

VYSOKÁ ŠKOLA EKONOMICKÁ V PRAZE

Fakulta informatiky a statistiky

Katedra informačního a znalostního inženýrství

Obor: informační a znalostní inženýrství



Konverze formátovacích objektů do ODF

Diplomová práce

Petr Bodnár

Vedoucí práce: Ing. Jiří Kosek

Praha 2008

Prohlášení

Prohlašuji, že jsem vypracoval samostatně diplomovou práci na téma *Konverze formátovacích objektů do ODF*. Použitou literaturu a další podkladové materiály uvádím v přiloženém seznamu literatury.

V Praze dne 30. června 2008

Petr Bodnár

Poděkování

Rád bych poděkoval vedoucímu své práce Ing. Jiřímu Koskovi za podporu při její tvorbě a mnohé cenné podněty.

Obsah

1. Úvod	6
1.1. Cíl práce	6
1.2. Struktura práce	7
1.3. Konvence použité v této práci	7
1.4. Očekávané znalosti čtenáře	8
2. XSL FO	9
2.1. Účel	9
2.2. Historie	9
2.3. Schéma pro XSL FO	10
2.4. Základní struktura dokumentu	10
2.4.1. Formátovací vlastnosti	11
2.5. Základní elementy	11
2.5.1. Rozvržení stránky	11
2.5.2. Odstavcové elementy	13
2.5.3. Seznamy	14
2.5.4. Tabulky	15
2.5.5. Inline elementy	16
2.5.6. Obrázkové elementy	16
2.5.7. „out-of-line“ elementy	16
2.5.8. Dynamické elementy	17
2.5.9. Ostatní elementy	17
3. ODF	19
3.1. Účel a historie	19
3.2. Schéma pro ODF	19
3.3. Základní struktura dokumentu	19
3.3.1. ODF dokument jako archiv	20
3.3.2. Formátovací vlastnosti	21
3.4. Základní elementy	21
3.4.1. Rozvržení stránky	21
3.4.2. Odstavcové elementy	22
3.4.3. Seznamy	23
3.4.4. Tabulky	24
3.4.5. Inline elementy	25
3.4.6. Obrázkové elementy	25
3.4.7. „out-of-line“ elementy	26
3.4.8. Dynamické elementy	26
3.4.9. Ostatní elementy	27
4. Srovnání formátů, možnosti konverze	28
4.1. Umístění formátovacích vlastností	28
4.2. Zápis hodnot formátovacích vlastností	28
4.2.1. Element <fo:wrapper>	29
4.3. Umístění statického obsahu	29
4.4. Formát stránky	29
4.5. Bloky	30
4.6. Seznamy	30
4.7. Tabulky	31
4.8. Elementy <fo:inline> a <text:span>	32
4.9. Vodící čára	32

4.10. Obrázky	32
4.11. Plovoucí objekty	33
4.12. Poznámky pod čarou	34
4.13. Dynamické elementy	34
4.14. „Živý“ text v záhlaví a zápatí	36
4.15. Vybrané formátovací vlastnosti	36
4.15.1. Vlastnosti keep-*	36
4.15.2. „Bílé znaky“	37
4.15.3. Barva	37
4.15.4. Vlastnost font-family	38
4.15.5. Vlastnost style:join-border	38
5. Implementace konverze	39
5.1. Proč právě XSLT	39
5.2. Vytvořené XSLT styly	40
5.2.1. Struktura stylů	40
5.2.2. Použité konvence a přístupy	41
5.2.3. Šablony pro celý dokument	42
5.2.4. Šablony pro spočítání hodnot vlastností	43
5.2.5. Šablony pro ODF styly	43
5.2.6. Komprimace ODF stylů	43
5.2.7. Šablony pro formát stránky	44
5.2.8. Šablony pro blokové elementy	44
5.2.9. Šablony pro seznamy	45
5.2.10. Šablony pro tabulky	45
5.2.11. Šablony pro inline elementy	45
5.2.12. Šablony pro vodící čáru	46
5.2.13. Šablony pro obrázky	46
5.2.14. Šablony pro plovoucí objekty	46
5.2.15. Šablony pro poznámky pod čarou	47
5.2.16. Šablony pro dynamické elementy	47
5.3. Rozhraní ukázkové aplikace	47
5.3.1. XML filtr v OpenOffice Writer	47
5.3.2. Webové rozhraní	50
5.3.3. Příkazová řádka	51
6. Závěr	53
Literatura	54
Obecný rejstřík	55
Rejstřík elementů	56
Rejstřík atributů	58
Slovník	60
A. Implementované FO elementy	61
B. Implementované FO vlastnosti	64
C. PHP kód ukázkové aplikace	75
D. Ukázkové FO dokumenty	78

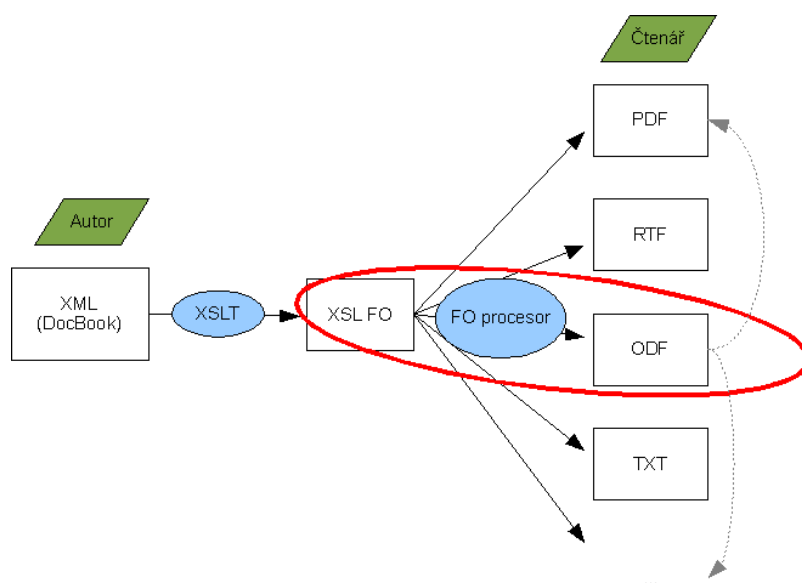
Kapitola 1

Úvod

1.1. Cíl práce

XML formát XSL FO, zkráceně tzv. formátovací objekty, nám umožňuje popsat vzhled textového dokumentu bez ohledu na konkrétní výstupní formát dokumentu. Soubor ve formátu XSL FO (dále jen „FO dokument“) je zpravidla automaticky vytvořen v rámci publikační fáze tvorby dokumentu.

Daný dokument je nejprve vytvořen v určitém XML formátu vhodným pro psaní dokumentů. Takovým formátem je např. DocBook¹, při jehož použití se v průběhu psaní nemusíme starat o formátování dokumentu. Při publikaci se z dokumentu za pomoci různých nástrojů nejprve vytvoří FO dokument. Zdarma či komerčně dostupnými programy, tzv. FO procesory, lze v dalším kroku z FO dokumentu vygenerovat dokument ve formátu, který již lze zobrazit, případně vytisknout, v některém běžně používaném prohlížeči či editoru. Následující obrázek znázorňuje výše popsaný proces.



Obrázek 1.1. Proces XSL transformace. Zvýrazněná část odpovídá cíli práce, tj. převodu z XSL FO do ODF.

Výstupní formát, který je *cílem této práce*, je formát OpenDocument (ODF). Ten, kromě jiných aplikací, jako svůj hlavní formát pro ukládání dokumentů používá open source kancelářský balík OpenOffice²

¹ <http://www.docbook.org/>

² <http://www.openoffice.org/>

(od verze 2.0). Jedná se v podstatě o konkurenta proprietárních formátů používaných kancelářským balíkem MS Office. ODF dokument získaný převodem z FO dokumentu lze zobrazit, editovat a případně i vytisknout v aplikaci OpenOffice Writer. Tento editor také umí exportovat dokument do nejrůznějších formátů (včetně kvalitního exportu do PDF). Nabízí se tak příležitost využít převod do ODF a následný export namísto některého existujícího FO procesoru, který dokáže s menším či větším úspěchem převést FO dokument do daného formátu. Získáme tím navíc možnost dodatečných úprav dokumentu před finálním exportem.

V době psaní této práce existoval nejspíše jen jediný FO procesor, který zvládal konverzi do ODF, a to komerční XMLmind XSL-FO Converter³ (označovaný též zkratkou XFC; s jistými omezeními zdarma pro osobní použití). Open source projekt Apache FOP⁴ se teprve chystal na implementaci (podpora ODF formátu byla na tzv. „seznamu přání“).

1.2. Struktura práce

Před samotným popisem možností konverze dokumentu z XSL FO do ODF se tato práce nejdříve věnuje oběma formátům. Popisuje jejich účel, historii a základní elementy, které lze použít pro popis struktury a vzhledu dokumentu. Toto umožňuje lépe si uvědomit rozdíly mezi těmito formáty, a připravit si tak půdu pro návrh všech dílčích transformací, které jsou nutné pro úspěšnou celkovou konverzi. Rozsah popisu obou formátů je přitom zpravidla záměrně omezen pouze na ty části, které jsou relevantní vzhledem k cíli této práce, tj. převodu z XSL FO do ODF.

Z dostupných možností implementace převodu bylo vybráno použití široce rozšířené a praxí ověřené technologie XSLT. Následující kapitola se proto věnuje popisu této implementace. Tzv. „XSLT styly“ umožňují relativně jednoduše definovat, jak budou jednotlivé části vstupního XML souboru (zde XSL FO) převedeny na části výstupního souboru (zde ODF). XSLT styly vytvořené v rámci této práce lze použít s XSLT procesory libxslt a Xalan-Java. První z nich je obsažen v jazyku PHP od verze 5 a druhý používá OpenOffice pro importování souborů pomocí XSLT. Výsledná ukázková aplikace, která je součástí této práce, tak umožňuje spustit konverzi z příkazové řádky, přes webové rozhraní napsané v PHP nebo přímo při otevírání FO dokumentu v OpenOffice. Popis této „integrace“ vytvořených XSLT stylů do různých prostředí tak následuje za popisem vytvořených XSLT stylů.

1.3. Konvence použité v této práci

V textu této práce je použito několik na první pohled více či méně zřejmých konvencí. Jsou to zejména tyto:

- Názvy XML elementů jsou pro snadné rozpoznání psané <jako otevírací tagy neproporcionálním písmem>.
- První výskyt názvu XML elementu, který je součástí bližšího popisu daného elementu, je navíc zvýrazněn <tučným písmem>.
- Názvy XML atributů stejně jako jejich hodnoty jsou psané neproporcionálním písmem.
- Také různé ukázky kódu jsou psány neproporcionálním písmem.
- Každou vlastnost určitého formátovacího objektu lze v drtivé většině případů ztotožnit s některým atributem uvedeným na elementu představujícím daný formátovací objekt. Obdobně to lze říci také

³ <http://www.xmlmind.com/foconverter/>

⁴ <http://xmlgraphics.apache.org/fop/>

o formátu ODF. Pojmy „vlastnost“ a „atribut“ jsou tak v textu střídavě používány v závislosti na tom, který je v daném kontextu vhodnější. Význam však zůstává ve většině případů stejný nebo téměř stejný i po myšleném zaměnění pojmů.

- Na některých místech v textu je pro odkázání se na více objektů, jejichž část názvu je stejná, použita tzv. hvězdičková konvence.

1.4. Očekávané znalosti čtenáře

Od čtenáře této práce se očekává alespoň minimální znalost XML. Část práce popisující implementaci konverze mezi jednotlivými formáty se pak bude lépe číst těm, kteří již navíc znají XSLT a související technologie. Také znalost HTML a kaskádových stylů (CSS) může napomoci v rychlejším zorientování se v některých pasážích textu.

Kapitola 2

XSL FO

2.1. Účel

XML formát XSL FO (zkráceně též tzv. formátovací objekty) je součástí jazyka XSL, což je v podstatě sada technologií umožňujících transformaci a formátování XML dokumentů. Částí, která má za úkol transformaci dokumentů, je jazyk XSLT. O formátování, tj. o určení vzhledu dokumentů, se stará právě XSL FO.

Formátovací objekty nám umožňují relativně přesně popsat vzhled textového dokumentu bez ohledu na konkrétní výstupní formát dokumentu, jako je např. PDF. Dokument přitom může být napsán prakticky v jakémkoli jazyce, protože XSL FO již od začátku počítá s asijskými a dalšími „složitějšími“ jazyky. Pomocí formátovacích objektů lze definovat téměř vše, co se týče vzhledu dokumentu, od rozvržení jednotlivých stránek (např. velikost okrajů, záhlaví a zápatí), přes vlastnosti odstavců, tabulek a vložených obrázků až po font a barvu textu. Formátovací objekty jsou přitom, na rozdíl od formátů jako je např. HTML, používány k popisu *stránkovaného* dokumentu určeného zpravidla pro tisk.

Klasický scénář použití XSL je tento (viz také obrázek 1.1): Autor nejdříve vytvoří svůj dokument (např. článek, dokumentaci k programu nebo třeba i celou knížku) v určitém XML formátu, např. ve formátu DocBook. Pomocí XSLT (přesněji řečeno pomocí XSLT procesoru, kterému předáme XML dokument a tzv. XSLT styl) se z původního XML dokumentu vygeneruje FO dokument popisující vzhled dokumentu. Následuje vygenerování dokumentu v určitém výstupním formátu tzv. FO procesorem. Posledním krokem, který je vlastně i cílem celého procesu, ale kterým se tu však již vzhledem k tématu práce nebudeme zabývat, je distribuce dokumentu v daném výstupním formátu (či formátech) potencionálním čtenářům, ať už elektronickou nebo tištěnou formou.

Nejčastějším formátem generovaným z XSL FO je všeobecně známý formát PDF (Portable Document Format). Dalšími obvyklými cílovými formáty/jazyky jsou RTF (Rich Text Format) nebo nejrůznější formáty určené pro tiskárny (např. PostScript, AFP nebo PCL). Teoreticky je však množina výstupních formátů neomezená, z FO dokumentu lze např. vytvořit obrázek nebo obyčejný textový soubor.

2.2. Historie

XSL bylo vyvinuto W3C konsorciem a jeho první verze označená číslem 1.0 pochází z října 2001¹.

Další a zatím také poslední verze specifikace je verze 1.1 z prosince 2006, která s sebou přináší několik vylepšení (např. možnost mít více „flow“ na jedné stránce, podpora automatické tvorby rejstříků, snadnější odkaz na číslo poslední stránky), nicméně samotná podstata a použití formátovacích objektů zůstávají stejné. Vzhledem k prozatím menšímu rozšíření této verze a také s přihlédnutím k rozsahu této práce se v dalším textu budeme věnovat pouze verzi 1.0.

¹<http://www.w3.org/TR/2001/REC-xsl-20011015/>

FO procesory momentálně neimplementují všechny možné elementy a funkce definované v XSL FO a ani v budoucnu se z důvodu relativně velkých nároků na implementaci tohoto formátu nedá očekávat plná podpora všech funkcí. Komerční procesory přitom zpravidla implementují více funkcí než ty zdarma dostupné.

2.3. Schéma pro XSL FO

Specifikace XSL FO nedává k dispozici žádné XML schéma, oproti kterému by šlo FO dokumenty validovat, případně pomocí kterého by mohly XML editory pomáhat s tvorbou nebo dodatečnou úpravou FO dokumentů. Důvody jsou vysvětleny například v článku „Relax NG schema for XSL FO²“. Ve zkratce se dá říci, že důvodem je příliš velká komplexnost požadavků na validní dokument, které jsou obtížně převoditelné do některého z XML schémat. Společnost RenderX nicméně vytvořila DTD schéma, XSLT styl a Relax NG schéma, které lze ve většině případů bez problémů použít. Uvedená schémata jsou k dispozici ke stažení na adrese <http://www.renderx.com/tools/validators.html>.

2.4. Základní struktura dokumentu

Příklad 2.1 ukazuje nejjednodušší možný XSL FO dokument (jen atribut `margin` lze vynechat, případně element `<fo:block>` nahradit jiným blokovým elementem).

Příklad 2.1. Nejjednodušší FO dokument. Na stránku s výchozími rozměry umístí text „Ahoj světe!“.

```
<?xml version="1.0"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="my-page">
      <fo:region-body margin="1in"/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-reference="my-page">
    <fo:flow flow-name="xsl-region-body">
      <fo:block>Ahoj světe!</fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

Vidíme, že celý dokument je obsažen v kořenovém elementu **<fo:root>**. Jmenný prostor všech FO elementů je přitom <http://www.w3.org/1999/XSL/Format>. Element **<fo:root>** obsahuje vždy minimálně dva elementy – **<fo:layout-master-set>** a **<fo:page-sequence>**. V elementu **<fo:layout-master-set>** jsou definována rozvržení stránek dokumentu, na která se pak odkazuje jeden nebo více **<fo:page-sequence>** elementů, které obsahují obsah stránek dokumentu.

V elementu **<fo:page-sequence>** může být obsaženo kromě elementu **<fo:flow>**, který obsahuje samotný text dokumentu, také jeden nebo více elementů **<fo:static-content>**, ve kterých může být uveden obsah záhlaví, zápatí apod., který se má opakovat na každé stránce rodičovského **<fo:page-sequence>** elementu. Elementy **<fo:flow>** a **<fo:static-content>** musí obsahovat atribut `flow-name`,

² http://www.idealliance.org/papers/dx_xmle04/papers/03-02-02/03-02-02.html

který odkazuje na název jednoho z `<fo:region-*>` elementů (Každý `<fo:region-*>` element má své předdefinované jméno začínající na `xsl-`).

Oba zmiňované elementy smí obsahovat pouze tzv. „blokové elementy“, tj. `<fo:block>`, `<fo:block-container>`, `<fo:table-and-caption>`, `<fo:table>` a `<fo:list-block>`, případně také tzv. „neutrální“ a „out-of-line“ elementy. Tzv. inline (řádkové) elementy jako např. `<fo:inline>`, `<fo:page-number>` nebo `<fo:external-graphic>` mohou být pouze potomky těchto blokových elementů. Popis uvedených blokových a řádkových elementů je uveden níže.

2.4.1. Formátovací vlastnosti

Formátovací objekty vycházejí z velké části z kaskádových stylů (CSS), konkrétně od nich přebírají většinu atributů pro definování vzhledu a dalších vlastností elementů (např. barva, font, styl a velikost písma; zarovnání, okraje a pozadí elementů atd.). Tj. názvy i význam vlastností jsou stejné nebo jen mírně odlišné od CSS. Hodnoty vlastností mohou být i v FO absolutní nebo relativní (např. vzhledem k velikosti písma elementu nebo vzhledem k šířce stránky). V zápisu hodnot vlastností lze navíc použít matematické operátory a také funkce speciálně definované pro FO (např. funkce `from-parent()` vracející hodnotu vlastnosti spočítanou na rodičovském elementu).

Také dědění vlastností od předků daného elementu funguje v FO obdobně jako v CSS. Na rozdíl od CSS však FO neumožňuje definovat pojmenovanou sadu vlastností, na kterou by se pak elementy mohly odkazovat. To znamená, že u každého elementu je vždy nutné formou atributů uvést všechny potřebné vlastnosti, není možné se odkázat na něco, jako je třída v CSS. Toto omezení nicméně zase až tolik nevádí, jelikož FO dokumenty jsou zpravidla generovány automaticky pomocí XSLT transformace a nejsou psány ručně.

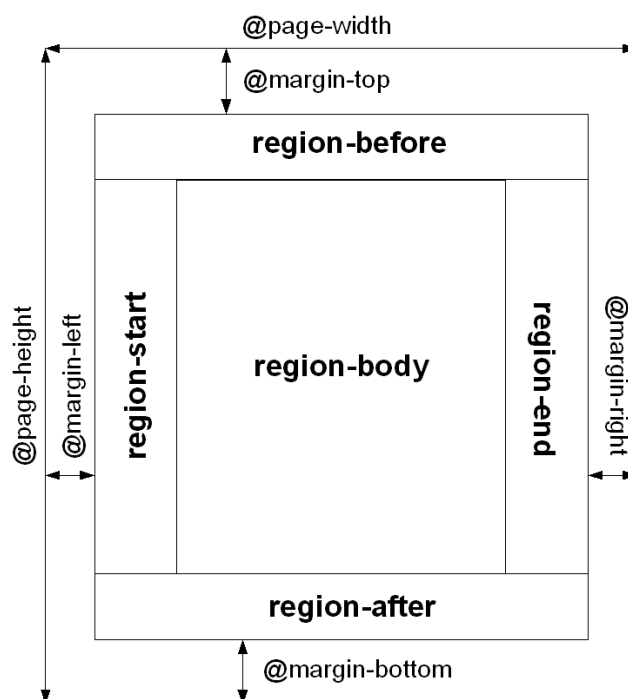
2.5. Základní elementy

2.5.1. Rozvržení stránek

Jak již bylo uvedeno výše, rozvržení (anglicky layout) stránek se definuje v elementu `<fo:layout-master-set>`. Ten obsahuje elementy `<fo:simple-page-master>` pro definici vzhledu jedné stránky, případně `<fo:page-sequence-master>` pro definici vzhledu sekvence stránek.³ Element `<fo:page-sequence>` pak atributem `master-reference` odkazuje na jeden z těchto `<fo:*-master>` elementů, který se má použít pro formátování dané sekvence stránek.

Vzhled dané stránky je tedy definován v elementu `<fo:simple-page-master>`. Jeho atributy určují šířku a výšku stránky a velikost okrajů. Element `<fo:simple-page-master>` musí povinně obsahovat element `<fo:region-body>`, který specifikuje vlastnosti (typicky mezery a ohraničení) části stránky, kam bude umístěn text dokumentu generovaný příslušným `<fo:flow>` elementem. Naopak nepovinně může obsahovat elementy `<fo:region-before>`, `<fo:region-after>`, `<fo:region-start>` a `<fo:region-end>`, které obdobně jako `<fo:region-body>` určují vlastnosti dalších oblastí stránky – „záhlaví“, „zápatí“, „levého okraje“ a „pravého okraje“. Výše uvedené ilustruje následující obrázek.

³V elementu `<fo:page-sequence-master>` lze kromě jiného určit různý vzhled lichých/sudých, prázdných/neprázdných a prvních/posledních/ostatních stránek.



Obrázek 2.1. Rozvržení stránky v XSL FO (obsah `<fo:simple-page-master>` elementu)

Následující příklad ukazuje, jak vytvořit dokument, který bude umístěn na stránku o velikosti 15 x 10 cm s okraji 1 cm, s textem vzdáleného 10 bodů od 1 bod širokého černého ohraničení a s textem „Titulek“ v 1 cm vysokém záhlaví:

Příklad 2.2. Ukázkový FO dokument se záhlavím

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="my-page" page-width="15cm" page-height="10cm" ►
margin="1cm">
      <fo:region-body border="1pt solid black" padding="10pt" />
      <fo:region-before display-align="before" extent="1cm" />
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-reference="my-page">
    <fo:static-content flow-name="xsl-region-before">
      <fo:block>Titulek</fo:block>
    </fo:static-content>

    <fo:flow flow-name="xsl-region-body">
      <fo:block>První odstavec dokumentu.</fo:block>
      <fo:block>Druhý odstavec dokumentu.</fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

Poslední příklad této kapitoly ukazuje, jak definovat různý vzhled pro první stránku a pro liché a sudé stránky dokumentu. Novými elementy jsou zde **<fo:repeatable-page-master-alternatives>** a v něm obsažené **<fo:conditional-page-master-reference>**. Pro layout stránky se použije první element **<fo:conditional-page-master-reference>**, kterému daná stránka svými vlastnostmi (tj. zda je lichá apod.) odpovídá.

Příklad 2.3. Různý vzhled první stránky a lichých a sudých stránek v XSL FO

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:layout-master-set>
      <fo:simple-page-master master-name="first-page">
        <fo:region-body margin="1in" border="5pt solid blue" padding="5pt"/>
      </fo:simple-page-master>
      <fo:simple-page-master master-name="odd-page">
        <fo:region-body margin="1in" border-right="2pt solid black" padding-right="5pt" ►
      </fo:simple-page-master>
      <fo:simple-page-master master-name="even-page">
        <fo:region-body margin="1in" border-left="2pt solid black" padding-left="5pt"/>
      </fo:simple-page-master>
      <fo:page-sequence-master master-name="my-sequence">
        <fo:repeatable-page-master-alternatives>
          <fo:conditional-page-master-reference page-position="first"
            master-reference="first-page"/>
          <fo:conditional-page-master-reference odd-or-even="odd"
            master-reference="odd-page"/>
          <fo:conditional-page-master-reference odd-or-even="even"
            master-reference="even-page"/>
        </fo:repeatable-page-master-alternatives>
      </fo:page-sequence-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="my-sequence">

    ...

  </fo:root>
```

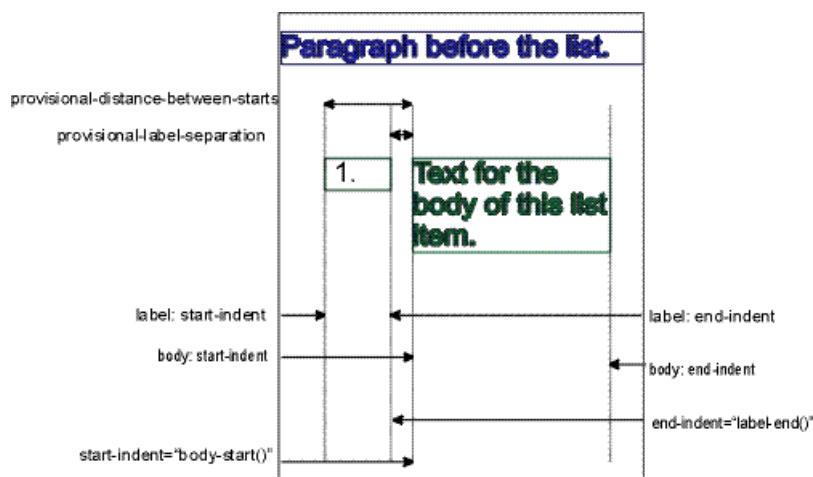
2.5.2. Odstavcové elementy

Pro odstavce, nadpisy a další elementy, pro které je vyžadováno oddělení „řádkovým zlomem“ od ostatních elementů, se používá element **<fo:block>**. Ten může obsahovat již přímo nějaký text a inline elementy. Element **<fo:block>** však může obsahovat též další blokové elementy, a sloužit tak jako „obalovací kontejner“ těchto elementů. Podobný účel plní s jistými rozdíly element **<fo:block-container>**. Tento element může kromě jiného obsahovat atribut `writing-mode`, kterým lze změnit orientaci textu vnořených blokových elementů.

2.5.3. Seznamy

Pro zápis seznamů slouží element `<fo:list-block>`, který obsahuje pro každou položku seznamu vnořený element `<fo:list-item>`. Element `<fo:list-item>` pak obsahuje element `<fo:list-item-label>` pro zápis návěští (např. číslo pro číslovaný seznam, nebo obrázek představující odrážku nečíslovaného seznamu) následovaný elementem `<fo:list-item-body>` pro zápis obsahu dané položky seznamu. Jak `<fo:list-item-label>`, tak `<fo:list-item-body>` obsahují jeden nebo více blokových elementů.

Při zápisu seznamu zpravidla chceme, aby mezi začátkem návěští a začátkem obsahu položky byla určitá vzdálenost a aby mezi koncem návěští a začátkem obsahu položky byla určitá minimální vzdálenost. Právě pro tento účel se používají atributy `provisional-distance-between-starts` a `provisional-label-separation` umístěné na elementu `<fo:list-block>`, spolu s atributy `end-indent` na `<fo:list-item-label>` a `start-indent` na `<fo:list-item-body>`, pro jejichž hodnoty lze použít speciální FO funkce `label-end()` a `body-start()`. Tento na první pohled možná složitý koncept ilustruje následující obrázek převzatý z XSL specifikace a také jednoduchý příklad číslovaného seznamu.



Obrázek 2.2. Definice mezer mezi návěští a obsahem položky FO seznamu (převzato z XSL specifikace)

Příklad 2.4. Jednoduchý číslovaný seznam v XSL FO

```
<fo:list-block provisional-distance-between-starts="24pt" ►
provisional-label-separation="8pt">
  <fo:list-item>
    <fo:list-item-label end-indent="label-end()">
      <fo:block>1.</fo:block>
    </fo:list-item-label>
    <fo:list-item-body start-indent="body-start()">
      <fo:block>Text první položky</fo:block>
    </fo:list-item-body>
  </fo:list-item>
  <fo:list-item>
    <fo:list-item-label end-indent="label-end()">
      <fo:block>2.</fo:block>
    </fo:list-item-label>
```

```

    <fo:list-item-body start-indent="body-start()">
      <fo:block>Text druhé položky</fo:block>
    </fo:list-item-body>
  </fo:list-item>
</fo:list-block>

```

2.5.4. Tabulky

Pro definici obsahu, který má být formátován jako tabulka, se používá element **<fo:table>**, jehož model obsahu je (fo:table-column*, fo:table-header?, fo:table-footer?, fo:table-body+). Nepovinné elementy **<fo:table-column>** slouží k definici společných vlastností buněk v daném sloupci (zejména atribut column-width pro určení šířky buněk).

Řádky uvedené v elementu **<fo:table-header>** jsou považovány za záhlaví tabulky a na výstupu by měly být opakovány na začátku každé části tabulky, která začíná na nové stránce nebo v novém sloupci. Ovšem jen za podmínky, že atribut table-omit-header-at-break elementu **<fo:table>** není nastaven na hodnotu true. To samé platí analogicky pro zápatí tabulky definované elementem **<fo:table-footer>** a atribut table-omit-footer-at-break. Konečně element **<fo:table-body>** pak obsahuje řádky tabulky, které jsou považovány za „tělo“ tabulky.

Každý řádek tabulky je reprezentován elementem **<fo:table-row>** a obsahuje jednu nebo více buněk, ty se zapisují elementem **<fo:table-cell>**. Alternativně nemusí být buňky vnořené v elementech **<fo:table-row>**. V tom případě se pro indikaci začátku nového řádku, resp. konce stávajícího řádku použije atribut starts-row, resp. ends-row na dané buňce.

Podobně jako v HTML lze pro určitou buňku nastavit, že pro její obsah se má vyhradit více sloupců či řádků tabulky. V XSL FO jsou k tomuto účelu určeny atributy number-columns-spanned a number-rows-spanned.

Příklad 2.5. FO tabulka se dvěma sloupci. Obsah druhého řádku je vyhrazen pro jedinou buňku.

```

<fo:table width="300pt" border="2pt solid black" border-collapse="collapse">
  <fo:table-column column-width="30%" />
  <fo:table-column column-width="70%" />
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell border="1pt solid black">
        <fo:block text-align="center">30%</fo:block>
      </fo:table-cell>
      <fo:table-cell border="1pt solid black">
        <fo:block text-align="center">70%</fo:block>
      </fo:table-cell>
    </fo:table-row>
    <fo:table-row>
      <fo:table-cell border="1pt solid black" number-columns-spanned="2">
        <fo:block text-align="center">100%</fo:block>
      </fo:table-cell>
    </fo:table-row>
  </fo:table-body>
</fo:table>

```

V XSL FO lze navíc použít ještě element **<fo:table-and-caption>**, který slouží jako kontejner pro nadpis tabulky (**<fo:table-caption>**) a samotnou tabulku (**<fo:table>**).

2.5.5. Inline elementy

Inline elementy jsou opakem blokových elementů, nejsou tedy od sousedních elementů odděleny „řádkovým zlomem“, nýbrž jsou součástí řádku daného bloku. Patří sem elementy **<fo:inline>**, **<fo:inline-container>**, **<fo:leader>**, **<fo:character>** a **<fo:bidirectional-override>**, ale i elementy uvedené v kapitole 2.5.6 – „Obrázkové elementy“ a 2.5.8 – „Dynamické elementy“.

Element **<fo:inline>** je obdobou **** elementu z HTML a umožňuje nastavit formátovací vlastnosti pro vnořený text a elementy. Element **<fo:inline-container>** má pak pro inline elementy obdobnou úlohu jako **<fo:block-container>** pro blokové elementy.

Element **<fo:leader>** se typicky používá k vygenerování „spojovací (vodící) čáry“ mezi dvěma elementy, např. k často používanému oddělení nadpisu a čísla kapitoly v položce obsahu dokumentu pomocí sekvence teček. Element **<fo:character>** slouží pro zápis znaku, který je namapován na „kresbu znaku“ (anglicky „glyph“). Posledně jmenovaný element **<fo:bidirectional-override>** se používá pro změnu směru vypsání vnořeného textu.

2.5.6. Obrázkové elementy

Pro vložení obrázku slouží prázdný element **<fo:external-graphic>** odkazující atributem **src** na soubor obsahující daný obrázek. Alternativou je element **<fo:instream-foreign-object>**, jehož jediný dětský element, který musí mít jiný jmenný prostor než „XSL“, reprezentuje obsah daného obrázku, resp. jiného typu objektu. Takto lze například vložit přímo do dokumentu obrázek ve formátu SVG. Pokud je potřeba, aby byl obrázek zobrazen jako samostatný odstavec, pak stačí obrázek vnořit do elementu **<fo:block>**.

Kromě běžných atributů jsou u obrázkových elementů používány zejména tyto atributy:

- **width** a **height** pro specifikaci plochy dostupné pro obrázek
- **content-width** a **content-height** pro specifikaci velikosti samotného obrázku. Pokud mají oba hodnotu **auto** (výchozí), pak je použita původní velikost obrázku. Pokud jeden z nich má jinou hodnotu než **auto**, pak musí být druhý rozměr obrázku dopočítán tak, aby byl zachován původní poměr stran obrázku.
- **text-align** pro určení horizontálního zarovnání obrázku v dostupné ploše
- **display-align** pro určení vertikálního zarovnání obrázku v dostupné ploše
- **clip** pro určení ořiznutí obrázku, pokud by měl „přetéct“ dostupnou plochu

2.5.7. „out-of-line“ elementy

Tzv. „out-of-line“ elementy zde rozumíme elementy, které nejsou součástí běžného toku textu. Jedná se o poznámky pod čarou a o „plovoucí“ objekty.

Element **<fo:footnote>** reprezentuje poznámku pod čarou a jeho model obsahu je (**fo:inline**, **fo:footnote-body**). Obsah **<fo:inline>** elementu (popis viz výše) je vložen přímo do textu na místo výskytu elementu **<fo:footnote>**, zatímco obsah **<fo:footnote-body>** je vložen do speciální oblasti stránky určené pro poznámky pod čarou. Právě tuto čáru, obecně jakýkoli oddělovač poznámek od

zbytku textu, lze definovat elementem `<fo:static-content>` (popis viz výše) s hodnotou atributu `flow-name` rovnou `xsl-footnote-separator`.

Pro „plovoucí“ objekty, tj. objekty, které ostatní objekty „obtékají“, se používá element `<fo:float>`, který slouží jako obálka pro vnořené blokové elementy. Typickým příkladem použití je obrázek, který je textem odstavce obtékán zleva či zprava. Pozici plovoucího objektu vzhledem k ostatním objektům přitom určuje atribut `float`, který lze na rozdíl od CSS ve formátovacích objektech použít právě jen u elementu `<fo:float>`. Dalším relevantním atributem je atribut `clear`, kterým lze od elementu, na kterém je použit, vynutit konec obtékání předcházejících plovoucích objektů.

Speciální hodnotou `float` atributu použitelnou v FO dokumentech je `before`, která znamená, že daný plovoucí objekt bude umístěn v části stránky mezi záhlavím a samotným textem stránky. Obdobně jako u poznámek pod čarou, i pro tyto „plovoucí“ objekty lze přitom definovat oddělovač, v tomto případě elementem `<fo:static-content>` s hodnotou atributu `flow-name` rovnou `xsl-before-float-separator`.

2.5.8. Dynamické elementy

Do této skupiny elementů lze zařadit elementy `<fo:basic-link>`, `<fo:page-number>` a `<fo:page-number-citation>`.

Element `<fo:basic-link>`, jak už název napovídá, reprezentuje odkaz, který v interaktivním prohlížeči dokumentu umožní rychlý přechod na odkazovaný objekt. Odkazovaným objektem může být buď nějaký externí zdroj (použit atribut `external-destination` obsahující URI zdroje), nebo element v rámci stejného FO dokumentu (použit atribut `internal-destination` obsahující hodnotu atributu `id` odkazovaného elementu).

Prázdný element `<fo:page-number>` slouží k vložení aktuálního čísla stránky, na níž je element použit. Prázdný element `<fo:page-number-citation>` má podobný účel. Liší se pouze v tom, že číslo stránky se vztahuje k elementu, jehož atribut `id` se shoduje s atributem `ref-id` elementu `<fo:page-number-citation>`.

Speciální skupinou dynamických elementů jsou elementy určené pro změnu vlastností části výsledného dokumentu v závislosti na aktuálním stavu prohlížeče, daném interakcí s uživatelem. Těmito elementy lze vytvořit například klikatelný „rozbalovací strom“ nebo objekt, který po najetí kurzorem myši změní svůj vzhled. Konkrétně se jedná o elementy `<fo:multi-toggle>`, `<fo:multi-switch>` a `<fo:multi-properties>`.

2.5.9. Ostatní elementy

Element `<fo:wrapper>` slouží čistě jako „nosič“ vlastností, které zdědí jeho potomci. Tento element se může v FO dokumentu vyskytovat téměř na libovolném místě.

Elementy `<fo:marker>` a `<fo:retrieve-marker>` se zpravidla používají pro „živý“ (anglicky „running“) text v záhlaví nebo zápatí stránky obsahující například nadpis aktuální kapitoly nebo první a poslední slovo na stránce (např. v případě dokumentu obsahujícího slovník). Element `<fo:marker>` slouží k označení elementů, které se stanou „kandidáty“ na zobrazení na místě výskytu souvisejícího `<fo:retrieve-marker>` elementu. Který z `<fo:marker>` elementů bude nakonec vybrán pro zobrazení ve výsledném dokumentu, závisí na jeho pozici ve výsledném dokumentu a na hodnotách atributů `marker-class-name` (na elementu `<fo:marker>`), `retrieve-class-name`, `retrieve-position` a `retrieve-boundary` (na elementu `<fo:retrieve-marker>`).

Posledním elementem, který zde zmíníme, je prázdný element `<fo:initial-property-set>`. Jeho atributy se použijí pro formátování prvního řádku bloku generovaného elementem `<fo:block>`, ve kterém je `<fo:initial-property-set>` obsažen.

Kapitola 3

ODF

3.1. Účel a historie

XML formát ODF, což je zkratka za OpenDocument Format, případně Open Document Format, slouží k reprezentaci dokumentů tzv. kancelářských aplikací. Konkrétně lze ODF použít pro ukládání textových dokumentů, prezentací, dokumentů tabulkových procesorů, kreslicích, ale i dalších nástrojů. Formát ODF kromě prvků (elementy, atributy) popisujících vzhled dokumentu definuje též prvky souvisejících s editací dokumentu (na rozdíl od XSL FO).

Formát ODF byl vyvinut konsorciem OASIS a jeho první verze označená 1.0 pochází z roku 2005. Formát z velké části vychází z formátu „OpenOffice.org XML“ (zkratka OOo) používaném sadou kancelářských aplikací open source projektu OpenOffice. V roce 2006 byla vydána zatím poslední verze 1.1, která obsahuje zejména vylepšení týkající se přístupnosti dokumentů (např. alternativní text pro obrázky) a opravy chyb (překlepů, nejasností apod.) předchozí verze. Další verze s označením 1.2 opravující zejména často kritizované nedostatky formátu by měla být dokončena v roce 2008.

Na rozdíl od proprietárních formátů používaných např. kancelářskými aplikacemi MS Office (formáty DOC, XLS a PPT) je ODF otevřeným formátem. To znamená, že formát může využít a implementovat jakákoli aplikace a také že do jeho dalšího vývoje se může (teoreticky) zapojit jakýkoli subjekt. Nejznámějším produktem používajícím ODF je přitom již zmiňovaný open source kancelářský balík OpenOffice, dále např. KOffice nebo Google Docs.

V roce 2006 firma Microsoft jako „protitah“ představila svůj vlastní otevřený formát „Office Open XML“, který se v dubnu 2008 stal také ISO standardem. Nyní tedy pro kancelářské aplikace existují dva konkurenční otevřené formáty a je otázkou, který se prosadí více a zda na tom nakonec koncoví uživatelé „vydělají“.

3.2. Schéma pro ODF

Schéma formátu je součástí ODF specifikace¹ a je k dispozici v jazyku Relax-NG. V případě potřeby lze dostupnými nástroji (např. open source konvertorem Trang) jednoduše převést schéma do jazyka DTD nebo W3C XML Schema.

3.3. Základní struktura dokumentu

Příklad 3.1 ukazuje nejjednodušší možný textový ODF dokument (jen element `<text:p>` lze vynechat).

¹ <http://www.oasis-open.org/specs/index.php#opendocumentv1.1>

Příklad 3.1. Nejjednodušší textový ODF dokument. Na stránku s výchozími rozměry umístí text „Ahoj světe!“.

```
<?xml version="1.0"?>
<office:document xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
  xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
  mimetype="application/vnd.oasis.opendocument.text"
>
  <office:body>
    <office:text>
      <text:p>Ahoj světe!</text:p>
    </office:text>
  </office:body>
</office:document>
```

Jmenný prostor většiny ODF elementů se skládá z prefixu `urn:oasis:names:tc:opendocument:xmlns`, z daného jména a čísla verze, vše oddělené dvojtečkou. Vidíme, že celý dokument je obsažen v kořenovém elementu **<office:document>**. Jeho atribut `mimetype` definuje MIME typ dokumentu, v našem případě textový dokument. Povinnému elementu **<office:body>**, který obsahuje „tělo“ daného dokumentu, mohou volitelně předcházet elementy pro zápis metainformací a stylů daného dokumentu (konkrétně elementy **<office:meta>**, **<office:settings>**, **<office:scripts>**, **<office:font-face-decls>**, **<office:styles>**, **<office:automatic-styles>** a **<office:master-styles>**).

V závislosti na typu dokumentu obsahuje element **<office:body>** odpovídající vnořený element. Pro textové dokumenty, které jsou právě cílem této práce, je to element **<office:text>**. Ten již obsahuje jednotlivé nadpisy (element **<text:h>**), odstavce (element **<text:p>**), tabulky a další elementy reprezentující text dokumentu.

Obsah záhlaví a zápatí určité sekvence stránek přitom není součástí elementu **<office:body>**, nýbrž je obsažen v definici stylů dokumentu – více viz kapitola 3.4.1 – „Rozvržení stránky“.

3.3.1. ODF dokument jako archiv

Výše uvedený příklad ukazující ODF dokument jako jeden XML soubor je jednou ze dvou možností, jak může být ODF dokument uložen.

Tou druhou a více používanou variantou je uložení ODF dokumentu jako více souborů zabalených do ZIP archivu. Dokument tak má díky kompresi menší velikost. Každý ze souborů přitom obsahuje určitou část dokumentu:

- Soubor s názvem `mimetype` obsahuje MIME typ dokumentu (viz výše).
- Soubor s názvem `manifest.xml` ve složce `META-INF` obsahuje informace o jednotlivých souborech v archivu. (Součástí mohou být také údaje o zašifrování daného souboru v archivu.)
- Soubor s názvem `content.xml` obsahuje v kořenovém elementu **<office:document-content>** obsah dokumentu a automatické styly.
- Soubor s názvem `styles.xml` obsahuje v kořenovém elementu **<office:document-styles>** „klasické“ (viz dále) a automatické styly (vč. elementu **<office:master-styles>**).

- Soubor s názvem `meta.xml` obsahuje v kořenovém elementu `<office:document-meta>` metainformace o dokumentu.
- Soubor s názvem `settings.xml` obsahuje v kořenovém elementu `<office:document-settings>` specifická aplikační nastavení.
- Další soubory jako např. obrázky použité v dokumentu.

Variantu ODF dokumentu jako jeden XML soubor používá aplikace OpenOffice Writer pouze při importu a exportu dokumentů prostřednictvím tzv. XML filtrů (viz dále). Pro ukládání a otevírání ODF dokumentů používá standardně jen variantu ZIP archivu. Z důvodu jednoduchosti nicméně budeme i nadále pro příklady používat první variantu.

3.3.2. Formátovací vlastnosti

Formát ODF odděluje samotný obsah dokumentu od formátovacích (stylových) informací. To znamená, že jediné informace o formátování, které nalezneme v potomcích elementu `<office:body>`, jsou jména stylů přiřazených jednotlivým elementům atributem `style-name` (majícím různý jmenný prostor pro různé typy elementů). Samotné styly, obsahující formátovací a další vlastnosti elementů, jsou definovány zvlášť jako potomci elementů `<office:styles>` (pro „klasické“ styly, které vidí a může použít uživatel textového editoru) a `<office:automatic-styles>` (pro všechny ostatní, „automaticky generované“, styly).

Určitý styl je nejčastěji reprezentován elementem `<style:style>`. Jeho jméno, na které se lze z jiných elementů odkazovat, je uvedeno v atributu `style:name` a samotné formátovací vlastnosti jsou uvedeny jako atributy vnořených elementů. Toto umožňuje logické členění vlastností podle jejich typu. Například pro „textové“ vlastnosti se použije element `<style:text-properties>`, pro „odstavcové“ vlastnosti element `<style:paragraph-properties>` a pro vlastnosti tabulky element `<style:table-properties>`. Kde je to možné, tam formát ODF přebírá názvy a význam jednotlivých atributů z existujících standardů, zejména pak z XSL, CSS a SVG.

Každý element `<style:style>` navíc obsahuje atribut `style:family`, který určuje typ elementu, pro který lze daný styl použít a současně určuje, které ze `<style:*-properties>` elementů lze vnořit do tohoto elementu.

3.4. Základní elementy

3.4.1. Rozvržení stránky

Následující příklad ukazuje dokument, který bude umístěn na stránku o velikosti 15 x 10 cm s okrajem 1 cm, s textem vzdáleného 10 bodů od 1 bod širokého černého ohraničení a s textem „Titulek“ v 1 cm vysokém záhlaví. Formát stránky je definován elementem `<style:page-layout>`, na který se odkazuje element `<style:master-page>`, který definuje „pouze“ obsah záhlaví a zápatí stránky. Styl elementu, od kterého se má daný formát stránky (včetně záhlaví a zápatí) použít, pak atributem `style:master-page-name` odkazuje na daný `<style:master-page>` element.

Obsah záhlaví stránek je obsažen v elementu `<style:header>`, obsah zápatí stránek pak v elementu `<style:footer>`. Pokud bychom chtěli použít jiný obsah záhlaví pro liché stránky, pak bychom tento obsah umístili do elementu `<style:header-left>` (analogicky pro zápatí).

Příklad 3.2. Ukázkový ODF dokument se záhlavím

```
<?xml version="1.0"?>
<office:document xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0" ►
xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0" ►
xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0" ►
xmlns:fo="urn:oasis:names:tc:opendocument:xmlns:xsl-fo-compatible:1.0" ►
mimetype="application/vnd.oasis.opendocument.text">
<office:automatic-styles>
  <style:page-layout style:name="PL1">
    <style:page-layout-properties fo:page-width="15cm" fo:page-height="10cm" ►
fo:margin-left="1cm" fo:margin-right="1cm" fo:margin-top="1cm" fo:margin-bottom="1cm" ►
fo:border="1pt solid #000000" fo:padding="10pt"/>
    <style:header-style>
      <style:header-footer-properties fo:min-height="1cm"/>
    </style:header-style>
    <style:footer-style>
      <style:header-footer-properties/>
    </style:footer-style>
  </style:page-layout>
  <style:style style:family="paragraph" style:name="P1" ►
style:master-page-name="MP1">
    <style:paragraph-properties/>
    <style:text-properties/>
  </style:style>
</office:automatic-styles>

<office:master-styles>
  <style:master-page style:name="MP1" style:page-layout-name="PL1">
    <style:header>
      <text:p>Titulek</text:p>
    </style:header>
    <style:footer/>
  </style:master-page>
</office:master-styles>

<office:body>
  <office:text>
    <text:p text:style-name="P1">První odstavec dokumentu.</text:p>
    <text:p>Druhý odstavec dokumentu.</text:p>
  </office:text>
</office:body>
</office:document>
```

3.4.2. Odstavcové elementy

Každý odstavec dokumentu je reprezentován elementem **<text:p>**. Odstavec je již dále „nedělitelný“, tj. nesmí obsahovat žádné blokové elementy jako např. tabulky nebo seznamy.

Pro odstavec, který má navíc význam nadpisu, se používá element **<text:h>**.

3.4.3. Seznamy

Seznamy se v ODF zapisují s jistými omezeními podobně jako v XSL FO. Celý seznam je reprezentován elementem **<text:list>** a položka seznamu elementem **<text:list-item>**. V elementu **<text:list-item>** je již vnořen samotný obsah dané položky reprezentovaný odstavcovými elementy nebo vnořenými seznamy. V obsahu položky nelze použít tabulku.

Návěští všech položek daného seznamu je definováno v přiřazeném stylu (atributem **text:style-name** elementu **<text:list>**), pro jehož zápis se používá element **<text:list-style>**. V něm je pak typicky pro každou úroveň seznamu vnořený element **<text:list-level-style-number>** a v něm pak ještě element **<style:list-level-properties>**. Prvně zmiňovaný element definuje, co se objeví v návěští položky (tj. nějaký symbol pro odrážku nebo automaticky generované číslo). Druhý element pak kromě jiného určuje minimální šířku návěští (atributem **text:min-label-width** analogickým atributu **provisional-distance-between-starts** v XSL FO) a minimální vzdálenost návěští od obsahu položky (atributem **text:min-label-distance** analogickým atributu **provisional-label-separation** v XSL FO).

Příklad 3.3. Jednoduchý číslovaný seznam v ODF

```
<?xml version="1.0"?>
<office:document xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
  xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
  xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0"
  mimetype="application/vnd.oasis.opendocument.text"
>
<office:automatic-styles>
  <text:list-style style:name="L1">
    <text:list-level-style-number text:level="1" style:num-format="1" ►
    style:num-suffix=".">
      <style:list-level-properties text:min-label-width="24pt" ►
      text:min-label-distance="8pt"/>
    </text:list-level-style-number>
  </text:list-style>
</office:automatic-styles>
<office:body>
  <office:text>
    <text:list text:style-name="L1">
      <text:list-item>
        <text:p>Text první položky</text:p>
      </text:list-item>
      <text:list-item>
        <text:p>Text druhé položky</text:p>
      </text:list-item>
    </text:list>
  </office:text>
</office:body>
</office:document>
```

3.4.4. Tabulky

Také tabulky se v ODF zapisují s jistými rozdíly obdobně jako v XSL FO. Tabulka je reprezentována elementem **<table:table>**. V něm musí být pro každý sloupec tabulky přítomen element **<table:table-column>** definující společné vlastnosti buněk v daném sloupci. Řádky tvořící záhlaví tabulky jsou umístěny v elementu **<table:table-header-rows>** a všechny ostatní řádky v elementu **<table:table-rows>**.

Každý řádek tabulky je reprezentován elementem **<table:table-row>**, který ve vnořených elementech **<table:table-cell>** obsahuje jednotlivé buňky tabulky. Každá buňka smí obsahovat blokové elementy včetně vnořených tabulek.

Stejně jako v XSL FO, i v ODF lze pro buňky, které zabírají více sloupců nebo řádků, použít atributy `number-columns-spanned` a `number-rows-spanned`. Specifikace ODF však navíc vyžaduje, aby na každém místě, kde by byla „překrytá“ (anglicky „covered“) buňka, byl vložen element **<table:covered-table-cell>**. (Editor OpenOffice Writer nicméně dokáže otevřít i dokument, který toto pravidlo porušuje.)

Příklad 3.4. ODF tabulka se dvěma sloupci. Obsah druhého řádku je vyhrazen pro jedinou buňku.

```
<?xml version="1.0"?>
<office:document xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
  xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
  xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0"
  xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0"
  xmlns:fo="urn:oasis:names:tc:opendocument:xmlns:xsl-fo-compatible:1.0"
  mimetype="application/vnd.oasis.opendocument.text"
>
<office:automatic-styles>
  <style:style style:family="paragraph" style:name="P_1">
    <style:paragraph-properties fo:text-align="center"/>
  </style:style>
  <style:style style:family="table" style:name="TB_1">
    <style:table-properties table:align="left" style:width="300pt" ►
table:border-model="collapsing"/>
  </style:style>
  <style:style style:family="table-column" style:name="TC_1">
    <style:table-column-properties style:rel-column-width="30*" />
  </style:style>
  <style:style style:family="table-column" style:name="TC_2">
    <style:table-column-properties style:rel-column-width="70*" />
  </style:style>
  <style:style style:family="table-cell" style:name="TD_1">
    <style:table-cell-properties fo:border="1pt solid #000000"/>
  </style:style>
  <style:style style:family="table-row" style:name="TR_1">
    <style:table-row-properties />
  </style:style>
</office:automatic-styles>

<office:body>
```



```

<office:text>
  <table:table table:style-name="TB_1">
    <table:table-column table:style-name="TC_1"/>
    <table:table-column table:style-name="TC_2"/>
    <table:table-rows>
      <table:table-row table:style-name="TR_1">
        <table:table-cell table:style-name="TD_1">
          <text:p text:style-name="P_1">30%</text:p>
        </table:table-cell>
        <table:table-cell table:style-name="TD_1">
          <text:p text:style-name="P_1">70%</text:p>
        </table:table-cell>
      </table:table-row>
      <table:table-row table:style-name="TR_1">
        <table:table-cell table:style-name="TD_1" ►
table:number-columns-spanned="2">
          <text:p text:style-name="P_1">100%</text:p>
        </table:table-cell>
        <table:covered-table-cell/>
      </table:table-row>
    </table:table-rows>
  </table:table>
</office:text>
</office:body>
</office:document>

```

3.4.5. Inline elementy

Inline (řádkové) elementy jsou opakem blokových elementů, nejsou tedy odděleny od sousedních elementů „řádkovým zlomem“, nýbrž jsou součástí řádku daného bloku. Patří sem element `<text:span>` a také elementy uvedené v kapitolách 3.4.6 – „Obrázkové elementy“ a 3.4.8 – „Dynamické elementy“.

Element `<text:span>` je obdobou `` elementu z HTML a umožňuje nastavit formátovací vlastnosti pro vnořený text a elementy (samozřejmě opět prostřednictvím přiřazeného stylu).

3.4.6. Obrázkové elementy

Pro zápis obrázku lze v ODF použít element `<draw:image>` vnořený v elementu `<draw:frame>`.

Element `<draw:frame>` se používá jako kontejner nejen pro obrázky, ale i další objekty jako plovoucí textové rámce nebo objekty vložené do dokumentu z jiných aplikací. Rozměry a další vlastnosti obsaženého objektu jsou definovány pomocí atributů tohoto elementu (např. `svg:width` a `svg:height` pro rozměry objektu) a jeho stylem.

Element `<draw:frame>` se může objevit všude tam, kde odstavcové a další blokové elementy, ale i uvnitř odstavce. Záleží na tom, jak chceme daný objekt „ukotvit“ – zda ke stránce, k odstavci, nebo jako znak odstavce. Způsob ukotvení objektu explicitně určuje atribut `text:anchor-type` uvedený na odpovídajícím stylovém elementu.

Element `<draw:image>` pak slouží čistě pro specifikaci dat obrázku, a to buď uvedením atributu `xlink:href` obsahujícím URL obrázku (Element je potom prázdný.), nebo vnořeným elementem `<office:binary-data>`, který obsahuje data obrázku v kódování BASE64 (běžně používané kódování pro binární objekty v XML souborech).

3.4.7. „out-of-line“ elementy

Tzv. „out-of-line“ elementy zde rozumíme elementy, které nejsou součástí běžného toku textu. Jedná se o poznámky pod čarou a o „plovoucí“ objekty.

Element **<text:note>** reprezentuje poznámku pod čarou, případně poznámku na konci dokumentu, a má podobný obsah jako element **<fo:footnote>** v XSL FO. První vnořený element je **<text:note-citation>** a jeho obsah je vložen přímo do textu na místo výskytu elementu **<text:note>**. Druhý vnořený element je **<text:note-body>**, jehož obsah je vložen buď do speciální oblasti stránky určené pro poznámky pod čarou, nebo na konec dokumentu. To záleží na hodnotě atributu **text:note-class**, která může být buď **footnote**, nebo **endnote**.

Pro nastavení automatického číslování a některých dalších vlastností obou typů poznámek lze v ODF použít element **<text:notes-configuration>** vnořený v elementu **<office:styles>**. Vlastnosti čáry oddělující poznámky od zbytku textu pak lze zadat elementem **<style:footnote-sep>** vnořeným v elementu **<style:page-layout-properties>**.

Pro „plovoucí“ objekty, tj. objekty, které ostatní objekty „obtékají“, lze použít element **<draw:frame>**, o kterém již byla řeč v kapitole 3.4.6 – „Obrázkové elementy“. Nás zde bude zajímat zejména možnost vnoření elementu **<draw:text-box>**, jenž může obsahovat všechny elementy, které se smí normálně vyskytovat v obsahu textového dokumentu.

Pozici plovoucího objektu vzhledem k ostatním objektům lze přitom ovlivnit na připojeném stylu pomocí atributů **style:horizontal-pos** a **style:horizontal-rel** pro horizontální a pomocí atributů **style:vertical-pos** a **style:vertical-rel** pro vertikální umístění. To, z které strany bude okolní text plovoucí objekt „obtékat“, určuje atribut **style:wrap**, který má tedy přibližně opačný význam než atribut **float** ve formátovacích objektech nebo CSS.

3.4.8. Dynamické elementy

Do této skupiny elementů lze pro účely této práce zařadit elementy **<text:a>**, **<text:page-number>**, **<text:page-count>**, **<text:bookmark>** a **<text:bookmark-ref>**.

Element **<text:a>** reprezentuje hypertextový odkaz a používá se obdobně jako HTML element **<a>**. Atribut **xlink:href** obsahuje absolutní nebo relativní URL cíle odkazu. Další atributy **office:name**, **office:title** a **office:target-frame-name** slouží podobnému účelu jako jejich protějšky **name**, **title** a **target** v HTML.

Element **<text:page-number>** slouží k vložení aktuálního čísla stránky, případně lze vložit i relativní číslo stránky vzhledem k aktuální stránce.

Element **<text:page-count>** slouží k vložení celkového počtu stránek dokumentu.

Prázdným elementem **<text:bookmark>** a jeho atributem **text:name** lze definovat cíl odkazu. Odkaz na definovaný cíl se pak vloží do dokumentu pomocí elementu **<text:bookmark-ref>** a jeho atributem **text:ref-name**. Atributem **text:reference-format** (možné hodnoty **page**, **chapter**, **direction** a **text**) lze navíc určit, jaký text se objeví na místě odkazu.

Dalším elementem, který lze také považovat za dynamický, je element **<text:chapter>**, který lze použít pro „živý“ text v záhlaví nebo zápatí stránky. Na jeho místě se v závislosti na hodnotě atributu **text:display** zobrazí číslo, název nebo číslo i název aktuální kapitoly. Kapitola začíná nadpisem (tj. elementem **<text:h>**) určité úrovně a končí nadpisem další kapitoly. Za aktuální kapitolu se přitom považuje kapitola, která jako první končí, nebo která jako první začíná na dané stránce (vyhodnocováno v uvedeném pořadí).

3.4.9. Ostatní elementy

Mezi ostatní elementy, které zde stojí za povšimnutí, lze zařadit element **<text:tab>**. Ten reprezentuje znak tabulátoru, jenž má za následek posunutí a zarovnání následujícího textu k další „zarážce“ definované pro daný odstavec. Jednotlivé zarážky se definují v rámci stylu odstavce vnořeným elementem **<style:tab-stop>**, který má tyto atributy:

- `style:position` – pozice zarážky od levého okraje odstavce
- `style:type` – typ odrážky určující zarovnání textu vzhledem k jejímu umístění (hodnoty `left`, `center`, `right`, `char` (spolu s atributem `style:char` pro zarovnání textu vzhledem k zadanému znaku))
- `style:leader-text` – „vodící text“ mezi texty oddělenými tabulátorem (výchozí hodnota mezera; aplikace může použít jen 1. znak, pokud nepodporuje celý řetězec; `style:leader-text` má přednost před `style:leader-type` a `style:leader-style`)
- `style:leader-type`, `style:leader-style` a další – vlastnosti „vodící čáry“ (pokud není specifikován atribut `style:leader-text`)

Kapitola 4

Srovnání formátů, možnosti konverze

V předcházejících kapitolách jsme se věnovali samostatně formátu XSL FO a poté formátu ODF. V této kapitole přikročíme ke srovnání obou formátů, a to hlavně s přihlédnutím k možnostem konverze z výchozího XSL FO formátu do cílového ODF formátu.

Jak už bylo zmíněno v kapitole o ODF, tento formát z velké části vychází z již existujících standardů. Co se týče jeho podobnosti s XSL FO, tak ta je dána především použitím mnoha stejných či podobných formátovacích vlastností, zapsaných formou atributů. Dále také některé objekty, jako např. tabulky nebo seznamy, se na první pohled zapisují obdobným způsobem. Může se tedy zdát, že konverze by mohla být relativně jednoduchá, nicméně při detailnějším srovnání uvidíme, že převod jistých částí výchozího dokumentu nebude zdaleka tak triviální a v některých případech ani (plně) proveditelný.

4.1. Umístění formátovacích vlastností

V obou formátech jsou vlastnosti objektů reprezentovány pomocí atributů. Ve většině případů lze přitom jeden atribut ve výchozím dokumentu převést na jeden atribut v cílovém dokumentu. Zatímco v XSL FO jsou však atributy umístěny přímo na elementu odpovídajícím danému objektu, v ODF jsou atributy uvedeny na příslušných `<style:*-properties>` elementech vnořených v definici stylu elementem `<style:style>`, na který se element reprezentující daný objekt odkazuje atributem `style-name`.

Vzhledem ke komplexnosti formátovacích objektů a jejich vlastností (a také kvůli možnosti vnořování objektů) nelze předem jednoduše určit, že určité dva objekty budou mít ve výsledném dokumentu stejné vlastnosti. To znamená, že téměř pro každý ODF element, který bude výsledkem konverze, bude potřeba nejdříve vygenerovat jemu přiřazený styl s unikátním jménem. Abychom se zbavili mnoha duplicitních stylů pro elementy, které budou mít nakonec stejné vlastnosti, budeme muset

- buď rozpoznat vznik duplicitního stylu již během konverze (např. za použití kolekce typu `mapa` běžně dostupné v moderních programovacích jazycích) a použít již existující styl,
- nebo duplicitně vygenerované styly dodatečně „zkomprimovat“ použitím jen např. prvně vygenerovaného unikátního stylu a vynecháním duplicitních.

4.2. Zápis hodnot formátovacích vlastností

Z důvodu jistého ulehčení zápisu a tvorby „vzhledem k pozdějším změnám robustnějších“ dokumentů umožňují formátovací objekty použít v hodnotách vlastností kromě relativních jednotek a procent také matematických operátorů a speciálních funkcí. Protože formát ODF nic takového neumožňuje, je potřeba při konverzi atributu spočítat jeho výslednou hodnotu.

Dalším rozdílem mezi oběma formáty je zápis některých formátovacích vlastností jako např. velikost okrajů a ohraničení objektů. V XSL FO je pro zápis jedné takové vlastnosti možné použít „absolutního“

nebo „relativního“ atributu. Např. pro specifikaci ohraničení předcházející odstavci lze v XSL FO použít absolutní atribut `border-top` nebo relativní atribut `border-before`. Relativní atribut je přitom „univerzálnější“ v tom smyslu, že skutečná výsledná pozice ohraničení záleží na aktuálním „režimu psaní“, který lze určit atributem `writing-mode`. Výchozí hodnota `lr-tb` znamená, že text je psán/čten zleva doprava a shora dolů.

Ačkoli formát ODF také podporuje atribut `writing-mode`, pro zápis uvedených vlastností poskytuje pouze „absolutní“ atributy. Relativní atributy je tedy při konverzi z formátovacích objektů potřeba převést na absolutní v závislosti na aktuální hodnotě vlastnosti `writing-mode`. Podobná situace je u absolutních a relativních hodnot některých atributů.

Poslední hlavní rozdíl týkající se zápisu hodnot formátovacích vlastností je použití tzv. zkratkových (anglicky „shorthand“) vlastností. Zkratkovými vlastnostmi jsou v XSL FO např. atributy `font` nebo `padding`, které oba umožňují zadat v jednom řetězci hodnoty více konkrétnějších vlastností. Ne všechny zkratkové vlastnosti jsou v ODF k dispozici, a při konverzi je tedy takové vlastnosti logicky nutné převést na existující ODF vlastnosti.

4.2.1. Element `<fo:wrapper>`

Pro FO element `<fo:wrapper>` nesoucím vlastnosti, které zdědí jeho potomci, neexistuje v ODF žádná obdoba. Při konverzi tak tento element nebude převeden na žádný ODF element, přičemž ale vnořené elementy musí být zpracovány tak, jako by na nich byly uvedeny dané zděděné vlastnosti. Navíc v případě, že nějaký text (zde chápán jako řetězec znaků) je přímým potomkem `<fo:wrapper>` elementu, je potřeba tento text vnořit do `<text:span>` elementu se stylem obsahujícím převedené vlastnosti `<fo:wrapper>` elementu.

4.3. Umístění statického obsahu

Statický obsah, kterým je zpravidla myšlen obsah záhlaví a zápatí stránek, je v XSL FO součástí sekvence stránek umístěné v elementu `<fo:page-sequence>`. Konkrétně je obsažen v elementech `<fo:static-content>`. V ODF formátu se používá odlišný přístup – obsah záhlaví a zápatí je definován v elementech `<style:header>` a `<style:footer>` vnořených ve stylovém elementu `<style:master-page>`. Tento přístup umožňuje snadné znovu-použití určitého vzhledu stránek včetně statického obsahu v různých částech dokumentu.

Implementace konverze statického obsahu je tedy zřejmá – pro každý element `<fo:page-sequence>` ve zdrojovém dokumentu vygenerovat odpovídající `<style:master-page>` element včetně vnořených elementů pro statický obsah pro každý odpovídající `<fo:static-content>`.

4.4. Formát stránky

V XSL FO je formát stránky definován v elementu `<fo:simple-page-master>`. Jeho atributy definují rozměry stránky a vnořené elementy `<fo:region-body>`, `<fo:region-before>` a `<fo:region-after>` definují formát samotného obsahu, záhlaví a zápatí stránky.

Elementu `<fo:simple-page-master>` odpovídá v ODF element `<style:page-layout>`. Jeho atributům a vnořenému elementu `<fo:region-body>` pak odpovídá element `<style:page-layout-properties>`, elementům `<fo:region-before>` a `<fo:region-after>` elementy `<style:header-style>` a `<style:footer-style>` (oba s vnořeným `<style:header-footer-properties>` elementem). Většinu vlastností stránky tak lze relativně snadno a rychle převést z FO do ODF. Výjimku tvoří oblasti stránky definované pomocí elementů `<fo:region-start>` a `<fo:region-end>`, pro které není ve formátu ODF žádná obdoba.

Aby se ve formátovacích objektech pro danou sekvenci stránek aplikoval určitý formát, je potřeba se na něj odkázat atributem `master-reference` elementu `<fo:page-sequence>`. Ve výsledném ODF dokumentu se nicméně nebudeme odkazovat na odpovídající `<style:page-layout>` element, nýbrž na `<style:master-page>` element vygenerovaný pro daný `<fo:page-sequence>` element (blíže viz kapitola 4.3 – „Umístění statického obsahu“). Jak vyžaduje ODF, odkážeme se na něj atributem `style:master-page-name` umístěným na stylu prvního elementu vygenerovaného v rámci daného `<fo:page-sequence>` elementu.

Poznámka: Situace je o něco složitější v případě použití podmíněných formátů stránek (viz např. element `<fo:conditional-page-master-reference>`), rozbor možností této konverze by však již byl nad rámec textu této práce. Obecně však lze říci, že formát ODF nenabízí v této oblasti tolik možností jako formát XSL FO a převod by byl složitý.

4.5. Bloky

Blokové elementy lze v XSL FO v podstatě neomezeně zanořovat. Jeden `<fo:block>` element tak může sloužit např. jako kontejner ohraničující vnořené `<fo:block>` a další elementy.

ODF na druhou stranu má v porovnání s formátovacími objekty relativně striktní požadavky na strukturu dokumentu. Ty jsou kromě jiného způsobeny hlavně určením formátu pro kancelářské textové editory, jejichž omezení a požadavky právě ODF formát reflektuje. Ve srovnání s XSL FO jsou tedy v ODF tato omezení:

- *Na nejvyšší úrovni textu mohou být pouze odstavce, seznamy a tabulky.* Podobně v XSL FO spočívá toto omezení ve výskytu pouze blokových elementů, takže konverze by v tomto směru měla být relativně přímočará.
- *Odstavec je již dále „nedělitelný“, tj. nesmí obsahovat žádné blokové elementy jako např. tabulky nebo seznamy.* Zde je velký rozdíl oproti XSL FO, kde blokové elementy, tedy hlavně `<fo:block>`, lze téměř libovolně do sebe zanořovat.

Takže např. dvěma po sobě jdoucím `<fo:block>` elementům (dále jen „blokům“) obsahujícím nějaký text a vnořeným v rodičovském bloku, který sám žádný text neobsahuje, budou v ODF odpovídat dva odstavce, tj. dva elementy `<text:p>`. Pro rodičovský blok se při konverzi žádný odstavec nevygeneruje. Pokud ovšem tento blok má nějaké vlastnosti, je nutné tyto vlastnosti vhodně „rozdělit“ mezi vnořené elementy. Např. pokud rodičovský blok definuje určitou mezeru před blokem, pak po konverzi by měl tuto mezeru zdědit první vnořený blok, z kterého po konverzi vznikne odstavec.

Z výše uvedeného tedy vyplývá, že konverze nemůže být v tomto směru úplná, protože některé vlastnosti „obalujících“ bloků nelze vždy plně nebo vůbec převést na vygenerované „jedno-úrovňové“ odstavce.

- *Některé inline elementy, jako např. `<text:bookmark>`, mohou být umístěny pouze v odstavci.* To znamená, že pokud budeme pro některý formátovací objekt na základě jeho atributu `id` generovat ODF element `<text:bookmark>`, pak tento element je potřeba vnořit do nejbližšího následujícího odstavce, který vznikne při konverzi, pokud už však samozřejmě daný formátovací objekt není součástí generovaného odstavce.

4.6. Seznamy

Zápis seznamů je v obou formátech na první pohled podobný, nicméně jsou zde tyto podstatné rozdíly:

- Návěští položky seznamu je v XSL FO součástí elementu `<fo:list-item>`, kdežto v ODF je definován jako text (konstantní nebo automaticky generovaný) nebo obrázek ve stylu připojeném k elementu `<text:list>` reprezentujícímu celý seznam. To kromě jiného znamená, že v ODF nelze jednoduše elementem `<text:list>` zapisovat seznamy definic (např. v HTML definované elementem `<dl>`).
- Tělo položky seznamu v ODF (element `<text:list-item-body>`) nemůže obsahovat tabulky (na rozdíl od elementu `<fo:list-item-body>` v XSL FO).
- ODF neumožňuje použít v hodnotách atributů FO funkce `label-end()` a `body-start()`. K těmto funkcím je potřeba přistoupit různě v závislosti na zvolené „strategii“ konverze (viz dále).

FO element `<fo:list-block>` je tedy možné úspěšně převést na ODF element `<text:list>`, pouze pokud ve zdrojovém seznamu nenalezneme něco, co by naplňovalo jeden z výše uvedených rozdílů. Takto převoditelným seznamem tedy bude např. číselný seznam nebo seznam se stejnou odrážkou u všech položek, který navíc neobsahuje v žádné položce seznamu tabulku. Výhodou tohoto převodu je víceméně kompletní převod vlastností `provisional-distance-between-starts` a `provisional-label-separation`. V případě potřeby lze navíc seznam dále jednoduše editovat v textovém editoru. Z hlediska implementace je zde nejspíš nejsložitější částí převodu rozpoznání daného číslování a ověření neexistence uvedených rozdílů mezi formáty.

V opačném případě lze seznam nejvěrněji převést nejspíše pouze *pomocí tabulky se dvěma sloupci*. Pro každou položku seznamu se vygeneruje jeden řádek tabulky. Buňka v prvním sloupci přitom bude obsahovat návěští položky a buňka v druhém sloupci tělo položky. Takto lze jednoduše převést i seznamy s návěstím skládajícím se z více blokových elementů, stejně jako seznamy s položkami obsahujícími tabulky. Nevýhodou tohoto převodu je, že nelze jednoduchým způsobem určit přesnou šířku sloupců tabulky tak, aby byly vždy správně převedeny vlastnosti `provisional-distance-between-starts` a `provisional-label-separation`. Další menší komplikací je, že vlastnosti jako např. ohraničení nebo mezery nad a pod položkou seznamu je potřeba převést na vlastnosti buněk řádku vygenerovaného pro danou položku.

4.7. Tabulky

Až na výjimky mají tabulky v obou formátech podobnou strukturu. Jeden element FO tabulky tak lze jednoduše nahradit jedním ODF elementem se stejným nebo podobným názvem. Mezi hlavní rozdíly v zápisu tabulek patří následující:

- FO element `<fo:table-column>` je nepovinný, zatímco v ODF je element `<table:table-column>` potřeba uvést pro všechny sloupce tabulky. Při konverzi je tedy nutné zjistit aktuální počet sloupců tabulky, porovnat ho s počtem sloupců „pokrytých“ elementy `<fo:table-column>` a pro „nepokryté“ sloupce dovygenerovat odpovídající `<table:table-column>` elementy.
- FO elementu `<fo:table-body>` odpovídá ODF element `<table:table-rows>`.
- FO elementu `<fo:table-header>` odpovídá ODF element `<table:table-header-rows>`, ovšem jen za podmínky, že atribut `table-omit-header-at-break` elementu `<fo:table>` není nastaven na hodnotu `true` (výchozí je `false`). V opačném případě se opět použije element `<table:table-rows>`.
- Pro FO element `<fo:table-footer>` neexistuje v ODF žádný odpovídající protějšek, proto při konverzi nezbývá nic jiného než opět použít ODF element `<table:table-rows>`.
- Formát ODF vyžaduje, aby na každém místě, kde by byla „překrytá“ (anglicky „covered“) buňka, je potřeba vložit element `<table:covered-table-cell>`. Implementace, která chce splnit tuto podmínku,

tedy musí pro každý konvertovaný řádek odvodit z atributů buněk `number-columns-spanned` a `number-rows-spanned` místa mezi vygenerovanými `<table:table-cell>` elementy, kam je potřeba vložit odpovídající `<table:covered-table-cell>` element(y).

- Zatímco v XSL FO lze specifikovat *ohraničení buňky, řádku i celé tabulky*, v ODF je to možné jen pro buňku. Pokud je tedy například na FO tabulce uvedeno nějaké ohraničení, je potřeba, aby toto ohraničení na odpovídajících stranách „podědily“ buňky v prvním a posledním řádku stejně jako buňky v prvním a posledním sloupci tabulky. Analogicky je nutné postupovat u `padding-*` vlastností buněk.

4.8. Elementy `<fo:inline>` a `<text:span>`

FO element `<fo:inline>` a ODF element `<text:span>` jsou v obou formátech v podstatě ekvivalentní. Oba definují vlastnosti textu a oba lze libovolně zanořovat. Konverze tedy v tomto případě spočívá v jednoduchém nahrazení `<fo:inline>` elementu `<text:span>` elementem.

Element `<text:span>` lze nicméně použít i v rámci převodu některých dalších inline elementů, na jejichž protějšcích ODF neumožňuje specifikovat formátovací vlastnosti (více viz konkrétní kapitoly).

4.9. Vodící čára

Element `<fo:leader>`, který v FO reprezentuje vodící (spojovací) čáru typicky mezi dvěma elementy na řádce, lze do ODF převést jeho nahrazením elementem `<text:tab>`, který představuje znak tabulátoru. Ve stylu přiřazeném danému odstavci, ve kterém se `<fo:leader>` vyskytuje, se pak elementem `<style:tab-stop>` definuje pozice zarážky pro tabulátor a znak (případně čára) sloužící jako výplň mezi „spojenými“ elementy.

Pokud má daný odstavec nastavenou vlastnost `text-align-last` na hodnotu `justify`, pak je vhodné nastavit zarovnání textu doprava vzhledem k zarážce (atributem `style:type` elementu `<style:tab-stop>`). Navíc, abychom zachovali formátovací vlastnosti uvedené na elementu `<fo:leader>`, je nutné obalit vygenerovaný `<text:tab>` element `<text:span>` elementem.

Výše uvedený převod má jednu podstatnou „slabinu“, a sice že není možné při konverzi jednoduše odvodit pozici tabulátorové zarážky ze zadané šířky `<fo:leader>` elementu. Implementující kód by musel buď tuto pozici nějakým způsobem odhadnout, nebo ji nastavit na nějakou „rozumnou“ hodnotu (např. na určitou malou vzdálenost od pravého okraje textu, jak bývá obvyklé pro řádky klasického obsahu dokumentu).

Nebo implementace může zvolit přístup, který používá např. FO procesor XFC. Jedná se o přidání rozšiřujícího atributu `tab-position` k elementům `<fo:leader>` ve zdrojovém dokumentu. Tímto atributem lze přímo specifikovat absolutní nebo relativní pozici zarážky, která se pak vezme a použije pro konverzi.

Zcela odlišným, přesto pravděpodobně možným přístupem k převodu je nahrazení `<fo:leader>` elementu ODF textovým rámcem, kterému je možné nastavit přesnou šířku na základě šířky uvedené na `<fo:leader>` elementu. Tento přístup nicméně umožňuje implementovat pouze vodící čáry, za kterými je text zarovnán doleva, případně za kterými již žádný text není (např. v případě čáry oddělující odstavce).

4.10. Obrázky

V XSL FO je možné vložit do dokumentu obrázek elementem `<fo:external-graphic>`, případně elementem `<fo:instream-foreign-object>`. Oba elementy lze v ODF reprezentovat elementem

`<draw:frame>`, který obsahuje všechny vlastnosti obrázku, s vnořeným elementem `<draw:image>`, který slouží čistě pro specifikaci dat obrázku.

ODF nespecifikuje, že pokud nejsou na elementu uvedeny rozměry, tak se mají použít původní rozměry obrázku. Proto pokud nejsou ve zdrojovém dokumentu uvedeny absolutní rozměry obrázku, pak je potřeba zjistit původní rozměry obrázku, a to z dat obsažených v případě elementu `<fo:external-graphic>` v externím souboru a v případě elementu `<fo:instream-foreign-object>` v jeho potomcích. Převzaté nebo dopočítané rozměry obrázku se pak zapíší jako atributy `svg:width` a `svg:height` na elementu `<draw:frame>`.

Pro zápis samotných dat obrázku je v ODF možné použít dva způsoby:

- *Zapsat data obrázku přímo do výsledného XML dokumentu.* To se provede zapsáním dat obrázku v kódování BASE64 do elementu `<office:binary-data>` vnořeného do elementu `<draw:image>`.
- *Zapsat data obrázku do externího souboru v některém podporovaném formátu.* Na tento soubor se pak odkážeme atributem `xlink:href` elementu `<draw:image>`.

Pokud by se daný obrázek neměl celý vejít do vymezené plochy, pak je možné v ODF použít atribut `clip`, který umožňuje čtyřmi čísly specifikovat oříznutí obrázku ze všech čtyř stran. Záporná čísla přitom znamenají přidání dodatečné plochy vedle obrázku v daném směru. To umožňuje implementovat „padding“ vlastnosti obrázku i jeho zarovnání v rámci vymezené plochy (FO atributy `text-align` a `display-align`).

4.11. Plovoucí objekty

Element `<fo:float>` reprezentuje v XSL FO plovoucí objekt. Jeho nejbližším ekvivalentem v ODF je textový rámec, tj. element `<draw:frame>` s vnořeným elementem `<draw:text-box>`, ve kterém je teprve vnořen samotný textový obsah.

Také v případě plovoucích objektů je mezi oběma formáty několik rozdílů, které je potřeba vzít při konverzi v potaz:

- *V ODF není žádná obdoba FO atributu `clear`.* Tento atribut (stejně jako v CSS) umožňuje zrušit obtékání plovoucího objektu od elementu, na kterém je uveden. Pravděpodobně jediným ODF atributem, který lze pro konverzi použít, je atribut `style:number-wrapped-paragraphs` určující počet odstavců, které budou daný plovoucí objekt obtékat. Tento počet je možné odvodit nalezením prvního bloku, který následuje za plovoucím objektem a který má nastavenou hodnotu atributu `clear` na hodnotu shodnou s typem obtékání daného objektu (atribut `float`).¹
- *V ODF může být plovoucí objekt obtékán pouze odstavci.* Aby došlo k obtékání, je při převodu potřeba vygenerovaný textový rámec umístit na začátek prvního odstavce (element `<text:p>`), který je vygenerován za obtékaným objektem, a v přiřazeném stylu nastavit hodnotu atributů `style:horizontal-rel` a `style:vertical-rel` na `paragraph`.
- Pokud je hodnota atributu `float` nastavena na `none`, nemá dojít k obtékání. V tom případě *zpracujeme obsah `<fo:float>` elementu bez generování textového rámce*. Stejně postupujeme i v případě, že není možné z nějakého důvodu vygenerovat textový rámec, který by šel vnořit do odstavce následujícího za plovoucím objektem (viz předchozí bod).

¹OpenOffice Writer nicméně zdá se podporuje pouze hodnotu 1, pro vyšší hodnoty se chová, jako by bylo zadáno `no-limit`.

- *V ODF neexistuje obdoba hodnoty `before` atributu `float`* (pro plovoucí objekt umístěný mezi záhlavím a textem stránky). Pokud má formátovací objekt atribut s touto hodnotou, pak je nejspíš nejlepší opět použít přístup uvedený v předchozím bodě, tj. zpracovat obsah elementu bez generování textového rámce.
- *V ODF musí být zadány rozměry plovoucího objektu, jinak jsou nulové.* Toto lze naštěstí vyřešit jednoduše přidáním atributů `fo:min-width` a `fo:min-height` s hodnotou `0cm` na vygenerovaný `<draw:frame>` element. To způsobí, že textový rámec bude mít takové rozměry, aby se do něj vešel celý jeho obsah.

4.12. Poznámky pod čarou

XSL FO umožňuje zapsat elementem `<fo:footnote>` poznámku pod čarou. V ODF se pro zápis poznámky používá obdobný element `<text:note>`, u kterého lze navíc atributem `text:note-class` určit, zda se jedná o poznámku pod čarou (hodnota `footnote`), nebo o poznámku umístěnou na konec dokumentu (hodnota `endnote`). Při konverzi z XSL FO tedy logicky použijeme první hodnotu, tj. `footnote`.

Vnořenému elementu `<fo:inline>`, jenž se používá pro text, který bude vložen přímo na místo výskytu poznámky, odpovídá v ODF element `<text:note-citation>`. Na rozdíl od FO však tento element neobsahuje žádné formátovací vlastnosti a může obsahovat pouze text bez vnořených elementů. Pro plnou konverzi je tedy nezbytné před vygenerovaný `<text:note>` element dát element `<text:span>` jako výsledek transformace příslušného `<fo:inline>` elementu a element `<text:note-citation>` vygenerovat prázdný – např. zápisem „`<text:note-citation text:label="​"> </text:note-citation>`“, kde entita „`​`“ zastupuje „mezeru nulové šířky“, což je přesně to, co potřebujeme (Zadáním prázdného řetězce, tj. „“, by jinak došlo k automatickému vygenerování čísla poznámky.).

Vnořenému elementu `<fo:footnote-body>`, jenž reprezentuje samotný text poznámky, odpovídá v ODF element `<text:note-body>`. Zde kromě jiného názvu elementu není potřeba provádět žádnou další konverzi.

4.13. Dynamické elementy

Převod tzv. dynamických elementů z XSL FO do ODF je zpravidla relativně přímočarý. Jedinou větší komplikací je fakt, že ODF element `<text:bookmark>` pro označení cíle nějakého odkazu musí být vždy obsažen v nějakém odstavci, tj. v elementu `<text:p>` (více viz kapitola 4.5 – „Bloky“). Způsob zápisu jednotlivých dynamických elementů v obou formátech a zároveň nástin konverze uvádí následující tabulka.

Tabulka 4.1. Dynamické elementy – porovnání zápisu

XSL FO	ODF
<code><fo:page-number /></code>	<code><text:span></code> <code><text:page-number /></code> <code></text:span></code>

XSL FO	ODF
<pre><fo:page-number-citation ref-id="cil-odkazu" /> ... <fo:... id="cil-odkazu">...</fo:...></pre>	<pre><text:span> <text:bookmark-ref ► text:ref-name="cil-odkazu" text:reference-format="page"> 0 </text:bookmark-ref> </text:span> ... <text:p> <text:bookmark text:name="cil-odkazu"/> ... </text:p></pre>
<pre><fo:basic-link internal-destination="cil-odkazu"> text odkazu </fo:basic-link> ... <fo:... id="cil-odkazu">...</fo:...></pre>	<pre><text:a xlink:href="#cil-odkazu"> <text:span>text odkazu</text:span> </text:a> ... <text:p> <text:bookmark text:name="cil-odkazu"/> ... </text:p></pre>
<pre><fo:basic-link ► external-destination="url(http://example.org)"> text odkazu </fo:basic-link></pre>	<pre><text:a xlink:href="http://example.org"> <text:span>text odkazu</text:span> </text:a></pre>

Poznámky k tabulce:

- Pro větší přehlednost není u elementů `<text:span>` uveden atribut `text:style-name`, který za normálních okolností musí být přítomen, pokud chceme pro daný element definovat formátovací vlastnosti.
- Text „0“ v elementu `<text:bookmark-ref>` slouží jako výchozí hodnota čísla stránky a bude nastavena na správnou hodnotu po otevření dokumentu v ODF editoru, typicky po zadání příkazu „Aktualizovat pole“.
- Přesná pozice elementu (případně elementů) vzniklého konverzí elementu, který je cílem odkazu, záleží na typu tohoto elementu, proto není výsledný element v příkladech explicitně uveden. Např. pro tabulku by byl vygenerován element `<table:table>` a element `<text:bookmark>` vygenerovaný pro atribut `id` této tabulky by byl nejspíše vložen na začátek prvního odstavce, který by byl vygenerován v rámci tabulky.

4.14. „Živý“ text v záhlaví a zápatí

Pro FO elementy `<fo:marker>` a `<fo:retrieve-marker>` neexistuje v ODF žádná přímá obdoba. „Živý“ text lze nicméně s jistými omezeními (viz níže) vložit do záhlaví nebo zápatí stránky elementem `<text:chapter>`, jemuž při konverzi nastavíme atribut `text:display` na hodnotu `name`. Tato hodnota atributu znamená, že se na místě elementu `<text:chapter>` vloží do dokumentu název aktuální kapitoly.

Název kapitoly je přitom vzat z textu uvedeném v příslušném `<text:h>` elementu, který tedy musí vzniknout jako výsledek konverze `<fo:marker>` elementu. Vygenerovaný element `<text:h>` ovšem samozřejmě nesmí být viditelný. To lze v ODF zajistit atributem `text:display` s hodnotou `none` na elementu `<style:text-properties>` připojeného stylu.² Je zřejmé, že uvedená implementace „živého“ textu klade tato omezení na zdrojový FO dokument:

- Podporována může být jen jedna hodnota atributů `marker-class-name` a `retrieve-class-name`, tj. jen jeden typ „živého“ textu, např. „název kapitoly“.
- Podporován může být jen `<fo:retrieve-marker>` element, jehož atributy `retrieve-position` a `retrieve-boundary` odpovídají způsobu zjišťování aktuální kapitoly ve formátu ODF.
- Jakékoli formátovací vlastnosti textu definovaného elementem `<fo:marker>` budou při obdržení textu elementem `<text:chapter>` ignorovány.

4.15. Vybrané formátovací vlastnosti

V následujících odstavcích jsou popsány vybrané formátovací vlastnosti, jejichž transformace do ODF je složitější nebo jinak zajímavá.

4.15.1. Vlastnosti `keep-*`

V XSL FO jsou pro uvedení toho, jak mají elementy „držet pohromadě“ (anglicky „keep together“), k dispozici atributy `keep-with-next`, `keep-with-previous` a `keep-together`. Navíc k nim existují tzv. komponentní atributy, jejichž název se skládá vždy z názvu „hlavního“ atributu následovaného řetězcem „within-page“, „within-column“ nebo „within-line“ (tj. např. atribut `keep-with-next.within-page`). Přípustné hodnoty všech uvedených atributů jsou `always`, `auto`, případně číslo udávající „sílu soudržnosti“ elementů.

Naproti tomu v ODF je možností, jak specifikovat soudržnost elementů, logicky o dost méně. Při konverzi je potřeba vzít v úvahu tyto rozdíly:

- *V ODF neexistuje obdoba atributu `keep-with-previous`. Požadovaného efektu lze nicméně relativně jednoduše dosáhnout tak, že si při konverzi „domyslíme“ atribut `fo:keep-with-next` se stejnou hodnotou na předcházejícím elementu.*
- *V ODF nejsou k dispozici výše uvedené „komponentní“ atributy (tj. např. `keep-with-next.within-page`). ODF totiž nerozlišuje mezi „stránkovou“ a „sloupcovou“ soudržností elementů. Ke komponentám „within-page“ a „within-column“ tak lze při konverzi přistoupit tak, jako by byl zadán jen odpovídající „hlavní“ atribut.*

²Editor OpenOffice Writer paradoxně skryje pouze text, který má nastaven hodnotu stylového atributu `text:display` na `true`. Je tedy otázka, zda se jedná o chybu v tomto programu, nebo o něco jiného (testováno s verzí OpenOffice 2.3.0).

Co se týče komponenty „within-line“, která umožňuje vynutit, že určitá část textu bude na jedné řádce, tak její převod do ODF bude pravděpodobně problematický. Bylo by nejspíše nutné nahradit mezery nedělitelnými mezerami a zakázat dělení slov.

- *Hodnoty keep-* atributů mohou v ODF nabývat pouze hodnoty always, nebo auto.* Pokud je tedy na FO elementu atributem keep-* uvedena „síla soudržnosti“, je potřeba ji při konverzi vyhodnotit a odvodit pro daný element jednu z povolených hodnot.
- *Atribut keep-with-next lze použít pouze na elementech <style:paragraph-properties> a <style:table-properties>.* Vlastnost tedy nelze aplikovat na řádky tabulky.
- *Atribut keep-together lze použít pouze na elementech <style:paragraph-properties> a <style:table-row-properties>.* Vlastnost tedy nelze aplikovat na celou tabulku.
- *Pokud je některý keep-* atribut uveden na FO bloku, který slouží jako kontejner pro vnořené bloky, je při konverzi do ODF potřeba, aby vnořené bloky „zdědily“ odpovídajícím způsobem tento atribut.* Např. atribut keep-together na „kontejnerovém“ bloku lze do ODF převést „domyšlením si“ atributu keep-with-next na všech vnořených blocích (z kterých se stanou např. odstavce a tabulky).

4.15.2. „Bílé znaky“

V XSL FO lze atributem white-space (a dalšími atributy, za něž je vlastně white-space „zkratkový“ atribut) určit, jak se má přistupovat k tzv. „bílým znakům“ v textu. Bílým znakem přitom může být mezera, tabulátor nebo znak konce řádku.

V ODF žádný takový atribut ani jeho obdoba není. Místo toho formát definuje pro každý z bílých znaků speciální element, jenž lze použít na místě, na kterém chceme, aby byl daný znak přítomen (a zobrazen). Těmito elementy jsou:

- <text:s> reprezentující znak mezery,
- <text:tab> reprezentující znak tabulátoru
- a <text:line-break> reprezentující znak nového řádku.

Elementy <text:s> a <text:tab> navíc umožňují atributem text:c zadat počet opakování daného znaku (defaultně 1).

4.15.3. Barva

V ODF je možné zadat barvu pouze jejím nezkráceným hexadecimálním kódem, tj. např. „#0000ff“ pro modrou barvu. Implementace konverze si tedy musí poradit s převodem ze všech těchto způsobů zápisu barev, které jsou přípustné ve formátovacích objektech:

- #00f – zkrácený hexadecimální zápis
- rgb(0, 0, 255) – desítkový zápis
- rgb(0%, 0%, 100%) – procentuální zápis
- blue – název barvy
- system-color(...) – název systémové barvy

- `rgb-icc(0, 0, 255, ...)` – barva z tzv. barevného profilu
- `transparent` – text, pozadí nebo jiná část objektu, v závislosti na použitém atributu, bude průhledná

4.15.4. Vlastnost `font-family`

Formát XSL FO (stejně jako CSS) umožňuje v atributu `font-family` uvést více názvů fontů oddělených čárkou s tím, že aplikace zobrazující dokument by měla použít první z uvedených fontů, který je k dispozici. V ODF lze naproti tomu ve stejném atributu zadat pouze název jednoho fontu.

4.15.5. Vlastnost `style:join-border`

V ODF dokumentu lze použít atribut `style:join-border`, který sice není součástí ODF specifikace (minimálně do verze 1.1), ale je používán např. editorem OpenOffice Writer. Tento atribut, pokud je nastaven na hodnotu `true` (v editoru OpenOffice Writer výchozí), způsobí, že pokud po sobě následující objekty (např. odstavce) mají nastaveny stejná ohraničení, pak tato ohraničení se stanou jedním společným ohraničením těchto objektů. Takto lze tedy mít jedno souvislé ohraničení okolo více po sobě jdoucích objektů.

Kapitola 5

Implementace konverze

V předcházející kapitole jsme srovnávali formáty XSL FO a ODF a probrali možnosti jejich konverze na obecné úrovni. To znamená, že jsme se zaměřili na to, „co“ je třeba provést, nikoli na to, „jak“ to bude provedeno.

Právě otázce „jak“ se budeme věnovat v této kapitole, ve které nejdříve popíšeme různé možnosti implementace této konverze a následně popíšeme implementaci pomocí XSLT, kterou jsme zvolili jako hlavní technologii pro naši ukázkovou aplikaci.

5.1. Proč právě XSLT

Jelikož se jedná o převod z jednoho XML formátu do druhého, pro implementaci se nabízí v podstatě dvě hlavní možnosti:

- *použití některého programovacího jazyka*, který nabízí dostatečné funkce pro práci s XML dokumenty. Dnes již jsou tyto funkce dostupné ve většině moderních programovacích jazycích, ať už v rámci standardních nebo rozšiřujících knihoven. Za všechny jmenujme např. jazyky Java a C#. Pro relativně pohodlnou manipulaci s dokumentem se přitom zpravidla používá API v podobě rozhraní DOM, naopak pro rychlé sekvenční čtení a zpracování XML dokumentů rozhraní SAX. Nějakou dobu jsou též v některých jazycích dostupné funkce pro vyhodnocování výrazů jazyka XPath, které může výrazně usnadnit programové zpracování XML dokumentu.
- *použití stylového jazyka XSLT*. Jak už bylo zmíněno v kapitole o formátu XSL FO, jazyk XSLT je přímo určen pro převod XML dokumentů mezi různými formáty. Transformace z výchozího do cílového dokumentu je přitom popsána tzv. XSLT stylem, což je sám o sobě také XML dokument. Samotnou transformaci nad daným dokumentem a předaným stylem provádí tzv. XSLT procesor, což může být buď samostatná aplikace, nebo knihovna funkcí dostupná v určitém programovacím jazyku.

Styl obsahuje kromě jiného tzv. šablony, kdy každá šablona je zavolána na určitou část výchozího dokumentu a obsah (resp. výsledek) této šablony se pak stává součástí transformací vzniklého cílového dokumentu. V XSLT stylech lze použít prvky programovacích jazyků jako podmíněné bloky nebo rozdělení kódu do procedur (v XSLT pomocí tzv. pojmenovaných šablon). Pro ukládání mezivýsledků lze použít proměnné, i když jen omezeným způsobem. Jednou nastavenou proměnnou totiž již nelze změnit. I proto se při složitějších transformacích často používá jako jediné dostupné řešení rekurzivní volání šablon. V XSLT stylech se velmi hojně využívá jazyk XPath pro adresování částí XML dokumentu a pro provádění výpočtů nad nimi.

Která z variant je tedy lepší? Na tuto otázku pravděpodobně neexistuje jednoznačná odpověď. Pro obě alternativy existují zdarma dostupné a zároveň kvalitní nástroje, takže cena zde nemůže být kritériem. Co se týče rychlosti, tak program napsaný v některém programovacím jazyku bude nejspíše o něco

rychlejší než XSLT styl, který musí být vykonán některým z XSLT procesorů.¹ XSLT dále oproti programovacím jazykům poskytuje slabší vyjadřovací možnosti. Ty jsou však v XSLT vyvažovány relativně větší přehledností a čitelností stylů, které popisují konverzi bez žádného „zbytečného“ kódu navíc. Jazyk XSLT je také snadnější na naučení se.

Pokud srovnáme počet dostupných funkcí pro zpracování řetězců, čísel a dalších datových typů, stejně jako i počet dalších funkcí, tak zde jasně vedou programovací jazyky. XSLT nicméně poskytuje prostředky pro volání funkcí definovaných právě v některém programovacím jazyku. Tím lze tuto nevýhodu více-méně „neutralizovat“. Přidáním závislosti na určitém programovacím jazyku nicméně přicházíme o jednu z výhod XSLT, a tou je snadná přenositelnost jak mezi operačními systémy, tak i mezi aplikacemi.

Pro naši ukázkovou implementaci konverze jsme se nakonec rozhodli pro použití stylového jazyka XSLT, a to zejména z těchto důvodů:

- Minimálně jako ilustrace možností konverze mezi oběma formáty je použití XSLT přehlednější a nevyžaduje od čtenáře znalost konkrétního programovacího jazyka a API použitého pro práci s XML.
- Vytvořené XSLT styly lze snadno zaintegrovat do existujících aplikací. Užitečná je zejména možnost snadné integrace stylů do editoru OpenOffice Writer, ve kterém je pak možné otevírat soubory přímo ve formátu XSL FO (viz dále).
- Rychlost transformace není pro ukázkové účely kritickým faktorem.
- Předchozí zkušenosti autora s tvorbou XSLT stylů.

5.2. Vytvořené XSLT styly

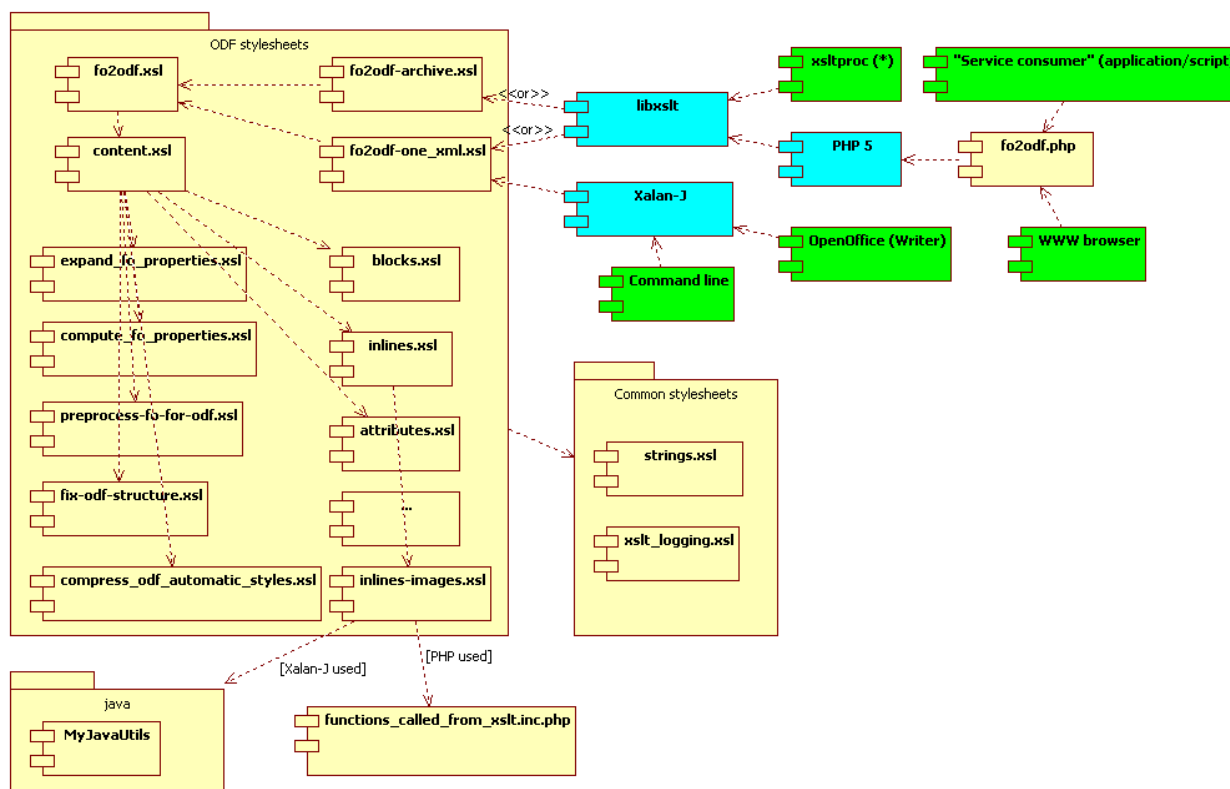
V rámci tvorby XSLT stylů pro konverzi byly též připraveny dávkové soubory pro jejich jednoduché spuštění z příkazové řádky a menší webová aplikace napsaná v PHP pro spouštění konverze z internetového prohlížeče. Celá výsledná implementace pak byla označena anglickým názvem „FO2ODF Converter“ (znamenající „Konvertor z FO do ODF“). V textu se na ni odkazujeme jako na ukázkovou aplikaci, i když se v podstatě jedná spíše o více samostatných funkčních celků postavených kolem vytvořených XSLT stylů.

Následuje popis těchto stylů stejně jako popis vytvořených rozhraní. Implementace převodu jednotlivých částí zdrojového dokumentu je zpravidla zřejmá již z kódu vytvořených XSLT šablon, i proto se zde omezíme pouze na základní popis těchto šablon a na některé složitější části konverze.

5.2.1. Struktura stylů

Umístění vytvořených stylů do souborů včetně jejich návaznosti na XSLT procesory a aplikace, které je mohou využívat, ilustruje následující obrázek. Z obrázku je patrné, že kromě stylových souborů (s příponou „xsl“), byly pro část transformace obrázků do ODF vytvořeny také odpovídající rozšiřující funkce v jazyku Java a stejné funkce též v jazyku PHP. Důvodem vytvoření funkcí právě v těchto programovacích jazycích je možnost spuštění vytvořených stylů procesorem Xalan-Java nebo knihovnou libxslt obsaženou v jazyku PHP od verze 5.

¹I když např. Xalan-Java již s jistými omezeními umožňuje XSLT styl zkompileovat a vytvořit z něj spustitelný Java program.



Obrázek 5.1. Struktura aplikace „FO2ODF Converter“ – vytvořené styly a jejich návaznost na XSLT procesory a aplikace

5.2.2. Použité konvence a přístupy

- *Modularita.* Z důvodu relativně vysoké komplexnosti a velikosti výsledného XSLT kódu byly styly rozděleny do logických celků, na nejvyšší úrovni rozdělených do samostatných souborů. To umožňuje lepší spravovatelnost i větší přehlednost kódu, než kdyby vše bylo obsaženo v jednom souboru.
- *Využití XSLT 1.0 spolu s EXSLT².* Některé funkce a elementy, které jsou pro naši implementaci konverze nezbytně nutné, případně ji přinejmenším podstatně zjednodušují (např. definice vlastních funkcí přímo v XSLT, nebo víceprůchodové zpracování dokumentu), jsou dostupné v XSLT teprve od verze 2.0. Jelikož tuto verzi XSLT podporuje zatím jen minimum XSLT procesorů (zejména procesor Saxon³), tak je ve stylech využito rozšiřujících funkcí definovaných (de-facto) standardem EXSLT, které jsou implementovány většinou XSLT procesorů.
- *Vícefázová (víceprůchodová) konverze.* Jak již dříve vyplynulo z porovnání obou formátů, zejména konverze formátovacích vlastností bude nutně zahrnovat převod zkratkových vlastností na konkrétní vlastnosti, spočítání hodnot jednotlivých vlastností a také jejich konečné umístění na často předem těžko odhadnutelných elementech. To vše v XSLT v podstatě není možné provést jedním jediným průchodem zdrojového dokumentu, který by zároveň se vším všudy vygeneroval i cílový dokument v požadovaném formátu. Proto je celá konverze rozdělena do několika fází, kdy v každé se soustředíme

² <http://www.exslt.org/>

³ <http://saxon.sourceforge.net/>

jen na danou oblast konverze a využíváme výsledků získaných v předchozích fázích. Víceprůchodovou konverzi přitom umožňuje EXSLT funkce `node-set()`.

- *Přehlednost má přednost před rychlostí.* Při psaní implementace se soustředíme hlavně na přehlednost kódu, až poté na jeho rychlost, pokud ta není dostačující. K přehlednosti kódu patří i vysvětlující komentáře uvedené přímo u daného kódu.
- *Logování důležitých událostí.* Logování poskytuje užitečné informace jak pro konečného uživatele (např. varování o omezené podpoře převodu některé části formátovacích objektů), tak pro ladění vytvářených stylů. Logování je provedeno použitím XSLT elementu `<message>` s tím, že byl připraven jednoduchý logovací framework, který umožňuje, aby se na výstupu objevily pouze logovací zprávy od jisté úrovně (např. jen závažné chyby).
- *Automatizované testy, kde to má smysl.* Pro funkce a šablony vytvořené v rámci psaní stylů jsou tam, kde to má smysl, vytvořeny automatizované testy ve stylu xUnit testů známých z programovacích jazyků (např. JUnit pro jazyk Java).⁴ To nám umožňuje více důvěřovat takto otestovanému kódu, stejně jako provádět změny v tomto kódu bez větších obav vzniku nových chyb.
- *Snadná parametrizace konverze.* XSLT styly jsou napsány tak, aby šlo průběh a výsledek konverze ovlivnit pokud možno bez nutnosti změny stylů. Pro tento účel se standardním způsobem používají XSLT parametry (element `<param>`) a případně i pojmenované sady atributů (element `<attribute-set>`).

5.2.3. Šablony pro celý dokument

Ukázková aplikace umožňuje konverzi FO dokumentu jak do jednoho XML souboru ve formátu ODF, tak do ODF archivu. V obou případech je přitom konverze až na menší rozdíly ve výsledné struktuře souborů stejná. Pro každou z alternativ byl proto vytvořen jeden stylový soubor, který importuje společnou část stylů pro konverzi (soubor `fo2odf.xml`).

V prvním případě (soubor `fo2odf-one_xml.xml`) šablona pro kořenový element generuje element `<office:document>`, zatímco v druhém případě (soubor `fo2odf-archiv.xml`) je vygenerováno několik souborů (`styles.xml` s kořenovým elementem `<office:document-styles>`, `content.xml` s kořenovým elementem `<office:document-content>` a další), které tvoří obsah výsledného ODF archivu.

Hlavní šablonou, ve které se odehrává většina procesu konverze a která je v obou případech volána, je šablona s názvem `document-content-part-with-styles` definovaná v souboru `content.xml`. Zde jsou postupně v jednotlivých fázích (za pomoci EXSLT funkce `node-set()`) provedeny přibližně tyto kroky:

- „expanze“ zkratkových vlastností na konkrétní,
- spočítání hodnot vlastností (tj. především spočítání relativně zadaných hodnot a hodnot zadaných výrazy),
- zkopírování děděných vlastností na potomky elementů,
- úprava vstupního dokumentu pro následnou snazší konverzi do ODF,
- samotná transformace do ODF, tj. vygenerování ODF elementů z FO elementů,

⁴Pro testy byl použit jednoduchý ale účinný framework pro psaní testů v xUnit stylu převzatý z projektu XSLT Standard Library [<http://xsltsl.sourceforge.net/>].

- dodatečná úprava výsledků transformace tak, aby byl vytvořen validní ODF dokument,
- a na závěr volitelně (na základě XSLT parametru) „komprese“ vygenerovaných ODF stylů.

5.2.4. Šablony pro spočítání hodnot vlastností

Po fázi „expanze“ zkratkových vlastností následuje spočítání hodnot vlastností. Jedná se především o vyhodnocení výrazů obsahujících matematické operátory a speciální FO funkce. Původní hodnoty atributů jsou přitom nahrazeny spočítanými hodnotami. Z důvodu usnadnění výpočtu i dalšího zpracování dokumentu jsou v rámci tohoto procesu také hodnoty všech „délkových“ atributů převedeny na společnou jednotku, a tou jsou body (tj. číslo bezprostředně následované řetězcem pt).

Aby uvedené vyhodnocení bylo možné provést v XSLT, je v šablonách pro zpracování atributů použito rozšiřující EXSLT funkce `evaluate()`. Ta dokáže předaný řetězec vyhodnotit, jako kdyby se jednalo o XPath výraz. Aby byl této funkci opravdu předán validní XPath výraz a výsledek vyhodnocení byl použitelný, je předtím hodnota zdrojového atributu upravena následujícím způsobem:

- Všechna volání FO funkcí jsou nahrazena voláním funkcí s podobným názvem, které jsou pro tento účel definovány přímo v XSLT stylu pomocí EXSLT elementu `<func:function>`. Takže např. volání „`from-parent(color)`“ je nahrazeno voláním „`my:fo-function-from-parent('color')`“.
- Všechny výrazy specifikující „délku“, tj. čísla bezprostředně následovaná označením absolutní nebo relativní jednotky, jsou nahrazeny voláním funkce `my:to-points()`, která se pokusí převést danou délku na číslo udávající počet bodů. Např. výraz „`10cm`“ je tak nahrazen výrazem „`my:to-points(10, 'cm')`“.

5.2.5. Šablony pro ODF styly

ODF stylový element `<style:style>` je během transformace vygenerován vždy ještě před ODF elementem, který se na něj odkazuje, zpravidla hlavně na základě atributů zpracovávaného formátovacího objektu. Vygenerované stylové elementy jsou v následující fázi přesunuty do elementu `<style:automatic-styles>`, aby byl vygenerován validní dokument. Název každého stylu musí být unikátní, a proto je pro obdržení unikátního názvu využita standardní XSLT funkce `generate-id()`, již je jako parametr předán aktuálně konvertovaný FO element.

5.2.6. Komprimace ODF stylů

Jak již bylo dříve uvedeno, pro menší velikost výsledného ODF souboru je vhodné vygenerované ODF styly „komprimovat“, a to buď již v průběhu transformace, nebo na jejím konci. Pro XSLT implementaci připadá v úvahu v podstatě jen druhá varianta. Nahrazení vygenerovaných stylů jejich „zkomprimovanou verzí“ probíhá přibližně následovně:

- Pro každý stylový element `<style:style>` je získána jeho textová reprezentace spojením názvů a hodnot atributů a vnořených elementů do jednoho řetězce. Pouze atribut `style:name` elementu `<style:style>` je vynechán.
- Použitím EXSLT funkce `set:distinct()` aplikované na takto získané textové reprezentace stylů obdržíme množinu unikátních stylů.
- Pro každý unikátní styl a styly, které jsou jeho „duplikáty“, je získán pomocný element nesoucí informaci o nahrazení daného stylu unikátním stylem, konkrétně element s atributy pro původní a nový název stylu.

- Na obsah výsledného ODF dokumentu, na jehož začátek jsou dočasně umístěny pomocné elementy z předchozího kroku⁵, jsou opět aplikovány šablony, v rámci kterých jsou z výstupu vynechány duplicitní styly a hodnoty atributů nesoucích název stylu jsou nahrazeny novými hodnotami.

5.2.7. Šablony pro formát stránky

Pro každý element `<fo:page-sequence>` je vygenerován ODF element `<style:master-page>` odkazující se na ODF element `<style:page-layout>` vzniklý konverzí elementu `<fo:simple-page-master>`. Odpovídající `<fo:simple-page-master>` element je přitom dohledán na základě hodnoty atributu `master-reference` daného `<fo:page-sequence>` elementu.

Pokud žádný takový element `<fo:simple-page-master>` není nalezen, je hledán element `<fo:page-sequence-master>` se stejným jménem. Jelikož, jak již víme, následné vyhledání správného `<fo:simple-page-master>` elementu na základě aktuální stránky není snadné, je pro jednoduchost vybrán první `<fo:simple-page-master>`, na který je v rámci daného `<fo:page-sequence-master>` elementu odkazováno.

V šabloně pro vygenerování `<style:master-page>` elementu jsou do něj vnořeny elementy `<style:header>` a `<style:footer>`, každý s výsledkem konverze aplikované na obsah elementů `<fo:static-content>`. Pro element `<style:header>` přitom musí mít element `<fo:static-content>` atribut `flow-name` s hodnotou `xsl-region-before`, pro element `<style:footer>` pak s hodnotou `xsl-region-after`. Taková implementace není sice přesná, nicméně pro většinu FO dokumentů by měla být dostačující.

Aby se od ODF elementu, který je vygenerován jako první element vzniklý z transformace obsahu (tj. elementu `<fo:flow>`) daného `<fo:page-sequence>` elementu, aplikoval vygenerovaný `<style:master-page>`, je ve fázi „dodatečné úpravy výsledků transformace“ stylový element `<style:style>`, který jako první následuje za elementem `<style:master-page>`, upraven tak, že je mu přidán atribut `style:master-page-name` odkazující na příslušný `<style:master-page>` element.

5.2.8. Šablony pro blokové elementy

Jelikož elementy `<fo:block>` mohou obsahovat jak text odstavce, tak další blokové elementy, není převod do ODF jednoduchý. Ve fázi „úprava vstupního dokumentu“ je proto struktura FO dokumentu upravena tak, že každý element `<fo:block>` obsahuje buď pouze blokové elementy, nebo pouze text odstavce (včetně inline elementů). Do `<fo:block>` elementů obsahujících smíšený obsah jsou tedy vloženy „umělé“ `<fo:block>` elementy pro každý souvislý text tvořící ve výsledku odstavec. To umožňuje snadnější generování ODF odstavcových elementů `<text:p>` ve fázi „samotné transformace do ODF“.

Všechny `<fo:block>` elementy, kromě těch na nejvyšší úrovni, jsou navíc opatřeny logickými atributy `temp-is-first` a `temp-is-last` určujícími, zda je daný blok prvním nebo posledním blokem vnořeným v rodičovském bloku. Ve fázi „samotné transformace do ODF“ pak lze snadněji převzít vlastnosti nadřazených blokových elementů. Tyto vlastnosti jsou přitom pouze jednoduše zkopírovány, správně by nicméně hodnoty udávající např. mezeru mezi bloky, měly být sčítány.

⁵Toto umístění pomocných elementů na začátek množiny elementů, na kterou jsou poté aplikovány šablony, umožňuje snadný přístup k pomocným elementům z každé zavolané šablony. Tím se vyhneme poněkud těžkopádnému předávání pomocných elementů, sloužících zde v podstatě jako parametry konverze, za použití rekurzivního volání pojmenovaných šablon.

5.2.9. Šablony pro seznamy

Jak již víme, převod seznamů do formátu ODF lze s různým stupněm úspěšnosti uskutečnit asi dvěma různými přístupy. Pro ukázkovou aplikaci jsme zvolili nejspíše ten jednodušší, a sice převod seznamu na tabulku o dvou sloupcích.

Už ve fázi přípravy dokumentu pro samotnou konverzi jsou tak všechny `<fo:list-block>` elementy nahrazeny elementem `<fo:table>`. Pro každý element `<fo:list-item>` je pak vygenerován element `<fo:table-row>` se dvěma vnořenými `<fo:table-cell>` elementy. První z nich nahrazuje element `<fo:list-item-label>` a druhý element `<fo:list-item-body>`. Pro snadnost a rychlost implementace jsou přitom všechny atributy `<fo:list-block>` a `<fo:list-item>` elementů jednoduše zkopírovány na jejich tabulkové protějšky.

Jedinou složitější konstrukcí a zároveň v některých případech i „kamenem úrazu“ je zde zjištění šířky vytvořené tabulky a jejích sloupců. Šířka tabulky je spočítána jako rozdíl mezi atributem `page-width` elementu `<fo:simple-page-master>` efektivního pro aktuální `<fo:page-sequence>` element a součtem jeho atributů `margin-left` a `margin-right` a atributů `padding-left`, `padding-right`, `border-left-width` a `border-right-width` nalezených na vnořeném `<fo:region-body>` elementu. Od šířky je navíc odečtena hodnota atributu `margin-left`, pokud je na seznamu uvedena.

Uvedený výpočet by šlo ještě zpřesnit, např. použitím odpovídajícího `<fo:region-*` elementu na základě toho, jestli je zpracováván seznam potomkem `<fo:flow>`, nebo některého `<fo:static-content>` elementu.

Ve fázi samotné transformace dokumentu do ODF je pak tabulka, která nahradila daný seznam, zpracována jako jakákoli jiná FO tabulka.

5.2.10. Šablony pro tabulky

Šablony pro tabulkové elementy zpravidla jednoduše nahrazují jeden FO element za analogický ODF element. Šablony pro elementy `<fo:table-and-caption>` a `<fo:table-caption>`, pro které v ODF není žádná obdoba, pak pouze zavolají šablony na své potomky.

Požadavek ODF na přítomnost elementu `<table:table-column>` pro všechny sloupce tabulky je přitom implementován tak, že po převodu `<fo:table-column>` elementů následuje ještě dovygenerování případných zbývajících `<table:table-column>` elementů. Pro jednoduchost je tento element vytvořen pro každou buňku prvního řádku tabulky, jejíž pozice v řádku je větší než součet hodnot atributů `number-columns-repeated` na elementech `<fo:table-column>` vstupní tabulky.

Co se sloupcového překrývání buněk týče, tak v šabloně pro buňku je za odpovídajícím `<table:table-cell>` elementem vygenerováno právě `("number-columns-spanned" - 1)` elementů `<table:covered-table-cell>`. Na implementaci složitější řádkové překrývání buněk specifikované atributem `number-rows-spanned` je vyřešeno rekurzivním spočítáním míst, na které je nutné vložit `<table:covered-table-cell>` element v aktuálně zpracovávaném řádku.

5.2.11. Šablony pro inline elementy

Šablona pro inline element `<fo:inline>` je jedna z těch snadných. Vygeneruje element `<text:span>` spolu se stylem nesoucím výsledek konverze atributů daného `<fo:inline>` elementu. Obsah elementu `<text:span>` je pak výsledkem aplikace šablon na potomky `<fo:inline>` elementu. Jelikož při konverzi několika dalších inline elementů (viz dále) je také potřeba podobným způsobem vygenerovat odpovídající `<text:span>` element, používají tyto šablony pojmenovanou šablonu `generate-odf-text-span` vytvořenou právě za tímto účelem.

5.2.12. Šablony pro vodící čáru

Uskutečnit plný převod elementu `<fo:leader>`, představujícího vodící čáru, do ODF je, jak již už bylo zmíněno, relativně složité. V ukázkových stylech je vybrán převod elementu na odstavcové tabulátory, který navíc předpokládá pouze jeden výskyt `<fo:leader>` elementu v jednom odstavci (což bývá obvyklý případ). První vytvořená šablona se aplikuje na samotný `<fo:leader>` element a vygeneruje prázdný ODF element `<text:tab>` obalený elementem `<text:span>` pomocí pojmenované šablony `generate-odf-text-span`.

Druhá šablona, která se účastní převodu vodící čáry, je šablona pro `<fo:block>`, která při generování stylového elementu pro výsledný `<text:p>` element vloží do tohoto stylu element `<style:tab-stops>` s jedním `<style:tab-stop>` elementem odvozeným z vlastností dané vodící čáry a vlastnosti `text-align-last` daného odstavce. Pozice tabulátoru je přitom určena buď použitím nějaké „rozumné“ konstantní hodnoty, nebo využitím rozšiřujícího atributu `tab-position`, pokud je přítomen (viz kapitola srovnávající oba formáty).

5.2.13. Šablony pro obrázky

Pro převod obrázků jsme použili přístup, kdy data obrázku jsou přímo vložena do výsledného XML souboru jako řetězec v kódování BASE64. Jelikož jazyk XSLT nenabízí žádné funkce pro čtení ani jinou manipulaci s obrázky, byly pro tento účel vytvořeny rozšiřující funkce v jazyku Java pro procesor Xalan-Java a obdobné funkce v jazyku PHP pro webové rozhraní konvertoru. Konkrétně se jedná o funkci pro zjištění rozměrů obrázku a o funkci pro získání dat obrázku v kódování BASE64.

Zavolání připravených funkcí z XSLT je komplikováno jedinou věcí, a tou je skutečnost, že v případě relativního URL obrázku je pro úspěšné načtení obrázku potřeba z tohoto relativního URL získat URL absolutní. URL obrázku přitom může být relativní vzhledem k adresáři, ve kterém je umístěn zpracovávaný dokument, nebo vzhledem k hodnotě atributu `xml:base` uvedeném na obrázkovém elementu nebo na některém z jeho předků.

Vytvořené XSLT styly si částečně dokáží poradit pouze s prvně uvedeným případem. Při použití PHP je umístění zdrojového dokumentu známo již v samotném PHP skriptu volajícím XSLT transformaci a v případě spuštění transformace procesorem Xalan-Java je hledáno umístění zdrojového dokumentu v rozšiřujícím atributu `xml:base`, pokud je uveden na elementu `<fo:root>`. Nabízí se též předání cesty k adresáři zdrojového dokumentu XSLT parametrem. V případě použití dávkových souborů lze adresář zjistit a předat dokonce automaticky. Toto však není v ukázkové aplikaci implementováno.

A jak probíhá samotná konverze obrázků? Šablona aplikovaná na element `<fo:external-graphic>` (element `<fo:instream-foreign-object>` v ukázkové aplikaci neimplementován) se nejprve pokusí zjistit rozměry obrázku. V případě úspěchu se pokusí načíst obsah obrázku a pokračuje spočítáním výsledných rozměrů obrázku a rozměrů plochy vyhrazené pro obrázek na základě atributů obrázkového elementu a zjištěných skutečných rozměrů obrázku. Následuje příprava hodnot ODF atributů (zejména `svg:width`, `svg:height` a `fo:clip`) na základě zjištěných rozměrů a vygenerování odpovídajícího `<draw:frame>` elementu s vnořeným obrázkem a přidruženého stylového elementu.

5.2.14. Šablony pro plovoucí objekty

Šablona pro element `<fo:float>` dělá víceméně přesně to, co bylo popsáno v porovnání obou formátů. To znamená, že odpovídající element `<draw:frame>` je vygenerován pouze v případě, že hodnota atributu `float` se nerovná `none` nebo `before`. Jinak jsou pouze zavolány šablony na potomky daného `<fo:float>` elementu.

Požadavek ODF na vnoření vygenerovaného `<draw:frame>` elementu v odstavci, tj. `<text:p>` elementu, je implementován ve fázi „dodatečná úprava výsledků transformace“: pokud za `<draw:frame>` elementem následuje odstavec, je šablonou pro `<draw:frame>` element vynechán a při zavolání šablony pro daný odstavec zkopírován na začátek odstavce.

Co se týče převodu atributu `clear` na ODF atribut `style:number-wrapped-paragraphs`, tak ten není z důvodu větší složitosti implementován.

5.2.15. Šablony pro poznámky pod čarou

Šablona pro element `<fo:footnote>` je opět jedna z těch jednodušších. Je v ní vygenerován element `<text:span>` jako výsledek konverze vnořeného `<fo:inline>` elementu následovaný elementem `<text:note>`, který obsahuje element `<text:note-citation>` s prázdným textem a dále element `<text:note-body>` s výsledkem konverze obsahu odpovídajícího `<fo:footnote-body>` elementu.

5.2.16. Šablony pro dynamické elementy

Šablony pro dynamické elementy jsou zpravidla velmi krátké a vycházejí přímo z tabulky 4.1 – „Dynamické elementy – porovnání zápisu“ – pro element `<fo:page-number>` je vygenerován element `<text:page-number>`, pro `<text:page-number-citation>` element `<text:bookmark-ref>` a pro `<fo:basic-link>` element `<text:a>`. Pomocí pojmenované šablony `generate-odf-text-span` (viz výše) je zároveň vytvořen obalující element `<text:span>` jako „nosič“ formátovacích vlastností, pokud jsou na zdrojovém elementu nějaké uvedeny.

Pro každý formátovací objekt, který má nastaven atribut `id`, je ještě před dalším zpracováním objektu vygenerován ODF element `<text:bookmark>` s atributem `text:name` s hodnotou atributu `id`. Aby vznikl validní ODF dokument, je pak ve fázi „dodatečná úprava výsledků transformace“ provedeno přesunutí každého `<text:bookmark>` elementu do nejbližšího následujícího odstavce, pokud ještě v žádném není, a to obdobným způsobem, který byl uveden u popisu šablon pro plovoucí objekty.

5.3. Rozhraní ukázkové aplikace

Abychom z FO dokumentu získali ODF dokument, je potřeba určitým způsobem předat FO dokument spolu s vytvořeným XSLT stylem tzv. XSLT procesoru. Procesor provede zadanou konverzi a vrátí výsledný dokument. Pro demonstraci začlenění XSLT stylů do různých typů aplikací byly spolu se styly připraveny dávkové soubory pro spuštění konverze z příkazové řádky a jednoduchá webová aplikace napsaná v jazyku PHP. V dalších odstavcích bude kromě těchto dvou rozhraní jako první popsán postup instalace vytvořených stylů do editoru OpenOffice Writer.

5.3.1. XML filtr v OpenOffice Writer

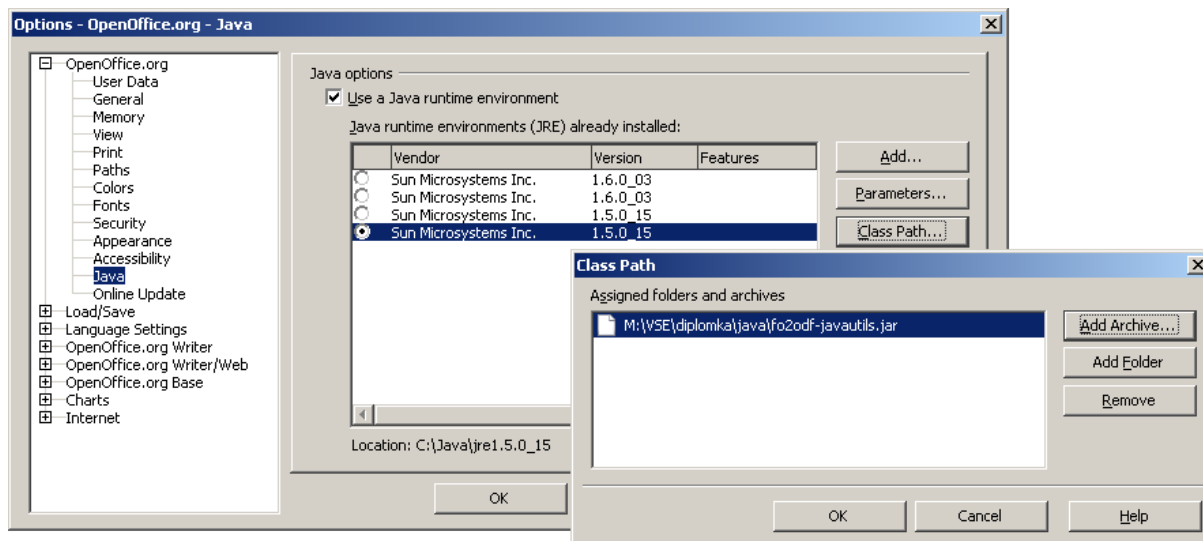
V kancelářském balíku OpenOffice lze pro otevření souboru v libovolném XML formátu přímo v editoru použít tzv. XML filtry. XML filtr specifikuje, pro jaký typ dokumentů bude použit a jaký XSLT styl bude použit pro import dokumentu, tj. pro otevření dokumentu přímo v editoru. Případně lze ve filtru nastavit i (nebo pouze) styl pro export dokumentu do daného formátu. OpenOffice používá pro provedení konverze XSLT procesor Xalan-Java⁶ (zkráceně též Xalan-J), který je součástí standardní instalace OpenOffice. Procesor je napsaný v jazyku Java, a pro svůj běh proto vyžaduje běhové prostředí JDK nebo JRE verze 1.3, 1.4 nebo 1.5.⁷

⁶ <http://xml.apache.org/xalan-j/>

⁷ <http://java.sun.com/javase/downloads/>

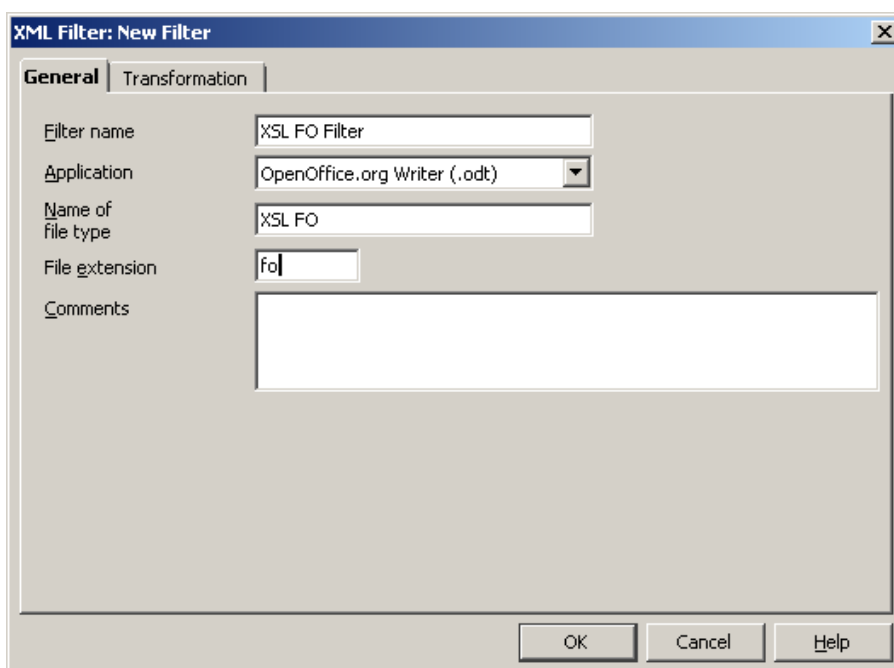
Zde je postup vytvoření XML filtru pro FO dokumenty:

- Nejprve zkontrolujeme, jestli je nainstalováno JDK nebo JRE požadované verze a jestli je nastaveno jeho použití v OpenOffice. To umožňuje po spuštění některého z OpenOffice editorů dialog „Options – OpenOffice.org – Java“. Pokud chceme při otevření FO dokumentu vidět i obrázky, je navíc potřeba v dialogu „Class Path“ nastavit cestu ke knihovnímu souboru `fo2odf-javautils.jar`, který je součástí ukázkové aplikace a obsahuje všechny potřebné funkce. Po provedených změnách je potřeba restartovat OpenOffice. Jedno z možných nastavení ukazuje následující obrázek.

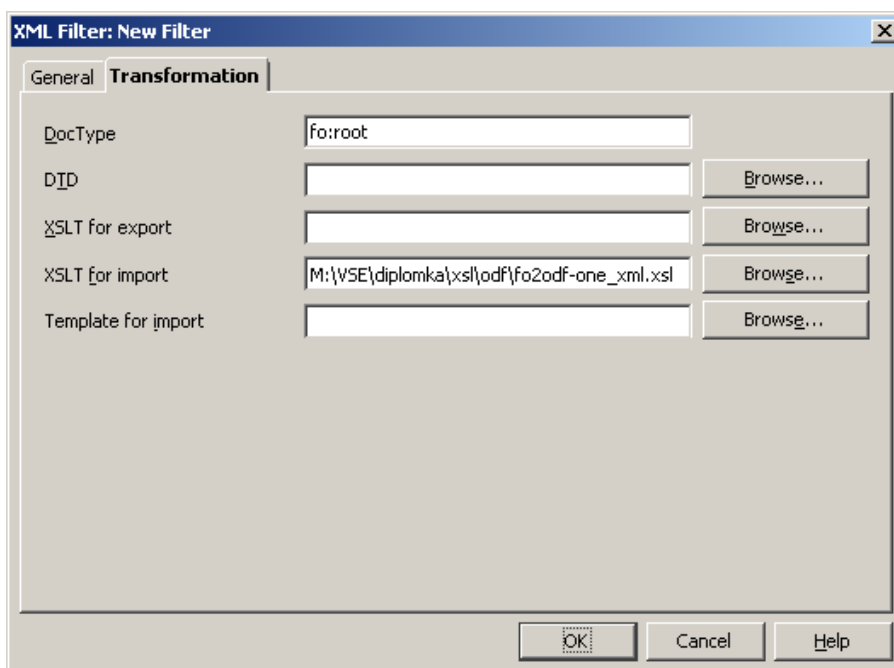


Obrázek 5.2. Nastavení JRE v OpenOffice

- Dále vytvoříme nový importní XML filtr. V menu vybereme „Tools – XML Filter Settings...“. V otevřeném dialogu klikneme na tlačítko „New...“. Objeví se dialog pro vytvoření nového XML filtru. Vyplnění polí na jednotlivých záložkách tohoto dialogu ilustrují následující obrázky.



Obrázek 5.3. Dialog pro vytvoření nového XML filtru pro FO dokumenty – záložka „General“



Obrázek 5.4. Dialog pro vytvoření nového XML filtru pro FO dokumenty – záložka „Transformation“

- Vytvoření filtru potvrdíme kliknutím na tlačítko „OK“. Pokud jsme postupovali správně, měl by se v dialogu pro otevření souboru nyní nacházet typ „XSL FO“ a po otevření souboru s příponou „fo“, který obsahuje jako kořenový element element `<fo:root>`, by mělo dojít k automatickému spuštění XSLT transformace a následnému zobrazení dokumentu přímo v editoru OpenOffice Writer.

Uvedený postup byl úspěšně odzkoušen v OpenOffice verze 2.2 a 2.3. Otevírání FO dokumentů tímto způsobem může být užitečné, nicméně má to tu nevýhodu, že OpenOffice neposkytuje žádné jednoduché prostředky jak nastavit parametry konverze ani přístup k chybovým a dalším hlášením vzniklým během XSLT transformace. Zmíněné nedostatky snad budou odstraněny v budoucích verzích tohoto kancelářského balíku.

5.3.2. Webové rozhraní

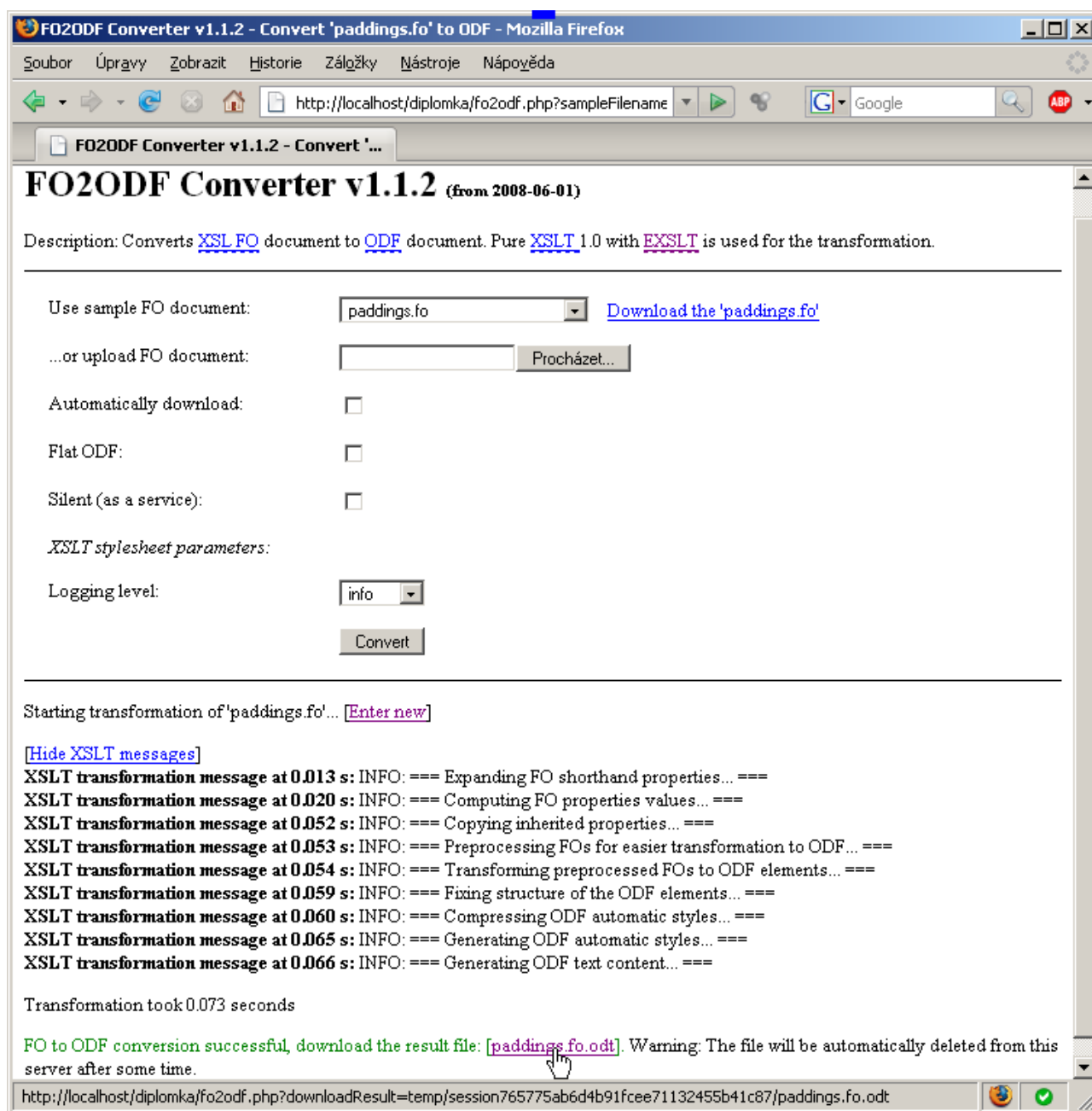
Součástí ukázkové aplikace jsou PHP⁸ skripty, které tvoří jednoduché webové rozhraní pro spouštění konverze nad FO dokumenty z internetového prohlížeče. Toto rozhraní umožňuje pomocí formuláře nastavit, jestli má být výsledkem konverze ODF archiv nebo jeden ODF XML soubor, jestli se mají zobrazovat jen XSLT zprávy od určité úrovně a další. Skripty přitom využívají pro přístup k programovému rozhraní XSLT procesoru PHP třídu `XSLTProcessor`. Konkrétní ukázka této části PHP kódu je uvedena v příloze C – „*PHP kód ukázkové aplikace*“.

Pro instalaci stačí PHP skripty a XSLT styly zkopírovat do adresáře na webovém serveru (např. oblíbený open source server Apache⁹), na kterém je nainstalováno PHP alespoň verze 5.1 obsahující knihovnu libxslt¹⁰ verze 1.1.0 nebo vyšší. Nastavení PHP skriptů lze přitom změnit ve skriptu `serverSettings.inc.php`. Následující obrázek ukazuje, jak vypadá popisované webové rozhraní v prohlížeči po úspěšném dokončení převodu FO dokumentu.

⁸ <http://www.php.net/>

⁹ <http://httpd.apache.org/>

¹⁰ <http://xmlsoft.org/XSLT/>



Obrázek 5.5. Webové rozhraní pro konverzi z FO do ODF

5.3.3. Příkazová řádka

Nejrychlejším způsobem, jak lze konverzi FO dokumentu provést, je spuštění některého z vytvořených dávkových souborů z příkazové řádky operačního systému MS Windows (testováno pouze ve Windows XP).¹¹

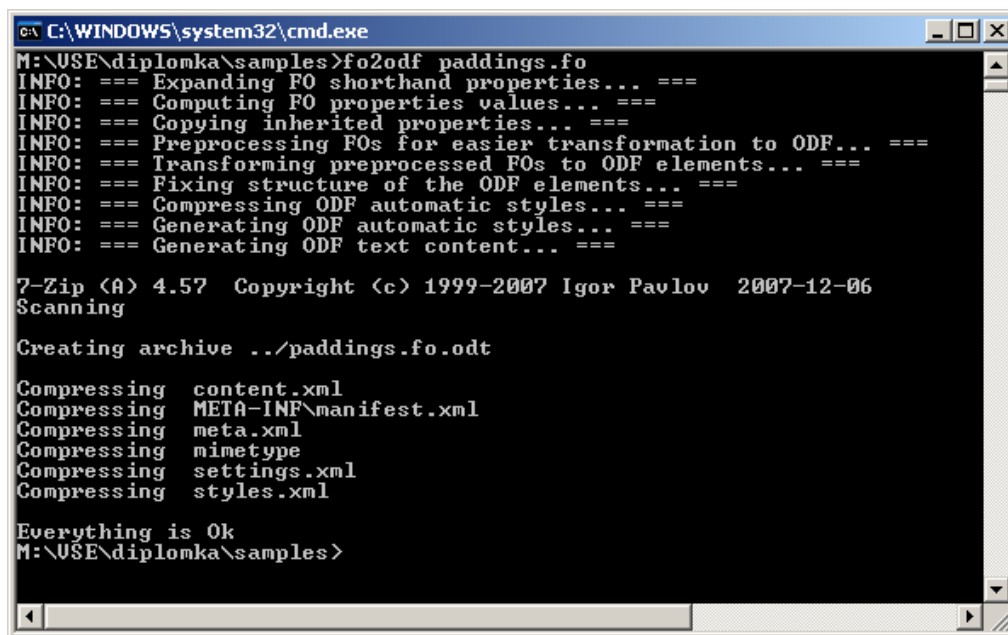
Dávkové soubory volají XSLT procesor xsltproc (nastavba pro příkazovou řádku nad knihovnou libxslt), nebo procesor Xalan-Java. Oba procesory jsou uloženy ve složce „lib“ ukázkové aplikace. Jediným

¹¹Pro operační systém UNIX by šly nicméně obdobné dávkové soubory také relativně jednoduše připravit.

krokem při instalaci je tak přidání cesty k adresáři dávkových souborů do systémové proměnné *PATH*. Pak už je možné spustit konverzi z jakéhokoli adresáře.

Pro zabalení vygenerovaných souborů do ODF archivu, což je vlastně soubor ve formátu ZIP, používají dávkové soubory open source program 7-zip. Dávkové soubory samozřejmě není v případě potřeby problém upravit, aby používaly jiný archivovací program (např. program „jar“ obsažený ve standardní distribuci JDK).

Poslední obrázek této kapitoly zobrazuje příkazovou řádku po úspěšném dokončení převodu FO dokumentu.



```
C:\WINDOWS\system32\cmd.exe
M:\USE\diplomka\samples>fo2odf paddings.fo
INFO: === Expanding FO shorthand properties... ===
INFO: === Computing FO properties values... ===
INFO: === Copying inherited properties... ===
INFO: === Preprocessing FOs for easier transformation to ODF... ===
INFO: === Transforming preprocessed FOs to ODF elements... ===
INFO: === Fixing structure of the ODF elements... ===
INFO: === Compressing ODF automatic styles... ===
INFO: === Generating ODF automatic styles... ===
INFO: === Generating ODF text content... ===

7-Zip (A) 4.57 Copyright (c) 1999-2007 Igor Pavlov 2007-12-06
Scanning

Creating archive ../paddings.fo.odt

Compressing content.xml
Compressing META-INF\manifest.xml
Compressing meta.xml
Compressing mimetype
Compressing settings.xml
Compressing styles.xml

Everything is Ok
M:\USE\diplomka\samples>
```

Obrázek 5.6. Konverze z FO do ODF z příkazové řádky

Kapitola 6

Závěr

V této diplomové práci jsem popsal formáty XSL FO a ODF, uvedl jejich srovnání a nastínil možnosti konverze dokumentů z formátu XSL FO do formátu ODF. Na ukázkové aplikaci, která je obsažena na přiloženém CD, jsem pak ilustroval implementaci tohoto převodu pomocí technologie XSLT a možnosti relativně jednoduchého začlenění vytvořených XSLT stylů do existujících aplikací. Ukázal jsem, že pomocí XSLT verze 1.0 lze provádět i složitější transformace, při jejichž implementaci se nicméně neobejdeme bez použití rozšiřujících XSLT funkcí definovaných de-facto standardem EXSLT.

Na základě srovnání formátů lze říci, že převod formátovacích objektů a vlastností do ODF je z velké části proveditelný a vzhledem k jisté podobnosti formátů neztídká i relativně přímočarý. Kompletní převod FO dokumentů, které využívají pokročilejších možností nabízených formátem XSL FO (např. „živý“ text v záhlaví nebo podmíněný formát stránek), nicméně může být realizován jen obtížně nebo vůbec ne, zpravidla proto, že v ODF neexistuje obdoba určitého formátovacího objektu nebo vlastnosti. Stupeň převoditelnosti se však samozřejmě může měnit na základě změn v budoucích verzích obou formátů. Například další verze ODF by měla být schválena a pravděpodobně i implementována příslušnými kancelářskými balíky již tento rok.

Vzhledem ke komplexnosti formátovacích objektů a omezenému rozsahu tohoto textu jsme se plně nestačili věnovat např. možnostem převodu směru psaní textu (FO atribut `writing-mode`), víceloupcovým sekcím a některým dalším formátovacím objektům a vlastnostem. Také ukázková aplikace neimplementuje všechny nastíněné části konverze, pro mnoho dokumentů ovšem bude výsledek téměř jistě uspokojivý. Je však potřeba upozornit na to, že transformace delších a složitějších dokumentů pomocí vytvořených XSLT stylů může trvat řádově i desítky vteřin (ať už při spuštění procesorem Xalan-Java, nebo libxslt). Pro častější používání by se tak vyplatilo styly ještě, třeba i za cenu menší přehlednosti, optimalizovat, aby se alespoň trochu přiblížily rychlosti existujících FO procesorů.

Literatura

- [1] BALL, S.: *XSLT Standard Library 1.2.1*. 2004. Dostupné na WWW: <<http://xslt.sourceforge.net/>>
- [2] EXSLT.org: *Extensions to XSLT*. Dostupné na WWW: <<http://exslt.org/>>
- [3] KOSEK, J.: *XML pro každého*. Grada Publishing, Praha 2000. ISBN 80-7169-860-1. Dostupné na WWW: <<http://www.kosek.cz/xml/xmlprokazdeho.pdf>>
- [4] KOSEK, J.: *XSLT v příkladech*. Dostupné na WWW: <<http://www.kosek.cz/xml/xslt/index.html>>
- [5] OASIS: *Open Document Format for Office Applications (OpenDocument) v1.1*. Únor 2007. Dostupné na WWW: <<http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1.pdf>>
- [6] OpenOffice.org: *OpenOffice.org XML Project Wiki Pages*. Dostupné na WWW: <<http://wiki.services.openoffice.org/wiki/Xml>>
- [7] PACHMAN, J.: *Převod FO do WordML*. Vysoká škola ekonomická v Praze, srpen 2004. Dostupné na WWW: <<http://fo2wordml.sourceforge.net>>
- [8] RenderX Inc.: *XSL Formatting Objects Tutorial*. 2003. Dostupné na WWW: <<http://www.renderx.com/tutorial.html>>
- [9] W3C: *Extensible Stylesheet Language (XSL) 1.0*. W3C doporučení. Říjen 2001. Dostupné na WWW: <<http://www.w3.org/TR/2001/REC-xsl-20011015/>>
- [10] W3C: *XSL Transformations (XSLT) 1.0*. W3C doporučení. Listopad 1999. Dostupné na WWW: <<http://www.w3.org/TR/1999/REC-xslt-19991116>>
- [11] WALSH, N.: *The Design of the DocBook XSL Stylesheets*. Duben 2001. Dostupné na WWW: <<http://nwalsh.com/docs/articles/dbdesign/>>
- [12] Wikipedia: *OpenDocument*. Dostupné na WWW: <<http://en.wikipedia.org/wiki/OpenDocument>>

Obecný rejstřík

C

content.xml, 20
CSS, 11, 21

D

DocBook, 9

E

EXSLT, 41

F

FO procesor, 9, 10
FO2ODF Converter, 40
formátovací objekty (Viz XSL FO)

M

manifest.xml, 20
meta.xml, 21

O

ODF, 19
 dynamické elementy, 26
 elementy pro seznamy, 23
 elementy pro tabulky, 24
 formátovací vlastnosti, 21
 inline elementy, 25
 obrázkové elementy, 25
 odstavcové elementy, 22
 ostatní elementy, 27
 out-of-line elementy, 26
 rozvržení stránky, 21
 XML schéma, 19
ODF archiv, 20
OpenOffice, 19
 XML filtr, 47

P

PDF, 9

S

settings.xml, 21
styles.xml, 20

T

třída XSLTProcessor, 50

X

XSL, 9
XSL FO, 9
 dynamické elementy, 17
 elementy pro seznamy, 14
 elementy pro tabulky, 15
 formátovací vlastnosti, 11
 inline elementy, 16
 obrázkové elementy, 16
 odstavcové elementy, 13
 ostatní elementy, 17
 out-of-line elementy, 16
 rozvržení stránky, 11
 XML schéma, 10
XSLT, 9, 39

Rejstřík elementů

A

text:a, **26**, 47
attribute-set, 42
office:automatic-styles, 20, **21**, 43

B

fo:basic-link, **17**, 47
fo:bidirectional-override, **16**
office:binary-data, 25, 33
fo:block, 10, 11, **13**, 16, 18, 30, 44, 46
fo:block-container, 11, **13**, 16
office:body, **20**, 21
text:bookmark, **26**, 30, 34, 35, 47
text:bookmark-ref, **26**, 35, 47

C

fo:conditional-page-master-reference, **13**, 30
table:covered-table-cell, **24**, 31, 32, 45

D

dl, 31
office:document, **20**, 42
office:document-content, 20, 42
office:document-meta, 21
office:document-settings, 21
office:document-styles, 20, 42

E

fo:external-graphic, 11, **16**, 32, 33, 46

F

fo:float, **17**, 26, 33, 34, 46
fo:flow, **10**, 11, 44, 45
office:font-face-decls, 20
style:footer, **21**, 29, 44
style:footer-style, 29
fo:footnote, **16**, 26, 34, 47
fo:footnote-body, **16**, 34, 47
style:footnote-sep, **26**
draw:frame, **25**, **26**, 33, 34, 46, 47
func:function, 43

H

text:h, 20, **22**, 26, 36
style:header, **21**, 29, 44
style:header-footer-properties, 29

style:header-left, 21
style:header-style, 29

Ch

text:chapter, **26**, 36
fo:character, **16**

I

draw:image, **25**, 33
fo:initial-property-set, **18**
fo:inline, 11, **16**, 32, 34, 45, 47
fo:inline-container, **16**
fo:instream-foreign-object, **16**, 32, 33, 46

L

fo:layout-master-set, **10**, 11
fo:leader, **16**, 32, 46
text:line-break, 37
text:list, **23**, 31
fo:list-block, 11, **14**, 31, 45
fo:list-item, **14**, **23**, 31, 45
fo:list-item-body, **14**, 31, 45
fo:list-item-label, **14**, 45
style:list-level-properties, **23**
text:list-level-style-number, **23**
text:list-style, **23**

M

fo:marker, **17**, 36
style:master-page, **21**, 29, 30, 44
office:master-styles, 20
message, 42
office:meta, 20
fo:multi-properties, 17
fo:multi-switch, 17
fo:multi-toggle, 17

N

text:note, **26**, 34, 47
text:note-body, **26**, 34, 47
text:note-citation, **26**, 34, 47
text:notes-configuration, **26**

P

text:p, 19, 20, **22**, 30, 33, 34, 44, 46, 47
text:page-count, **26**
style:page-layout, **21**, 29, 30, 44
style:page-layout-properties, 26, 29
fo:page-number, 11, **17**, **26**, 47

fo:page-number-citation, **17**, 47
fo:page-sequence, **10**, 11, 29, 30, 44, 45
fo:page-sequence-master, **11**, 44
style:paragraph-properties, 21, 37
param, 42

R

fo:region-after, 11, 29
fo:region-before, 11, 29
fo:region-body, **11**, 29, 45
fo:region-end, 11, 29
fo:region-start, 11, 29
fo:repeatable-page-master-alternatives, **13**
fo:retrieve-marker, **17**, 36
fo:root, **10**, 46, 49

S

text:s, 37
office:scripts, 20
office:settings, 20
fo:simple-page-master, **11**, 12, 29, 44, 45
span, 16, **25**, 29, 32, 34, 35, 45, 46, 47
fo:static-content, **10**, 17, 29, 44, 45
style:style, **21**, 28, 43, 44
office:styles, 20, **21**, 26

T

text:tab, **27**, 32, 37, 46
style:tab-stop, **27**, 32, 46
style:tab-stops, 46
fo:table, 11, **15**, 16, **24**, 31, 35, 45
fo:table-and-caption, 11, **16**, 45
fo:table-body, **15**, 31
fo:table-caption, 16, 45
fo:table-cell, **15**, **24**, 32, 45
fo:table-column, **15**, **24**, 31, 45
fo:table-footer, **15**, 31
fo:table-header, **15**, 31
table:table-header-rows, **24**, 31
style:table-properties, 21, 37
fo:table-row, **15**, **24**, 45
style:table-row-properties, 37
table:table-rows, **24**, 31
office:text, **20**
draw:text-box, **26**, 33
style:text-properties, 21, 36

W

fo:wrapper, **17**, 29

Rejstřík atributů

A

text:anchor-type, 25

B

xml:base, 46
border-before, 29
border-left-width, 45
border-right-width, 45
border-top, 29

C

text:c, 37
clear, 17, 33, 47
clip, 16, 33, 46
column-width, 15
content-height, 16
content-width, 16

D

text:display, 26, 36
display-align, 16, 33

E

end-indent, 14
ends-row, 15
external-destination, 17

F

style:family, 21
float, 17, 26, 33, 34, 46
flow-name, 10, 17, 44
font, 29
font-family, 38

H

height, 16, 25, 33, 46
style:horizontal-pos, 26
style:horizontal-rel, 26, 33
xlink:href, 25, 26, 33

Ch

style:char, 27

I

id, 17, 30, 35, 47
internal-destination, 17

J

style:join-border, 38

K

keep-together, 36, 37
keep-with-next, 36, 37
keep-with-next.within-page, 36
keep-with-previous, 36

L

style:leader-style, 27
style:leader-text, 27
style:leader-type, 27

M

margin, 10
margin-left, 45
margin-right, 45
marker-class-name, 17, 36
style:master-page-name, 21, 30, 44
master-reference, 11, 30, 44
mimetype, 20
fo:min-height, 34
text:min-label-distance, 23
text:min-label-width, 23
fo:min-width, 34

N

style:name, 21, 26, 43, 47
text:note-class, 26, 34
number-columns-repeated, 45
number-columns-spanned, 15, 24, 32
number-rows-spanned, 15, 24, 32, 45
style:number-wrapped-paragraphs, 33, 47

P

padding, 29
padding-left, 45
padding-right, 45
page-width, 45
style:position, 27
provisional-distance-between-starts, 14, 23, 31
provisional-label-separation, 14, 23, 31

R

ref-id, 17
text:ref-name, 26
text:reference-format, 26
retrieve-boundary, 17, 36

retrieve-class-name, 17, 36

retrieve-position, 17, 36

S

src, 16

start-indent, 14

starts-row, 15

style-name, 21, 23, 28, 35

T

tab-position, 32, 46

table-omit-footer-at-break, 15

table-omit-header-at-break, 15, 31

target, 26

office:target-frame-name, 26

temp-is-first, 44

temp-is-last, 44

text-align, 16, 33

text-align-last, 32, 46

office:title, 26

style:type, 27, 32

V

style:vertical-pos, 26

style:vertical-rel, 26, 33

W

white-space, 37

width, 16, 25, 33, 46

style:wrap, 26

writing-mode, 13, 29, 53

Slovník

CSS (Cascading Style Sheets)	Stylový jazyk pro popis vzhledu XML dokumentu. Nejznámějším typem dokumentu, pro který CSS se používá, je HTML.
DocBook	XML, případně SGML, jazyk pro zápis dokumentace, původně zejména technické (softwarové a hardwarové) dokumentace.
XML (eXtensible Markup Language)	Obecný značkovací jazyk (formát) umožňující definovat a používat vlastní značkovací jazyk (formát), a to zejména pro zápis a přenos strukturovaných dat (v podobě tzv. XML dokumentů). XML je zjednodušenou podmnožinou jazyka SGML („Standard Generalized Markup Language“).
PHP (Hypertext Preprocessor)	Skriptovací jazyk používaný pro dynamické generování dokumentů (typicky HTML stránek) na straně webového serveru (multiplatformní open source software).
ODF (OpenDocument Format)	Formát pro reprezentaci textových dokumentů, prezentací, dokumentů tabulkových procesorů a jiných aplikací. Tento formát používají zejména aplikace kancelářského balíku OpenOffice.
URI (Uniform Resource Identifier)	Jednotný identifikátor zdroje – univerzální schéma užívané pro adresování zdrojů.
URL (Uniform Resource Locator)	Jednotný ukazatel na zdroj – podmnožina URI, která identifikuje dokument místem jeho uložení.
XPath (XML Path Language)	Jazyk pro adresování částí XML dokumentu a pro provádění základních výpočtů nad nimi.
XSL (XML Stylesheet Language)	Sada technologií (jazyků) umožňujících transformaci a formátování XML dokumentů. Viz též XSLT, XSL FO, XPath.
XSL FO (XSL Formatting Objects)	XML jazyk pro popis formátování (vzhledu) stránkovaných textových dokumentů.
XSLT (XSL Transformations)	XML jazyk pro transformaci XML dokumentů do jiných XML nebo textových dokumentů.

Příloha A

Implementované FO elementy

Tabulka A.1. Implementované FO elementy (objekty)

Formátovací objekt	Implementováno	Poznámka
Objekty pro deklarace, pro rozvržení a obsah stránek		
fo:root	ano	
fo:declarations	ne	
fo:color-profile	ne	
fo:page-sequence	ano	
fo:layout-master-set	ano	
fo:page-sequence-master	ne	Použije se první odkazovaný fo:simple-page-master.
fo:single-page-master-reference	ne	Použije se první odkazovaný fo:simple-page-master.
fo:repeatable-page-master-reference	ne	Použije se první odkazovaný fo:simple-page-master.
fo:conditional-page-master-reference	ne	Použije se první odkazovaný fo:simple-page-master.
fo:simple-page-master	ano	
fo:region-body	ano	
fo:region-before	ano	
fo:region-after	ano	
fo:region-start	ne	ODF podporuje pouze záhlaví a zápatí.
fo:region-end	ne	ODF podporuje pouze záhlaví a zápatí.
fo:flow	ano	
fo:static-content	ano	Pouze pro regiony „body“, „before“ a „after“.
fo:title	ne	V ODF lze nastavit titulek pouze pro celý dokument.
Blokové objekty		
fo:block	ano	Omezený převod formátovacích vlastností při použití vnořených bloků.

Formátovací objekt	Implementováno	Poznámka
fo:block-container	ano	Jen přibližná implementace, bez podpory změny orientace textu.
Objekty pro seznamy		
fo:list-block	ano	S omezeními vyplývajícími z implementace pomocí převodu seznamu na tabulku.
fo:list-item	ano	
fo:list-item-body	ano	
fo:list-item-label	ano	
Objekty pro tabulky		
fo:table-and-caption	ano	Jednoduše jen zpracován obsah elementu.
fo:table	ano	
fo:table-column	ano	
fo:table-caption	ano	Jednoduše jen zpracován obsah elementu.
fo:table-header	ano	
fo:table-footer	ano	Bez podpory opakování zápatí (omezení ODF).
fo:table-body	ano	
fo:table-row	ano	
fo:table-cell	ano	
Inline objekty		
fo:inline	ano	
fo:inline-container	ne	Pouze převeden obsah elementu.
fo:leader	ano	Šířkové a některé další atributy nepodporovány (dáno převodem na ODF tabulátor).
fo:character	ne	
fo:bidirectional-override	ne	
Objekty pro obrázky		
fo:external-graphic	ano	
fo:instream-foreign-object	ne	
„out-of-line“ objekty		
fo:float	ano	Bez podpory atributu clear. Obtékání pouze na úrovni odstavců (omezení ODF). Hodnota before atributu float není podporována (omezení ODF).
fo:footnote	ano	
fo:footnote-body	ano	
Dynamické objekty		
fo:basic-link	ano	
fo:page-number	ano	

Formátovací objekt	Implementováno	Poznámka
fo:page-number-citation	ano	
fo:multi-switch	ne	
fo:multi-case	ne	
fo:multi-toggle	ne	
fo:multi-properties	ne	
fo:multi-property-set	ne	
Ostatní objekty		
fo:wrapper	ano	
fo:marker	ne	
fo:retrieve-marker	ne	
fo:initial-property-set	ne	

Příloha B

Implementované FO vlastnosti

Následující tabulka obsahuje seznam všech formátovacích vlastností definovaných v XSL FO. Vlastnosti a jejich hodnoty, které byly implementovány v ukázkové aplikaci, jsou psány *kurzívou*. Sloupec „Úroveň“ pak udává, ke které úrovni implementace (základní, rozšířená, kompletní) daná vlastnost dle XSL specifikace patří. Vlastnosti, u nichž je ve sloupci „Výchozí hodnota“ uvedeno „viz konkrétní vlastnosti“, jsou tzv. zkratkové vlastnosti.

Tabulka B.1. Implementované FO vlastnosti

Vlastnost	Hodnoty	Výchozí hodnota	Dědičná	Úroveň
absolute-position	auto absolute fixed inherit	auto	ne	kompletní
active-state	link visited active hover focus	prázdný řetězec	hodnota je povinná	rozšířená
alignment-adjust	auto baseline before-edge text-before-edge middle central after-edge text-after-edge ideographic alphabetic hanging mathematical <procenta> <délka> inherit	auto	ne	základní
alignment-baseline	auto baseline before-edge text-before-edge middle central after-edge text-after-edge ideographic alphabetic hanging mathematical inherit	auto	ne	základní
auto-restore	true false	false	ano	rozšířená
azimuth	<angle> [[left-side far-left left center-left center center-right right far-right right-side] behind] leftwards rightwards inherit	center	ano	základní
<i>background</i>	[<background-color> <background-image> <background-repeat> <background-attachment> <background-position>] <i>inherit</i>	viz konkrétní vlastnosti	ne	kompletní
background-attachment	scroll fixed inherit	scroll	ne	rozšířená
<i>background-color</i>	<barva> <i>transparent</i> <i>inherit</i>	transparent	ne	základní
background-image	<URI> none inherit	none	ne	rozšířená

Vlastnost	Hodnoty	Výchozí hodnota	Dědičná	Úroveň
background-position	[[<procenta> <délka>] {1,2} [[top center bottom] [left center right]]] inherit	0% 0%	ne	kompletní
background-position-horizontal	<procenta> <délka> left center right inherit	0%	ne	rozšířená
background-position-vertical	<procenta> <délka> top center bottom inherit	0%	ne	rozšířená
background-repeat	repeat repeat-x repeat-y no-repeat inherit	repeat	ne	rozšířená
baseline-shift	baseline sub super <procenta> <délka> inherit	baseline	ne	základní
blank-or-not-blank	blank not-blank any inherit	any	ne	rozšířená
block-progression-dimension	auto <délka> <procenta> <rozmezi> inherit	auto	ne	základní
border	[<border-width> <border-style> <barva>] inherit	viz konkrétní vlastnosti	ne	kompletní
border-„strana“-color	<barva> inherit	the value of the 'color' property	ne	základní
border-„strana“-precedence	force <celé číslo> inherit	viz specif.	ne	základní
border-„strana“-style	none hidden dotted dashed solid double groove ridge inset outset inherit	none	ne	základní
border-„strana“-width	thin medium thick <délka> <podmíněná délka> inherit	medium	ne	základní
border-„strana“-color	<barva> inherit	hodnota vlastnosti 'color'	ne	základní
border-„strana“	[<border-width> <border-style> <barva>] inherit	viz konkrétní vlastnosti	ne	kompletní
border-collapse	collapse collapse-with-precedence separate inherit	collapse	ano	rozšířená
border-color	[<barva> transparent] {1,4} inherit	viz konkrétní vlastnosti	ne	kompletní
border-separation	<length-bp-ip-direction> inherit	0pt	ano	rozšířená
border-spacing	<délka> <délka>? inherit	0pt	ano	kompletní

Vlastnost	Hodnoty	Výchozí hodnota	Dědičná	Úroveň
<i>border-style</i>	<code><border-style> {1,4} inherit</code>	viz konkrétní vlastnosti	ne	kompletní
<i>border-width</i>	<code><border-width> {1,4} inherit</code>	viz konkrétní vlastnosti	ne	kompletní
<i>bottom</i>	<code><délka> <procenta> auto inherit</code>	auto	ne	rozšířená
<i>break-after</i>	<code>auto column page even-page odd-page inherit^a</code>	auto	ne	základní
<i>break-before</i>	<code>auto column page even-page odd-page inherit^a</code>	auto	ne	základní
<i>caption-side</i>	<code>before after start end top bottom left right inherit</code>	before	ano	kompletní
<i>case-name</i>	<code><název></code>	hodnota je povinná	hodnota je povinná	rozšířená
<i>case-title</i>	<code><řetězec></code>	hodnota je povinná	hodnota je povinná	rozšířená
<i>character</i>	<code><znak></code>	hodnota je povinná	hodnota je povinná	základní
<i>clear</i>	<code>start end left right both none inherit</code>	none	ne	rozšířená
<i>clip</i>	<code><tvar> auto inherit</code>	auto	ne	rozšířená
<i>color</i>	<code><barva> inherit</code>	závislé na prohlížeči	ano	základní
<i>color-profile-name</i>	<code><název> inherit</code>	hodnota je povinná	ne	rozšířená
<i>column-count</i>	<code><číslo> inherit</code>	1	ne	rozšířená
<i>column-gap</i>	<code><délka> <procenta> inherit</code>	12.0pt	ne	rozšířená
<i>column-number</i>	<code><číslo></code>	viz specif.	ne	základní
<i>column-width</i>	<code><délka> <procenta></code>	viz specif.	ne	základní
<i>content-height</i>	<code>auto scale-to-fit <délka> <procenta> inherit</code>	auto	ne	rozšířená
<i>content-type</i>	<code><řetězec> auto</code>	auto	ne	rozšířená

Vlastnost	Hodnoty	Výchozí hodnota	Dědičná	Úroveň
<i>content-width</i>	<i>auto</i> <i>scale-to-fit</i> <i><délka></i> <i><procenta></i> <i>inherit</i>	auto	ne	rozšířená
country	<i>none</i> <i><země></i> <i>inherit</i>	none	ano	rozšířená
cue	<i><cue-before></i> <i><cue-after></i> <i>inherit</i>	viz konkrétní vlastnosti	ne	kompletní
cue-after	<i><URI></i> <i>none</i> <i>inherit</i>	none	ne	základní
cue-before	<i><URI></i> <i>none</i> <i>inherit</i>	none	ne	základní
destination-placement-offset	<i><délka></i>	0pt	ne	rozšířená
direction	<i>ltr</i> <i>rtl</i> <i>inherit</i>	ltr	ano	základní
<i>display-align</i>	<i>auto</i> <i>before</i> <i>center</i> <i>after</i> <i>inherit</i>	auto	ano	rozšířená
dominant-baseline	<i>auto</i> <i>use-script</i> <i>no-change</i> <i>reset-size</i> <i>ideographic</i> <i>alphabetic</i> <i>hanging</i> <i>mathematical</i> <i>central</i> <i>middle</i> <i>text-after-edge</i> <i>text-before-edge</i> <i>inherit</i>	auto	ne	základní
elevation	<i><úhel></i> <i>below</i> <i>level</i> <i>above</i> <i>higher</i> <i>lower</i> <i>inherit</i>	level	ano	základní
empty-cells	<i>show</i> <i>hide</i> <i>inherit</i>	show	ano	rozšířená
<i>end-indent</i>	<i><délka></i> <i><procenta></i> <i>inherit</i>	0pt	ano	základní
ends-row	<i>true</i> <i>false</i>	false	ne	rozšířená
extent	<i><délka></i> <i><procenta></i> <i>inherit</i>	0.0pt	ne	rozšířená
<i>external-destination</i>	<i><URI></i>	prázdný řetězec	ne	rozšířená
<i>float</i>	<i>before</i> <i>start</i> <i>end</i> <i>left</i> <i>right</i> <i>none</i> <i>inherit</i> ^b	none	ne	rozšířená
<i>flow-name</i>	<i><název></i>	prázdný řetězec	hodnota je povinná	základní
<i>font</i>	[[<i><font-style></i> <i><font-variant></i> <i><font-weight></i>]? <i><font-size></i> [/ <i><line-height></i>]? <i><font-family></i>] <i>caption</i> <i>icon</i> <i>menu</i> <i>message-box</i> <i>small-caption</i> <i>status-bar</i> <i>inherit</i>	viz konkrétní vlastnosti	ano	kompletní
<i>font-family</i>	<i><řetězec názvů fontů seřazených dle priority></i> <i>inherit</i> ^c	závislé na prohlížeči	ano	základní
font-selection-strategy	<i>auto</i> <i>character-by-character</i> <i>inherit</i>	auto	ano	kompletní
<i>font-size</i>	<i><xx-small></i> ... <i>medium</i> ... <i>xx-large></i> <i><larger></i> <i><smaller></i> <i><délka></i> <i><procenta></i> <i>inherit</i>	medium	ano, dědí se spočíta-	základní

Vlastnost	Hodnoty	Výchozí hodnota	Dědičná	Úroveň
			ná hodnota	
font-size-adjust	<číslo> none inherit	none	ano	rozšířená
font-stretch	normal wider narrower ultra-condensed extra-condensed condensed semi-condensed semi-expanded expanded extra-expanded ultra-expanded inherit	normal	ano	rozšířená
font-style	normal italic oblique backslant inherit	normal	ano	základní
font-variant	normal small-caps inherit	normal	ano	základní
font-weight	normal bold bolder lighter 100 200 300 400 500 600 700 800 900 inherit	normal	ano	základní
force-page-count	auto even odd end-on-even end-on-odd no-force inherit	auto	ne	rozšířená
format	<řetězec>	1	ne	základní
glyph-orientation-horizontal	<úhel> inherit	0deg	ano	rozšířená
glyph-orientation-vertical	auto <úhel> inherit	auto	ano	rozšířená
grouping-separator	<znak>	no separator	ne	rozšířená
grouping-size	<číslo>	no grouping	ne	rozšířená
height	<délka> <procenta> auto inherit ^d	auto	ne	základní
hyphenate	false true inherit	false	ano	rozšířená
hyphenation-character	<znak> inherit ^e	Unicode znak s kódem U+2010	ano	rozšířená
hyphenation-keep	auto column page inherit	auto	ano	rozšířená
hyphenation-ladder-count	no-limit <číslo> inherit	no-limit	ano	rozšířená
hyphenation-push-character-count	<číslo> inherit	2	ano	rozšířená
hyphenation-remain-character-count	<číslo> inherit	2	ano	rozšířená
id	<id>	viz spec. cif.	ne, viz spec. cif.	základní
indicate-destination	true false	false	ne	rozšířená
initial-page-number	auto auto-odd auto-even <číslo> inherit	auto	ne	základní
inline-progression-dimension	auto <délka> <procenta> <rozmezí> inherit	auto	ne	základní

Vlastnost	Hodnoty	Výchozí hodnota	Dědičná	Úroveň
<i>internal-destination</i>	<idref>	prázdný řetězec	ne	rozšířená
<i>intrusion-displace</i>	auto none line indent block inherit	auto	ano	rozšířená
<i>keep-together</i>	auto always inherit ^f	auto	ano	rozšířená
<i>keep-with-next</i>	auto always inherit ^g	auto	ne	základní
<i>keep-with-previous</i>	auto always inherit ^h	auto	ne	základní
<i>language</i>	none <jazyk> inherit	none	ano	rozšířená
<i>last-line-end-indent</i>	<délka> <procenta> inherit	0pt	ano	rozšířená
<i>leader-alignment</i>	none reference-area page inherit	none	ano	rozšířená
<i>leader-length</i>	<rozmezí> <procenta> inherit	12pt (viz specif.)	ano	základní
<i>leader-pattern</i>	space rule dots use-content inherit	space	ano	základní
<i>leader-pattern-width</i>	use-font-metrics <délka> <procenta> inherit	use-font-metrics	ano	rozšířená
<i>left</i>	<délka> <procenta> auto inherit	auto	ne	rozšířená
<i>letter-spacing</i>	normal <délka> <rozmezí> inherit	normal	ano	rozšířená
<i>letter-value</i>	auto alphabetic traditional	auto	ne	základní
<i>linefeed-treatment</i>	ignore preserve treat-as-space treat-as-zero-width-space inherit	treat-as-space	ano	rozšířená
<i>line-height</i>	normal <délka> <číslo> <procenta> <rozmezí> inherit	normal	ano	základní
<i>line-height-shift-adjustment</i>	consider-shifts disregard-shifts inherit	consider-shifts	ano	rozšířená
<i>line-stacking-strategy</i>	line-height font-height max-height inherit	max-height	ano	základní
<i>margin</i>	<šířka> {1,4} inherit	viz konkrétní vlastnosti	ne	kompletní
<i>margin-„, strana“</i>	<délka> <procenta> inherit	0pt	ne	základní
<i>marker-class-name</i>	<název>	prázdný řetězec	hodnota je povinná	rozšířená
<i>master-name</i>	<název>	prázdný řetězec	hodnota je povinná	základní
<i>master-reference</i>	<název>	prázdný řetězec	hodnota je povinná	základní

Vlastnost	Hodnoty	Výchozí hodnota	Dědičná	Úroveň
max-height	<délka> <procenta> none inherit	0pt	ne	kompletní
maximum-repeats	<číslo> no-limit inherit	no-limit	ne	rozšířená
max-width	<délka> <procenta> none inherit	none	ne	kompletní
media-usage	auto paginate bounded-in-one-dimension unbounded	auto	ne	rozšířená
min-height	<délka> <procenta> inherit	0pt	ne	kompletní
min-width	<délka> <procenta> inherit	závislé na prohlížeči	ne	kompletní
<i>number-columns-repeated</i>	<číslo>	1	ne	základní
<i>number-columns-spanned</i>	<číslo>	1	ne	základní
<i>number-rows-spanned</i>	<číslo>	1	ne	základní
odd-or-even	odd even any inherit	any	ne	rozšířená
orphans	<celé číslo> inherit	2	ano	základní
overflow	visible hidden scroll error-if-overflow auto inherit	auto	ne	základní
padding	<šířka> {1,4} inherit	viz konkrétní vlastnosti	ne	kompletní
<i>padding-„strana“</i>	<délka> <podmíněná délka> inherit	0pt	ne	základní
page-break-after	auto always avoid left right inherit	auto	ne	kompletní
page-break-before	auto always avoid left right inherit	auto	ne	kompletní
page-break-inside	avoid auto inherit	auto	ano	kompletní
<i>page-height</i>	auto indefinite <délka> inherit	auto	ne	základní
page-position	first last rest any inherit	any	ne	rozšířená
<i>page-width</i>	auto indefinite <délka> inherit	auto	ne	základní
pause	[<čas> <procenta>]{1,2} inherit	závislé na prohlížeči	ne	kompletní
pause-after	<čas> <procenta> inherit	závislé na prohlížeči	ne	základní
pause-before	<čas> <procenta> inherit	závislé na prohlížeči	ne	základní
pitch	<frekvence> x-low low medium high x-high inherit	medium	ano	základní
pitch-range	<číslo> inherit	50	ano	základní

Vlastnost	Hodnoty	Výchozí hodnota	Dědičná	Úroveň
play-during	<URI> mix? repeat? auto none inherit	auto	ne	základní
position	static relative absolute fixed inherit	static	ne	kompletní
precedence	true false inherit	false	ne	rozšířená
<i>provisional-distance-between-starts</i>	<délka> <procenta> <i>inherit</i>	24.0pt	ano	základní
<i>provisional-label-separation</i>	<délka> <procenta> <i>inherit</i>	6.0pt	ano	základní
reference-orientation	0 90 180 270 -90 -180 -270 inherit	0	ano (viz specif.)	rozšířená
<i>ref-id</i>	<idref> <i>inherit</i>	prázdný řetězec	hodnota je povinná	rozšířená
<i>region-name</i>	<i>xsl-region-body</i> xsl-region-start xsl-region-end <i>xsl-region-before</i> <i>xsl-region-after</i> xsl-before-float-separator xsl-footnote-separator <název>	viz specif.	hodnota je povinná	základní
relative-align	before baseline inherit	before	ano	rozšířená
relative-position	static relative inherit	static	ne	rozšířená
rendering-intent	auto perceptual relative-colorimetric saturation absolute-colorimetric inherit	auto	ne	rozšířená
retrieve-boundary	page page-sequence document	page-sequence	ne	rozšířená
retrieve-class-name	<název>	prázdný řetězec	hodnota je povinná	rozšířená
retrieve-position	first-starting-within-page first-including-carryover last-starting-within-page last-ending-within-page	first-starting-within-page	ne	rozšířená
richness	<číslo> inherit	50	ano	základní
right	<délka> <procenta> auto inherit	auto	ne	rozšířená
role	<řetězec> <URI> none inherit	none	ne	základní
rule-style	none dotted dashed solid double groove ridge inherit	solid	ano	základní
rule-thickness	<délka>	1.0pt	ano	základní
scaling	uniform non-uniform inherit	uniform	ne	rozšířená
scaling-method	auto integer-pixels resample-any-method inherit	auto	ne	rozšířená
score-spaces	true false inherit	true	ano	rozšířená
script	none auto <script> inherit	auto	ano	rozšířená

Vlastnost	Hodnoty	Výchozí hodnota	Dědičná	Úroveň
show-destination	replace new	replace	ne	rozšířená
size	<délka>{1,2} auto landscape portrait inherit	auto	ne	kompletní
source-document	<URI> [<URI>]* none inherit	none	ne	základní
<i>space-after</i>	<délka> <rozmězí> <i>inherit</i>	0pt	ne	základní
<i>space-before</i>	<délka> <rozmězí> <i>inherit</i>	0pt	ne	základní
space-end	<rozmězí> <procenta> inherit	0ptr	ne	základní
space-start	<rozmězí> <procenta> inherit	0pt	ne	základní
span	none all inherit	none	ne	rozšířená
speak	normal none spell-out inherit	normal	ano	základní
speak-header	once always inherit	once	ano	základní
speak-numeral	digits continuous inherit	continuous	ano	základní
speak-punctuation	code none inherit	none	ano	základní
speech-rate	<číslo> x-slow slow medium fast x-fast faster slower inherit	medium	ano	základní
<i>src</i>	<URI> <i>inherit</i>	none, value required	ne	základní
<i>start-indent</i>	<délka> <procenta> <i>inherit</i>	0pt	ano	základní
starting-state	show hide	show	ne	rozšířená
starts-row	true false	false	ne	rozšířená
stress	<číslo> inherit	50	ano	základní
suppress-at-line-break	auto suppress retain inherit	auto	ne	rozšířená
switch-to	xsl-preceding xsl-following xsl-any <název>[<název>]*	xsl-any	ne	rozšířená
table-layout	auto fixed inherit	auto	ne	rozšířená
table-omit-footer-at-break	true false	false	ne	rozšířená
<i>table-omit-header-at-break</i>	<i>true</i> <i>false</i>	false	ne	rozšířená
target-presentation-context	use-target-processing-context <URI>	use-target-processing-context	ne	rozšířená
target-processing-context	document-root <URI>	document-root	ne	rozšířená
target-stylesheet	use-normal-stylesheet <URI>	use-normal-stylesheet	ne	rozšířená

Vlastnost	Hodnoty	Výchozí hodnota	Dědičná	Úroveň
<i>text-align</i>	<i>start</i> <i>center</i> <i>end</i> <i>justify</i> <i>inside</i> <i>outside</i> <i>left</i> <i>right</i> <řetězec> <i>inherit</i>	start	ano	základní
<i>text-align-last</i>	<i>relative</i> <i>start</i> <i>center</i> <i>end</i> <i>justify</i> <i>inside</i> <i>outside</i> <i>left</i> <i>right</i> <i>inherit</i>	relative	ano	rozšířená
<i>text-altitude</i>	<i>use-font-metrics</i> <délka> <procenta> <i>inherit</i>	<i>use-font-metrics</i>	ne	rozšířená
<i>text-decoration</i>	<i>none</i> [[<i>underline</i> <i>no-underline</i>] [<i>overline</i> <i>no-overline</i>] [<i>line-through</i> <i>no-line-through</i>] [<i>blink</i> <i>no-blink</i>]] <i>inherit</i>	<i>none</i>	ne – viz specif.	rozšířená
<i>text-depth</i>	<i>use-font-metrics</i> <délka> <procenta> <i>inherit</i>	<i>use-font-metrics</i>	ne	rozšířená
<i>text-indent</i>	<délka> <procenta> <i>inherit</i>	0pt	ano	základní
<i>text-shadow</i>	<i>none</i> [<barva> <délka> <délka> <délka>? ,]* [<barva> <délka> <délka> <délka>?] <i>inherit</i>	<i>none</i>	no, viz specif.	rozšířená
<i>text-transform</i>	<i>capitalize</i> <i>uppercase</i> <i>lowercase</i> <i>none</i> <i>inherit</i>	<i>none</i>	ano	rozšířená
<i>top</i>	<délka> <procenta> <i>auto</i> <i>inherit</i>	<i>auto</i>	ne	rozšířená
<i>treat-as-word-space</i>	<i>auto</i> <i>true</i> <i>false</i> <i>inherit</i>	<i>auto</i>	ne	rozšířená
<i>unicode-bidi</i>	<i>normal</i> <i>embed</i> <i>bidirectional-override</i> <i>inherit</i>	<i>normal</i>	ne	rozšířená
<i>vertical-align</i>	<i>baseline</i> <i>middle</i> <i>sub</i> <i>super</i> <i>text-top</i> <i>text-bottom</i> <procenta> <délka> <i>top</i> <i>bottom</i> <i>inherit</i>	<i>baseline</i>	ne	kompletní
<i>visibility</i>	<i>visible</i> <i>hidden</i> <i>collapse</i> <i>inherit</i>	<i>visible</i>	ano	rozšířená
<i>voice-family</i>	<řetězec názvů hlasů seřazených dle priority> <i>inherit</i>	<i>závislé na prohlížeči</i>	ano	základní
<i>volume</i>	<číslo> <procenta> <i>silent</i> <i>x-soft</i> <i>soft</i> <i>medium</i> <i>loud</i> <i>x-loud</i> <i>inherit</i>	<i>medium</i>	ano	základní
<i>white-space</i>	<i>normal</i> <i>pre</i> <i>nowrap</i> <i>inherit</i>	<i>normal</i>	ano	kompletní
<i>white-space-collapse</i>	<i>false</i> <i>true</i> <i>inherit</i>	<i>true</i>	ano	rozšířená
<i>white-space-treatment</i>	<i>ignore</i> <i>preserve</i> <i>ignore-if-before-linefeed</i> <i>ignore-if-after-linefeed</i> <i>ignore-if-surrounding-linefeed</i> <i>inherit</i>	<i>ignore-if-surrounding-linefeed</i>	ano	rozšířená
<i>widows</i>	<celé číslo> <i>inherit</i>	2	ano	základní
<i>width</i>	<délka> <procenta> <i>auto</i> <i>inherit</i> ¹	<i>auto</i>	ne	základní
<i>word-spacing</i>	<i>normal</i> <délka> <rozměr> <i>inherit</i>	<i>normal</i>	ano	rozšířená
<i>wrap-option</i>	<i>no-wrap</i> <i>wrap</i> <i>inherit</i>	<i>wrap</i>	ano	základní

Vlastnost	Hodnoty	Výchozí hodnota	Dědičná	Úroveň
writing-mode	lr-tb rl-tb tb-rl lr rl tb inherit	lr-tb	ano (viz specif.)	základní
xml:lang	<jazyk a/nebo země> inherit	viz konkrétní vlastnosti	ano	kompletní
z-index	auto <celé číslo> inherit	auto	ne	rozšířená

^aHodnoty column, even-page a odd-page převedeny, jako by bylo zadáno page.

^bHodnota start vzata jako left a hodnota end vzata jako right.

^cVzat pouze první z uvedených názvů fontů.

^dVlastnost height implementována jen pro elementy <fo:table-row> a <fo:external-graphic>.

^eODF neumožňuje nastavit znak rozdělovače.

^fVlastnost keep-together lze v ODF nastavit pouze pro odstavce a řádky tabulky.

^gVlastnost keep-with-next lze v ODF nastavit pouze pro odstavce a tabulky.

^hVlastnost keep-with-previous implementována převodem na vlastnost keep-with-next.

ⁱVlastnost width implementována jen pro elementy <fo:table>, <fo:table-cell> a <fo:external-graphic>.

Příloha C

PHP kód ukázkové aplikace

Následující příklad ukazuje část PHP skriptu, která se stará o samotné spuštění XSLT transformace nad zadaným FO souborem a o uložení výsledného ODF souboru. Méně podstatné části kódu jsou přitom vypuštěny a nahrazeny komentářem „// ...“. Pro načtení souboru s XSLT stylem stejně jako pro načtení zadaného FO souboru je použita třída `DOMDocument` dostupná v PHP od verze 5. Načtené dokumenty jsou pak předány třídě `XsltProcessor` (standardně obsažena v PHP 5), jejíž metoda `transformToDoc()` vrátí jako výsledek transformace opět instanci třídy `DOMDocument`.

Pro zachycení zpráv vzniklých během transformace je použita vlastní funkce, a to zavoláním funkce `set_error_handler()`. Posledním důležitějším detailem je zde použití vlastní třídy `SafeMode`, která zapouzdřuje některé funkce pro práci se souborovým systémem tak, aby se uvedený skript nemusel starat o to, zda je na serveru, kde běží, zapnuta PHP direktiva `safe_mode` (detaily viz např. manuál k PHP).

Příklad C.1. PHP kód pro XSLT transformaci dokumentu

```
// ...
do {
    // ...
    $outputFilename = $inputFilename . ($flatODF ? ".oft" : ".odt");
    // ...
    // Load the stylesheet as a DOM document
    $xsl = new DOMDocument();
    $stylesheetFilename = $flatODF ? "fo2odf-one_xml.xsl" : "fo2odf-archive.xsl";
    $orig = ini_set("track_errors", 1);
    if (!$xsl->load("$stylesheetsDir/$stylesheetFilename")) {
        $errors[] = "Failed to load the stylesheet file '$stylesheetFilename'. ►
$errorForAdminStr";
        break;
    }

    // Load the input XML file as a DOM document
    $inputdom = new DOMDocument();
    if (!$inputdom->load($fileUploaded ? $uploadedFilepath : ►
"$samplesDir/$sampleFilename")) {
        $errors[] = "Failed to load the source FO document '$inputFilename' - it ►
is probably not a valid XML file.";
        break;
    }
    ini_set("track_errors", $orig);
}
```

```
// Create the XSLT processor and import the stylesheet
$proc = new XsltProcessor();
if (!$proc->hasExsltSupport()) {
    $errors[] = "EXSLT support not available! $errorForAdminStr";
    break;
}
$proc->registerPHPFunctions();
$proc->importStylesheet($xsl);

// Prepare temporary directories for transformation
$transfDir = "$tempSessionDir/transformed";
if (!$flatODF) {
    SafeMode::mkdir($transfDir . "/META-INF", 0777, true);
    $proc->setParameter("", "output-dir", $transfDir . "/");
    $proc->setParameter("", "php-safe_mode", SafeMode::isActive() ? "yes" : "no");
}

// Set stylesheet parameters
$proc->setParameter("", "global-log-level", $loggingLevel);

// Perform the XSLT transformation
// ...
set_error_handler("myErrorHandler", E_WARNING);
$xsltError = ""; // modified by the error handler
// ...
$newdom = $proc->transformToDoc($inputdom);
// ...
restore_error_handler();
// ...
if ($newdom === false || $xsltError) {
    $errors[] = "XSLT transformation failed";
    break;
}

if ($flatODF) {
    // --- RETURN FLAT ODF DOCUMENT - 1 XML file
    $newdom->encoding = "utf-8"; // without this line entities would be used for ►
    $newdom->save(SafeMode::getFilepath($outputFilepath));
} else {
    // --- MAKE THE ODF ZIP ARCHIV
    $zip = new ZipArchive();
    if ($zip->open(SafeMode::getFilepath($outputFilepath), ZIPARCHIVE::OVERWRITE) ►
    !== true) {
        $errors[] = "Cannot open file for creating the ODF zip file. ►
$errorForAdminStr";
        break;
    }
    $odfFiles = array("mimetype", "content.xml", "meta.xml",
```

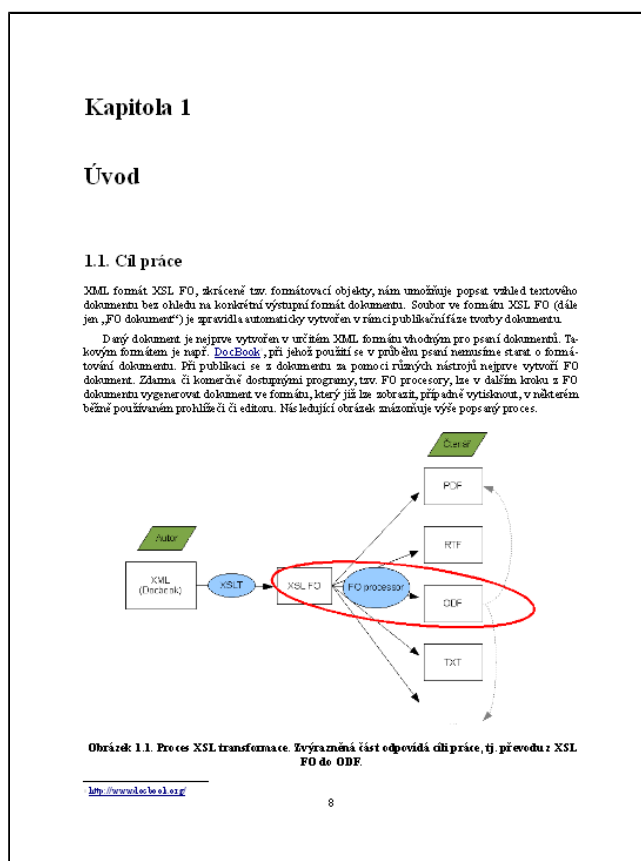
```
        "settings.xml", "styles.xml", "META-INF/manifest.xml",
    );
    foreach ($odfFiles as $odfFile) {
        $zip->addFile(SafeMode::getFilepath("$transfDir/$odfFile"), "$odfFile");
    }
    if ($zip->status != 0) {
        $errors[] = "Error appeared while adding files to the ODF zip file. ►
$errorForAdminStr";
    }
    $zip->close();
}
// ...
} while (false);
// ...
```

Příloha D

Ukázkové FO dokumenty

Tato příloha obsahuje snímky FO dokumentů nebo jejich částí po jejich převodu do ODF pomocí vytvořených XSLT stylů a otevření v editoru OpenOffice Writer. Pro srovnání jsou též u některých z nich uvedeny také snímky získané konverzí dokumentu do PDF procesorem XEP, který se dá považovat v podstatě za referenční implementaci formátovacích objektů, a jeho otevřením v prohlížeči Acrobat Reader.

Následující obrázky ukazují, jak vypadají různé části textu této práce po převodu do ODF. Text práce byl totiž napsán ve formátu DocBook, takže, jak již bylo zmiňováno, bylo možné převést ho do formátu XSL FO a odtud do PDF nebo dalších formátů.



Obrázek D.1. Stránka úvodní kapitoly s obrázkem a poznámkou pod čarou – ODF

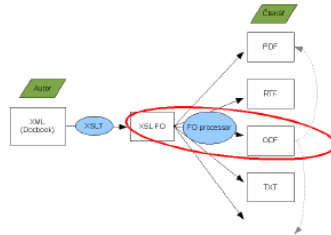
Kapitola 1

Úvod

1.1. Cíl práce

XML formát XSL FO¹, skráceně tzv. formátovací objekty, nám umožňuje popsat vzhled textového dokumentu bez ohledu na konkrétní výstupní formát dokumentu. Soubor ve formátu XSL FO (dále jen „FO dokument“) je zpravidla automaticky vytvořen v rámci publikačního tvorbu dokumentu.

Daný dokument je nejprve vytvořen v určitém XML formátu vhodným pro psaní dokumentů. Takovým formátem je např. DocBook², při jehož použití se v průběhu psaní nemusíme starat o formátování dokumentu. Při publikaci se z dokumentu za pomoci různých nástrojů nejprve vytvoří FO dokument. Zdarma či komerčně dostupnými programy, tzv. FO procesory, lze v dalším kroku z FO dokumentu vygenerovat dokument ve formátu, který již lze zobrazit, případně vytisknout, v některém běžně používaném prohlížeči či editoru. Následující obrázek znázorňuje výše popsaný proces.



Obrázek 1.1. Proces XSL transformace. Zvýrazněná část odpovídá cíli práce, tj. převodu z XSL FO do ODF.

Výstupní formát, který je cílem této práce, je formát OpenDocument (ODF). Ten kromě jiných aplikací jako svůj kľeňformát pro ukládání dokumentů používá open source kancelářský balík OpenO-

¹ <http://www.docbook.org/>

Obrázek D.2. Stránka úvodní kapitoly s obrázkem a poznámkou pod čarou – PDF

1.2. Struktura práce

Před samotným popisem možnosti konverze dokumentu z formátu XSL FO do formátu ODF se tato práce nejdříve věnuje oběma formátům. Popisuje jejich účel, historii a základní elementy, které lze použít pro popis struktury a vzhledu dokumentu. Toto umožňuje lépe si uvědomit rozdíly mezi těmito formáty, a připravit si tak půdu pro návrh všech dílčích transformací, které jsou nutné pro úspěšnou celkovou konverzi. Rozsah popisu obou formátů je přitom zpravidla záměrně omezen pouze na ty části, které jsou relevantní vzhledem k cíli této práce, tj. převodu z XSL FO do ODF.

Z dostupných možností implementace převodu bylo vybráno použití široce rozšířené a praxí ověřené technologie XSLT. Následující kapitola se proto věnuje popisu této implementace. Tzv. „XSLT stylé“ umožňují relativně jednoduše definovat, jak budou jednotlivé části vstupního XML souboru (zde XSL FO) převedeny na části výstupního souboru (zde ODF). XSLT stylé vytvořené v rámci této práce lze použít s XSLT procesory libxslt a Xalan-Java. První z nich je obsažen v jazyku PHP od verze 5 a druhý používá OpenOffice pro importování souborů pomocí XSLT. Výsledná ukázková aplikace, která je součástí této práce, tak umožňuje spustit konverzi z příkazové řádky, přes webové rozhraní napsané v PHP nebo přímo při otevírání FO dokumentu v OpenOffice. Popis této „integrace“ vytvořených XSLT stylů do různých prostředí je tak následuje za popisem vytvořených XSLT stylů.

1.3. Konvence použité v této práci

V textu této práce je použito několik na první pohled více či méně zřejmých konvencí. Jsou to zejména tyto:

- Názvy XML elementů jsou pro snadné rozpoznání psané <jako> otevírací tagy neproporcionálním písmem.
- První výskyt názvu XML elementu, který je součástí bližšího popisu daného elementu, je navíc zvýrazněn <tučným písmem>.
- Názvy XML atributů stejně jako jejich hodnoty jsou psány neproporcionálním písmem.
- Také různé ukádky kódu jsou psány neproporcionálním písmem.
- Každou vlastnost určitého formátovacího objektu lze v dříve věšně případy ztotožnit s některým atributem uvedeným na elementu představujícím daný formátovací objekt. Obdobně to lze říci také o formátu ODF. Pojmy „vlastnost“ a „atribut“ jsou tak v textu střídavě používány v závislosti na

² <http://www.xmlmind.com/docbook/>
<http://www.xmlmind.com/docbook/>
<http://xml.apache.org/doc/>

Obrázek D.3. Různý formát textu a seznam s odrážkami – ODF

1.2. Struktura práce

Před samotným popisem možnosti konverze dokumentu z formátu XSL FO do formátu ODF se tato práce nejprve věnuje oběma formátům. Popisuje jejich účel, historii a základní elementy, které lze použít pro popis struktury a vzhledu dokumentu. Toto umožňuje lépe si uvědomit rozdíly mezi těmito formáty, a připravit si tak půdu pro návrh všech dílčích transformací, které jsou nutné pro úspěšnou celkovou konverzi. Rozsah popisu obou formátů je přitom zpravidla záměrně omezen pouze na ty části, které jsou relevantní vzhledem k cíli této práce, tj. převodu z XSL FO do ODF.

Z dostupných možností implementace převodu bylo vybráno použití široce rozšířené a praxí ověřené technologie XSLT. Následující kapitola se proto věnuje popisu této implementace. Tzv. „XSLT stylů“ umožňují relativně jednoduše definovat, jak budou je jednotlivé části vstupního XML souboru (zde XSL FO) převedeny na části výstupního souboru (zde ODF). XSLT stylů vytvořené v rámci této práce lze použít s XSLT procesory libxslt a Xalan-Java. První z nich je obsažen v jazyku PHP od verze 5 a druhý používá OpenOffice pro importování souborů pomocí XSLT. Výsledná ukázková aplikace, která je součástí této práce, tak umožňuje spustit konverzi z příkazové řádky, přes webové rozhraní napsané v PHP nebo přímo při otevírání FO dokumentu v OpenOffice. Popis této „integrace“ vytvořených XSLT stylů do různých prostředí tak následuje za popisem vytvořených XSLT stylů.

1.3. Konvence použité v této práci

V textu této práce je použito několik na první pohled více či méně zjevných konvencí. Jsou to zejména tyto:

- Názvy XML elementů jsou pro snadné rozpoznání psané < jako otevírací tagy neproporcionálním písmem.
- První výskyt názvu XML elementu, který je součástí bližšího popisu daného elementu, je navíc zvýrazněn tačným písmem.
- Názvy XML atributů stejně jako jejich hodnoty jsou psané neproporcionálním písmem.
- Také různé ukázkový kód jsou psány neproporcionálním písmem.

⁵<http://www.xml.org/>

⁶<http://www.xmlind.com/ocawater/>

⁷<http://xmlgraphics.apache.org/fop/>

Obrázek D.4. Různý formát textu a seznam s odrážkami – PDF

Tabulka 4.1. Dynamické elementy - porovnání zápisu	
XSL FO	ODF
<code><fo:page-number /></code>	<code><text:span> <text:page-number /> </text:span></code>
<code><fo:page-number-citation ref-id="cil-odkazu" /></code> ... <code><fo:... id="cil- odkazu">...</fo:...></code>	<code><text:span> <text:bookmark-ref text:ref- name="cil-odkazu" text:reference- format="page"> 0 </text:bookmark-ref> </text:span></code> ... <code><text:p> <text:bookmark text:name="cil- odkazu"/> ... </text:p></code>
<code><fo:basic-link internal-destination="cil- odkazu"> text odkazu </fo:basic-link></code> ... <code><fo:... id="cil- odkazu">...</fo:...></code>	<code><text:a xlink:href="#cil-odkazu"> <text:span>text - odkazu</text:span> </text:a></code> ... <code><text:p> <text:bookmark text:name="cil- odkazu"/> ... </text:p></code>

Obrázek D.5. Tabulka se záhlavím a předformátovaným textem – ODF

Na závěr ještě ukázka implementace různých formátovacích vlastností textu (soubor font-attributes.fo v adresáři samples na přiloženém CD):

Font family: Palatino, Arial, Courier New.
 Font family (generic): serif sans-serif monospace.
 Font family (multiple fonts for fallback): "Arial, Courier New", "WrongName, Courier New".

Font size: 8 pts, 12 pts, 16 pts, 2 ems, 200%, **2 cms**.

Font weight: **bold**.
 Font style: *italic*, *oblique*, backslant (not in ODF).
 Font variant: SMALL CAPS.
 Text transform: Capitalize, UPPERCASE, lowercase, None.
 Text decoration: underline, ~~strike-through~~.
 Text decoration2: parent underlined, child not decorated and this red child underlined too.
 Text decoration3: blinking (no-blink), ~~line-through~~ plus underline [without line through] (none) ~~(overline fallback)~~.
 Baseline shift: -1 pt, +2 pts, ^{superscript}, _{subscript}, 20% ^{superscript}, 20% _{subscript}.
 Foreground color: **red**, #808080.
 Fallbacks: **green** from **rgb-icc(0, 255, 0, 'name', 1, 2)**, system-color('name').
 Background color: background-color="#F0F0F0", **background="repeat-x red"**.
 Letter-spacing: normal, 1 0 p t, 0 . 5 e m, -2t
 Font shorthands: **font="900 italic small-caps 15pt Times"**, font="status-bar", font="caption".

Obrázek D.6. Různé formátovací vlastnosti textu – ODF

Font family: Palatino, Arial, Courier New.
 Font family (generic): serif sans-serif monospace.
 Font family (multiple fonts for fallback): "Arial, Courier New", "WrongName, Courier New".

Font size: 8 pts, 12 pts, 16 pts, 2 ems, 200%, **2 cms**.

Font weight: **bold**.
 Font style: *italic*, *oblique*, backslant (not in ODF).
 Font variant: small caps.
 Text transform: Capitalize, UPPERCASE, lowercase, None.
 Text decoration: underline, ~~strike-through~~.
 Text decoration2: parent underlined, child not decorated and this red child underlined too.
 Text decoration3: blinking (no-blink), ~~line-through~~ plus underline [without line through] (none) ~~(overline fallback)~~.
 Baseline shift: -1 pt, +2 pts, subscript, ^{superscript}, 20% subscript, 20% superscript.
 Foreground color: **red**, #808080.
 Fallbacks: **green** from **rgb-icc(0, 255, 0, 'name', 1, 2)**, system-color('name').
 Background color: background-color="#F0F0F0", **background="repeat-x red"**.
 Letter-spacing: normal, 1 0 p t, 0 . 5 e m, -2t
 Font shorthands: **font="900 italic small-caps 15pt Times"**, font="status-bar", font="caption".

Obrázek D.7. Různé formátovací vlastnosti textu – PDF