XML na papír

Bakalářská práce

Dušan Fencl

Vedoucí bak. práce: Ing. Válav Novák, Csc.

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

2010

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval/a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Libniči, dne 5. ledna 2010

.....
Dušan Fencl

Anotace

Práce se zabývá XSL-FO technologií a snaží se přiblížit tuto technologii běžným uživatelům. Obsah práce je zaměřen na tvorbu tištěných výstupů ve firmách pomocí různých formátovacích nástrojů. V práci lze nalézt mnoho příkladů, které názorně ukazují možnosti vytvoření tištěného výstupu pomocí formátovacích objektů. Zabývá se problematikou vytvoření elektronické faktury z XML do PDF a porovnává kvalitu a správné zobrazení výstupů vytvořené pomocí FO procesorů.

Klíčová slova

XML, XSLT, XSL-FO, formátovací objekty, XSLT procesory, XSL-FO procesory, šablona faktury

Abstract

This thesis is focused on XSL-FO technology and tries to introduce this technology to its users. The content of this thesis is focusing in creation of printed outputs in companies using various formatting equipments. In this thesis, you may find lots of different examples, which clearly demonstrate possibilities of creating printed outputs with assistance of formatting objects. It's also focused on difficulties of electronic invoice creation from XML to PDF and compares the quality and its correct appearance of outputs formed by FO processors.

Keywords

XML, XSLT, XSL-FO, formatting objects, XSLT processors, XSL-FO processors, template invoice

Poděkování

Rád bych poděkoval své rodině a přítelkyni za morální podporu při psaní práce, dále panu Ing. Pavlu Tylovi za poskytnutí materiálů pro tvorbu této práce a panu Ing. Jirkovi Koskovi za rady při počeštění XSL-FO procesorů.

Obsah

Úvod ... **10**

Cíle práce ... **12**

Metodika ... **13**

1. XSLT (Extensible Stylesheet Language Transformations) ... **14**

1.1 Deklarace XSLT ... **14**

1.2 XSLT šablona ... **15**

1.3 Nejdůležitější XSLT elementy ... **15**

2. XSLT procesory ... **17**

2.1 Popis práce XSLT procesoru ... **18**

2.2 Instalace a popis práce s XSLT procesorem ... **18**

2.3 Nejčastěji používané XSLT procesory ... **19**

2.3.1 Saxon ... **19**

2.3.2 Xalan ... **21**

2.3.3 libxslt/xsltproc ... **22**

2.3.4 XT ... **23**

2.4 Další XSLT procesory ... **23**

2.5 Shrnutí XSLT procesory ... **24**

3. XSL-FO(XSL- Formatting Objects) ... **26**

3.1 Budoucnost XSL-FO ... **26**

3.2 Vytvoření pozvánky za pomoci XSL-FO příručky ... **28**

4. XSL-FO procesory ... **29**

4.1 XSL Formatter ... **30**

4.2 PassiveTeX ... **32**

4.3 XEP ... **32**

4.4 XFC ... **34**

4.5 Formátovací procesor FOP ... **35**

4.5.1 Instalace FOPu ... **36**

4.5.2 Počeštění Fopu **38**
4.5.3 Spouštění FOPu **39**
4.5.4 Přidání dalších truetype fontů do FOPu **40**
4.5.5 Instalace a nastavení programu Barcode4J **41**
4.6 Shrnutí XSL-FO procesory **43**
5. Tvorba faktury pomocí XSL-FO **44**
5.1 Faktura a její náležitosti **44**
5.2 Vizuální návrh faktury **45**
5.3 Předloha elektronické faktury **46**
5.3.1 Popis rozvržení e-faktury **46**
5.3.2 Výběr formátovacích objektů pro návrh e-faktury **48**
5.3.3 Řešení problémových částí návrhu **50**
5.3.4 Podtržení menu a poslední položky **52**
5.3.5 Čárový kód **53**
5.3.6 Digitální podpis **53**
5.4 Tvorba šablony faktury **54**
5.4.1 Vytvoření layoutu stránky **55**
5.4.2 Vytvoření hlavní šablony **56**
5.4.3 Vytvoření šablony pro položky **56**
5.5 Převod XML faktury do PDF **58**
5.5.1 Převedení ve FOPu **58**
5.5.2 Převedení v XEPu **58**
5.5.3 Převedení v AHFormatteru **59**
5.6 Porovnání výstupu FO procesorů **59**
5.7 Závěrečné hodnocení **60**
6. Převod FO do ODF nebo OOXML **62**
6.1 Převod .fo souboru do ODF **62**
6.2 Převod .fo souboru do OOXML **63**
Závěr **65**
Příloha A **66**

XSL-FO příručka ... **66**
1. Struktura XSL-FO dokumentu ... **67**
2. Tvorba XSL-FO oblastí ... **70**
3. Tvorba XSL-FO toků ... **72**
4. Tvorba obsahu a formátování na úrovni bloku ... **73**
5. Tvorba tabulek ... **76**
6. Tvorba XSL-FO seznamu ... **78**
7. Vkládání obrázků a SVG ... **80**
8. Obtékání textu a obrázku ... **82**
9. Absolutní pozicování pomocí kontejnerů ... **83**
10. Tvorba odkazů, obsahu a číslování stránek ... **85**
11. Poznámky pod čarou, text v záhlaví a zápatí ... **89**
12. Tvorba stromu dokumentu ... **92**
Příloha B ... **94**
Obsah CD ... **94**
Seznam zkratek ... **95**
Seznam použité literatury ... **97**
Internetové zdroje ... **99**

Úvod

V dnešní době se s pojmem XML setkáváme velmi často. Velkou popularitu si zasloužilo především díky své univerzálnosti. Obsahuje pouze text, který je nezávislý na platformě či aplikaci. To je sice hezké, ale pro prezentaci obsahu je nevhodné. Za tímto účelem byla vyvinuta skupina jazyků XSL. Nejznámějším jazykem této skupiny je bezesporu XSLT, které slouží pro prezentaci obsahu XML dokumentu na internetu v podobě HTML či XHTML. Velká popularita těchto jazyků je nesporná, neboť existuje celá řada publikací či odborných příruček.

Co když ale potřebujeme přenositelnou formu prezentovaného obsahu? Pro tyto účely byl vyvinut jazyk XSL-FO, který pracuje s formátovacími objekty určujícími vzhled výstupu. Pomocí formátovacích objektů je možné převádět XML dokument do jiných formátů. XSL-FO není bohužel příliš jednoduchou záležitostí a tak není divu, že odborných publikací je velmi poskrovnu a většina z nich je napsána v anglickém jazyce. Bohužel XSL-FO částečně zastiňuje i jiná forma možnosti převedení XML dokumentu do tištěné podoby. Touto možností je DocBook, který byl původně vyvinut za účelem tvorby dokumentace k softwaru a hardwaru. Součástí Docbooku jsou i formátovací objekty, ale také řada jiných technologií. Pokud potřebujeme stoprocentně typograficky kvalitní výstup, je úprava Docbooku nezbytná a vyžaduje poměrně značné vědomosti v oblastech XML, XSD, DTD, SGML, CSS, DSSL, XSL-FO či XSLT.

Malé množství informací a časté dotazy na internetu na téma XSL-FO mě vedly k napsání bakalářské práce, která se zabývá touto problematikou. Práce je určena čtenářům, kteří již mají alespoň základní znalosti z oblasti XML, XSLT a XPath. Součástí bakalářské práce je příručka XSL-FO, pomocí které

by měl každý zvládnout základní operace a správné používání formátovacích objektů. Práce obsahuje poměrně velké množství nejčastěji používaných formátovacích nástrojů, popisuje jejich možnosti využití, nastavení a porovnává jejich výstupy. XSL-FO se nejvíce využívá pro tvorbu tištěných výstupů ve větších firmách. Mezi nejčastější tištěné výstupy ve firmách patří faktury, prezentace vlastních produktů či služeb, manuály, zprávy týkající se firmy. Jednoduché prezentace výrobků či manuálů lze nalézt na internetu. Rozhodl jsem se vytvořit pomocí XSL-FO šablonu, která vytvoří z XML faktury fakturu elektronickou. Na výrobě faktury se dají dobře ukázat přednosti formátovacích objektů. Dalším důvodem, proč jsem si vybral právě fakturu, byly časté dotazy s otázkou: „Jakým způsobem lze vytvořit elektronickou fakturu z XML faktury?“ Za tímto účelem je práce prezentována také na internetu na stránkách **http://xsl-fo.aspone.cz.**

Cíle práce

Prostudoval jsem poměrně značné množství prací či článků snažících se zachytit oblast XSL-FO, ale ve většině z nich jsem se setkal s informacemi, které byly příliš povrchní. Hlavním cílem je vytvořit práci, která čtenáři poskytne dostatečné informace k tomu, aby mohl bezproblémově vytvořit pomocí formátovacích nástrojů .fo soubory a převést je na výstup např. na obrazovku, papír či jiné medium. Řadu zájemců při poznávání oblasti XSL-FO odradí netriviální nastavení nebo nevhodné vybrání XSLT či XSLFO nástrojů. Proto dalším cílem práce bude zmapování a vybrání nejvhodnějších XSLT a XSL-FO nástrojů. Pokusím se vyřešit problematiku s nastavováním správných cest, které jsou potřebné pro chod nástroje, se zobrazováním českých znaků a přidáním možnosti dělení slov.

Dalším cílem práce je vytvořit příručku, která bude obsahovat nejčastěji používané formátovací objekty a jejich atributy. U každé kapitolky uvedu názorný příklad použití formátovacích objektů. Vytvořím xsl-fo šablonu pro možnost vytváření pozvánek, která poslouží jako vhodná ukázka použití formátovacích objektů pro práci s textem. V celé jedné kapitole bych rád věnoval pozornost možnosti vytvoření xsl-fo šablony k vytvoření elektronické faktury a následnému převodu do PDF. Pomocí vybraných nástrojů porovnám výsledky výstupů u těchto příkladů.

Metodika

Informace potřebné pro zpracování této bakalářské práce byly převážně získány z literatury a internetových zdrojů (viz Seznam použité literatury). Některé materiály a podklady k této práci mi věnoval pan Ing. Pavel Tyl.

Při rozhodování jak práci koncipovat mi v mnohém pomohlo vlastní začátečnické zkoumání XSL-FO. Několik užitečných rad ohledně počeštění XSL-FO procesorů mi poskytl pan Ing. Jirka Koska. Práce byla koncipována tak, jak jsem vlastními krůčky poznával svět formátovacích objektů. Snažil jsem se zmapovat současný stav poznání XSLT procesorů. Vybral jsem nejznámější a nejpoužívanější procesory. U každého procesoru jsem vytvořil dávkovací soubor pro jednoduchou obsluhu. Každý procesor jsem otestoval pomocí rozsáhlé šablony, kterou jsem použil na převod své bakalářské práce.

1. XSLT (Extensible Stylesheet Language Transformations)

XSLT je jazyk, který se používá pro transformaci XML do jiného XML dokumentu nebo transformaci XML do formátů HTML či XHTML. Je součástí širší specifikace jazyka XSL. XSL(Extensible Style Language) nezahrnuje pouze transformační jazyk, ale také jazyk XSL-FO, který je mnohem složitější než jazyk XSLT a zabývá se formátováním výstupu. XSLT původně vznikl proto, aby usnadnil práci s formátovacími objekty. Transformační a formátovací části jazyka XSL mohou fungovat nezávisle jedna na druhé. XSLT 1.0 bylo přijato za standart konsorciem W3C 16. listopadu 1999. V současné době již existuje XSLT 2.0. XSLT je úzce spjato s jazykem XPath, který slouží pro hledání informací v XML dokumentu. Transformace XML dokumentu probíhá tak, že k XML dokumentu je připojena XSLT šablona a za pomoci internetového prohlížeče či XSLT procesoru dojde k transformaci.

1.1 Deklarace XSLT

Každý XSLT styl začíná deklarací XML. Kořenem stylu XSLT je element <xsl:stylesheet>, ve kterém je uvedena deklarace jmenného prostoru. Pro tento prostor je deklarován prefix xsl, který je použit u všech XSLT elementů a jeho hodnota je nastavena na http*://www.w3.org/1999/XSL/Transform*.

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

1.2 XSLT šablona

Šablony umožňují určit, jak celou transformaci realizovat. Každý element <*xsl:template*> odpovídá jednomu určitému uzlu, na který je aplikován. Šablona se připojuje pomocí atributu *match*.

Příklad 1.1: Ukázka XSLT šablony

```
<?xml version="1.0">

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

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

<xsl:template match="faktura ">
<P><xsl:apply-templates/></P>
</xsl:template>
</xsl:stylesheet>
```

Aby mohla být šablona aplikována na XML dokument, je třeba ji svázat s tímto dokumentem. Do XML dokumentu je třeba vložit pod XML deklaraci:

```
<?xml-stylesheet type="text/xsl" href="nazevsablony.xsl">
```

1.3 Nejdůležitější XSLT elementy

Mezi nejdůležitější XSLT elementy patří element <*xsl:template*>. Jak už jsme si řekli, tento element se používá k tvorbě šablon a pomocí atributu match je připojen k příslušnému XML elementu. Pokud je hodnota atributu *match="/"* je šablona aplikována na kořen XML dokumentu. Element <xsl:apply-templates> aplikuje šablonu na uzel, ke kterému je připojena nebo na synovské uzly aktuálního elementu. Na předchozím příkladu se

do odstavce aplikují všechny synovské elementy elementu <*faktura*>. Pokud použijeme atribut *select*, můžeme vybrat konkrétní uzel.

Dalším velmi důležitým elementem je <*xsl:value-of* >, který se používá k vybrání hodnoty elementu v XML dokumentu a přidání do výstupu transformace. Opět se nejčastěji používá ve spojení atributu *select*. Pokud potřebujeme zpracovat více elementů najednou, použijeme element <*xsl:for-each*>. Ke třídění výstupu se používá element <*xsl:sort*>, který se vkládá dovnitř elementu <*xsl:for-each*>. Na příkladu 1.2 je vidět zpracování všech položek ve faktuře a položky se setřídí podle kódu.

Příklad 1.2: Ukázka použití elemetů <xsl:for-each>,<xsl:value-of > a <xsl:sort>

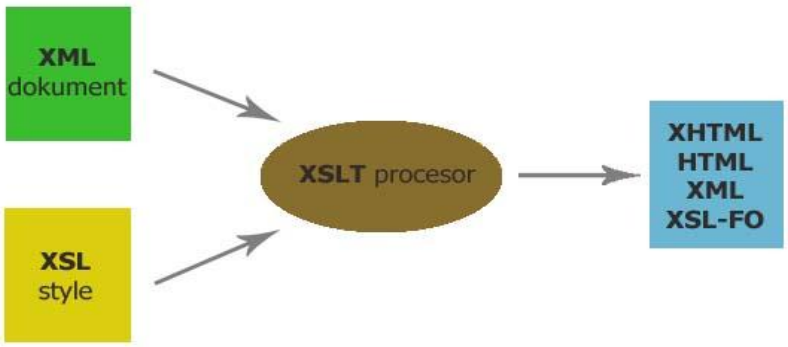
```
<xsl:template match="faktura ">
<xsl:for-each select="polozka ">
<xsl:sort select="kod"/>
<xsl:value-of select="."/>
</xsl:for-each>
</xsl:template>
```

Dalšími zajímavými elementy jsou <*xsl:copy*>, který kopíruje zdrojový uzel na výstup, element <*xsl:if*>, který testuje hodnotu a na základě zjištěných výsledků vykonává různé akce a element <xsl:choose>, který se používá vždy ve spojení s <*xsl:when*> a <*xsl:otherwise*> pro otestování vícenásobné podmínky.

```
<xsl:choose>
 <xsl:when test="podmínka">
  <!--výstup-->
 </xsl:when>
 <xsl:otherwise>
  <!--druhý výstup-->
 </xsl:otherwise>
</xsl:choose>
```

2 XSLT procesory

Procesor XSLT je program, který se používá k transformaci XML dokumentů pomocí jazyka XSLT. XSL dokument (XSL stylesheet) je procesorem aplikován na vstupní XML dokument a výsledkem je výstupní dokument, který může být ve speciálním světě formátovaných objektů XSL:FO (Formatting Objects nebo Flow Objects), nebo také dokument ve formátu HTML,XHTML, případně dokument využívající jiný XML prostor (XML namespace). Dnes již existuje celá řada XSLT procesorů, které umožňují zpracování XML dokumentů na základě XSLT stylu, ale jen několik z nich plně podporuje XSLT 1.0 standard, ba dokonce XSLT 2.0. Procesor může být přímo součástí XML editoru nebo ho můžeme použít jako samostatný program. Může také běžet přímo v internetovém prohlížeči nebo na serveru, kde dojde ke konverzi na HTML. V této kapitole se budeme zabývat těmi nejčastěji používanými samostatnými XSLT procesory a vybereme si jeden, který budeme používat pro transformaci XML dokument na XSL: FO dokument.



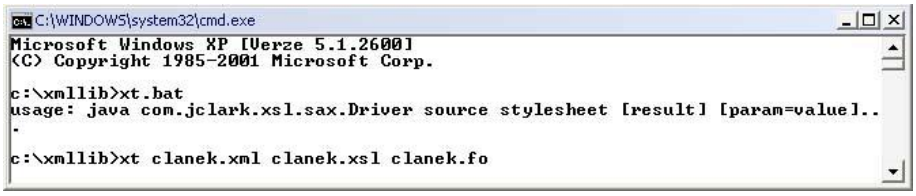
Obrázek 2.1: Transformace pomocí XSLT procesoru

2.1 Popis práce XSLT procesoru

Zpracování probíhá tak, že procesor prochází celý strom všech uzlů vstupního dokumentu a pro každý uzel hledá šablonu (pravidlo) ke zpracování tohoto uzlu. Šablonu vyjádříme elementem XSLT *xsl:template*. Jestliže procesor žádnou šablonu nenalezne, použije se interní šablona procesoru, která obsahuje zkopírování uzlu do výstupního dokumentu a spustí zpracování podřízených uzlů. Takto vlastně dochází rekurzivně k hierarchickému zpracování. Pokud ale definujete vlastní šablony, v podstatě převezmete nad danými uzly řízení a můžete je zpracovat, jak se vám zlíbí. V případě, kdy nějakému uzlu odpovídá více šablon, je vybrána šablona s nejvyšší prioritou. Ta je ovlivněna buď vaším vlastním nastavením, k čemuž slouží atribut *priority* instrukce *xsl:template*, nebo je vypočítaná vlastní logikou procesoru - ten šablonám přiřazuje hodnoty od -0,5 do +0,5. Přitom jsou upřednostňovány šablony s přesnější specifikací.

2.2 Instalace a popis práce s XSLT procesorem

Procesory XSLT se obvykle spouštějí z příkazové řádky. Procesor očekává prostřednictvím argumentu zdrojový XML dokument, dokument obsahující definici stylu XSLT a název cílového dokumentu, který se vytvoří po transformaci. Na obrázku 2.2 je ukázka transformace XML dokumentu do FO dokumentu pomocí procesoru XT od Jamese Clarka.



Obrázek 2.2: Transformace dokumentu pomocí XT

XSLT procesory bývají většinou napsány v jazyce Java nebo v jazyce C++. Pro jejich chod musíte mít nainstalovanou Javu od společnosti Sun Microsystem, kterou si můžete stáhnout na stránkách Java Sun. Pro použití procesoru v jazyce C++ musíte mít nainstalovaný .NET Framework. Ten si můžete stáhnout na stránkách Microsoftu. Některé procesory vyžadují pro chod ještě instalaci virtuálního stroje jazyka Java (VM). Download najdete na stránkách Java Virtual Machine.

Jak už jsem se zmiňoval výše, XSLT procesorů je velmi mnoho, ale ne všechny plně podporují XSLT 1.0, ba dokonce XSLT 2.0. Nejvíce používanými XSLT procesory jsou Saxon, Xalan a xsltproc.

2.3 Nejčastěji používané XSLT procesory

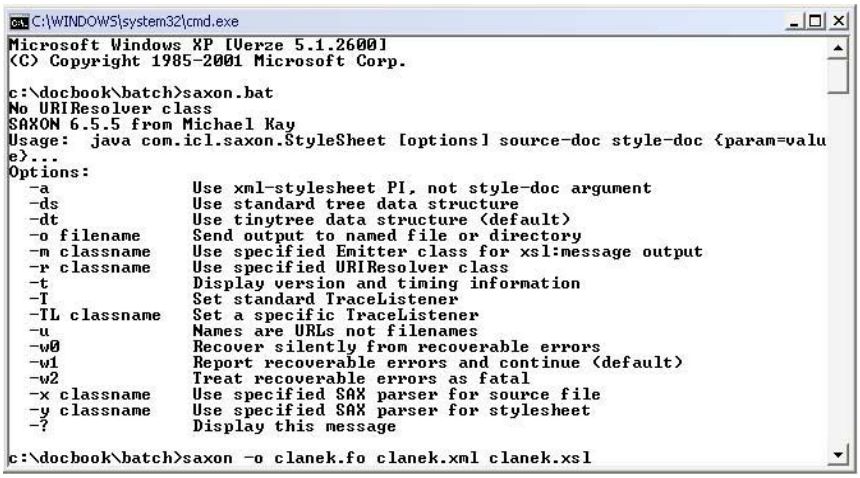
2.3.1 Saxon

Saxon je rychlý XSLT a XQuery procesor. Autorem open source Saxonu napsaném v Javě je Michael Kay. Dnes na vývoji Saxonu pracuje společnost Saxonica, která byla založena právě Michaelem Kayem. Saxon je dostupný v open source i komerční verzi. Open source verze neobsahuje některé nástroje, ale i tak plně dostačuje pro normální práci s XML dokumenty. Obě verze Saxonu 9.1 plně podporují XSLT 2.0, XPath 2.0, XQuery 1.0. Samozřejmě nechybí ani plná podpora XSLT 1.0, XPath 1.0, navíc komerční verze nabízí podporu XML Schema. Open source i komerční verzi je možné si stáhnout na stránkách Saxonu pro platformu Java, ale i pro platformu .NET Framework.

Procesor Saxon je navíc doporučován pro práci s DocBook stylesheety. Saxon je také dostupný včetně zdrojového kódu. Dalším užitečným rozšířením je možnost vytvoření více výstupních dokumentů, množinové operace s uzly, víceprůchodové zpracování dokumentů, možnost definice vlastních funkcí.

Dále umožňuje kombinovat javový kód s XSLT,--- měnit obsah proměnných, podporuje na výstupu i kódování iso-8859-2 a windows-1250. Na vstupu lze použít libovolný parser, dokáže zpracovat až 200MB XML dat. Procesor Saxon se neustále vyvíjí a jsou vydávány nové verze.

Na CD v sekci XSLT procesory jsem připravil několik verzí Saxonu. Verzi Saxonu 9.1.0.6 v jazyce Javy i C++. Saxon 6.5.5.0 s dávkovacím souborem a podporou XML katalogů a XInclude a instant Saxon 6.5.3.0. Název "Instant" byl vybrán ze dvou důvodů: snadné stahování a instalace, provedení a rychlost. Tato verze je ořezaná verze plného balíčku Saxonu. Neobsahuje zdrojový kód, API dokumentaci a ukázkové aplikace. Saxon Instant je spustitelná verze na platformě Win32. Plně podporuje XSLT 1.0 a XPath 1.0. Pro spuštění verze je potřeba mít nainstalován virtuální stroj Javy (VM). Na obrázku 2.3 je ukázka transformace XML dokumentu do FO dokumentu pomocí procesoru Saxon.

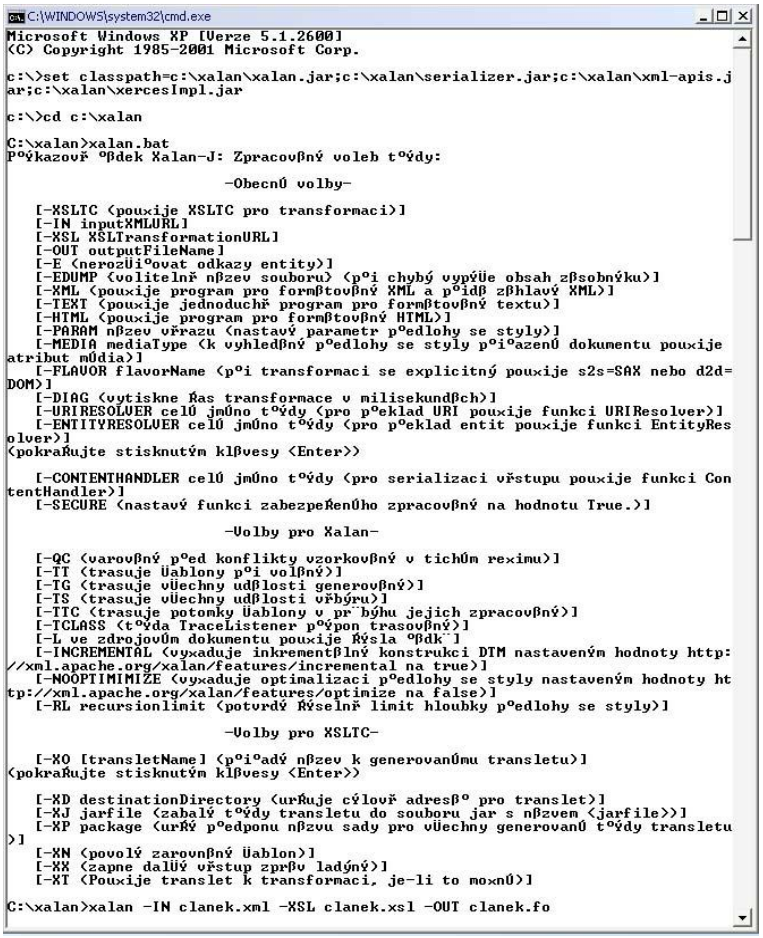


Obrázek 2.3: Transformace dokumentu pomocí Saxonu

2.3.2 Xalan

Dříve se jmenoval LotusXSL a byl produktem firmy IBM, dnes je ale k dispozici jako součást open source projektu XML Apache. Stejně jako Saxon i Xalan je dostupný v jazyce Java a C++. Java verze je poměrně pomalá. Plně podporuje XSLT 1.0 a XPath 1.0. Xalan je dostupný včetně zdrojového kódu. Open source verzi s parserem Xerces si můžete stáhnout na stránkách Xalanu. U Xalanu je možnost vytvoření více výstupních dokumentů, možnost definice vlastních funkcí v libovolném jazyce, který podporuje rozhraní BSF, JavaScript, VBScript, ReXX, Python, Perl, atd., na vstupu lze použít libovolný parser, navíc verze C++ na vstupu podporuje mnoho kódování včetně těch českých, po překompilování lze využít různá kódování i na výstupu. Poslední verze C++ vyšla v roce 2005, verze v Javě v roce 2007. Na CD v sekci XSLT procesory se nachází Java i C++ verze Xalanu. Samozřejmě nechybí ani dávkovací soubor pro Java verzi. K C++ verzi je přiložen parser Xerces a pro správný chod Xalanu jsem přidal do složky bin knihovnu *xerces-c_2_7.dll*.

Pro spuštění Xalanu je nutné nastavit prostředí proměnné. V proměnné CLASSPATH je třeba nastavit cestu k souborům *xalan.jar, serializer.jar, xercesImpl.jar, xml-apis.jar*. Pokud byste si chtěli vyzkoušet příklady, které jsou součástí instalační verze Xalanu, musíte nastavit ještě cestu k souboru xalansample.jar, který se nachází ve složce samples. Nastavení a transformace XML dokumentu by mohla vypadat například takto:



Obrázek 2.4: Transformace dokumentu pomocí Xalanu

2.3.3 libxslt/xsltproc

Libxslt je XSLT C knihovna vyvinutá pro GNOME projekt. Její součástí je xsltproc, který napsal Daniel Veillard v jazyce C++. Xsltproc je považován za nejrychlejší XSLT procesor. Plně podporuje XSLT 1.0 a XPath 1.0. Xsltproc je dostupný včetně zdrojového kódu a je open source. Podporuje většinu z EXSLT, umožňuje definice vlastních rozšíření v C, s knihovnou *iconv* podporuje velké množství kódování, obsahuje podporu XML Catalogs.

Aby mohl xsltproc vůbec fungovat je třeba si stáhnout několik souborů, kde jsou uloženy potřebné knihovny pro běh procesoru. Ty lze stáhnout na stránkách Zlatkovic.com. Je tedy třeba mít v jednom souboru tyto knihovny: *libxml2.dll, libexslt.dll, xsltproc.exe, iconv.dll, zlib1.dll, libxslt.dll, xmllint.exe*. Xsltproc je taktéž připraven na CD v sekci XSLT procesory. Jak je patrné z obrázku 2.5, transformace je velmi jednoduchá.

Obrázek 2.5: Transformace dokumentu pomocí xsltproc

2.3.4 XT

XT je velmi rychlý procesor od Jamese Clarka napsaný v Javě. Jedná se o první používaný procesor. Podporuje pouze částečně XSLT 1.0 a XPath 1.0. Stejně jako předchozí procesory i tento podporuje více výstupních dokumentů, množinové operace s uzly či možnost definice vlastních rozšíření. Na CD v sekci XSLT procesory jsem připravil upravenou verzi s podporou kódování windows-1250 a iso 8859-2 od pana Koska. Pro snadnější obsluhu jsem vytvořil dávkovací soubor. Součástí je také spustitelná verze na platformě Win32. K jejímu spuštění je třeba mít nainstalován virtuální stroj jazyka Java (VM). Nejnovější verzi si můžete stáhnout na stránkách BLNZ.com.

2.4 Další XSLT procesory

Dalšími poměrně známými XSLT procesory jsou MSXML od Microsoftu a Sablotron.

MSXML je COM komponenta napsaná v C++. Plně podporuje XSLT 1.0, XPath1.0 a také XML Schema 1.0. MSXML je velmi rychlý, je součástí

různých produktů společnosti Microsoft, například systémů Microsoft Windows, aplikací Microsoft Internet Explorer, sad Microsoft Office nebo serverů Microsoft SQL Server. Nevýhodou je uzavřený kód Microsoftu, úzce spjatý s jejich parserem.

Sablotron je rychlý, kompaktní a přenosný XSLT procesor napsaný v C++. Jedná se o open source projekt, který plně podporuje XSLT 1.0 a XPath 1.0. Sablotron klade velký důraz na přenositelnost, je možné ho spustit na systému Solaris, FreeBSD, OpenBSD, HP-UX, IRIX a všech Win32 systémech.

2.5 Shrnutí XSLT procesory

K vybrání vhodného XSLT procesoru nám poslouží následující tabulka 2.1.

Tabulka 2.1: Shrnutí XSLT procesorů

vlastnosti	Saxon	Xalan	xsltproc	XT	MSXML	Sablotron
platforma	Java, C++, EXE	Java, C++	C++	Java, EXE	C++	C++
rychlost	rychlá	pomalá, rychlá	extra rychlá	velmi rychlá	rychlá	rychlá
podpora	plně XSLT 2.0 XPath 2.0 XQuery 1.0	plně XSLT 1.0 XPath 1.0	plně XSLT 1.0 XPath 1.0	částečně XSLT 1.0 XPath 1.0	plně XSLT 1.0 XPath 1.0 XML Schema 1.0	plně XSLT 1.0 XPath 1.0
typ verze	open source, komerční	open source	open source	open source	součástí produktů Microsoft	open source
aktualizace	velmi často	naposledy 2007	často	naposledy 2005	často	občas

Při výběru vhodného XSLT procesoru hraje nejdůležitější roli podpora jazyků. Pokud potřebujete transformovat XML dokument a šablona bude obsahovat XSLT 2.0 elementy, vhodnou variantou je použít procesor Saxon.

V dnešní době hraje rychlost velmi důležitou roli. K urychlení transformace obrovského množství dat se používají k procesorům přídavné akcelerátory. Pro práci s malým množstvím dat, ztrácí rychlost na důležitosti.

Z tabulky je patrné, že nejvhodnějším kandidátem je Saxon, který v komerční verzi Enterprise podporuje také XML Schema 1.0 či možný update na XSLT 2.1, XQuery 1.1 a XSD 1.1. Nejjednoduší práce je s procesorem Saxon Instant, u kterého nemusíte nic nastavovat. Velmi dobře se mi pracovalo také s procesorem xsltproc. Nedoporučuji používat procesor XT, který je velmi zastaralý a nepodporuje plně XSLT 1.0.

3 XSL-FO(XSL- Formatting Objects)

XSL-FO je jazyk založený na XML a popisuje formátování XML dat pro výstup na obrazovku, papír či jiné medium. XSL-FO je druhou částí jazyka XSL a bylo přijato konsorciem W3C 15. října 2001. Nejnovější specifikace XSL-FO 1.1, přijatá 6. prosince 2006 konsorciem W3C obsahuje 81 formátovacích objektů, které dále rozšiřuje 281 atributů. Máme tedy nepřeberné množství možností k rozvržení a naformátování obsahu stránek. Bohužel jsme částečně omezováni formátovacími nástroji, které některé elementy či atributy nepodporují. K pochopení práce s formátovacími objekty je třeba mít základní znalosti XML, jmenných prostorů XML, XSLT a XPath. Výhodou je taktéž znalost CSS, neboť mnoho atributů formátovacích objektů je shodných s vlastnostmi CSS nebo se jim podobá. FO dokumenty můžeme vytvářet přímo v XML editoru a poté za pomoci XSL-FO procesoru převést do jiného výstupu (PDF, PS, RTF). Další možností je vytvořit XSLT šablonu, která obsahuje formátovací objekty, a za pomoci XSLT procesoru ji aplikovat na XML dokument.

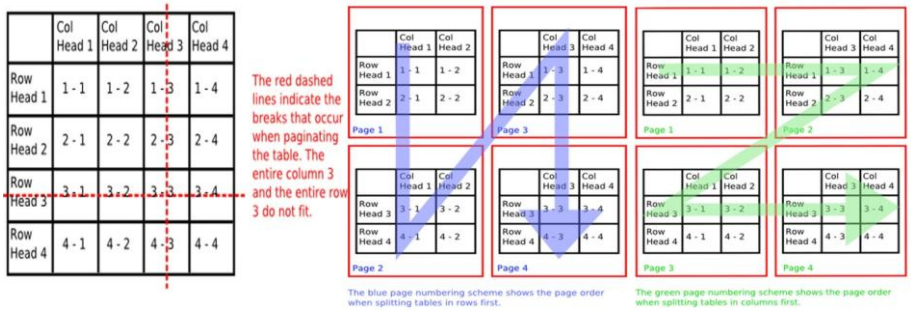
3.1 Budoucnost XSL-FO

Prvotní zmínka o přípravě na nové verzi XSL-FO pochází již z roku 2006. První veřejné zasedání se konalo 21. října 2006 v německém Heidelbergu. Na zasedání se řešila především problematika spojená se SVG grafikou a přání větší integrace MathML. Zasedání se zúčastnili také japonští inženýři z firmy Antenna House, kteří se snažili prosadit větší kompatibilitu mezi XSL-FO a CSS, zejména pak mezi XSL-FO1.1 a CSS3. Bohužel tento návrh byl nejenom zamítnut, ale také celá třetina zúčastněných byla pro úplné zrušení kompatibility.

První oficiální návrh byl vydán W3C dne 26. března 2008. V návrhu je několik zásadních novinek, které považuji za velký krok kupředu:

- **vytváření neobdélníkových oblastí s vnitřním obsahem textu,**
- **přidání textu do cesty (uchycení textu ke křivce),**
- **text proudící kolem ilustrací, regionů a dalších objektů,**
- **obtékání neobdélníkových oblastí textem a určování priorit mezi objekty,**
- **vylepšené možnosti copyfitingu, možnost vytváření marginálií a kapitál,**
- **svislé zarovnání v rámci stránky nebo sloupce.**

Další výrazné změny se týkají tabulek a seznamů. U tabulek bude možné vkládat záhlaví a zápatí. Pokud se tabulka nevejde na jednu stránku, tabulka je rozdělena do více stránek. Následující obrázek ukazuje možnost rozdělení tabulky:



Obrázek 3.1: Transformace dokumentu pomocí xsltproc

V návrhu nechybí ani opakování obsahu v buňce při rozdělování tabulky či mobilní přesahující hranice tabulky pro naznačení pokračování tabulky na další stránce. Další zajímavou novinkou je, pokud na stránce máme

např. 3 svislé sloupce textu a 1 vertikální sloupec textu, který překrývá svislé sloupce, automaticky se vytvoří více sloupců tak, aby se text nepřekrýval.

První návrh obsahuje více než 150 změn, což považuji za velmi výrazný pokrok v oblasti XSL-FO. Druhý a prozatím poslední návrh XSL-FO 2.0 vyšel 29. září 2009. V návrhu přibyly především první XSL definice pro formátovací objekty a jejich vlastnosti. PDF soubor s návrhem XSL-FO2.0 lze nalézt na přiloženém CD. Trochu mě mrzí, že při psaní této práce nemohu využít již XSL-FO 2.0. Pokud se podíváme na rozmezí vydání prvního a druhého návrhu, troufám si tvrdit, že bychom se velmi brzy dočkat XSL-FO 2.0 standardu.

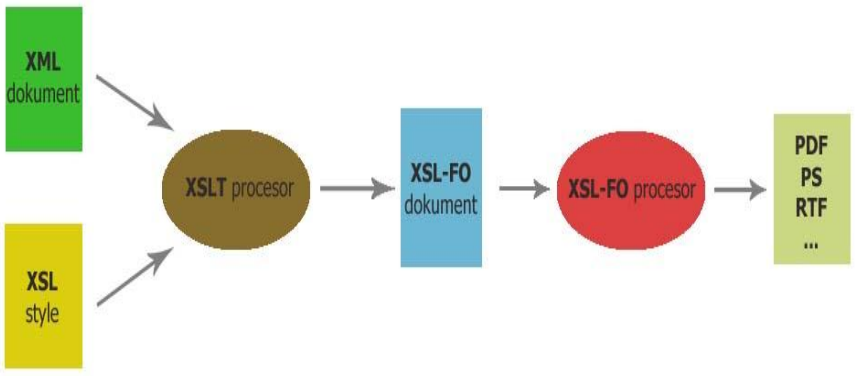
3.2 Vytvoření pozvánky za pomoci XSL-FO příručky

Jak jsem již v úvodu zmíňil - pro práci s formátovacími objekty jsem vytvořil XSL-FO příručku, ve které můžete najít nejčastěji používané formátovací objekty.

Jako vhodný příklad pro ukázku využití XSL-FO příručky v praxi, jsem si vybral vytvoření šablony pozvánky, pomocí které jsem vhodně rozvrhl text na stránce. Jelikož šablona obsahuje elementy, které jsou obsaženy v příručce, rozebírat zdrojový kód by bylo neefektivní. Pokud si chcete prohlédnout zdrojový kód nebo vytvořený .fo soubor *pozvanka*, naleznete ho na CD v sekci Pozvanka.

4 XSL-FO procesory

XSL-FO procesor je program pro transformaci XML dokumentu pomocí XSLT šablony, která obsahuje tzv. FO (Formatting Objects) značky. Vznikne mezivýstup XSL-FO dokument, který je transformován na tištěný výstup, jiný výstupní formát nebo je pouze zobrazen na obrazovce. Nejčastějšími výstupními formáty, které umí FO procesory vytvářet, jsou PDF (Portable Document Format), PS (Post Script), RTF (Rich Text Format). Další možností, jak z XML dokumentu vytvořit např. PDF, je použít samostatný XSLT procesor, pomocí kterého vytvoříme XSL-FO dokument. Tento dokument poté zpracujeme FO procesorem. Doporučil bych používat tuto variantu. XSLT procesory, které jsou součástí FO procesorů, mohou být buď zastaralé, nebo nemusí podporovat některé XSLT značky. Další nespornou výhodou je kontrola FO dokumentu pomocí XML editoru. Třetí možností je přímo napsat FO dokument a převést FO procesorem. Tuto variantu doporučuji používat pouze na krátké dokumenty nebo k vyzkoušení si práce s formátovacími objekty. Na obrázku 4.1 je znázorněn princip činnosti FO procesoru.



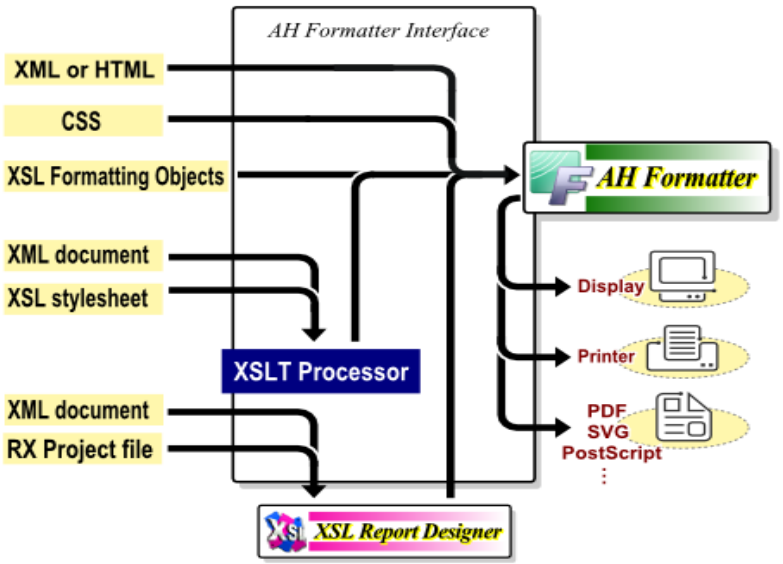
Obrázek 4.1: Vygenerování tištěného výstupu pomocí XSL-FO procesoru

V dřívější době bychom mohli spočítat množství XSL-FO procesorů na prstech jedné ruky. Hlavními příčinami bylo dokončení XSL-FO standardu o téměř dva roky později než XSLT standardu a také jeho větší a komplikovanější rozsah. Dnes sice existuje celá řada FO procesorů, ale většina z nich je komerčních (XSL Formatter, XEP, Xinc, XML2PDF, XPP a jiné). Kvalita výstupu a množství podporovaných FO značek se u řady FO procesorů liší. Obecně mohu říci, že komerční procesory mají kvalitnější výstup a podporují mnohem větší množství FO značek než open source procesory. Světlou výjimkou je procesor FOP, který může směle konkurovat některým komerčním produktům. Tento procesor budu využívat pro názorné ukázky při vytváření příkladů na formátovací objekty. Řekneme si, jak nainstalovat nejnovější verzi FOPu, také si zde ukážeme kompilaci vývojové verze FOP Trunk, kterou mi na transformaci bakalářské práce do PDF doporučil pan Ing. Pavel Tyl, a podíváme se také na některé nejznámější FO procesory. **Na CD v sekci XSL-FO procesory můžete nalézt FOP Trunk a FOP 0.95, zkušební verzi XEP 4.15, AHFormatter V5 Evolution a XMLmind 4.3.**

4.1 XSL Formatter

XSL Formatter je komerční software od společnosti Antenna House. Jedná se o vynikající, bohužel však méně známý, program pro práci s formátovacími objekty. Nejnovější verze Antenna House Formatter V5.0 podporuje rozvržení stránek buď pomocí CSS, nebo pomocí XSL-FO pro vytvoření PDF, PS, SVG či prohlédnutí výstupu na obrazovce. AH Formatter téměř plně podporuje všechny elementy XSL-FO 1.1 standardu. Nechybí ani podpora MathML. Součástí programu je rozhraní příkazové řádky a možnost ovládat ho jako COM objekt. GUI rozhraní působí velmi přívětivě a přehledně. Součástí programu je také XSLT procesor MSXML 4.0 či MSXML 3.0, samozřejmě

můžete použít svůj vlastní XSLT procesor (např. Saxon). Převod XML dokumentu je velmi rychlý. Převod 300 stránkového PDF z XML dokumentu mi trval 5 sekund. Kvalita výstupu do PDF je vynikající. Výstup může být v PDF ve verzích 1.3 až 1.7, PDF/X či PDF/A, podpora rastrové grafiky ve formátech JPEG, JPEG2000, PNG, TIFF, GIF, podpora vektorové grafiky EMF, WMF, SVG, MathML, Excel Graf. Součástí je také modul pro podporu Pantone barev, které lze vkládat do výstupu a převádět na CMYK či RGB vyjádření. K dispozici je i Barcode Option pro vkládání čárových kódů. Velkou výhodou je podpora více než 50 jazyků a dělení slov u 40 jazyků. Velikost vstupu podle výrobce je neomezená. XSL Formatter je možné spustit na operačních systémech Win32,Win64, Solaris 10, na Linux32,64bit verzích postavených na GCC 3.4. X, HP-UX či AIX. Obrázek 4.2 znázorňuje interface AH Formatteru.



Obrázek 4.2: Rozhraní Antenna House Formatteru

Je možné si stáhnout 90 denní Evolution verzi na stránkách Anenna House. Nejdříve musíte vyplnit formulář, kde se mimo jiné ptají, k jakým účelům budete tuto verzi používat. Po odeslání formuláře obdržíte mail s adresou,

kde si můžete stáhnout tuto verzi. Verze Evolution AH Formatter V5.0 je také uložena na CD v sekci XSL-FO procesory. Jedná se o ořezanou verzi Standart. Na každé stránce ve spodní části je umístěn vodoznak a URL Antenna House.

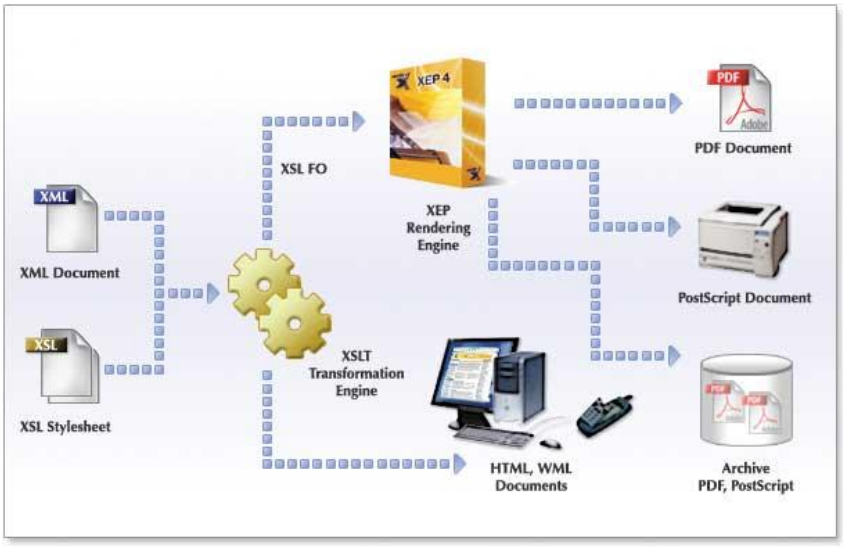
4.2 PassiveTeX

PassiveTeX od Sebastiana Rahtze je open source, který je implementován jako makro pro TeX, které rovnou načítá dokument FO a sází jej. Pro jeho činnost je potřeba moderní instalace TeXu. Pro načítání XML se používá XML parser napsaný v TeXu – xmltex. Každý formátovací objekt má v souboru obsaženou sekvenci texových příkazů, kterými se má nahradit. Kvůli tomu PassiveTeX dnes nepodporuje FO úplně a asi ani nikdy nebude. Formátovací modely TeXu a FO jsou odlišné a jednoduché prostředky xmltexu neumožňují tyto rozdíly zcela překonat. S pomocí varianty TeXu pdfTeX lze s pomocí PassiveTeXu generovat přímo kvalitní výstupy v PDF. V praxi dnes nejvíce vadí nedokonalé zpracování tabulek a téměř nulová podpora pro relativní délkové jednotky a výrazy uvnitř vlastností. PassiveTeX umožňuje do souboru s FO vkládat přímo matematické výrazy zapsané v MathML. Využívá se přitom toho, že xmltex standardně obsahuje podporu pro sazbu MathML. Do dokumentů XML tedy můžeme vkládat vzorce v MathML a v XSLT stylu je beze změny nakopírovat mezi formátovací objekty. PassiveTeX si můžete stáhnout na stránkách Tei-c.org.ik.

4.3 XEP

XEP je komerční procesor vyvíjený americkou firmou RenderX. Jedná se o FO procesor napsaný v jazyce Java, který na vstupu přijímá XML dokumenty a ty převádí s pomocí XSL stylů do podoby XSL formátovacích objektů, z kterých pak generuje výstup ve formátech PostScript, PDF, SVG či AFP.

Vstup může obsahovat rastrovou grafiku formátů PNG, JPEG, GIF, TIFF či vektorovou grafiku formátů PDF, EPS, SVG, XEPOUT. Stejně jako tomu bylo u XSL Formatteru i procesor XEP téměř plně podporuje XSL-FO 1.1 standard. Bohužel XEP nepodporuje některé české znaky a české dělení slov. Upravenou verzi češtiny a české dělení slov s popisem instalace najdete na stránkách Jirky Koska. Také přímo nepodporuje MathML, můžeme ale použít konvertor z MathML do SVG a zobrazit jí jako obrázek. XEP je možno spouštět z příkazové řádky nebo pomocí XEP assistant, což je GUI rozhraní pro snazší používání XEPu. V tomto rozhraní si můžete prohlížet XML dokumenty, provádět veškerou konfiguraci, která je velmi rozmanitá či nastavit formátování (připojení stylu, zvolení výstupu, zvolení programu pro otevření výstupu). Na obrázku 4.3 můžete vidět způsob využití XEPu.



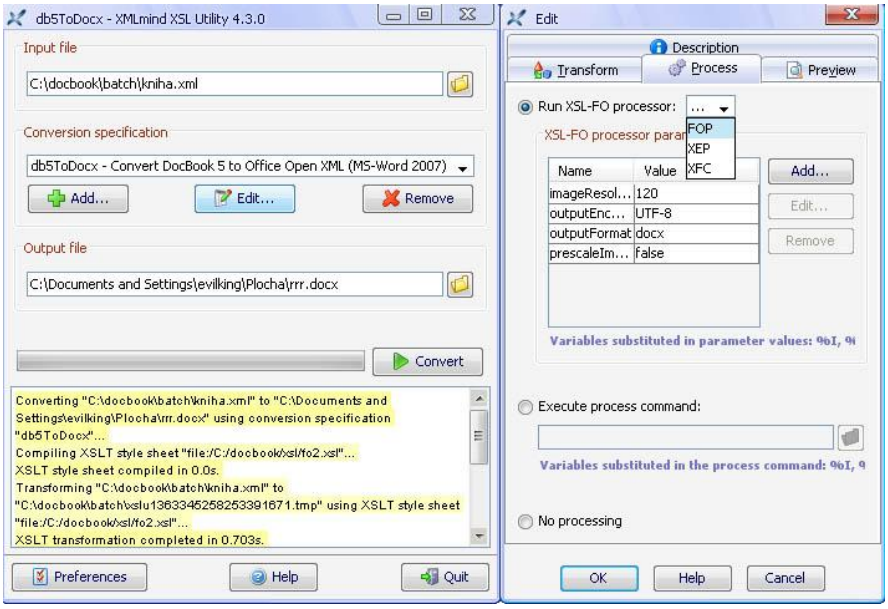
Obrázek 4.3: Využití XEPu

Na stránkách RenderX je možné vyplnit formulář a vybrat si nejenom typ trial verze, ale i mnoho užitečných doplňků. Po odeslání formuláře obdržíte do 24 hodin mail s odkazy na stáhnutí trial verze a doplňků. Zkušební verze vkládá na dolní okraj stránky reklamu na XEP. Plnohodnotně zobrazí prvních 11 stránek, každá další sudá stránka je vynechána. Pro vysoké školy je možné

získat Akademickou licenci plné verze XEPu Desktop, ale pouze se souhlasem RenderX.

4.4 XFC

XMLmind XSL-FO Converter je velmi rychlý XSL-FO procesor podobný procesorům Apache FOP, RenderX XEP nebo Antenna House XSL Formatter. Na rozdíl od výše zmíněných procesorů, který slouží především k převodu XSL-FO do PDF či PS, XMLmind XSL-FO Converter převádí XSL-FO do formátů: RTF (lze otevřít v aplikaci Word 2000 +), WordprocessingML (lze otevřít v aplikaci Word 2003 +), Office Open XML (.docx, lze otevřít v aplikaci Word 2007 +), OpenOffice (.odt, může být otevřen v OpenOffice.org 2 +). Po změně FO procesoru v nastavení, jak můžeme vidět na obrázku 4.4, lze XML dokument převést také na PDF. XFS sice nepodporuje plně XSL-FO standard, ale není problém si daný výstup poupravit podle sebe v Microsoft Office, či v Open Office. Vstupem může být buď přímo XSL-FO dokument, nebo XML dokument s XSLT šablonou. XFC je možné získat v Java i .Net verzi. Na vyzkoušení si můžete zdarma stáhnout Personal Edititon, která se liší od Profesional Edition pouze umístěním malého razítka Created by XMLmind XSL-FO Converter na spodním okraji listu. Toto razítko je ale téměř neviditelné a ve výstupním dokumentu nikterak neruší. Nejnovější verzi lze stáhnout na stránkách XMLmind.com.



Obrázek 4.4: Grafické rozhraní XMLmind

4.5 Formátovací procesor FOP

FOP (Formatting Objects Processor) je open source procesor společnosti Apache. Jedná se o aplikaci napsanou v jazyce Java. FOP je prvním XSL-FO procesorem vůbec. Jak už jsem výše zmiňoval, komerční FO procesory mají větší podporu FO značek než open source. FOP je tak trochu výjimkou, sice nemá tak dokonalou podporu jako XSL Formatter či XEP, ale pro převod docBookovských dokumentů, je to nejlepší, co lze dnes zdarma sehnat. Bohužel FOP nepodporuje češtinu, tu je třeba zvlášť přidat. Výstupním formátem FOPu může být PDF, PS, SVG, XML, AWT, MIF, částečně RTF a TXT. FOP také podporuje přímý výstup na tiskárnu. Primárním cílem FOPu je PDF. FOP podporuje grafické formáty BMP, GIF, JPEG, PNG, SVG, TIFF, WMF. Výstup do PS navíc ještě podporuje EPS.

Je možné si stáhnout jak binární verzi, která je již zkompilovaná, tak zdrojový kód a ten si zkompilovat pomocí Apache Ant. Nejnovější stabilní verzí je FOP 0.95, která podporuje velkou část z XSL-FO 1.1 standardu.

Pokud byste chtěli tuto verzi používat, musíte mít nainstalovánu Javu (alespoň verzi 1.4). Ve verzi FOP 0.95 došlo k opravě mnoha chyb se zobrazováním tabulek, k optimalizaci paměti a zvýšení výkonu při procházení FO stromem. Také je možné si stáhnout vývojovou verzi FOP Trunk. Při používání této verze je třeba sledovat stránku se seznamem známých chyb a omezení: xmlgraphics.apache.org. Nyní se pojďme podívat na instalaci, správné nastavení FOPu, instalaci češtiny a správné dělení slov, přidání dalších truetype fontů, instalaci a správné nastavení programu Barcode4J, který využijeme pro generování čárkového kódu při výrobě nejčastěji používaného tištěného výstupu ve firmách.

4.5.1 Instalace FOPu[1]

Jelikož je FOP Java aplikací, je potřeba mít nejdříve nainstalovánu Javu. Pro verzi 0.95, 0.95 beta či FOP Trunk je nutné mít Javu verzi 1.4 nebo vyšší. Poté je potřeba nastavit proměnnou JAVA_HOME v proměnném prostředí. Nejdříve si zkontrolujeme, zdali nemáme tuto proměnou již nastavenou. Spuštěním příkazu set v příkazové řádce se nám vypíší proměnné. Proměnná JAVA_HOME musí ukazovat do adresáře, ve kterém je nainstalována Java. Např. pokud máme Javu nainstalovanou v adresáři *C:\Program Files\Java\jdk1.6.0_13*, pak se do proměnné JAVA_HOME nastavuje cesta *c:\Program Files\Java\ jdk1.6.0_13*. Nejjednodušší způsob nastavení proměnné je doplnit do spouštěcí dávky řádek set JAVA_HOME=adresář Javy např. *JAVA_HOME= c:\Program Files\Java\ jdk1.6.0_13*. Druhou možností je nastavit JAVA_HOME jako proměnnou prostředí.

[1] Citováno z [10] viz Seznam použité literatury

Klikněte na: Start>Tento počítač > Vlastnosti systému > Upřesnit > Proměnné prostředí.

V okně Proměnné prostředí máme dvě možnosti vytvoření. Buď bude proměnná pouze pro jednoho uživatele, nebo bude systémová a uvidí ji všichni uživatelé. Doporučuji vytvořit v rámci systémových proměnných a po vytvoření provést restart systému. Pokud máme proměnnou nastavenou, stáhneme si jednu z verzí:

Tabulka 4.1: Aktuální verze FOPu

verze	Status	Adresa pro stažení
FOP 0.95	stabilní	http://public.picvi.com/apache/xmlgraphics/fop/
FOP 0.95b	nestabilní	http://archive.apache.org/dist/xmlgraphics/fop/binaries/
FOP Trunk	vývojová	http://svn.apache.org/repos/asf/xmlgraphics/fop/trunk/

Pro operační systém Windows doporučuji stáhnout ZIP archiv binární distribuce. Archiv rozbalíme kamkoli na disk. Pro pohodlnější práci s FOPem je v adresáři soubor fop.bat. Abychom ho mohli využívat odkudkoli z disku, musíme nastavit proměnnou Path v proměnném prostředí na aktuální umístění FOPu na disku (například C:\fop).

Abychom mohli sledovat vývoj Trunk verze, měli vždy nejnovější aktuální verzi FOPu a případně mohli pomoci s hledáním či odstraněním bugů, je vhodné si nainstalovat klienta pro verzovací systém. Oblíbeným klientem je například TortoiseSVN. Postup stažení FOP Trunk verze (pro Windows) programem TortoiseSVN:

1. Vytvoříme si prázdný adresář (například C:\Fop)

2. Klikneme na něj pravým tlačítkem a vybereme "SVN Checkout…" z kontextového menu.
3. Vložíme http://svn.apache.org/repos/asf/xmlgraphics/fop/trunk jako URL repositáře.
4. Klikneme OK a stahování začne.

Trunk verzi musíme poté zkompilovat. Použijeme k tomu (na radu v souboru build.xml) Apache ANT. Rozbalíme archiv na disk (u Windows 95 ME nejlépe do C:\). Pro plnou funkčnost ANTu musíme mít instalován JDK 1.4 či vyšší. Poté nastavíme následující proměnné prostředí:

set ANT_HOME=C:\ant
set JAVA_HOME=C:\j2sdk1.4.2_06
set PATH=%PATH%;%ANT_HOME%\bin

Poté již zkompilujeme Trunk verzi FOPu příkazem *ant* v adresáři s FOPem. Jsou-li při pozdější spouštění FOPu hlášeny neznáme chyby, přestože kompilace ANTem proběhla úspěšně, je dobré FOP Trunk zkompilovat znovu příkazem *ant clean*.

4.5.2 Počeštění FOPu[2]

Instalace pro FOP 0.93 a výše

Stáhneme si soubor fop-cs2.zip a rozbalíme jej do adresáře s instalací FOPu.

1. Při spouštění FOPu musíme určit cestu ke konfiguračnímu souboru s odkazy na metriky českých fontů. FOP je nutné spouštět

[2] Citováno z [11] viz Seznam použité literatury

s parametrem *c - c:\fop-0.93\conf\myfop.xconf* (adresář c:\fop-0.93 upravíme podle skutečného umístění FOPu).

2. V souboru *conf\myfop.xconf* upravíme cesty na začátku souboru tak, aby ukazovaly na adresář s instalací FOPu a na adresář, kde máme TTF fonty:

   - *<!ENTITY fop.home "file\\\c:\fop-0.93\">*
   - *<!ENTITY fonts.dir "file:\\\c:\windows\fonts">*

Pokud používáte Linux a nemáte české TTF fonty, můžete si je stáhnout z adresy http://corefonts.sourceforge.net. U verze Trunk je potřeba si stáhnout navíc OFFO vzory dělení slov. Vybereme vzory dělení pro verzi stable a z rozbaleného archivu překopírujeme soubor *fophyph.jar* do adresáře *C:\fop\lib*.

Zkoušel jsem tuto instalaci a funguje i na verzi 0.95 či FOP Trunk. Pokud máte problémy s některými znaky, je možné že používáte některé fonty, které nejsou zaregistrovány v *myfop.xconf*.

4.5.3 Spouštění FOPu[3]

FOP spouštíme z příkazové řádky. Napíšeme-li samotné slovo fop, získáme nápovědu s příklady syntaxí. Nejčastějšími variantami jsou tyto dvě:

fop -xml "cesta_k_souboru\vstup.xml"
-xsl "cesta_k_souboru\sablonaFO.xsl"
-pdf "cesta_k_souboru\vystup.pdf" -c C:\fop\conf\myfop.xconf

fop -fo "cesta_k_souboru\vstup.fo"
-pdf "cesta_k_souboru\výstup.pdf" -c C:\fop\conf\myfop.xconf

[3] Citováno z [10] viz Seznam použité literatury

4.5.4 Přidání dalších truetype fontů do FOPu[4]

Přidání dalších truetypových fontů do FOPu je jednoduché. Následující instrukci Javy je potřeba spustit pro každý soubor fontu v adresáři s FOPem. Příklad je pro font Bookman Old Style bookos.ttf :

Vytvoření metrického fontu z TTF fontu

```
java -cp "build\fop.jar;lib\avalon-framework-4.2.0.jar;
        lib\xml-apis-1.3.02.jar;lib\xercesImpl-2.7.1.jar;
        lib\xalan-2.7.0.jar;lib\commons-logging-1.0.4.jar;
        ib\commons-io-1.1.jar;lib\serializer-2.7.0.jar"
org.apache.fop.fonts.apps.TTFReader
C:\Windows\Fonts\bookos.ttf C:\fop\conf\Bookos.xml
```

Pak editujeme soubor myfop.xconf v podadresáři conf adresáře s FOPem. Tyto řádky přidáme mezi tagy <*fonts*> a </*fonts*>.

Připojení metrických fontů k FOPu

```
<font metrics-url="&fop.home;/conf/Bookos.xml" kerning="yes"
 embed-url="&fonts.dir;/Bookos.ttf">
 <font-triplet name="Bookos" style="normal" weight="normal"/>
</font>
<font metrics-url="&fop.home;/conf/Bookosb.xml" kerning="yes"
 embed-url="&fonts.dir;/Bookosb.ttf">
 <font-triplet name="Bookos" style="normal" weight="bold"/>
</font> . . .
<!-- další varianty pro proložené písmo, pro proložené tučné -->
```

FOP pak spouštíme samozřejmě s parametrem - c C:\fop\conf\myfop.conf. Atribut kerning=“yes“ zajišťuje opticky optimalizované vzdálenosti

[4] Citováno z [10] viz Seznam použité literatury

pro každou dvojici písmen zvlášť. U některých starších verzí (např 0.91beta) je vhodné nastavit u všech fontů kerning=“no“. Při transformaci se tak vyhneme obrovskému množství varovných hlášek “Kerning support is disabled until it is supported by the layout engine!“. U verzí 0.95 či Trunk jsem se s tím nesetkal.

4.5.5 Instalace a nastavení programu Barcode4J

Barcode4J je flexibilní generátor pro čárové kódy napsaný v jazyce Java. Jedná se o open source vyvíjený stejně jako FOP společností Apache. Barcode4J podporuje generování 1D čárových kódů (Code 39, Code 128, EAN-128, GS1-128, Codabar a jiné), 2D čárových kódů (PDF 147,DataMatrix). Dalšími podporovanými výstupy jsou například SVG, EPS, PNG, JPEG, Java2D(AWT).

Barcode4J lze využívat pro generování čárových kódů buď samostatně, nebo může sloužit jako rozšiřující část pro FO processor FOP či XSLT procesory Xalan a Saxon. Nás bude především zajímat rozšíření pro FOP. Na stránkách Barcode4J v sekci download lze stáhnout již druhou verzi Barcode4J.

Pro správné fungování s FOPem je třeba nastavit CLASSPATH pro soubory *barcode4j.jar* a *barcode4j-fop-ext.jar*. Je možné použít soubor *barcode4j-fop-ext-complete.jar* Cesta by mohla vypadat například takto:

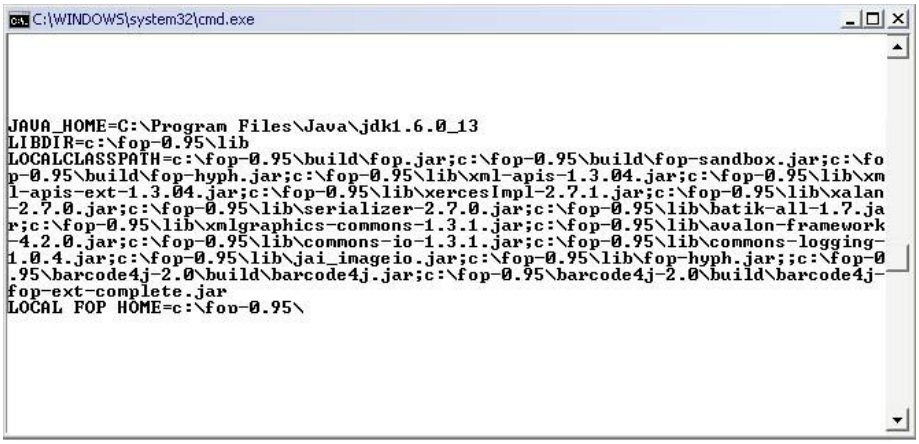
java -cp lib\avalon-framework-4.2.0.jar;lib\commons-cli-1.0.jar;build\barcode4j.jar org.krysalis.barcode4j.cli.Main

Místo *lib* je třeba nastavit skutečnou cestu umístění souborů. Pro práci s FOPem v příkazové řádce je třeba ve FOPu ještě upravit soubor *fop.bat* přidáním dvou řádek s nastavením cest

set
LOCALCLASSPATH=%LOCALCLASSPATH%;%LOCAL_FOP_HOME
%barcode4j-2.0\build\barcode4j.jar

set
LOCALCLASSPATH=%LOCALCLASSPATH%;%LOCAL_FOP_HOME
%barcode4j-2.0\build\barcode4j-fop-ext-complete.jar.

Pokud spustíte v příkazové řádce soubor *fop.bat* a poté napíšete příkaz *set*, mělo by se vám objevit nastavení cesty, ve které je zahrnuto i nastavení Barcode4J. Na obrázku 4.5 můžete vidět správné nastavení pro chod Barcode4J s FOPem.



Obrázek 4.5: Správné nastavení Barcode4J ve FOPu

Barcode4J je možné taktéž získat v podobě zdrojového kódu a pomocí Antu si jej překompilovat. Verzi 2.0 jsem přiložil s FOPem 0.95, který má již upravený *fop.bat*, na CD v sekci XSL-FO procesory. Součástí balíčku Barcode4J je mnoho příkladů psaní čárových kódů v XML, XSLT či pomocí formátovacích objektů, které lze použít pro FOP.

4.6 Shrnutí XSL-FO procesory

K vybrání vhodného XSL-FO procesoru nám poslouží následující tabulka 4.2.

Tabulka 4.2: Shrnutí XSL-FO procesorů

vlastnosti	AH Formatter	XEP	FOP	XFC
verze	komerční	komerční	open source	komerční
podpora XSL-FO 1.1	plná podpora	plná podpora	téměř plná podpora	částečná podpora
podpora CZ	ano	ne/ doinstalovat	ne/doinstalovat	ano
kvalita výstupu	výborná	výborná	velmi dobrá	X
velikost výstupu PDF	nejmenší	menší než u FOPu	vyšší než u XEPu	X
aktualizace	neustálý vývoj	neustálý vývoj	neustálý vývoj	neustálý vývoj

Bohužel PassiveTex se mi nepodařilo nastavit, proto není uveden v tabulce. XSL-FO procesorů na rozdíl od XSLT procesorů není mnoho. Z tabulky je patrné, že většina procesorů je komerčních. Jediným kvalitním open source procesorem je FOP. Proto věnuji tomuto procesoru ve své práci velkou pozornost. U procesoru XFC nelze porovnávat kvalitu výstupu PDF ani velikost, jelikož nepodporuje výstup do PDF.

Pro práci s XSL-FO doporučuji používat dvojici procesorů FOP a XFC. FOP pro převod .fo souborů do PDF a XFC pro převod .fo souboru do OOXML nebo Open Office.

5 Tvorba faktury pomocí XSL-FO

Až donedávna žila většina českých firem v přesvědčení, že daňové doklady lze uchovávat výhradně jako arch papíru opatřený razítkem a podpisem. V dnešní době český právní řád umožňuje vystavování faktur či daňových dokladů v elektronické podobě. Není nutný žádný formální proces přijetí faktury, evidence faktury či opětovného zadání faktury do účetního programu vedeného na počítači. Dále lze mezi výhody elektronických daňových dokladů, zejména faktur, určitě počítat i úsporu lidské práce a úsporu nákladů na poštovném a na papíře. V tabulce 5.1 můžete vidět efektivnost elektronické faktury vůči faktuře papírové.

Tabulka 5.1: Vývoj ukazatelů po zavedení elektronické fakturace (Zdroj: SPIS)

náklad	původní stav	nový stav	zlepšení
zpracování dokladu	5dní	1den	o 80% kratší
administrativní náklady	2USD	1USD	o 50% nižší
zamezení chyb	5%	0,05%	o 99% méně
zlepšení kontroly likvidace	3%	2,5%	o 15% méně
úspora pracovníků	30	15	o 50% méně

5.1 Faktura a její náležitosti

Faktura je běžný doklad, který musí podle Zákona o DPH povinně obsahovat předepsané náležitosti. Pokud některé kritérium nesplňuje, nelze ho považovat za daňový doklad.

Každý daňový doklad musí obsahovat:

- **označení kupujícího a prodávajícího** (obchodní firma nebo jméno a příjmení, název a dodatek ke jménu, sídlo, místo podnikání plátce, DIČ),
- **evidenční číslo daňového dokladu** (podle účetnictví nebo daňové evidence plátce),
- **datum vystavení dokladu**,
- **datum uskutečnění zdanitelného plnění** nebo datum přijetí platby (pokud se liší od data vystavení daňového dokladu),
- **jednotkovou cenu bez DPH**, případně slevu, pokud již není obsažena v jednotkové ceně,
- **základ daně**,
- **sazbu daně** (snížená 10 % nebo základní 20 %),
- **výši daně** uvedenou v korunách a haléřích, popř. zaokrouhlenou.

Na faktuře nesmí chybět žádný z těchto údajů. **Razítko ani podpis nepatří mezi základní náležitosti daňového dokladu** a nemusí být tedy na daňovém dokladu. Elektronický doklad však musí obsahovat elektronický podpis anebo elektronickou značku založenou na kvalifikovaném systémovém certifikátu. Dnes se pro vyšší důvěryhodnost používají digitální podpisy, které jsou založeny na asymetrické kryptografii.

5.2 Vizuální návrh faktury

Nejdůležitější částí při výrobě elektronické faktury je určitě její celkový vzhled a samozřejmě s tím i související přehlednost a pořizovací náklady na výtisk. Jak už jsme si výše říkali, faktura musí splňovat určitá pravidla proto, aby mohla být považována za daňový doklad. Rozmístění daných

náležitostí a vytvoření přívětivé podoby návrhu bývá leckdy problematické. Nakupuji přes internet velmi často, a tak nebyl problém si sehnat elektronické faktury. Pokud bych měl porovnat design e-faktur před cca 5 lety a dnes, je vidět velký pokrok kupředu. Odpovědět na otázku, proč tomu tak je, není nikterak jednoduché. Jednou z hlavních příčin je, že český právní řád umožňuje vystavování faktur v elektronické podobě, která plně nahrazuje papírovou formu. Řada firem zabývající se výrobou softwaru se začala soustředit na výrobu softwaru pro návrh šablon e-faktur. Popularita XML, jakožto jazyka pro přenos informací nezávislá na jakémkoliv prostředí, neustále stoupá a vznikají nové jazykové standardy založené na XML.

5.3 Předloha elektronické faktury

Pro návrh vytvoření designu jsem se rozhodoval mezi fakturami předních e-shopů prodávajících výpočetní techniku a příslušenství. Na výběr jsem měl e-faktury z e-shopů TNTrade, Czechcomputer, Alfacomp a Cybex. Nakonec jsem se rozhodl vyrobit pomocí XSL-FO e-fakturu Alfacompu pro její perfektní přehlednost a příjemný design.

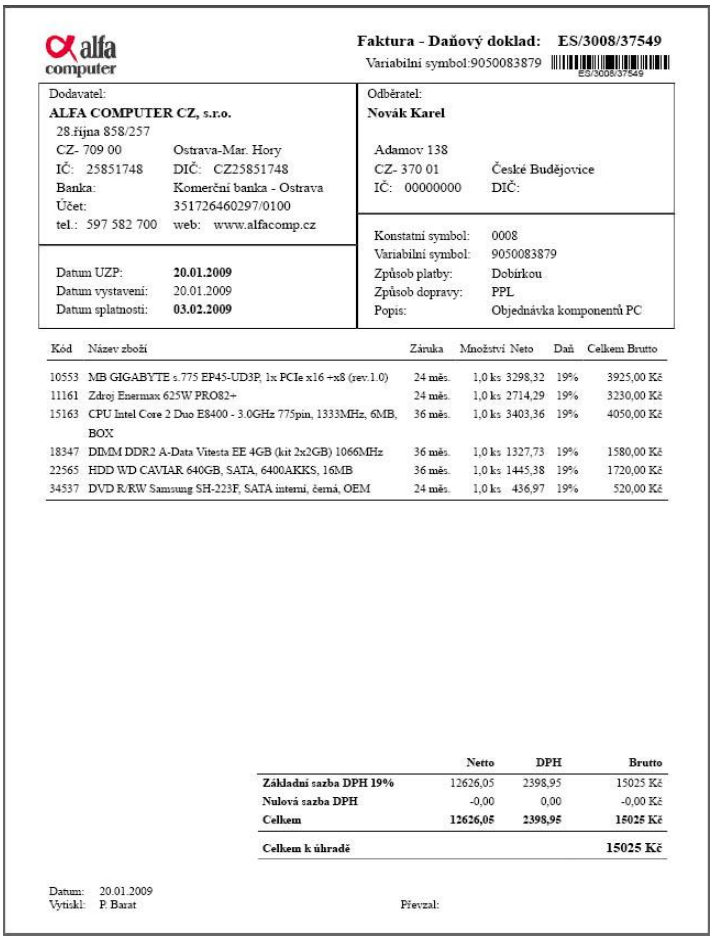
5.3.1 Popis rozvržení e-faktury

V prvních návrzích jsem se snažil využívat regiony pro rozdělení faktury na záhlaví, tělo a zápatí.

Do záhlaví jsem umístil logo firmy, evidenční číslo faktury, variabilní symbol a čárový kód. V těle jsem vytvořil 4 kontejnery s rámy, ve kterých jsou informace o dodavateli, odběrateli, datum UZP, vystavení a splatnosti, v posledním kontejneru je konstantní a variabilní symbol, způsob platby a popis faktury.

Ve středové části těla jsou vybrané položky zákazníka. Každá položka má svůj vlastní kód, název, množství, netto, daň a celkem brutto. Pod položkami je umístěný souhrn položek netto, DPH, brutto a celková cena. Hned pod souhrnem lze nalézt řádek o celkové úhradě.V zápatí je umístěno datum vystavení, informace o tom, kdo fakturu vyhotovil a je zde připravená část pro razítko a digitální podpis. Faktura je samozřejmě univerzální a po vytisknutí ji lze využít jako klasickou papírovou fakturu.

Abychom měli skutečnou představu, jak popisovaný návrh faktury vypadá, na obrázku 5.1 je její rozvržení.



alfa computer

Faktura - Daňový doklad: ES/3008/37549

Variabilní symbol:9050083879

ES/3008/37549

Dodavatel:		Odběratel:	
ALFA COMPUTER CZ, s.r.o.		**Novák Karel**	
28 října 858/257			
CZ- 709 00	Ostrava-Mar. Hory	Adamov 138	
IČ: 25851748	DIČ: CZ25851748	CZ- 370 01	České Budějovice
Banka:	Komerční banka - Ostrava	IČ: 00000000	DIČ:
Účet:	351726460297/0100		
tel.: 597 582 700	web: www.alfacomp.cz		

Datum UZP:	**20.01.2009**	Konstatní symbol:	0008
Datum vystavení:	20.01.2009	Variabilní symbol:	9050083879
Datum splatnosti:	**03.02.2009**	Způsob platby:	Dobírkou
		Způsob dopravy:	PPL
		Popis:	Objednávka komponentů PC

Kód	Název zboží	Záruka	Množství	Neto	Daň	Celkem Brutto
10553	MB GIGABYTE s.775 EP45-UD3P, 1x PCIe x16 +x8 (rev.1.0)	24 měs.	1,0 ks	3298,32	19%	3925,00 Kč
11161	Zdroj Enermax 625W PRO82+	24 měs.	1,0 ks	2714,29	19%	3230,00 Kč
15163	CPU Intel Core 2 Duo E8400 - 3.0GHz 775pin, 1333MHz, 6MB, BOX	36 měs.	1,0 ks	3403,36	19%	4050,00 Kč
18347	DIMM DDR2 A-Data Vitesta EE 4GB (kit 2x2GB) 1066MHz	36 měs.	1,0 ks	1327,73	19%	1580,00 Kč
22565	HDD WD CAVIAR 640GB, SATA, 6400AKKS, 16MB	36 měs.	1,0 ks	1445,38	19%	1720,00 Kč
34537	DVD R/RW Samsung SH-223F, SATA interní, černá, OEM	24 měs.	1,0 ks	436,97	19%	520,00 Kč

	Netto	DPH	Brutto
Základní sazba DPH 19%	12626,05	2398,95	15025 Kč
Nulová sazba DPH	-0,00	0,00	-0,00 Kč
Celkem	**12626,05**	**2398,95**	**15025 Kč**
Celkem k úhradě			**15025 Kč**

Datum: 20.01.2009
Vytiskl: P. Barat

Převzal:

Obrázek 5.1: Návrh elektronické faktury

Rozvržení částí do regionů se může zdát jako výhodné, ale u absolutního pozicování celého návrhu ztrácí na významu. Zbytečně bychom si zkomplikovali práci při vytváření šablony faktury. Proto jsem v konečné fázi návrhu zrušil region záhlaví a zápatí a výsledný návrh vložil do regionu těla. Na vzhledu faktury se vůbec nic nezmění a při psaní šablony si ušetříme několik desítek řádek. Výsledný návrh faktury si můžete prohlédnout po vygenerování souboru *Fakturafinal.fo* do PDF nebo ve FOPu příkazem *fop -awt Fakturafinal.fo* spustit náhled. **Na CD v sekci Faktura naleznete veškeré zdrojové kódy související s tvorbou e-faktury.**

5.3.2 Výběr formátovacích objektů pro návrh e-faktury

Formátovacích značek je poměrně velké množství, ale většina se zabývá vytvářením objektů pro psaní dokumentů. Na tuto práci však existuje mocnější nástroj - Docbook. XSL-FO bych doporučil používat jen u velmi krátkých dokumentů. Je možné tedy vytvářet grafické návrhy e-faktur pomocí formátovacích objektů? Po hlubším prozkoumání formátovacích objektů zjistíte, že jednou z možností, kterou považuji za nejdůležitější při vytváření grafického návrhu, je možnost absolutního pozicování.

Tuto možnost plně využívám ve svém návrhu a veškerý obsah faktury je obalen elementy *<fo:block-container>* s atributem *position="absolute"*. Pro správné rozmístění objektů na stránce jsem využil možnosti vytvoření rámu pomocí atributu *border="1pt solid"*. Na příkladu 5.1 je ukázka vytvoření hlavního rámu celé faktury. Začátek tvoření návrhu je možné zhlédnout po převedení procesorem FOP soubor *Faktura1.fo*.

Příklad 5.1: Vytvoření hlavního rámu e-faktury

```
<fo:block-container border-style="solid" border-width="1pt"
height="257mm"                    width="188mm"  top="2mm"
left="2mm" right="2mm" border-color="#5b5a58" padding="1pt"
position="absolute">
  <fo:block >
  </fo:block>
</fo:block-container>
```

Výšku a šířku rámu jsem nastavil pomocí atributů *height* a *width*. Velmi důležitými atributy elementu <*fo:block-container*> jsou *top, left, right, bottom,* pomocí kterých nastavujeme umístění kontejneru v regionu. Co jsou to regiony, se můžete dočíst v příručce XSL-FO, která je umístěna v Příloze B.

Řekli jsme si, že obsah faktury je obalen kontejnery, jak ale uspořádat text v těchto objektech? Pokud použijeme klasický element <*fo:block*>, vytvoříme tím obdélníkovou plochu přes celý řádek a nebudeme moci na stejný řádek nic dalšího umístit. Přímé použití v elementu <*fo:block-container*> je tedy nežádoucí. Další možností by bylo využít opět element <*fo:block-container*>. Návrh bychom nejspíše vytvořili, ale výsledný zdrojový kód by byl obrovský a hodně nepřehledný. Pro rozvržení obsahu v kontejneru využijeme tabulky, stejně jako kdysi při návrhu HTML stránek. Pomocí tabulky rozdělíme řádky na několik částí libovolné délky. Pro rozvržení obsahu v kontejneru využijeme tabulky, stejně jako kdysi při návrhu HTML stránek. Pomocí tabulky rozdělíme řádky na několik částí libovolné délky. Práce s tabulkami hraje tedy v návrhu zásadní roli. Formátovací objekty poskytují pro práci s tabulkami širokou základnu. Je možné vložit novou tabulku do buňky, slučovat buňky a rozdělit buňky. Velkou výhodou je taktéž možnost vytvoření nejenom rámečku pro tabulku či jednotlivých buněk, ale také můžeme vytvořit pouze část rámečku na jakékoliv straně tabulky.

5.3.3 Řešení problémových částí návrhu

V prvním a druhém kontejneru řeším problém s oddělením IČ, DIČ, web a tel. od čísla. Tato problematika se dá řešit dvěma způsoby. Prvním je vytvoření tabulky o 6 řádcích a 2 sloupcích. Na příkladu 5.2 je vidět vytvoření tabulky, která vypouští číslo, které je uloženo v XML souboru.

Příklad 5.2: První varianta rozvržení textu v kontejneru pro dodavatele

```
<fo:table>
   <fo:table-column column-width="35mm"/>
   <fo:table-column column-width="55mm"/>
     <fo:table-body>
       <fo:table-row>
         <fo:table-cell>
  <fo:block>28.října 858/257</fo:block>
         </fo:table-cell>
         <fo:table-cell>
   <fo:block/>
         </fo:table-cell>
       </fo:table-row>
            . . . . .
      <fo:table-row>
        <fo:table-cell>
          <fo:block>IČ: </fo:block>
        </fo:table-cell>
        <fo:table-cell>
          <fo:block>DIČ: </fo:block>
        </fo:table-cell>
      </fo:table-row>
  . . . . .
     </fo:table-body>
</fo:table>
```

Dodání čísla do buňky s IČ či DIČ se provede v šabloně, kde je spojeno XSLT s XSL-FO pomocí elementu *<xsl:value-of select=""/>*. Tato varianta nám sice ušetří práci a stačí vytvořit jednoduchou tabulku, ale text vybraný z elementu v xml se připojí hned za IČ a není možné ho poodsadit.

Druhou možností je vytvoření tabulky s 6 řádky a 4 sloupci. Tam, kde 4 sloupce nepotřebujeme, sloučíme je pomocí atributu *number-columns-spanned* elementu <*fo:table-cell*>. Následující příklad 5.3 znázorňuje vyřešení případu oddělení IČ od čísla a také možnost libovolného odsazení.

Příklad 5.3: Druhá varianta rozvržení textu v kontejneru pro dodavatele

```
<fo:table  table-layout="fixed">
   <fo:table-column column-width="9mm"/>
   <fo:table-column column-width="26mm"/>
   <fo:table-column column-width="12mm"/>
   <fo:table-column column-width="proportional-column-
width(1)"/>
     <fo:table-body>
      <fo:table-row>
       <fo:table-cell number-columns-spanned="2">
 <fo:block>28.října 858/257</fo:block>
       </fo:table-cell>
      </fo:table-row>
      <fo:table-row>
       <fo:table-cell number-columns-spanned="2">
         <fo:block>CZ-709 00</fo:block>
       </fo:table-cell>
       <fo:table-cell number-columns-spanned="2">
         <fo:block>Ostrava-Mar. Hory </fo:block>
       </fo:table-cell>
      </fo:table-row>
      <fo:table-row>
       <fo:table-cell>
 <fo:block>IČ:</fo:block>
       </fo:table-cell>
       <fo:table-cell>
 <fo:block>25851748</fo:block>
       </fo:table-cell>
 . . . . .
    </fo:table-body>
</fo:table>
```

Abychom nemuseli pracně dopočítávat šířku posledního sloupce, nastavíme hodnotu atributu *column-width="proportional-column-width(1)"*. Tento atribut smíme použít jedině tehdy, pokud u elementu <*fo:table*> použijeme atribut *table-layout="fixed"*. Tato možnost může být výhodná taktéž u tabulek, které mají rámeček. Automaticky nám podle šířky textu v poslední buňce nastaví ohraničení tabulky zprava. Nevhodná je u tabulek, kde chceme mít pevně danou šířku tabulky.

Při návrhu jsem se setkal také s možností přetečení kontejneru u dodavatele. Přetečení může být způsobené příliš dlouhým názvem města nebo banky. Dojde k automatickému zalomení na další řádek a kontejner přeteče přes spodní okraj. Z tohoto důvodu jsem nechal z pravé strany kontejneru dodavatele dostatečný prostor pro dlouhé názvy. Pokud by i přesto došlo k přetečení, je možné nastavit hodnotu atributu kontejneru *height="auto"*.

5.3.4 Podtržení menu a poslední položky

Pro podtržení menu jsem měl na výběr ze dvou možností. Tou první je využití elementu <*fo:leader*>, který se využívá pro potržení poznámek pod čarou nebo vytváří výplň seznamu. Pokud bych použil *<fo:leader leader-pattern="rule" leader-length="100%"/>*, docílil bych sice vytvoření podtrženého menu, ale odstup od něj by byl dost velký. Tato varianta proto není příliš vhodná. Druhou možností je využít vytvoření rámečku na spodní části tabulky či řádku. Tuto vlastnost nám umožnuje atribut *border-bottom="1px solid"*. Pokud tento atribut umístíme do elementu <*fo:table-body*>, podtržení se projeví až na konci tabulky. Vložením do elementu <*fo:table-row*> docílíme dotržení spodní strany řádku. V našem případě dojde k podtržení menu.

5.3.5 Čárový kód

Součástí faktury je také čárový kód. Žádný FO procesor nemá v základní výbavě podporu generování čárových kódů. Jediným open source řešením pro generování kódů je program Barcode4J, který může sloužit jako rozšíření pro FOP. Instalaci a nastavení jsme si ukázali již v kapitole zabývající se XSL-FO procesory. Zápis čárového kódu je u každého FO procesoru odlišný. Ve faktuře jsem použil Code128. Celý zápis čárového kódu se vkládá do elementu *<fo:instream-foreign-object>* Zápis pomocí formátovacích objektů si můžete prohlédnout na příkladu 5.4.

Příklad 5.4: Ukázka zápisu čárového kódu ve FOPu

```
<fo:instream-foreign-object>
 <bc:barcode xmlns:bc="http://barcode4j.krysalis.org/ns"
message="{cislo_faktury/text()}">
  <bc:code128>
   <bc:height>8mm</bc:height>
  </bc:code128>
 </bc:barcode>
</fo:instream-foreign-object>
```

Jak je z příkladu patrné stejně jako tomu je u vkládání SVG grafiky i zde má potomek elementu *<fo:instream-foreign-object>* jiný jmenný prostor než XSL-FO. **Pro vytváření čárového kódu jsem využil sílu XPath a čárový kód se automaticky vytváří z evidenčního čísla faktury.** Využil jsem možnosti vkládání textové hodnoty elementu do hodnoty atributu *message*, tím se zajistí automatické vytvoření nového čárového kódu a nemůže se stát, že by dvě faktury měly stejný čárový kód.

5.3.6 Digitální podpis

Digitální podpis je nejúčinnějším prostředkem pro zajištění integrity odesílaných dat a bezpečné ověření jejich odesílatele. Digitální klíč je vypočten z kryptografického kontrolního součtu na základě tajného klíče.

Kryptografický kontrolní součet je vypočítán z podepisovaných dat. Ověřování digitálního klíče probíhá tak, že příjemce ověří, že digitální klíč vyhovuje veřejnému klíči odesílatele. Pokud se shodují, je zaručeno, že dokument nebyl změněn třetí osobou. Příjemce má pouze veřejný klíč odesílatele, kterým ověří jeho digitální klíč.

Dne 10. června 2008 byl schválen konsorciem W3C XML Signature WG standart. XML Signature je standart pro přidání digitálního podpisu k XML dokumentu. Digitální podpis je zapisován v XML syntaxi. Za pomoci jazyka XPath je možné podepisovat pouze část XML dokumentu. Bohužel pochopení této poměrně složité oblasti by vyžadovalo rozsáhlé množství informací a překročilo by rámec tématu mé práce, proto se oblastí XML Signature již dále zabývat nebudu.

Digitální podpis je také možné připojit k vytvořené faktuře pomocí vhodného softwaru. Nejčastěji je faktura vygenerována do PDF souboru. Digitální podpis je možné připojit pomocí Adobe Reader, Acrobat Professional od verzí 8 nebo pomocí freewaru FreePDFSign. Finální verze faktury v PDF obsahuje digitální klíč vytvoření v Adobe Acrobat 8.0.

5.4 Tvorba šablony faktury

V šabloně se mísí kombinace XSLT a XSL-FO. Formátovací objekty vytvářejí návrh faktury a ve spojení s XSLT elementy nám vznikne po transformaci pomocí některého XSLT procesoru dokument, ve kterém došlo ke spojení XML dat s formátovacími objekty. Pokud je šablona správně navržena, výsledný dokument by měl být shodný s návrhem faktury. Šablona je uložena v souboru *SablonaFaktura.xsl*.

Každá šablona, ve které jsou použiti XSLT a XSL-FO elementy musí obsahovat deklaraci a definici pro jmenné prostory.

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

Tyto řádky nám říkají, že dokument bude v XML formátu a všechny elementy budou ze dvou jmenných prostorů.

5.4.1 Vytvoření layoutu stránky

Pomocí elementu <*xsl:template match="faktury"*> se vybere celý XML dokument, kde kořenovým elementem je element <*faktury*>. Poté dojde k vytvoření kořenového elementu <*fo:root*> pro formátovací objekty a následně k nastavení velikosti stránky a nastavení okrajů. V regionu těla se do elementu <*fo:block*> aplikují všechny vytvořené šablony.

Příklad 5.5: Vytvoření layoutu stránky

```
<xsl:template match="faktury">
 <fo:root>
  <fo:layout-master-set>
   <fo:simple-page-master master-name="page"
page-height="297mm"
page-width="210mm"
margin-left="11mm"
margin-right="30mm"
margin-top="5mm">
     <fo:region-body/>
   </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="page">
   <fo:flow flow-name="xsl-region-body"
font-family="TimesNewRoman"
font-size="12pt"
line-height="17pt" >
     <fo:block>
```

```
        <xsl:apply-templates/>
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
 </fo:root>
</xsl:template>
```

5.4.2 Vytvoření „hlavní“ šablony

„Hlavní“ šablona je připojena k elementu *<faktura>* v XML dokumentu. V této šabloně jsou aplikovány všechny vytvořené šablony. Zde se určuje pořadí zpracování všech šablon. Jelikož je v celém návrhu využito absolutní pozicování, nemusíme brát ohled na to, jak šablony půjdou po sobě. Pro lepší kontrolu s návrhem faktury seřadíme šablony podle vzniku návrhu.

Příklad 5.6: Šablona určující zpracování šablon

```
<xsl:template match="faktura">
   <xsl:apply-templates select="zahlavi"/>
   <xsl:apply-templates select="dodavatel"/>
   <xsl:apply-templates select="odberatel"/>
   <xsl:apply-templates select="datum"/>
   <xsl:apply-templates select="info"/>
   <xsl:apply-templates select="polozky"/>
   <xsl:apply-templates select="souhrn"/>
   <xsl:apply-templates select="zapati"/>
</xsl:template>
```

5.4.3 Vytvoření šablony pro položky

Šablona pro položky je nejdůležitější šablonou. Pomocí elementu *<xsl:for-each select="polozka">* zajistím zpracování všech elementů *<polozka>*. Elementem *<xsl:sort select="kod"/>* zajistím seřazení položek podle kódu od nejmenšího po nejvyšší. Hodnotu elementu zapíši do buňky pomocí elementu *<xsl:value-of select=" /text()"/>*.

Příklad 5.7: Ukázka části šablony pro položky

```
<fo:block border-bottom="1px solid " font-size="10pt" margin-
top="8pt" text-align="right">
<fo:table table-layout="fixed">
<fo:table-column column-width="11mm"/>
<fo:table-column column-width="96mm"/>
<fo:table-column column-width="15mm"/>
<fo:table-column column-width="15mm"/>
<fo:table-column column-width="13mm"/>
<fo:table-column column-width="10mm"/>
<fo:table-column ="proportional-column-width(1)"/>
<fo:table-body>
  <xsl:for-each select="polozka">
   <xsl:sort select="kod"/>
     <fo:table-row>
       <fo:table-cell>
          <fo:block><xsl:value-of
select="kod/text()"/></fo:block>
       </fo:table-cell>
       <fo:table-cell><fo:block text-align="left">
         <xsl:value-of select="nazev/text()"/></fo:block>
       </fo:table-cell>
       <fo:table-cell>
          <fo:block><xsl:value-of
select="zaruka/text()"/></fo:block>
       </fo:table-cell>
       <fo:table-cell>
          <fo:block><xsl:value-of
select="mnozstvi/text()"/></fo:block>
       </fo:table-cell>
           . . . . .
     </fo:table-row>
   </xsl:for-each>
</fo:table-body>
  </fo:table>
```

5.5 Převod XML faktury do PDF

Fakturu v XML můžeme pomocí XSLT šablony, která obsahuje formátovací objekty, převést buď přímo pomocí FO procesoru, nebo nejdříve provést transformaci XSLT procesorem a následný FO dokument převést FO procesorem do PDF. Variantu s využitím XSLT procesoru jsem používal při potřebě ověření shodnosti návrhu s vygenerovaným FO dokumentem. Nyní využije pouze FO procesor. Na převod jsem využil stabilní verzi FOP 0.95+Barcode4J, zkušební verzi XEP4.15 a Antenna House FormatterV5 Evolution. Výsledné PDF vytvořené těmito procesory si můžete prohlédnout na CD v sekci *Faktura*.

5.5.1 Převedení ve FOPu

Při převodu XML faktury do PDF ve FOPu se zabrazí v příkazové řádce jedno chybové hlášení o přetečení čárového kódu přes blok. Tato „chyba“ nemá žádný vliv na zobrazení faktury v PDF či jiném výstupním formátu. Dále zde můžeme najít info o nepodpoře atributu *table-layout*. Pokud by nebyl podporován tento atribut, nefungovala by správně ani automatická velikost posledního řádku podle velikosti textu. Tu jsem prověřil pomocí ohraničení každé buňky v posledním sloupci a vše fungovalo správně.

5.5.2 Převedení v XEPu

Převod v XEPu byl také úspěšný a vše proběhlo, jak jsem očekával. V hlášení o převodu se nejdříve prověřila validita souborů *faktura.xml* a *SablonaFaktura.xsl*. Ta proběhla v pořádku. Poté se zobrazila chyba o nemožnosti zobrazení čárového kódu. Samotná verze XEPu nepodporuje čárové kódy. Dále je v šabloně připravena možnost vložení razítka v podobě obrázku. XEP napíše chybové hlášení o nenalezení obrázku. V tomto případě se jedná sice o chybu, ale nemá žádný vliv na správné zobrazení a po vložení

razítka se již chybová hláška nebude po převodu zobrazovat. FOP se o této chybě vůbec nezmínil.

5.5.3 Převedení v AHFormatteru

Bohužel AHFormatter nemá žádné okno s hlášením o stavu převodu. Je možné, že plná verze touto vlastností disponuje. Pokud dokument nelze převést, zobrazí pouze hlášku o neúspěšném převodu.

5.6 Porovnání výstupu FO procesorů

Fakturu jsem původně navrhoval pro správné zobrazení ve FOPu. Když jsem získal i jiné FO procesory (XEP a AHFormatter), zkusil jsem převod faktury pomocí těchto procesorů a s hrůzou jsem sledoval dosti odlišný výstup mezi všemi procesory. FOP i přes některé chybové hlášky zobrazoval výstup správně. XEP již tak benevolentní nebyl. Některé chybové hlášky toleroval, jiné nikoliv. Troufám si tvrdit, že AHFormatter věrně zobrazuje zdrojový kód. Všechny chybové hlášky byly zobrazeny ve výstupu. Pro odstranění chyb a správné zobrazení výstupu jsem použil tento FO procesor. Bohužel nebylo možné vytvořit jednotný výstup ve všech třech procesorech. Problém je především se zobrazováním loga. Pokud si dobře prohlédnete všechny vytvořené faktury, zjistíte, že logo u faktury z FOPu je mnohem větší než u ostatních faktur. U faktury z XEPu je zase naopak dosti malé. Zlatou střední cestou je logo na faktuře z AHFormatteru. Původně jsem si myslel, že je chyba ve vykreslování grafiky na straně procesorů. Vytvořil jsem nový dokument, ve kterém byla pouze grafika. Kupodivu po převedení všechny tři procesory zobrazovaly shodný výsledek. Při větším přiblížení jsem narazil na horší podání barev u procesoru FOP. Světlé barvy, především pak light-red, jsou tmavší než u XEPu či AHFormatteru. U monitorů LCD TFT TN je tento jev hůře pozorovatelný. Tuto záhadu se mi nepodařilo objasnit.

Domnívám se, že problém je částečně na straně blokových kontejnerů a absolutního pozicování. FOP plně tyto vlastnosti nepodporuje. Druhým viditelným rozdílem je nulové zobrazení okrajů v rámečkových kontejnerech u faktury vytvořené v XEPu. V podpoře elementů a atributů XEPu je atribut *margin-left* podporován, bohužel v našem případě je tento atribut ignorován.

Důležitou vlastností je taktéž velikost vygenerovaného souboru. Faktura převedená ve FOPu má 72 kb, velikost faktury v AHFormatteru je 89 kb. Nejhůře dopadl XEP, který vygeneroval fakturu s velikostí 115 kb. Může se zdát, že rozdíly jsou zanedbatelné, ale při skladování faktur po několik let se tyto rozdíly silně prohlubují. Pokud do FOPu nenaimplementujeme češtinu, velikost faktury se 6 krát sníží. Totéž se dá říci o procesoru XEP, který taktéž v základu nepodporuje češtinu.

5.7 Závěrečné hodnocení

Ačkoliv mám velmi blízký vztah ke grafice, tvorba návrhu nebyla nikterak jednoduchá a vyžadovala neustálý náhled zobrazování. Jako velkým pomocníkem se ukázal XML editor XMLSpy 2009 od firmy Altova. V nastavení si můžete jednoduše nastavit vlastní upravenou verzi FO procesoru pro náhled zobrazení. Návrh jsem se snažil vytvořit také pomocí XSL-FO editoru StyleVision 2009 od Altovy, bohužel výsledný zdrojový kód byl velmi nepřehledný a obsahoval velké množství nevyžádaných atributů. V šabloně je použito poměrně malé množství XSLT elementů. Tabulky jsou taktéž poměrně netradičně tvořeny. V tomto případě si myslím, že tvorba tabulky pomocí několika xslt šablon je neefektivní a zdrojový kód činí zbytečně složitým. Za „vychytávku“ považuji automatické generování čárového kódu podle evidenčního čísla. XML soubor nemusí obsahovat element pro čárový kód. Při návrhu bylo velmi důležité správné navržení části s položkami, zvolení dostatečné velikosti sloupců pro jednotlivé části položky a zarovnání

jednotlivých sloupců. Co mě jako grafika trošku mrzí je nemožnost zakulacení rohů u rámečků. Společnost Ecrion sice ve svém tutoriálu o XSL-FO používá atribut *border-radius* pro zakulacení rohů, tento atribut bohužel v XSL-FO 1.1 standardu nenajdete. Nevím, kde by se dala ukázat větší názorná efektivnost využití XSL-FO než při tvorbě šablony faktury.

6 Převod FO do ODF nebo OOXML

ODF (OpenDocument Format) je XML formát, který byl vytvořen pro kancelářský balík OpenOffice.org. ODF lze použít pro ukládání textových dokumentů, prezentací, dokumentů tabulkových procesorů, kreslících, 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 je mezinárodním standardem (ISO/IEC 26300) od 30. listopadu 2006.

OOXML (Office Open XML) je rovněž XML formát pro ukládání kancelářských dokumentů od firmy Microsoft. Poprvé byl použit v kancelářském balíku Microsoft Office 2007. Formát Office Open XML je ZIP soubor, výsledkem jsou tak menší soubory než ty binární, které byly vytvářeny předchozími verzemi Microsoft Office. OOXML byl schválen mezinárodním standardem (ISO/IEC 29500) 2. dubna 2008.

6.1 Převod .fo souboru do ODF

ODF se s XSL-FO v mnohém podobají. Využívají mnoho společných nebo podobných formátovacích vlastností. 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. Bohužel některé objekty se liší velmi výrazně a některé dokonce nelze ani převést. V následující tabulce se můžeme podívat na zápis některých shodných elementů v XSLFO a ODF.

Tabulka 6.1: Zápis elemntů v XSL-FO a ODF

XSL-FO	ODF
<fo: block>	<text: p>
<fo:inline>	<text:span>
<fo:table-column>	<table:table-column>

Pro přímý převod .fo souborů můžeme využít XSL-FO procesor XFC. Pro vyzkoušení převodu z XSL-FO do ODF jsem použil .fo soubor pozvánky. Transformace proběhla v pořádku. Po otevření výstupního souboru v Open Office byl text rozvržen stejně jako v PDF souboru. Bohužel pokud použijeme absolutní pozicování, XFC tento element neumí převést.

Další možností jak převést .fo soubory do ODF, je využití programu FO2ODF Converter od Petra Bodnára. Tento konvertor můžete nálézt na přiloženém CD. Pokud nechcete nic instalovat, je možné převést tento soubor přímo na stránce FO2ODF Converter.

6.2 Převod .fo souboru do OOXML

Hlavním rozdílem mezi XSL-FO a OOXML je, že formátovací objekty nepoužívají mnoho elementů. K definování vlastností používají atributy. Naopak OOXML používá elementy téměř ke všemu, k definování struktur dokumentu i k definování vlastností. Dokument s formátovacími objekty nejprve definuje všechny možné rozvržení stránek a až poté následuje samotný obsah dokumentu, odkazující se na tato rozvržení. Naproti tomu formát WordML definuje rozvržení stránek až za vlastním obsahem dokumentu a to tak, že se každá definice vztahuje právě k předešlému obsahu. I přes mnoho rozdílů mezi jednotlivými formáty lze většinu formátovacích objektů převést do OOXML.

Podařilo se mi nalézt pouze jediný přímý konvertor z .fo souborů do OOXML. Tímto konvertorem je XMLmind s XFC procesorem. Pro vstup jsem použil stejně jako u ODF .fo soubor pozvánky. Výstup byl naprosto shodný s výstupem do ODF.

Závěr

V práci jsem se snažil zachytit současný stupeň poznání v oblasti XSL-FO. Mým cílem bylo napsat příručku, která obsahuje nejčastěji používané formátovací objekty a jejich atributy. Na názorných příkladech ukázat jejich správné použití. Celé zdrojové kódy těchto příkladů jsou uloženy na CD. Příručka je tvořena od nejdůležitějších elementů až po doplňky, které by se mohly hodit při vytváření krátkých dokumentů. Na názorných příkladech v podobě vytvoření šablony XSLT s formátovacími objekty pro vytvoření elektronické faktury ve formátu PDF a vytvoření šablony pro pozvánku jsem se snažil ukázat největší sílu XSL-FO a poukázat na některé nedostatky jak formátovacích objektů, tak XSL-FO procesorů. V práci jsem popsal kompletní nastavení nejznámějších a nejpoužívanějších XSLT a XSL-FO procesorů. Celá práce je napsána v XML a pomocí upravené šablony, kterou vytvořil pan Ing. Pavel Tyl, převedena do formátu PDF a XHTML.

XSL-FO je bezesporu výborný nástroj pro zpracování XML dokumentů do grafické podoby. Bohužel XSL-FO 1.1 standard je v době, kdy jsem psal tuto práci, zastaralý. Velmi brzo se dočkáme nového standardu XSL-FO 2.0, který přinese nové, moderní možnosti rozvržení XML dokumentů. Budoucnost tohoto standardu bude hodně záviset na podpoře zpracování formátovacích objektů XSL-FO procesory.

Příloha A

XSL-FO příručka

Jak už jsme si říkali, XSL-FO je mnohem rozsáhlejší a složitější než XSLT, které původně vzniklo za účelem práce s formátovacími objekty. Nejnovější specifikace XSL-FO 1.1, která byla přijata 6. prosince 2006 konsorciem W3C obsahuje 81 formátovacích objektů, které dále rozšiřuje 281 atributů. Máme tedy nepřeberné množství možností k rozvržení a naformátování obsahu stránek. Bohužel jsme částečně omezováni formátovacími nástroji, které některé elementy či atributy nepodporují.

Napsat kompletní příručku o formátovacích objektech by bylo velmi pracné a řekl bych i do jisté míry neefektivní pro ty, kteří s formátovacími objekty pracují poprvé. Co tedy lze od této příručky očekávat? V XSL-FO příručce si ukážeme základní a nejpoužívanější formátovací objekty a jejich vlastnosti. Na názorných příkladech si ukážeme např. rozvržení stránky, formátování textu, vytváření jednoduchých tabulek, seznamů, vkládání obrázků či vytváření poznámek pod čarou. **Na CD v sekci Example jsou uloženy některé .fo dokumenty, které používám v příručce jako názornou ukázku.**

1. Struktura XSL-FO dokumentu

Na začátek bychom se měli podívat, jak vlastně vypadá základní kostra FO dokumentu.

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

<fo:layout-master-set>
  <fo:simple-page-master master-name="A4">
    <!--Předloha -->
  </fo:simple-page-master>
</fo:layout-master-set>

<fo:page-sequence master-reference="A4">
  <!--Obsah stránek-->
</fo:page-sequence>
</fo:root>
```

Na první pohled je zřejmé, že se jedná o XML dokument, musí tedy vždy začínat XML deklarací:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Prvním formátovacím elementem je *<fo:root>*. Je to kořenový element XSL-FO dokumentu. Součástí kořenového elementu je deklarování jmenného prostoru XSL-FO. Tento element automaticky deklaruje prefix jmenného prostoru ”fo”:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <!-- XSL-FO dokument -->
</fo:root>
```

Potomky *<fo:root>* jsou jeden element *<fo:layout-master-set>* a posloupnost jednoho či více elementů *<fo:page-sequence>*. Formátovací objekt *<fo:layout-master-set>* obsahuje všechny předlohy použité

v dokumentu (předlohy posloupností stránek, předlohy stránek a předlohy oblastí):

```
<fo:layout-master-set>
  <!--Předlohy stránek -->
</fo:layout-master-set>
```

Element <*fo:simple-page-master*> je předloha stránky, která definuje skutečné rozvržení stránky a její geometrii. Každá předloha stránky musí mít jedinečný název (master-name):

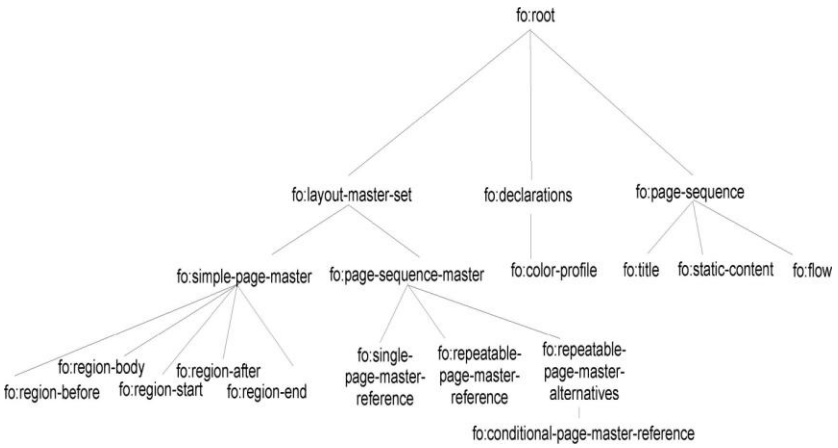
```
<fo:simple-page-master master-name="A4">
  <!--Předloha stránky-->
</fo:simple-page-master>
```

Element <*fo:page-sequence*> je posloupnost stránek. Co si pod tímto pojmem představit? Je to vlastně sada stránek, které sdílejí stejnou charakteristiku. Například kapitola v knize. Každý objekt <*fo:page-sequence*> se odkazuje na objekt <*fo:page-sequence-master*>, ve kterém dochází ke skutečnému rozložení stránky.

```
<fo:page-sequence master-reference="A4">
  <!--Obsah stránky -->
</fo:page-sequence>
```

Pozor! Hodnota “A4” atributu *master reference* není předdefinovaný formát stránky. Je to pouze jméno, lze ho nahradit jakýmkoliv jiným výrazem např. ”Page”, ”Template”.

Na obrázku O2 je zobrazena celá stromová struktura XSL-FO dokumentu.



Obrázek O2: Stromová struktura XSL-FO dokumentu

Nejpoužívanější atributy *<fo:simple-page-master>* jsou uvedeny v tabulce T1:

Tabulka T1: Nejpoužívanějších atributů elementu <fo:simple-page master>

Atribut	Hodnoty	Popis
margin-bottom	<margin-width> | inherit	nastavuje dolní okraj bloku
margin-left	<margin-width> | inherit	nastavuje levý okraj bloku
margin-right	<margin-width> | inherit	nastavuje pravý okraj bloku
margin-top	<margin-width> | inherit	nastavuje horní okraj bloku
master-name	<name>	nastavuje nebo vybírá předlohu
page-width	auto | indefinite | <length> |inherit	nastavuje šířku stránky
page-height	auto | indefinite | <length> |inherit	nastavuje výšku stránky

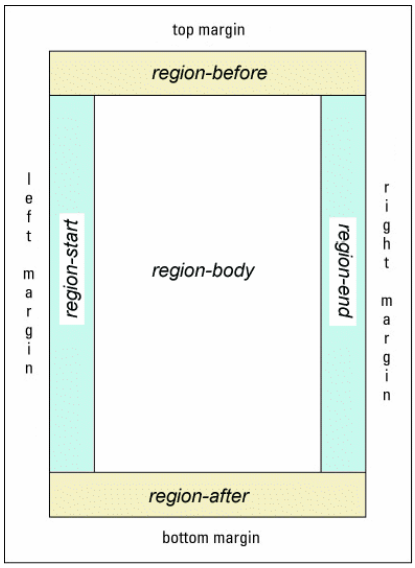
2. Tvorba XSL-FO oblastí

Ve specifikaci XSL-FO 1.1 má předloha stránky pět oblastí. Těmto oblastem se říká regiony. Centrální oblast odpovídá tělu stránky a je označována jako *region-body*. Horní část stránky neboli záhlaví je označována jako *region-before.* Dolní část stránky neboli pata je označována jako *region-after*. Levá část stránky se nazývá počáteční oblast neboli *region-start* a pravá část je nazývána *region-end.* Následující ukázka pochází ze souboru *Page_layout.fo*.

Příklad P1: Ukázka rozvržení stránky

```
<fo:layout-master-set>
  <fo:simple-page-master master-name="page" page-height="297mm"
                                                  page-width="210mm">
   <fo:region-body margin="2cm" background-color="yellow"
                                          border="solid thick orange"/>
   <fo:region-before extent="2cm" background-color="lightblue"
                                            border="solid thick blue"/>
   <fo:region-after extent="2cm" background-color="lightblue"
                                          border="solid thick blue"/>
   <fo:region-start extent="2cm" background-color="lightgreen"
                                          border="solid thick green"/>
    <fo:region-end extent="2cm" background-color="lightgreen"
                                          border="solid thick green"/>
  </fo:simple-page-master>
</fo:layout-master-set>
```

V elementu <*fo:simple-page-master master*> jsem nastavil velikost stránky, která odpovídá velikosti formátu A4. Aby bylo na příkladu dobře vidět rozvržení stránky, každý region má jinou barvu. Po transformaci FO dokumentu do PDF dostaneme podobný výsledek, jaký je vidět na obrázku O2.



Obrázek O2: Rozvržení stránky pomocí XSL-FO

Pro úpravu regionů existuje mnoho atributů, které slouží pro nastavení ohraničení, výplně a pozadí. V následující tabulce jsou nejčastěji používané atributy elementu *<fo:region-(body,before,after,start,end)>*.

T2: Atributy elementu <fo:region-(body,before,after,start,end)>

Atribut	Hodnoty	Popis
margin	<margin-width> | inherit	nastavuje všechny margin
extent	<length> | <percentage> | inherit	určuje šířku start a end region nebo výšku předchozí nebo následující oblasti
background-color	<color>|transparent| inherit	určuje barvu pozadí vybraného elementu
background-image	<uri-specification>|none| inherit	určuje obrázek pozadí vybraného elementu
border	[<border-width> || <borderstyle>|| [<color> | transparent]] | inherit	nastavení stejné výšky, barvy a stylu pro všechna ohraničení daného bloku

3. Tvorba XSL-FO toků

Označení objektů toku je odvozeno od toho, že text jimi „protéká“ a je uspořádán tak, aby jej zobrazovací software mohl na příslušné stránce dokonale zobrazit. Obsah stránky ošetřují právě objekty toku.

Existují dva typy objektů toku <*fo: static-content*> a <*fo:flow*>. Element <*fo: static-content*> se používá pro vkládání textu do záhlaví nebo pat, jde tedy o text, který se zobrazuje na každé stránce. Objekt <*fo:flow*> obsahuje text, který tvoří tělo dokumentu. Ukázka ze souboru *Flow_objects.fo* nám názorně demonstruje správné používání těchto formátovacích objektů.

Příklad P2: Ukázka použití objektů toku

```
<fo:page-sequence master-reference="page">
 <fo:static-content flow-name="xsl-region-before">

  <fo:block font-size="24pt" text-align="center" margin="5pt">
   Pokus o fo:static-content</fo:block>
 </fo:static-content>
 <fo:static-content flow-name="xsl-region-after">
  <fo:block font-size="24pt" text-align="right">
   Page <fo:page-number/> of <fo:page-number-citation ref-
id="last-page"/>
  </fo:block>
 </fo:static-content>
  <fo:flow flow-name="xsl-region-body">
   <fo:block font-size="24pt" text-align="center" font-
weight="bold">
     <!--Text-->
   </fo:block>
  <fo:block id="last-page"/>
 </fo:flow>
</fo:page-sequence>
```

U elementů <*fo:flow*> a <*fo: static-content*> se používá atribut *flow-name,* který určuje název toku.

4. Tvorba obsahu a formátování na úrovni bloku

Bloky jsou nejdůležitější částí XSL-FO. Používají se k vytváření obdélníkových zobrazovacích ploch, odlišujících se od zbývajících ploch dokumentu. Formátovací objekt <*fo:block*> se používá k formátování odstavců, nadpisů, titulů či popisu obrázků a tabulek. Nejlepší bude si opět ukázat názorný příklad ze souboru *Block_format.fo*.

Příklad P3a: Ukázka použití vlastností elementu <fo:block>

```
<fo:block font-size="36pt" font-weight="bold" text-align="center"
space-before="2in"
  space-after="1in" start-indent="1.5in" end-indent="1.5in" border-
color="red"
  border-style="solid" border-width="1mm" padding="1cm"
background-color="#aaaaff"><!--Text-->
</fo:block>
```

U elementu <*fo:block*> je možné použít více než 100 atributů. To zaručuje nepřeberné množství formátování bloku. Např. máme možnosti použít vlastnosti pro nastavení ohraničení, výplně, pozadí, určit vlastnosti písma či obsluhu dělení slov, nechybí ani vlastnosti okrajů bloků. Nejčastěji používanými atributy jsou *font-family, font-size, line-height*. Do tabulky T3 jsem se pokusil vybrat některé další nejpoužívanější atributy tohoto elementu.

Tabulka T3a: nejpoužívanějšími atributy v elementu <fo:block>

Atribut	Hodnoty	Popis
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit	nastavuje tučnost písma
border-width	<border-style>{1,4} | inherit	nastavuje šířku ohraničení
border-style	<border-style>{1,4} | inherit	ohraničení bloku
border-color	[<color> | transparent]{1,4} | inherit	určuje barvu všech 4 stran ohraničení
text-align	start | center | end | justify | inside | outside | left | right | <string> | inherit	zarovnání odstavce
text-align-last	relative | start | center | end | justify | inside | outside | left | right | inherit	zarovnání posledního řádku odstavce
text-indent	<length> | <percentage> |inherit	odsazení prvního řádku
text-decoration	none | [[underline | no-underline] || [overline | nooverline] || [line-through | no-line-through] || [blink | no-blink]] | inherit	ozdoby přidávané k textu
break-before	auto | column | page | even- page | odd-page | inherit	zalomení stránky před odstavcem
break-after	auto | column | page | even- page | odd-page | inherit	zalomení stránky za odstavcem
wrap-option	no-wrap | wrap | inherit	definuje zalomení slov
keep-togethet	<keep> | inherit	zakázání stránkového zlomu odstavce
keep-with-next	<keep> | inherit	zakázání zlomu mezi odstavcem a následujícím blokem

Tabulka T3b: nejpoužívanějšími atributy v elementu <fo:block>

Atribut	Hodnoty	Popis
letter-spacing	normal | <length> | <space> | inherit	velikost mezery mezi písmeny
word-spacing	normal | <length> | <space> | inherit	velikost mezery mezi slovy
hyphenate	false | true | inherit	dělení slov

Dalšími důležitými atributy jsou *padding, margin a background*.

Pro formátování části textu v bloku se používá element <*fo:inline*>. U vybrané části můžeme nastavit barvu pozadí, podtrhnout text, vytvořit rámeček a máme mnoho dalších možností. Podobnou vlastnost jako element <*fo:inline*> má element <*fo:character*>. Ten umožnuje manipulaci s jednotlivými znaky.

Příklad P3b: Formátování textu v bloku

```
<fo:block font="12pt Arial">
  Formátování <fo:inline font-weight="bold" color="red" text-
decoration="underline"
  background="yellow"> textu </fo:inline> je snadné.
</fo:block>
```

Pozor! FOP atribut *background* nepodporuje, je třeba použít *background-color*.

5. Tvorba tabulek

Tvorba tabulek pomocí formátovacích objektů je velmi podobná tabulkám v HTML. Jsou to obdélníkové mřížky řádků a sloupců, tvořené buňkami. Příklad P4 ukazuje část vytvořené tabulky. Celou tabulku si můžete prohlédnout v souboru *Table.fo*.

Příklad P4: Ukázka části tabulky pomocí XSL-FO

```
<fo:table table-layout="fixed" border-width="1mm" border-
style="solid">
  <fo:table-column column-width="3in"/>
  <fo:table-column column-width="3in"/>
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell padding="1mm" border-width="1mm" border-
style="solid">
<fo:block>Obsah</fo:block>
</fo:table-cell>
     <fo:table-cell padding="1mm" border-width="1mm" border-
style="solid">
       <fo:block>Obsah</fo:block>
      </fo:table-cell>
    </fo:table-row>
    <fo:table-row>
      <fo:table-cell padding="1mm" border-width="1mm" border-
style="solid">
       <fo:block>Obsah</fo:block>
       </fo:table-cell>
 <fo:table-cell padding="1mm" border-width="1mm" border-
style="solid">
       <fo:block>Obsah</fo:block>
      </fo:table-cell>
    </fo:table-row>
  </fo:table-body>
</fo:table>
```

Jak je z příkladu patrné, tabulka se skládá z několika formátovacích objektů. Základem je element <*fo:table*>, ten vytváří novou tabulku, která se skládá z nepovinného záhlaví, paty a jednoho nebo více těl buňky. Na vytvoření sloupců tabulky se používá element <*fo:table-column*>. K vytvoření těla tabulky slouží element <*fo:table-body*>. Tento element obsahuje elementy

<*fo:table-row*>, pomocí kterých se vytváří řádky tabulky. Elementy <*fo:table-row*> mohou obsahovat elementy <*fo:table-cell*>, v nichž jsou uložena zdrojová data. Dalšími elementy, které můžeme použít v tabulce, jsou <*fo:table-and-caption*>, <*fo:table-caption*>, <*fo:table-header*>, <*fo:table-footer*>. V tabulce T4 jsou nejpoužívanější atributy pro formátování tabulek.

Tabulka T4: nejpoužívanějšími atributy pro tabulku

Atribut	Hodnoty	Popis
table-layout	auto | fixed | inherit	algoritmus, který bude použit k rozvržení buněk, řádků a sloupců
column-width	<length> | <percentage>	určuje šířku sloupce
number-columns-spanned	<number>	počet sloučených buněk
number-rows-spanned	<number>	počet buněk sloučených vertikálně

6. Tvorba XSL-FO seznamu

Seznamy XSL-FO zobrazují, podobně jako seznamy v HTML, svislý seznam položek. Pro vytvoření seznamu potřebujeme následující elementy: *<fo:list-block>, <fo:list-item>, <fo:list-item-label>* a *<fo:list-item-body>*.

Základem je element *<fo:list-block>*, ten vytváří nový seznam a obsahuje elementy *<fo:list-item>*, které obsahují popisek položky a její tělo. V seznamu musí být alespoň jedna položka. K tomu, abychom mohli vytvořit popisek položky v seznamu, potřebujeme použít element *<fo:list-item-label>*. Tímto elementem očíslujete položky nebo vložíte odrážku. Tělo položky seznamu se vkládá do elementu *<fo:list-item-body>*. Data, která chceme dále formátovat, se vkládají do elementu *<fo:block>*. Pro lepší představu, jak takový seznam vypadá, se podívejme na část seznamu ze souboru *List.fo*. V tabulce T5 jsou nejpoužívanější atributy pro práci se seznamy.

Příklad P5: Ukázka části seznamu

```
<fo:list-block provisional-distance-between-starts=".5in" space-
after="12pt"
                start-indent="1in" font-family="Times" line-
height=".5in">
  <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>Nepokradeš</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>Nesesmylníš</fo:block>
    </fo:list-item-body>
  </fo:list-item>
```

Tabulka T5: nejpoužívanější atributy pro seznamy

Atribut	Hodnoty	Popis
provisional-distance-between-starts	<length>|<percentage>| inherit	vzdálenost mezi počátkem odsazení popisku a počátkem odsazení datového obsahu
provisional-label-separation	<length>|<percentage>| inherit	vzdálenost mezi koncem popisku a počátkem datového těla
start-indent	<length>|<percentage>| inherit	odsazení těla seznamu
end-indent	<length>|<percentage>| inherit	odsazení návěští zprava

7. Vkládání obrázků a SVG

Vkládání obrázků do dokumentu je poměrně snadné. Nesmíme si však zapomenout zjistit, které grafické formáty FO procesor podporuje. Stejně jako u HTML, tak i u formátovacích objektů, lze upravovat velikost vkládaných obrázků. K tomu jsou určeny atributy *content-height*, *content-width* a *scaling*. Vkládání obrázků se provádí pomocí elementu <*fo:external-graphic*>**.** Příklad P6 a ukazuje vložení obrázku do dokumentu.

Příklad P6a: Ukázka vložení obrázku

```
<fo:block text-align="center">
  <fo:external-graphic src="url('skica kovboj.jpg')"
    content-width="480px" content-height="320px"/>
</fo:block>
```

Vkládat SVG grafiku můžeme dvojím způsobem. Buď si jí vyrobíme v nějakém SVG editor (Inkscape, Sketsa) a obrázek vložíme pomocí <*fo:external-graphic*> jako v předchozím příkladu, nebo můžeme zapsat zdrojový kód SVG přímo do formátovacích objektů s použitím elementu <*fo:instream-foreign-object*>. Na příkladu P6b je vidět, že potomek elementu <*fo:instream-foreign-object* > má jiný jmenný prostor než XSL-FO.

Příklad P6b: Ukázka vložení zdrojového kódu SVG

```
<fo:instream-foreign-object content-height="2.2in">
  <svg xmlns="http://www.w3.org/2000/svg" width="480"
height="280">
    <linearGradient id="Grad1" gradientUnits="objectBoundingBox"
      x1="0" y1="0" x2="1" y2="1">
        <stop stop-color="rgb(238,130,238)" offset="0"/>
 <stop stop-color="blue" offset="0.2"/>
 <stop stop-color="lime" offset="0.4"/>
 <stop stop-color="yellow" offset="0.6"/>
 <stop stop-color="rgb(255,165,0)" offset="0.8"/>
       <stop stop-color="red" offset="1"/>
```

```
      </linearGradient>
      <rect x="20" y="20" width="440" height="80"
fill="url(#Grad1)"/>
      <text font-family="Arial" font-size="14" x="20" y="130">
  Multi-color linear gradient.
      </text>
  </svg>
</fo:instream-foreign-object>
```

Stejně jako tomu bylo u předchozích příkladů, celé části příkladů jsou uloženy v souboru *Graphic.fo*. V tabulce T6 jsou nejpoužívanější atributy pro práci s obrázky.

Tabulka T6: nejpoužívanější atributy elementu <fo:external-graphic>

Atribut	Hodnoty	Popis
content-width	auto | scale-to-fit | scale-down-to-fit | scale-up-to-fit |<length> | <percentage> |inherit	nastavuje velikost šířky obrázku
content-height	auto | scale-to-fit | scale-down-to-fit | scale-up-to-fit |<length> | <percentage> |inherit	nastavuje velikost výšky obrázku
scaling	uniform | non-uniform | inherit	
text-align	start | center | end | justify | inside | outside | left | right | <string> | inherit	určení horizontálního zarovnání obrázku v dostupné ploše
displey-align	auto | before | center | after | inherit	určení vertikálního zarovnání obrázku v dostupné ploše
clip	<shape> | auto | inherit	určení oříznutí obrázku
src	<uri specification> | inherit	url adrea obrázku

8. Obtékání textu a obrázku

Pro „plovoucí“ objekty (objekty, které obtékají ostatní objekty), 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. Následující příklad P7 ukazuje obtékání znaku textem a obtékání obrázku textem. Celý příklad naleznete v souboru *Float.fo*.

Příklad P7: Ukázka obtékání pomocí elementu <fo:float>

```
<fo:block text-align="justify">
  <fo:float float="start">
    <fo:block font-size="72pt" line-height="1" margin="5pt" text-
depth="0">L</fo:block>
  </fo:float>
  <!--Text-->
</fo:block>

<fo:float float="left">
  <fo:block margin-left="1cm" margin-top="30pt">
    <fo:external-graphic src="url('grafika.jpg')" content-
height="120px"
      content-width="160px" />
  </fo:block>
</fo:float>
  <fo:block font-family="Times" text-align="justify"  font-size="12pt"
margin-top="20pt">
  <!--Text-->
  </fo:block>
```

Nejpoužívanějším atributem elementu <*fo:float*> je atribut *float,* který určuje směr obtékání.

Pozor! FO procesor FOP element <*fo:float*> nepodporuje.

9. Absolutní pozicování pomocí kontejnerů

Občas je třeba určité položky rozmístit na stránce podle přesných pravidel. V XSL-FO můžeme definovat zobrazení jednotlivých položek pomocí relativních nebo absolutních souřadnic. Těchto vlastností můžeme dosáhnout pomocí elementu <*fo:block-container*>. Tento element podporuje vlastnosti pro definici absolutního umístění. Tuto vlastnost element <*fo:block*> neumí. Na příkladu P8 je vidět nastavení souřadnic, kde daný text bude umístěn na stránce a v tabulce T7 jsou nejpoužívanější atributy <*fo:block-container*>. Celý zdrojový příklad najdete v souboru Absolute_Position.fo.

Příklad P8: Ukázka absolutního pozicování textu

```
<fo:block-container  position="absolute" top="10pt"
   left="30pt" height="14pt" width="100%">
  <fo:block font="72pt Arial" color="silver">Under</fo:block>
</fo:block-container>
<fo:block>
 <fo:block>
  <!--Text-->
 </fo:block>
 <fo:block-container position="absolute" top="20pt"
    left="40pt" height="14pt" width="100%">
  <fo:block font="72pt Arial" color="red">Over</fo:block>
 </fo:block-container>
</fo:block>
```

Tabulka T7: nejpoužívanější atributy <*fo:block-container*>

Atribut	Hodnoty	Popis
absolute-position	auto | absolute | fixed | inherit	určuje, zda-li je umístění položky absolutní
writing-mode	lr-tb | rl-tb | tb-rl | tb-lr | bt-lr | bt-rl | lr-bt | rl-bt | lr-alternating-rl-bt | lr-alternatingrl-tb | lr-inverting-rl-bt | lr-inverting-rl-tb | tb-lr-in-lrpairs| lr | rl | tb | inherit	mění orientaci textu vnořených blokových elementů
top, left, right, bottom	<length> | <percentage> | auto | inherit	určuje vzdálenosti mezi horním, levým, pravým či spodním ohraničením a okrajem obsaženého bloku
reference-orientation	0 | 90 | 180 | 270 | -90 | -180 | -270 | inherit	určuje orientaci obsahu

Pozor! FOP umí absolutní pozicování pouze částečně. U starších verzí (starší než 0.94) se pro správné zobrazování píše *position="absolute"*, jelikož *absolute-position="absolute"* není v těchto verzích implementována.

10. Tvorba odkazů, obsahu a číslování stránek

Při tvorbě odkazů rozlišujeme dva druhy odkazů: externí a interní. Externí odkazy jsou odkazy směřující mimo dokument. Jedná se vlastně o jednoduchý hypertextový odkaz směřující například na některou webovou stránku. Interní odkazy jsou odkazy směřující z jednoho místa v dokumentu na jiné místo ve stejném dokumentu.

Externí odkazy se tvoří pomocí elementu <*fo:basic-link*> s povinným atributem *external-destination*. Na příkladu P9a je externí odkaz směřující na seznam.cz.

Příklad P9a: Ukázka externího odkazu

```
<fo:block>
   Hypertextový odkaz na:
   <fo:basic-link  color="blue" text-decoration="underline"
                  external-destination="url(http://www.seznam.cz)"
>
      Seznam.cz
   </fo:basic-link>
</fo:block>
```

Nejčastěji používanými atributy pro externí linky jsou *color* a *text-decoration*.

Interní odkazy se tvoří pomocí elementu <fo:basic-link> s povinným atributem *internal-destination*, který obsahuje hodnotu *id* odkazovaného atributu. Příklad P9b demonstruje vytvoření interního odkazu.

Příklad P9b: Ukázka interního odkazu

```
<fo:basic-link internal-destination="{generate-id(.)}">
 <xsl:value-of select="."/>
</fo:basic-link>
```

Pro tvorbu obsahu dokumentu se využívá element<*fo:leader*> s atributem *leader-pattern*. Pokud nastavíme hodnotu na *leader-pattern="dots"*, mezi odkazem na příslušnou stránku a číslem stránky se vytvoří řada teček. Pro správné seřazení čísel stránek pod sebe použijeme atribut *text-align-last="justify"*. Dá se tedy říci, že obsah dokumentu se tvoří více elementy: interními odkazy, výplněmi a číslováním stránek. V příkladu P9c je ukázka jedné řádky obsahu, pokud byste si chtěli prohlédnout celý obsah, podívejte se do souboru *Leader.fo*. Tabulka T8 obsahuje nejpoužívanější atributy elementu <*fo:leader*>.

Příklad P9c: Ukázka části obsahu

```
<fo:block text-align-last="justify">
  <fo:basic-link color="blue" internal-
destination="chapter1">Úvod</fo:basic-link>
  <fo:inline keep-together.within-line="always">
    <fo:leader leader-pattern="dots" />
    <fo:page-number-citation ref-id="chapter1" />
  </fo:inline>
</fo:block>
 . . . . . .
<fo:block id="chapter1" break-before="page" font-size="18pt">
   Úvod
</fo:block>
<fo:block>
   <!--Text-->
</fo:block>
```

Tabulka T8: nejpoužívanější atributy elementu <fo:leader>

Atribut	Hodnoty	Popis
leader-pattern	space | rule | dots | use-content | inherit	nastavení vzoru obsahu
leader-length	<length-range> | <percentage> | inherit	nastavuje délku obsahu
leader-pattern-width	use-font-metrics|<length>| <percentage> | inherit	určuje šířku vzoru obsahu
leader-alignment	none | reference-area | page |inherit	určuje zarovnání obsahu
rule-style	none | dotted | dashed | solid | double | groove | ridge | inherit	druh výplně, pokud se používá čára
rule-thickness	<length>	síla čáry

Číslování stránek je stejně jako obsah důležitá část při vytváření plnohodnotných dokumentů. Prázdný element *<fo:page-number>* se používá 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>*. Dobře patrné uplatnění je vidět na předchozím příkladu, kde se tento element využívá pro získávání čísla stránky v obsahu. Ale jak se vlastně vytváří čísla stránek v zápatí? Následující příklad P10 nám to objasní. Pokud byste chtěli vidět celý zdrojový kód s použitím číslování stránek, je uložen v souboru *Page_number.fo*.

Příklad P10: Ukázka vložení číslování stránek v zápatí

```
<fo:static-content flow-name="zapati">
  <fo:block text-align="right" border-top="1pt dashed silver">
        Strana
        <fo:page-number />
        z
        <fo:page-number-citation  ref-id="theEnd" />
  </fo:block>
</fo:static-content>
<fo:flow flow-name="xsl-region-body">
  <fo:block>
        <!--Text-->
  </fo:block>
   <fo:block break-before="page">
        <!--Text-->
   </fo:block>

   . . . . . .
<fo:block id="theEnd" />
```

Jak je na příkladu vidět, pro vytváření čísel stránek jsme použili oba dva elementy <*fo:page-number*> a <*fo:page-number-citation*>. Číslo vkládáme do zápatí (region-after), tudíž musíme použít element <*fo:static-content*>, jak jsme již zmiňovali v části pro tvorbu XSL-FO toků. Poslední číslo stránky získáme pomocí atributu *ref-id* s hodnotou *"theEnd"*.

Pro nastavení počátečního čísla stránky je možné použít atribut *initial-page-number* elementu <*fo:page-sequence*>. Tento atribut umožňuje například formátovat jednotlivé kapitoly knihy samostatně. Přesto budou stránky číslované správně.

11. Poznámky pod čarou, text v záhlaví a zápatí

Poznámky pod čarou se tvoří pomocí elementu <*fo:footnote*>. Jsou označovány za mimořádkové, jelikož přidávají text na konec stránky. Tělo poznámky je definováno elementem <*fo:footnote-body*>. Příklad P11 znázorňuje jednu poznámku pod čarou. Celý zdrojový kód poznámek pod čarou je uložen v souboru *Footnotes2.fo*. V souboru *Footnotes1.fo* je druhá verze poznámek pod čarou, kde jsou také použity atributy *column-gap* a *column-count* elementu <*fo:region-body*>, které se používají pro novinové sloupce.

Příklad P11: Ukázka poznámek pod čarou

```
<fo:static-content flow-name="xsl-footnote-separator">
  <fo:block>
   <fo:leader leader-pattern="rule" leader-length="100%"/>
  </fo:block>
 </fo:static-content>
 <fo:flow flow-name="xsl-region-body">
  <fo:block>
   <!--Text-->
   <fo:footnote>
    <fo:inline font-size="8pt" alignment-
baseline="hanging">1</fo:inline>
    <fo:footnote-body>
     <fo:block font-size="8pt">1.pokus o první poznámku pod čarou
<fo:block>
    </fo:footnote-body>
   </fo:footnote>
  </fo:block>
 </fo:flow>
```

Všimněme si hodnoty *"xsl-footnote-separator"* atributu *flow-name*. Stejně jako regiony i poznámky pod čarou mají vyhrazený XSL název toku. Do této oblasti je vkládán oddělovač (separátor), který odděluje poznámky od obsahu v *region-body*. Čáru můžeme vytvořit buď ručně, tak jako tomu je v souboru

Footnotes1.fo, nebo pomocí atributů *leader-pattern* a *leader-length* elementu <*fo:leader*>.

Například pro zobrazování kapitol v záhlaví či zápatí se používají atributy <*fo:marker*> a <*fo:retrieve-marker*>. Nejdříve pomocí elementu <*fo:marker*> vybereme kandidáty, které chceme zobrazit v záhlaví či zápatí. Poté použijeme element <*fo:retrieve-marker*> k načtení označené části či 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 markerclass-name (na elementu <*fo:marker*>), *retrieve-class-name*, *retrieve-position* a *retriever-boundary* (na elementu <*fo:retrieve-marker*>). Na následujícím příkladu P12 je ukázka vybrání části textu do záhlaví stránky. V tabulce T9 jsou nejpoužívanější atributy elementu <*fo:retrieve-marker*>. Celý zdrojový kód najdete v souboru *Markers.fo*.

Příklad P12: Ukázka použití výběru části textu do záhlaví

```
<fo:static-content flow-name="xsl-region-before">
  <fo:block>
    <fo:retrieve-marker retrieve-class-name="title"
                        retrieve-position="first-starting-within-page"
                        retrieve-boundary="page" />
  </fo:block>
</fo:static-content>
<fo:flow flow-name="xsl-region-body" font="10pt Verdana">
  <fo:block>
          <fo:block font="18pt 'Arial Narrow'">
            <fo:marker marker-class-name="title" >
              Title of Chapter 1
            </fo:marker>
              Chapter 1
          </fo:block>
            <!--Text-->
        </fo:block>
        <fo:block break-before="page">
          <fo:block font="18pt 'Arial Narrow'">
```

```
    <fo:marker marker-class-name="title" >
      Title of Chapter 2
    </fo:marker>
      Chapter 2
  </fo:block>
    <!--Text-->
 </fo:block>
</fo:flow>
```

Tabulka T9: nejpoužívanější atributy elementu <fo:retrieve-marker>

Atribut	Hodnoty	Popis
retrieve-class-name	<name>	pomocí shodných unikátních jmen se propojí fo:marker s fo:retrieve-marker
retrieve-position	first-starting-within-page | first-including-carryover | last-starting-within-page | last-ending-within-page	umožňuje vybrání určité oblasti, která byla označena elementem fo:marker
retrieve-boundary	page | page-sequence | document	určení oblasti výběru

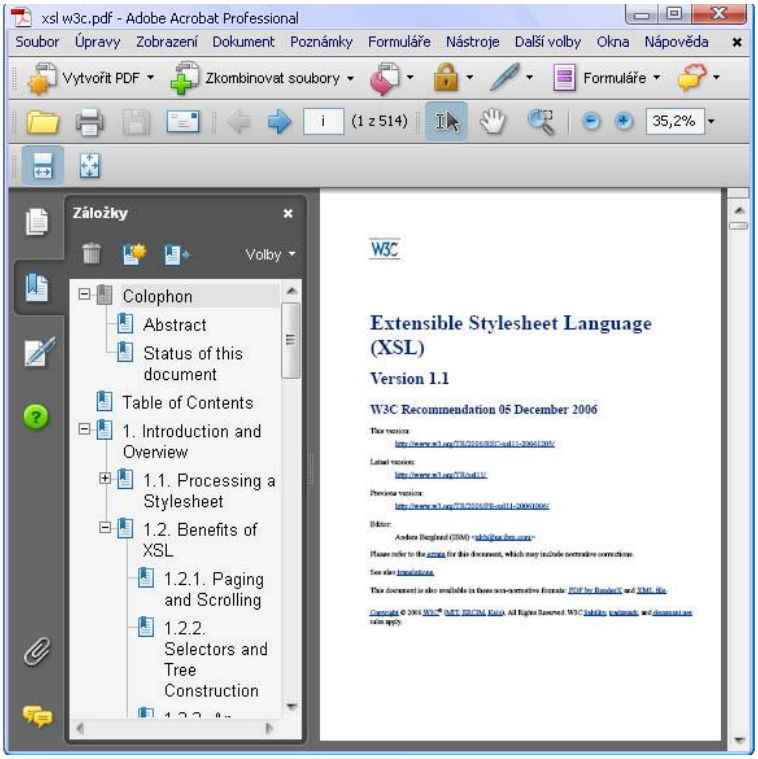
12. Tvorba stromu dokumentu

Stromová struktura dokumentu se tvoří pomocí tří elementů: <*fo:bookmark-tree*>, <*fo:bookmark*> a <*fo:bookmark-title*>. Jak je z příkladu P13 patrné, element <*fo:bookmark-tree*> vytváří novou stromovou strukturu. Tato struktura se vkládá mezi oblasti a toky. Zdrojový kód je uložen v souboru *Bookmarks.fo*.

Příklad P13: Ukázka stromové struktury

```
<fo:bookmark-tree >
  <fo:bookmark internal-destination="toc" >
    <fo:bookmark-title >Bookmarks Example</fo:bookmark-title>
    <fo:bookmark internal-destination="chapter1">
      <fo:bookmark-title>Úvod</fo:bookmark-title>
    </fo:bookmark>
    <fo:bookmark internal-destination="chapter2">
      <fo:bookmark-title>XSLT</fo:bookmark-title>
    </fo:bookmark>
  </fo:bookmark>
</fo:bookmark-tree>
<fo:page-sequence master-reference="LetterPage" font="10pt
Arial">
  <fo:flow flow-name="PageBody" font-family="Arial Narrow" font-
size="10pt">
    <fo:block id="toc">Stromová struktura</fo:block>
    <fo:block text-align-last="justify">
      <fo:basic-link color="blue" internal-
destination="chapter1">Úvod</fo:basic-link>
      <fo:inline keep-together.within-line="always">
        <fo:leader leader-pattern="dots" />
        <fo:page-number-citation ref-id="chapter1" />
      </fo:inline>
  </fo:block>
```

Element <*fo:bookmark*> využívá atribut *internal-destination*, který určuje cílový objekt toku. Dochází tak k propojení stromové struktury s obsahem dokumentu. Element <*fo:bookmark-title*> vytváří popisky stromové struktury. Obrázek O3 znázorňuje stromovou strukturu dokumentu v pdf.



Obrázek O3: Stromová struktura dokumentu v pdf

Příloha B

Obsah CD

- Složka Bakalářská práce (bakalářka v pdf, Šablona XSL-FO, Šablona XHTML)
- Složka Podpora software (Java a .Net platforma)
- Složka XSLT procesory (Saxon, Xalan, xsltproc, XT)
- Složka XSL-FO procesory (FOP, AHF Formarter v5, XEP 4.15, XMLmind 4.3)
- Složka Faktura (XML faktura, ŠablonaFaktura, FOPfaktura, XEPfaktura, AHF faktura, fakturaFO)
- Složka Pozvánka (XML poznamka, Šablona Pozvanka, pozvankaFOP, pozvankaXEP, pozvankaAH, pozvanakODF, pozvankaOOXML, pozvankaFO)
- Složka Příklady (zdrojové příklady použíté v příručce XSL-FO)

Seznam zkratek

AWT	Abstract Windows Toolkit
BMP	BitMap
BSF	Bean Scripting Framework
EMF	Enhanced Windows Metafile Format
EXSLT	Extensions for Extensible Specification Language
FO	Formattinf Objects
GIF	Graphics Interchange Format
JPEG	Joint Experts Group
MathML	Mathematical Markup Language
MIF	Maker Interchange Format
PDF	Portable Document Format
PNG	Portable Network Graphics
PS	PostScript
RTF	Rich Text Format
SVG	Scalable Vector Graphics
TIFF	Tagged Image File Format
TTF	Treu Type Font
W3C	World Wide Web Consortium
WMF	Windows Metafile Format
XML	Extensible Markup Language

XPath	XML Path Language
XQuery	XML Query Language
XSL	Extensible Stylesheet Language
XSL-FO	Extensible Stylesheet Language-Formatting Objects
XSLT	Extensible Stylesheet Language Transformations

Seznam použité literatury

[1] BODNÁR, P. (2008). *Konverze formátovacích objektů do ODF*, Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačního a znalostního inženýrství, 81 s..

[2] BRADLEY, N.: *XML, kompletní průvodce*. Praha: Grada Publishing, 2000. ISBN 80-7169-949-7.

[3] HAROLD, E. R., MEANS, W. S.: *XML v kostce: Pohotová referenční příručka*. Praha: Computer Press, 2002. ISBN 80-7226-712-4.

[4] HAROLD, E. R.: *XML Bible*. 2. vydání. New York: Hungry Minds, 2001. ISBN 07-6454-760-7.

[5] HOLZNER, S.: *Inside. XSLT*. Londýn: Macmillan Computer Publishing, 2001. ISBN 07-3571-136-4.

[6] HOLZNER, S.: *XSLT: Příručka internetového vývojáře*. Praha: Computer Press, 2002. ISBN 80-7226-600-4.

[7] MARCHAL, B.: *XML v příkladec*h. Praha: Computer Press, 2000. ISBN 80-7226-332-3.

[8] MLÝNKOVÁ, I. a kol.: *Technologie XML*. Praha: Nakladatelství Karolinum, 2006. ISBN 80-246-1272-0.

[9] PACHMAN, J. (2004). *Převod formátovacích objektů do formátu WordML*, Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačního a znalostního inženýrství, 73 s..

[10] TYL, P (2006). *Tvorba elektronických dokumentů na bázi XML*, Technická univerzita v Liberci, Fakulta mechatroniky a mezioborových inženýrských studií, 100 s..

Internetové zdroje

[11] KOSEK, J.: *Podpora češtiny pro FOP* [online]. 2002 – 2006 [cit. 2009-04-06]. Dostupné z WWW: <http://www.kosek.cz/sw/fop>.

[12] KOSEK, J.: *Vše o WWW* [online]. Prosinec 2009 [cit. 2009-04-06]. Dostupné z WWW: <http://www.kosek.cz>.

[13] SOUKUPOVÁ K.: *Elektronická fakturace, která ušetří čas i peníze, má zelenou* [online]. Listopad 2008 [cit. 2009-04-16]. Dostupné z WWW: <http://www.podnikatel.cz/clanky/elektronicka-fakturace-ma-zelenou>.

[14] STAYTON, B: *DocBook XSL (The Complete Guide)* [online]. Září 2007 [cit. 2009-03-1]. Dostupné z WWW: <http://www.sagehill.net/docbookxsl>.

[15] ŠTĚDROŇ, L.: *Jakou má elektronická fakturace oporu v zákoně?* [online]. Lupa.cz. 2004 [cit. 2009-04-17]. Dostupné z WWW: <http://www.lupa.cz/clanky/jakou-ma-elektronicka-fakturace-oporu-v-zakone>.

[16] *Apache FOP* [online]. Květen 2009 [cit. 2009-03-25]. Dostupné z WWW: <http://xmlgraphics.apache.org/fop>.

[17] ***Barcode4*** [online]. Březen 2009 [cit. 2009-04-2]. Dostupné z WWW: <http://barcode4j.sourceforge.net>.

[18] *Co s formátovacími objekty* [online]. [cit. 2009-03-11]. Dostupné z WWW:
<http://www.volny.cz/zampach/dbk/fo_pdf.html>.

[19] *Co s formátovacími objekty* [online]. 2008 [cit. 2009-04-23]. Dostupné z WWW:
< http://www.grafika.cz/art/sazba/renderx>.

[20] *Co s formátovacími objekty* [online]. 2008 [cit. 2009-04-26]. Dostupné z WWW:
<http://www.grafika.cz/art/sazba/xslformatter.html?vote=on&value=1>.

[21] ***Digitální podpis*** [online]. 2005 [cit. 2009-04-8]. Dostupné z WWW:
<http://www.alsoft.cz/products/Security/Security-Technology/Digital-Signature>.

[22] *FO2ODF Project Home Page* [online]. 2008 [cit. 2009-2-15]. Dostupné z WWW:
<http://fo2odf.sourceforge.net/>.

[23] *Interval* [online]. Leden 2010 [cit. 2009-02-16]. Dostupné z WWW:
<http://interval.cz>.

[24] *Nastavení systémové proměnné JAVA_HOME Converter* [online]. Duben 2004 [cit. 2009-02-17]. Dostupné z WWW:
<http://pichlik.sweb.cz/archive/2004_04_25_archive.html>.

[25] *OOXML versus ODF: lepší jeden příklad než tisíce stran dokumentace* [online]. Březen 2008 [cit. 2009-3-13]. Dostupné z WWW:
<http://jankuv.blog.lupa.cz/2008/03/15/ooxml-versus-odf-lepsi-jeden-priklad-nez-tisice-stran-dokumentace/>.

[26] *Open XML schválen mezinárodním standardem ISO* [online]. 2010 [cit. 2009-3-5]. Dostupné z WWW: <http://computerworld.cz/udalosti/open-xml-schvalen-mezinarodnim-standardem-iso-1436>.

[27] *Professional Formatting Solutions* [online]. Září 2009 [cit. 2009-03-21]. Dostupné z WWW: <http://www.antennahouse.com>.

[28] *Rozhovor: Jiří Kosek o OOXML, ODF a formátech obecně* [online].2009 [cit. 2009-2-15]. Dostupné z WWW: <http://www.abclinuxu.cz/clanky/rozhovory/rozhovor-jiri-kosek-o-ooxml-odf-a-formatech-obecne>.

[29] *SAXON: The XSLT and XQuery Processor* [online]. Říjen 2009 [cit. 2009-02-10]. Dostupné z WWW: <http://saxon.sourceforge.net/ >.

[30] *The Apache Xalan Project* [online]. Říjen 2009 [cit. 2009-03-21]. Dostupné z WWW: <http://xalan.apache.org>.

[31] *The coolest Interface to (Sub)Version Control* [online]. 2009 [cit. 2009-03-5]. Dostupné z WWW: <http://tortoisesvn.net>.

[32] *The XSLT C library for GNOME* [online]. [cit. 2009-03-21]. Dostupné z WWW: <http://xmlsoft.org/XSLT>.

[33] *W3C* [online]. 2010 [cit. 2009-3-1]. Dostupné z WWW: <http://www.w3.org>.

[34] *XML and XSL FO: Leading Products* [online]. 2009 [cit. 2009-04-23]. Dostupné z WWW: <http://www.renderx.com>.

[35] *XMLmind XSL-FO Converter* [online]. Prosinec 2009 [cit. 2009-03-23]. Dostupné z WWW: <http://www.xmlmind.com/foconverter/>.