Zpravodaj Československého sdružení uživatelů TeXu

Jiří Kosek
PassiveTeX

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 13 (2003), No. 1, 26–38

Persistent URL: http://dml.cz/dmlcz/149913

Závěr

Jazyk DSSSL je zajímavou možností pro formátování XML a SGML dokumentů, která umožňuje zcela oddělit obsah dokumentu od jeho vzhledu. Již dnes je však jasné, že mnohem lépe byl přijat novější stylový jazyk XSL a ve většině aplikací se jím DSSSL postupně nahrazuje.

Odkazy

[1] *ISO/IEC 10179. Information technology – Processing languages – Document Style Semantics and Specification Language (DSSSL).* ISO/IEC, 1996. URL: `ftp://ftp.ornl.gov/pub/sgml/WG8/DSSSL/`

Summary: JadeTEX

JadeTeX is a TeX macro package which is able to process SGML and XML documents according to a DSSSL stylesheet in conjunction with (Open)Jade DSSSL processor. This article briefly describes basic principles of the DSSSL language and its usage for formatting XML documents. Complete working example of a DSSSL stylesheet is shown in the article.

Jiří Kosek
`jirka@kosek.cz`

PassiveTEX

Jiří Kosek

PassiveTEX je implementace jazyka XSL FO založená na TEXu. PassiveTEX umí vysázet dokument, jehož obsah a vzhled je popsán speciálním dokumentem XML, který jako značky používá tzv. formátovací objekty. Pojďme se však nejprve stručně seznámit se samotným XSL, abychom lépe pochopili, jaká je role PassiveTEXu při zpracování dokumentů XML.

Princip XSL

Na jazyku pro formátování XML dokumentů se začalo pracovat v podstatě ve stejné době jako na samotném jazyce XML. Poměrně brzy se přišlo na to, že

pořádný stylový jazyk se skládá ze dvou částí. Stylový jazyk musí umět jednotlivým částem dokumentu přiřadit jejich formátovací vlastnosti. Před tímto přiřazením je však v mnoha případech potřeba provést ještě transformaci vstupního dokumentu. Transformace se použije pro takové operace jako vygenerování obsahu či rejstříku, očíslování kapitol a obrázků nebo pro sečtení položek na faktuře – záleží na našich potřebách a na vstupním dokumentu.

Tento přístup reflektuje i samotný jazyk XSL. Skládá se ze dvou samostatných částí – transformačního jazyka XSLT a tzv. formátovacích objektů (FO). Podobně jako XML i XSL je standardizováno v rámci W3C. XSLT bylo jako doporučení přijato v roce 1999 a formátovací objekty až o dva roky později (2001).

XSLT styl umožňuje popsat transformaci XML dokumentu do HTML, XML s jinou strukturou, případně do obyčejného textového souboru. Transformace je přitom popsána sadou šablon, které definují převod jednotlivých částí XML dokumentu, jež jsou identifikovány pomocí XPath výrazu. XPath je jednoduchý, ale velmi silný dotazovací jazyk nad stromovou reprezentací XML. Pokud chceme nějaký XML dokument transformovat pomocí XSLT, musíme mít k dispozici tzv. XSLT procesor. Mezi nejpoužívanější dnes patří asi Saxon, xsltproc a Xalan, ale existují mnohé další.[1]

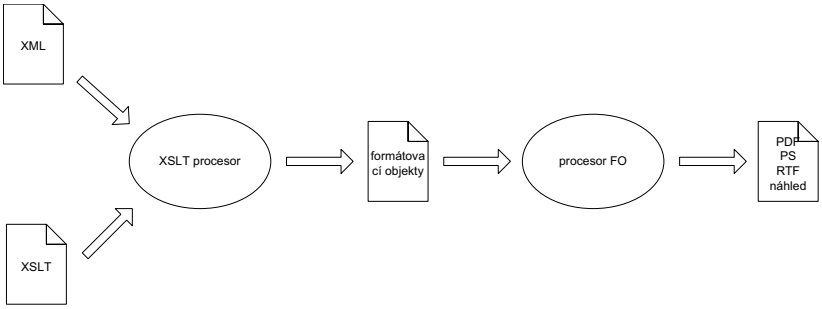
Formátovací objekty – druhá část XSL – jsou jen jakýsi abstraktní slovník, který umí velmi dobře definovat vizuální vzhled dokumentu. FO umožňují definovat layout stránek, a pak popsat formátování textu a dalších objektů, které se mají do těchto stránek naformátovat. Obsah stránek je přitom definován právě pomocí formátovacích objektů, které zastupují bloky textu, buňky tabulky nebo třeba obrázek. Každý objekt může mít několik desítek vlastností, které definují použité písmo, velikost okrajů okolo objektu, barvu textu i pozadí, vzhled rámečku, zakazují nebo povolují zlom stránky atd.

A jak formátovací objekty souvisí s formátováním XML? Celý trik spočívá v tom, že formátovací objekty se zapisují jako XML dokument, kde jednotlivým objektům odpovídají elementy a vlastnostem atributy. Díky tomu můžeme pro převedení XML do formátovacích objektů použít XSLT. Transformace převede sémantické značkování na formátovací objekty definující vzhled. Kromě toho se v tomto okamžiku může například vygenerovat obsah apod. Výsledný soubor FO se pak předá tzv. procesoru formátovacích objektů, který se postará o konečné zalomení formátovacích objektů do definovaných rozměrů stránky a výsledek zobrazí nebo uloží do nějakého vhodného formátu jako PDF nebo PostScript. V roli procesoru FO můžeme použít právě PassiveTEX.

Abychom si vše přiblížili na jednoduchém příkladě, předpokládejme, že máme následující XML dokument:

```
<odstavec>K tomuto účelu se používají
<pojem>stylové jazyky</pojem>,
```

[1] `http://www.xmlsoftware.com/xslt.html`



Obrázek 1: Princip zpracování XML pomocí XSL

```
které umožňují popsat, jak se mají jednotlivé elementy XML
dokumentu zobrazovat.</odstavec>
```

A my jej chceme pomocí XSL převést do tištěné podoby. Musíme si proto vytvořit XSLT styl, který bude generovat formátovací objekty. Dejme tomu, že chceme, aby se obsah elementu `odstavec` zobrazil jako samostatný odstavec, písmem o velikosti 12 bodů, zarovnaný do bloku, s odstavcovou zarážkou 18 bodů. Označené pojmy chceme zobrazit kurzívou. Do XSLT stylu proto přidáme dvě jednoduché šablony, které XML převedou na elementy odpovídající formátovacím objektům.

```
<xsl:template match="odstavec">
  <fo:block font-size="12pt" text-indent="18pt"
                             text-align="justify">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>

<xsl:template match="pojem">
  <fo:inline font-style="italic">
    <xsl:apply-templates/>
  </fo:inline>
</xsl:template>
```

Po zpracování dokumentu a stylu procesorem XSLT dostaneme dokument obsahující formátovací objekty:

```
<fo:block font-size="12pt" text-indent="18pt"
          text-align="justify">K tomuto účelu se používají
<fo:inline font-style="italic">stylové
jazyky</fo:inline>, které umožňují popsat, jak se mají jednotlivé
elementy XML dokumentu zobrazovat.</fo:block>
```

Vidíme, že během transformace se z XML dokumentu nesoucího nějakou sémantickou informaci stal jiný dokument, který již obsahuje jen informace o formátování. Procesor FO z něj dokáže vytvořit formátovaný dokument. Nejprve z něj načte elementy jako formátovací objekty, pak u všech formátovacích objektů doplní nespecifikované vlastnosti hodnotou zděděnou od rodičovského formátovacího objektu nebo implicitní hodnotou, vyhodnotí hodnoty výrazů apod. Výsledek se pak zalomí do oblastí vyhovujících všem omezením a práce je hotova.

Možnosti formátovacích objektů

Zatím jsme v ukázkách viděli jen dva formátovací objekty – `fo:block` a `fo:inline`, které patří mezi dva nejpoužívanější. První z nich umožňuje vytvářet samostatné bloky textu, které se zalamují do odstavce. Druhý pak umožňuje v části takového bloku změnit vybrané formátovací vlastnosti – např. změnit použitý řez písma.

Formátovacích objektů samozřejmě existuje mnohem více. Každý soubor FO na svém začátku nejdříve definuje předlohy stran – můžeme definovat rozměry stránky, velikost jejích okrajů apod. Těchto definic může být více a můžeme je sdružovat do předloh sekvencí stránek – tím lze snadno dosáhnout takových efektů, jako odlišný vzhled první stránky dokumentu či kapitoly nebo odlišnou velikost okrajů a záhlaví na sudých a lichých stránkách.

Za definicí předloh stránek pak následují jejich sekvence (`fo:page-sequence`). Pro každou sekvenci stránek můžeme určit, do jaké předlohy sekvence stránek se má sázet její obsah, jde definovat obsah pro záhlaví, zápatí, levý a pravý okraj. Ve formátovacím objektu `fo:flow` jsou pak obsaženy „běžné“ objekty jako právě `fo:block`, které již generují samotný obsah stránek.

Pro určité druhy objektů, které se mají ve výstupu objevit, nám již použití samotného `fo:block` nestačí. Pro tyto případy jsou k dispozici účelově zaměřené objekty.

K dispozici máme řadu objektů pro tvorbu tabulek. Model tabulek je podobný jako v HTML – tabulka (`fo:table`, `fo:table-and-caption`) se skládá z buněk (`fo:table-cell`), které jsou součástí řádky (`fo:table-row`) a řádka je součástí nějaké skupiny v tabulce (`fo:table-body`, `fo:table-footer`, `fo:table-header`). Pomocí vlastností pak můžeme nastavovat takové věci jako slučování buněk, rámečky okolo buněk a celé tabulky, barvy, pozadí, zarovnání obsahu buněk apod.

Samostatně existují i objekty pro seznamy. Celý seznam je uložen v `fo:list-block`. Každá položka seznamu je ohraničena pomocí `fo:list-item`. Položka seznamu pak má své návěstí (jako je odrážka, číslo, text) `fo:list-item-label` a tělo `fo:list-item-body`.

Obrázky se většinou vkládají jako odkaz na externí soubor pomocí

`fo:external-graphic`. Podporované formáty záleží na konkrétním procesoru FO. Pokud pro obrázky používáme nějaký formát založený na XML (např. SVG), můžeme je vložit přímo mezi formátovací objekty jako obsah elementu `fo:instream-foreign-object`. Při zpracování pak opět záleží na schopnostech konkrétního procesoru.

Celá řada objektů slouží ke vložení prvků, které se objeví na jiném místě, než jsou vloženy. Jedná se především o plovoucí objekty (`fo:float`), jež umožňují obtékání obrázků nebo jiných objektů textem zleva či zprava, případně ke vložení obrázku či tabulky na první vhodné místo v dokumentu. Do podobné kategorie spadají i poznámky pod čarou (`fo:footnote`, `fo:footnote-body`).

Existuje i několik objektů, které jsou vyhodnocovány a nahrazovány konkrétní hodnotou až v okamžiku formátování, protože ve fázi XSLT transformace nemáme dostatek informací. Typickým zástupcem je objekt pro vkládání aktuálního čísla strany `fo:page-number`. Při generování obsahu nebo křížových odkazů se naopak uplatní objekt `fo:page-number-citation`, který se nahradí číslem strany, na níž se vyskytuje formátovací objekt určený pomocí svého identifikátoru. Umožňuje-li to výstupní médium (např. PDF), můžeme z obsahu nebo křížových odkazů udělat jednoduše hypertextové odkazy pomocí `fo:basic-link`.

V mnoha dokumentech chceme mít v záhlaví/zápatí kromě čísla strany např. název kapitoly nebo podkapitoly. K dosažení tohoto efektu můžeme použít objekty `fo:marker` a `fo:retrieve-marker`. „Dynamicky“ se chová i objekt `fo:leader`, který vyplní volný prostor zadaným textem – lze jej použít například pro oddělení názvu kapitoly a čísla strany v obsahu tečkami.

Výčet objektů nebyl úplně kompletní, ale v praxi si s nimi bohatě vystačíme. V zasadě můžeme říci, že FO lze bez problémů použít pro formátování dokumentů obsahujících hladký text včetně obrázků, tabulek, obsahu, křížových odkazů, poznámek pod čarou, plovoucích záhlaví. Problémem není ani vícesloupcová sazba. Je to celkem pochopitelné, protože FO vycházejí z reálných požadavků na přípravu dokumentů.

Praktická ukázka

Ukažme si teď na jednoduchém příkladě kompletní funkční styl. Předpokládejme, že chceme definovat formátování pro články zapisované v XML. Každý článek má přitom následující strukturu:

```
<?xml version="1.0" encoding="iso-8859-2"?>
<!DOCTYPE clanek SYSTEM "clanek.dtd">
<clanek>
  <zahlavi>
    <rubrika>Aktuality</rubrika>
    <nazev>Státní správa bude používat TeX místo Wordu</nazev>
```

```
    <autor email="wild@duck.cz">Jan Novák</autor>
  </zahlavi>
  <perex>... text perexu ...</perex>
  <para>... text odstavce ...</para>
  <para>... text odstavce ...
        ... <em>zvýrazněný text</em> ...
        ... </para>
  <para>... text odstavce ...</para>
</clanek>
```

Chceme přitom nadpis článku vysázet větším písmem a vycentrovat, rubrika článku by se měla objevit v záhlaví a jméno a případný e-mail autora až na konci článku. Může nám k tomu posloužit následující styl:

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

<xsl:template match="/">
  <fo:root>
    <!-- Definice layoutu stránky -->
    <fo:layout-master-set>
      <!-- Rozměry stránky a její okraje -->
      <fo:simple-page-master master-name="my-page"
                             page-height="297mm"
                             page-width="210mm"
                             margin="1in">
        <!-- Tiskové zrcadlo - oblast pro samotný obsah stránky -->
        <fo:region-body margin-bottom="15mm" margin-top="15mm"/>
        <!-- Oblast pro záhlaví stránky -->
        <fo:region-before extent="10mm"/>
        <!-- Oblast pro zápatí stránky -->
        <fo:region-after extent="10mm"/>
      </fo:simple-page-master>
    </fo:layout-master-set>

    <!-- Definice obsahu stránky -->
    <fo:page-sequence master-reference="my-page"
                      font-family="Helvetica, sans-serif"
                      font-size="12pt"
                      language="cs"
                      hyphenate="true">
      <!-- Společný obsah všech stránek v záhlaví stránky -->
      <fo:static-content flow-name="xsl-region-before">
```

```
        <fo:block text-align="right" border-after-style="solid"
                  border-after-width="0.4pt">
          Rubrika: <xsl:value-of select="clanek/zahlavi/rubrika"/>
        </fo:block>
      </fo:static-content>
      <!-- Společný obsah všech stránek v zápatí stránky -->
      <fo:static-content flow-name="xsl-region-after">
        <fo:block text-align="center" font-style="italic">
          <xsl:text>- </xsl:text>
          <fo:page-number/>
          <xsl:text> -</xsl:text>
        </fo:block>
      </fo:static-content>
      <!-- Samotný text dokumentu -->
      <fo:flow flow-name="xsl-region-body">
        <!-- Zpracování všech elementů zdrojového dokumentu -->
        <xsl:apply-templates/>
        <!-- Jméno autora zpracujeme až za textem článku -->
        <xsl:apply-templates select="clanek/zahlavi/autor"/>
      </fo:flow>
    </fo:page-sequence>
  </fo:root>
</xsl:template>

<!-- Šablona pro záhlaví článku -->
<xsl:template match="zahlavi">
  <fo:block text-align="center"
            space-after="14pt">
    <xsl:apply-templates select="nazev"/>
  </fo:block>
</xsl:template>

<!--Šablona pro název článku -->
<xsl:template match="nazev">
  <fo:block font-size="24pt" font-weight="bold"
            space-after="12pt" line-height="28pt">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>

<!--Šablona pro autora článku -->
<xsl:template match="autor">
  <fo:block font-style="italic" text-align="end" space-before="6pt">
    <xsl:text>&#x2014; </xsl:text>
```

```
    <xsl:apply-templates/>
    <xsl:text> &#x2014; </xsl:text>
  </fo:block>
  <!-- Pokud má autor e-mail, zpracujeme ho -->
  <xsl:if test="@email">
    <fo:block font-size="10pt" text-align="end">
      <xsl:text>e-mail: </xsl:text>
      <fo:inline font-style="italic">
        <xsl:value-of select="@email"/>
      </fo:inline>
    </fo:block>
  </xsl:if>
</xsl:template>

<!--Šablona pro perex -->
<xsl:template match="perex">
  <fo:block text-align="justify" font-style="italic" font-weight="bold">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>

<!--Šablona pro odstavec -->
<xsl:template match="para">
  <fo:block text-indent="18pt" text-align="justify">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>

<!--Šablona pro zvýraznění -->
<xsl:template match="em">
  <fo:inline font-style="italic">
    <xsl:apply-templates/>
  </fo:inline>
</xsl:template>

</xsl:stylesheet>
```

Při zpracování tohoto stylu se nejprve načte celý dokument XML do paměti v podobě stromu, kde jednotlivé uzly odpovídají elementům XML. Strom se pak prochází a pro každý uzel se hledá odpovídající šablona (`xsl:template`). Uvnitř šablon pak generujeme formátovací objekty, které definují zobrazení daného elementu. V případě, že pro právě zpracovávaný element chceme i jeho podelementy zpracovat dalšími šablonami, použijeme instrukci `xsl:apply-templates`. Pro vložení hodnoty získané ze vstupního dokumentu do výstupu slouží instrukce

`xsl:value-of`. Kromě toho lze uvnitř šablon používat další konstrukce jako podmínky a cykly.

Při formátování dokumentu musíme nejprve dokument XML pomocí stylu XSLT převést do formátovacích objektů. K tomu budeme potřebovat nějaký XSLT procesor, např. Saxon[2] nebo xsltproc[3]. Dokument s FO získáme pomocí příkazu:

```
saxon -o dokument.fo dokument.xml clanek.xsl
```

resp.

```
xsltproc -o dokument.fo clanek.xsl dokument.xml
```

Pro převedení souboru FO do PDF nebo jiného formátu musíme použít procesor FO, v našem případě PassiveTEX. Ten je dnes součástí většiny novějších distribucí TEXu. Nemáme-li jej, je možné si poslední verzi stáhnout vždy z adresy `http://www.tei-c.org.uk/Software/passivetex/`.

```
pdfxmltex dokument.fo
```

Na obrázku 2 vidíme výsledné PDF. Jak si ještě dále řekneme, není PassiveTEX úplná implementace FO. To se projevilo například tím, že záhlaví je špatně zarovnáno a není odděleno čarou od tiskového zrcadla. Zpracujeme-li stejný dokument procesorem FO, který implementuje standard FO úplněji, dostaneme lepší výsledek (viz obrázek 3).

Implementace PassiveTEXu

PassiveTEX je implementován jako makro pro TEX, které rovnou načítá dokument FO a sází jej. Pro načítání XML (připomeňme ještě jednou, že FO mají podobu dokumentu XML) se používá XML parser napsaný v TEXu – xmltex. Technicky je to uděláno tak, že globální katalog xmltexu obsahuje záznam mapující zpracování elementů ze jmenného prostoru FO na makra podle souboru `fotex.xmt`:

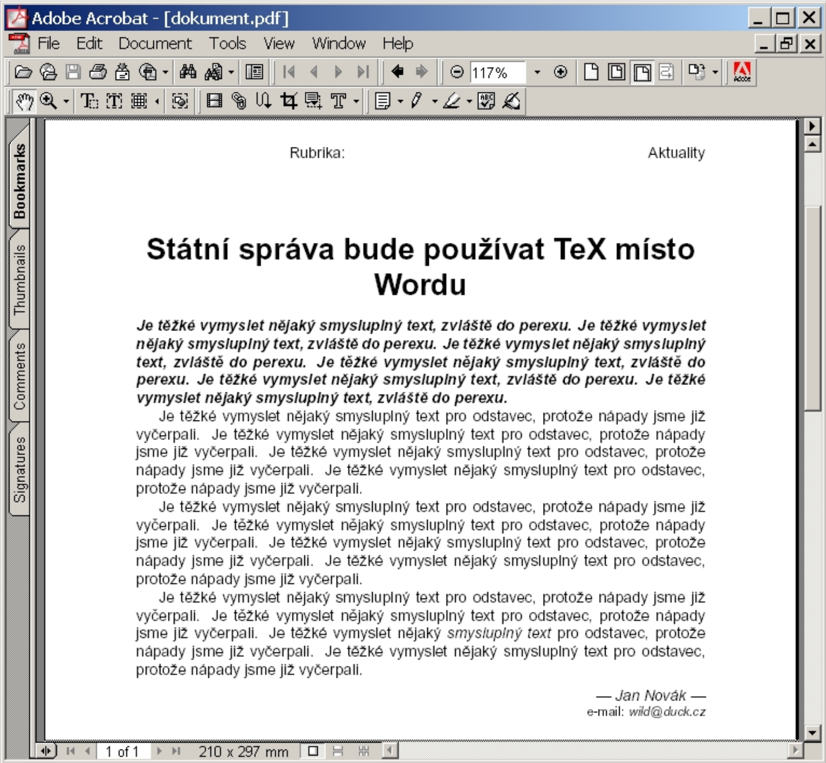
```
\NAMESPACE{http://www.w3.org/1999/XSL/Format}    {fotex.xmt}
```

Pro každý formátovací objekt je v souboru obsažena sekvence texových příkazů, kterými se má nahradit. Kvůli tomu PassiveTEX dnes nepodporuje FO úplně a asi ani nikdy nebude. Formátovací modely TEXu a FO jsou odlišné a jednoduché prostředky xmltexu neumožňují tyto rozdíly zcela překonat. V praxi dnes nejvíce vadí nedokonalé zpracování tabulek a téměř nulová podpora pro relativní délkové jednotky a výrazy uvnitř vlastností.

Používáme-li v dokumentech odkazy, nebo generujeme-li obsah, je potřeba PassiveTEX spustit opakovaně pro správné vygenerování čísel stran.

[2] `http://saxon.sourceforge.net`

[3] `http://xmlsoft.org/XSLT/xsltproc2.html`



Rubrika: Aktuality

Státní správa bude používat TeX místo Wordu

Je těžké vymyslet nějaký smysluplný text, zvláště do perexu. Je těžké vymyslet nějaký smysluplný text, zvláště do perexu. Je těžké vymyslet nějaký smysluplný text, zvláště do perexu. Je těžké vymyslet nějaký smysluplný text, zvláště do perexu. Je těžké vymyslet nějaký smysluplný text, zvláště do perexu. Je těžké vymyslet nějaký smysluplný text, zvláště do perexu.

Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali.

Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali.

Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký *smysluplný text* pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali.

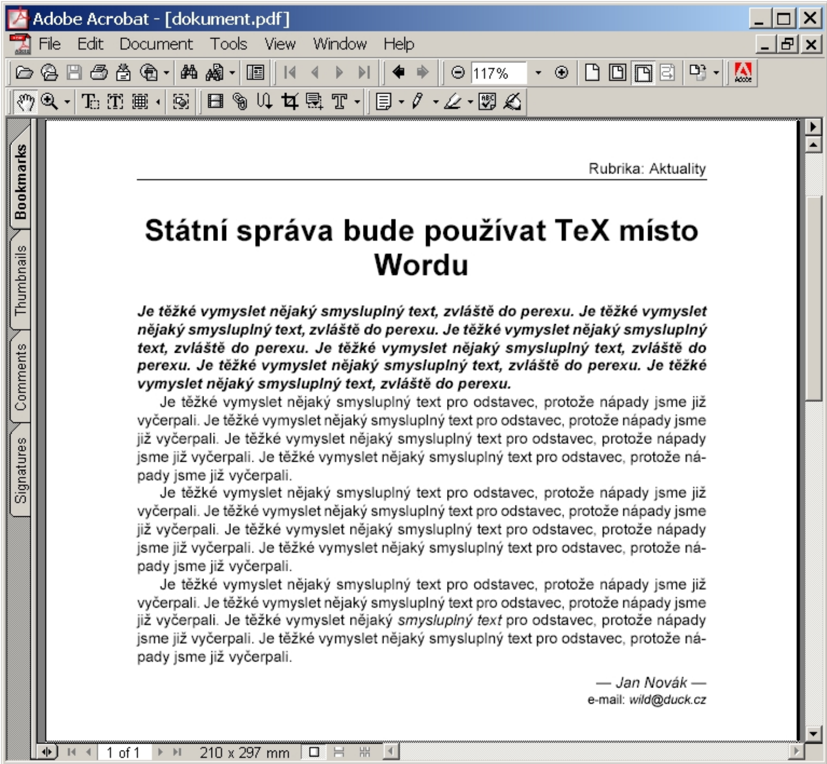
— *Jan Novák* —
e-mail: *wild@duck.cz*

Obrázek 2: Dokument XML zformátovaný pomocí PassiveTEXu

Rozšíření oproti standardu FO

PassiveTEX obsahuje rozšíření pro generování záložek v PDF. Předpokládejme následující vstupní dokument:

```
<kniha>
  <kapitola>
    <název>...</název>
    ...
  </kapitola>
  <kapitola>
    <název>...</název>
    ...
  </kapitola>
```



Rubrika: Aktuality

Státní správa bude používat TeX místo Wordu

Je těžké vymyslet nějaký smysluplný text, zvláště do perexu. Je těžké vymyslet nějaký smysluplný text, zvláště do perexu. Je těžké vymyslet nějaký smysluplný text, zvláště do perexu. Je těžké vymyslet nějaký smysluplný text, zvláště do perexu. Je těžké vymyslet nějaký smysluplný text, zvláště do perexu. Je těžké vymyslet nějaký smysluplný text, zvláště do perexu.

Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali.

Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali.

Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký *smysluplný text* pro odstavec, protože nápady jsme již vyčerpali. Je těžké vymyslet nějaký smysluplný text pro odstavec, protože nápady jsme již vyčerpali.

— *Jan Novák* —
e-mail: *wild@duck.cz*

Obrázek 3: XEP zvládá mnohem větší část FO a správně formátuje i záhlaví a zápatí stránky

```
  ...
</kniha>
```

Chceme-li mít názvy kapitol v záložkách, musíme každé kapitole přiřadit identifikátor a vložit do výsledného souboru FO rozšiřující element `bookmark`. Ten je v jiném jmenném prostoru, aby nekolidoval se standardními FO.

```
<xsl:template match="kapitola">
  <fo:block id="{generate-id(.)}">
    <fotex:bookmark xmlns:fotex="http://www.tug.org/fotex"
                    fotex-bookmark-level="1"
                    fotex-bookmark-label="{generate-id(.)}">
      <xsl:value-of select="název"/>
    </fotex:bookmark>
```

```
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
```

Sazba matematiky

PassiveTEX umožňuje do souboru s FO vkládat přímo matematické výrazy zapsané v MathML. Využívá se přitom toho, že xmltex standardně obsahuje podporu pro sazbu MathML. Do dokumentů XML tedy můžeme vkládat vzorce v MathML a v XSLT stylu je beze změny nakopírovat mezi formátovací objekty.

MathML má však jednu velkou nevýhodu a to je jeho ruční zápis. Formát je to poměrně upovídaný a vzorec, který lze v TEXu napsat na půlce řádky zabere v MathML spíše půl obrazovky. Rozumně se dá s MathML pracovat snad jen pomocí vizuálního editoru rovnic nebo pomocí nástrojů, které umějí provést transkripci z nějakého úspornějšího zápisu. PassiveTEX nám proto nabízí jednoduchou možnost, jak dokumenty zapisovat v XML se všemi s tím spojenými výhodami, ale matematické vzorce zapisovat dál v TEXu, jak jsme zvyklí.

Dosáhnout toho můžeme díky tomu, že v dokumentu s formátovacími objekty je rozeznávána speciální instrukce pro zpracování, která může obsahovat libovolný texový kód. Např.:

```
<?xmltex $a^2+b^2=c^2$?>
```

V dokumentu XML si pak stačí vzorce označit nějakým elementem:

```
<vzorec>a^2+b^2=c^2</vzorec>
```

A do stylu přidat šablonu, která z tohoto elementu vytvoří instrukci pro zpracování:

```
<xsl:template match="vzorec">
  <xsl:processing-instruction name="xmltex">
    <xsl:value-of select="."/>
  </xsl:processing-instruction>
</xsl:template>
```

Závěr

PassiveTEX zdaleka neimplementuje celý standard FO. Na druhou stranu jeho použití nám přináší kvalitní typografické algoritmy a podporu sazby matematiky. Dáme-li si pozor a ve stylech použijeme jen ty formátovací objekty a jejich vlastnosti, které PassiveTEX podporuje, můžeme z XML generovat velmi kvalitní tištěný výstup. PassiveTEX tak ukazuje jednu z cest, jak si může TEX zachovat své uživatele i v době, kdy je stále více vstupních dat a dokumentů dostupných právě v podobě XML.

Odkazy

[1] Sharon Adler – Anders Berglund – Jeff Caruso – Stephen Deach – Paul Grosso – Eduardo Gutentag – Alex Milowski – Scott Pernell – Jeremy Richman – Steve Zilles: *Extensible Stylesheet Language (XSL) – Version 1.0.* W3C, 2001. URL: `http://www.w3.org/TR/xsl`

[2] James Clark: *XSL Transformations (XSLT) Version 1.0.* W3C, 1999. URL: `http://www.w3.org/TR/xslt`

[3] Michel Goossens – Sebastian Rahtz: *PassiveTEX: from XML to PDF.* In: *TUGboat.* 3/2000. URL: `http://www.tug.org/TUGboat/Articles/tb21-3/tb68goos.pdf`

Summary: PassiveTEX

PassiveTEX is a TEX-based XSL-FO processor which is able to process XML documents according to an XSL stylesheet in conjuction with any XSLT processor. This article briefly discribes basic principles of the XSL language and its usage for formatting XML documents. Complete working example of an XSL stylesheet is shown in the article.

Jiří Kosek
`jirka@kosek.cz`

Ako TEXujeme

Ladislav Bittó

Vážení TEX-isti,

na základe našej ankety som Vám poslal jej výsledok. Dúfam, že nikoho som nevynechal a teda každý z Vás objaví svoj príspevok, ktorý je medzi dvoma horizontálnymi čiarami. Chcel som to naprogramovať tak, že sa Vášho mailu nedotknem (iba ich ukladať za sebou ako prichádzajú). Bohužiaľ nešlo to tak, lebo niektorí z Vás písali dlhšie komentáre, prehodili poradie, zmenili ste text otázky, niektorí poslali späť aj môj príklad. Aj ja som narobil zmätok tým, že v druhej výzve som vynechal položku prezerač. Na upozornenie som to hneď dal na vedomie, ale už sa to vlieklo. Podľa mňa, najväčší problém vznikol z toho dôvodu, že anketa sa rozbehla napriek tomu, že nebola ani vyhlásená. Ja som