík natvrdo začlenit do dokumentu. Postup je to nepohodlný a pro češtinu asi nebude fungovat zcela spolehlivě, protože FOP neumí do PDF vkládat korektní mapování z použitého písma do Unicode. Při extrakci čistého textu z PDF tak dostaneme poškozený text. Není to však velké omezení, protože pro kvalitní výstup je FOP stejně nepoužitelný.

Seskupování hesel podle českých zvyklostí

DocBookové styly přizpůsobují svůj výstup jazyku, ve kterém je dokument napsán. Aktivní jazyk je přitom možné určit pomocí atributu `lang`.

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE book PUBLIC '-//OASIS//DTD DocBook XML V4.2//EN'
                      'http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd'>
<book lang="cs">
  ...
</book>
```

Díky již dříve zmíněným omezením jazyka XSLT však nemohou styly vzít aktivní jazyk v úvahu při generování rejstříku. Naštěstí některé z XSLT procesorů umožňují definice uživatelských funkcí, pomocí kterých již lze implementovat seskupování závislé na jazyku. Protože tato část stylů používá nestandardní instrukce XSLT a mohla by způsobit problémy s kompatibilitou, není zahrnuta do standardního stylu.

Chceme-li český rejstřík generovat, musíme používat XSLT procesor, který podporuje definování uživatelských funkcí podle EXSLT[7] a tyto funkce lze použít v definici klíče (`xsl:key`). Těmto kritériím vyhovuje například Saxon. V době psaní článku však ještě nešlo použít xsltproc, protože obsahoval nějaké chyby spojené právě s inicializací klíčů obsahujících uživatelsky definované funkce.

Stačí si pak vytvořit styl s úpravami, který ve standardních stylech předefinuje některé šablony generující rejstřík. Kód již je hotový v souboru `autoidx-ng.xsl` a stačí jej proto sloučit se standardními styly.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">

<xsl:import
  href="http://docbook.sourceforge.net/release/xsl/current/fo/docbook.xsl"/>
<xsl:include
  href="http://docbook.sourceforge.net/release/xsl/current/fo/autoidx-ng.xsl"/>

<!-- Další úpravy, nastavení parametrů -->
</xsl:stylesheet>
```

[7] `http://www.exslt.org`

Úpravy kódu pro generování rejstříku jsou dostupné i pro HTML výstup, opět v souboru `autoidx-ng.xsl` v odpovídajícím adresáři.

Takto upravené styly přiřadí hesla správně do skupiny, a skupiny seřadí správně podle české abecedy. Je ošetřena i problematika písmena „ch“. Řazení hesel v jedné skupině je ponecháno na XSLT procesoru. Saxon[8] běžně české řazení nezvládá, ale můžeme jej o podporu českého řazení snadno rozšířit. Stačí, když do javové cesty (`CLASSPATH`) přidáme zkompilovanou třídu `Compare_cs`, jejíž kód je velmi jednoduchý.

```
package com.icl.saxon.sort;

import java.text.Collator;
import java.util.Locale;

public class Compare_cs extends TextComparer
{

    int caseOrder = UPPERCASE_FIRST;

    public int compare(Object a, Object b)
    {

        Collator csCollator = Collator.getInstance(new Locale("cs", "cz"));
        return csCollator.compare(a, b);
    }

    public Comparer setCaseOrder(int caseOrder)
    {
        this.caseOrder = caseOrder;
        return this;
    }

}
```

Více rejstříků v jednom dokumentu

V některých typech publikací je zapotřebí několika samostatných rejstříků – např. předmětného a jmenného. XSL styly si s tímto požadavkem snadno poradí. Při zápisu rejstříkového hesla stačí v atributu `role` uvést námi zvolený identifikátor rejstříku.

```
<para>Bohatství moderních společností je založeno na
informacích.<indexterm role="subj"><primary>informace</primary></indexterm>
O rozvoj informační teorie se ve 40. letech zasloužil Claude Shannon.
<indexterm role="name"><primary>Shannon, Claude</primary></indexterm></para>
```

[8] Máme teď na mysli verzi 6.5.3, která se používá s XSL styly pro DocBook, ne verzi 7.x, která implementuje návrh XSLT 2.0.

Na konec dokumentu pak vložíme dva elementy `index`, u kterých určíme jaká hesla do nich mají spadat.

```
<index role="subj"/>

<index role="name">
<title>Jmenný rejstřík</title>
</index>
```

Zda se bude generovat několik rejstříků podle obsahu atributu `role` ovlivňuje parametr `index.on.role`, který je standardně zapnutý.

Ze sémantiky až do rejstříku

Výhoda DocBooku a XML obecně oproti jiným technologiím pro přípravu dokumentů je možnost velmi jemného přiřazovaní významu jednotlivým částem textu. DocBook obsahuje několik desítek sémantických elementů, které umožňují odlišit jména souborů, od názvů funkcí, příkazů atd. Podívejme se na ukázku odstavce, který takto sémantické značkování používá.

```
<para>Příkaz <command>rm</command> je užitečný, ale používejte
jej opatrně. Některé soubory jako například
<filename>/etc/passwd</filename> jsou pro váš systém poměrně důležité.</para>
```

U větších tištěných příruček je velmi užitečné, pokud se všechny důležité termíny vyskytují v rejstříku. Mezi tyto termíny mohou v některých příručkách patřit i názvy příkazů nebo souborů. Pro zařazení hesel z předchozího příkladu do rejstříku bychom museli ručně doplnit odpovídající značkování.

```
<para>Příkaz <command>rm</command><indexterm><primary>rm</primary></indexterm>
<indexterm><primary>příkaz</primary><secondary>rm</secondary></indexterm>
je užitečný, ale používejte jej opatrně. Některé soubory jako například
<filename>/etc/passwd</filename>
<indexterm><primary>/etc/passwd</primary></indexterm>
jsou pro váš systém poměrně důležité.</para>
```

V rejstříku bychom pak dostali následující hesla

```
- Symboly -
/etc/passwd, 42

- P -
příkaz,
  rm, 42

- R -
rm, 42
```

Výsledek je sice užitečný, ale po pravdě řečeno, kdo by chtěl do vstupního dokumentu zapisovat mnoho redundantní informace. V tomto případě je naštěstí mapování sémantických značek na rejstříková hesla velmi jednoduché a

jednoznačné, takže je můžeme snadno algoritmizovat. Není problém napsat jednoduchý XSLT styl, který do dokumentu obsahujícího pouze sémantické značky doplní rejstříková hesla podle našich požadavků.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">

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

<!-- Příkazy se dají do rejstříku na dvě místa -->
<xsl:template match="command">
  <!-- Zkopírování původního elementu -->
  <xsl:copy-of select="."/>
  <!-- Vytvoření rejstříkových hesel -->
  <indexterm>
    <primary><xsl:value-of select="."/></primary>
  </indexterm>
  <indexterm>
    <primary>příkaz</primary>
    <secondary><xsl:value-of select="."/></secondary>
  </indexterm>
</xsl:template>

<!-- Každé jméno souboru se také přidá do rejstříku -->
<xsl:template match="filename">
  <!-- Zkopírování původního elementu -->
  <xsl:copy-of select="."/>
  <!-- Vytvoření rejstříkového hesla -->
  <indexterm>
    <primary><xsl:value-of select="."/></primary>
  </indexterm>
</xsl:template>

</xsl:stylesheet>
```

Zpracujeme-li nyní náš dokument tímto stylem, dostaneme dočasný dokument, kde budou pro všechny příkazy a jména souborů doplněná rejstříková hesla. Dočasný soubor pak můžeme zpracovat běžnými XSL styly pro DocBook. Celý proces generování dočasného dokumentu a jeho následného zpracování si můžeme zautomatizovat pomocí makefile, dávkových souborů nebo podobné techniky.

Docbookové styly nám však nabízejí možnost výše zmíněného automatického doplnění rejstříkových hesel přímo během zpracování dokumentu styly. Není

proto potřeba vůbec využívat dočasný soubor a zpracovávat dokument dvěma průchody.

Řešení využívá vlastnosti stylů, které se říká profilování. Profilování umožňuje podmíněně zpracovávat jen určité části dokumentu na základě hodnot uložených v atributech. V dokumentu tak můžeme např. označit, že některé kapitoly jsou určené pro začátečníky používající náš program v Linuxu a některé zase pro pokročilé uživatele Windows. Styly pro DocBook při zpracování dokumentu nejprve provedou odfiltrování nepotřebných částí dokumentu a pak provedou klasické zpracování. Filtrování je přitom interně implementováno jako speciální režim, ve kterém se provádí podmíněné kopírování dokumentu. Protože se profilování obvykle provádí pro elementy jako kapitola a podkapitola a málokdy pro sémantické inline elementy, můžeme v tomto režimu přidat šablony, které se kromě profilování postarají o doplnění rejstříkových hesel právě pro sémantické elementy.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">

<!-- Naimportování původního stylu -->
<xsl:import
href="http://docbook.sourceforge.net/release/xsl/current/fo/profile-docbook.xsl"/>

<!-- Příkazy se dají do rejstříku na dvě místa -->
<xsl:template match="command" mode="profile">
  <!-- Zkopírování původního elementu -->
  <xsl:copy-of select="."/>
  <!-- Vytvoření rejstříkových hesel -->
  <indexterm>
    <primary><xsl:value-of select="."/></primary>
  </indexterm>
  <indexterm>
    <primary>příkaz</primary>
    <secondary><xsl:value-of select="."/></secondary>
  </indexterm>
</xsl:template>

<!-- Každé jméno souboru se také přidá do rejstříku -->
<xsl:template match="filename" mode="profile">
  <!-- Zkopírování původního elementu -->
  <xsl:copy-of select="."/>
  <!-- Vytvoření rejstříkového hesla -->
  <indexterm>
    <primary><xsl:value-of select="."/></primary>
  </indexterm>
</xsl:template>

</xsl:stylesheet>
```

Nově připravený styl pak z pohledu uživatele zvládne během jednoho kroku do dokumentu doplnit rejstříková hesla a ještě jej zpracovat (např. převést do FO nebo HTML).

Závěr

V článku jsme se podrobně seznámili s možnostmi generování rejstříku pro dokumenty v DocBooku za použití standardních XSL stylů. Ukázali jsme si, jak generovat rejstříky vyhovující českým zvyklostem, jak do jednoho dokumentu vložit několik rejstříků a jak ze sémantického značkování automaticky generovat rejstříková hesla.

Odkazy

1. Jiří Kosek: *DocBook. Stručný úvod do tvorby a zpracování dokumentů.* URL: `http://www.kosek.cz/xml/db/`
2. Jiří Kosek: *XSL FO a jeho open-source implementace.* Str. 105-113. In: Jan Kasprzak – Petr Sojka: *SLT 2002.* ISBN 80-7302-043-2.
3. Robert Stayton: *DocBook XSL: The Complete Guide.* 2003. ISBN 0-9741521-1-0.
   Str. 394. URL: `http://www.sagehill.net/docbookxsl/index.html`
4. Norman Walsh – Leonard Muellner: *DocBook. The Definitive Guide.* 1999. ISBN 156592-580-7. Str. 648.
   URL: `http://www.docbook.org/tdg/en/html/docbook.html`

Multimediální publikování na DVD: projekt 10@FI[1]

Petr Vopálenský, Petr Sojka

Publikování a šíření multimediálních dat na nosičích DVD je stále častější. U každého většího publikačního projektu však prefabrikovaná řešení obvykle nestačí – je třeba řešení ušít na míru.

Článek diskutuje technologie, formáty a postupy vybrané a odzkoušené při tvorbě DVD připravovaného k desátému výročí Fakulty informatiky MU s názvem *10@FI*. Čtenář se seznámí s průběhem přípravy DVD od prvotní myšlenky a návrhu až po konečné vytvoření (GNU/Linux) bootovatelného obrazu DVD a jeho vylisování. Celý projekt vznikal převážně

[1] Aktualizovaná verze příspěvku prezentovaného na SLT 2004 [6]. Podporováno výzkumným záměrem MSM 143300003.