DocBook

Stručný úvod do tvorby a zpracování dokumentů

Jiří Kosek

DocBook: Stručný úvod do tvorby a zpracování dokumentů

Jiří Kosek

Dokument je dostupný ve třech verzích – HTML[1], PDF[2] a HTML Help[3]. Celý byl připraven pomocí DocBooku a volně dostupných stylů. Převodu do PDF se ještě zúčastnil formátovač XEP, pro kompilaci HTML Helpu byl použit HTML Help Workshop.

[1] http://www.kosek.cz/xml/db/
[2] http://www.kosek.cz/xml/db/db.pdf
[3] http://www.kosek.cz/xml/db/db.chm

Obsah

Seznam obrázků

Seznam tabulek

Seznam příkladů

Předmluva

Dokument, který právě teď držíte v ruce nebo čtete z obrazovky počítače, je asi prvním ucelenějším dokumentem o DocBooku v češtině. Rozhodně není dokonalý a je na něm co zlepšovat. Rozhodl jsem se ho však uvolnit v současné podobě, která je i tak pro mnoho uživatelů DocBooku užitečná, protože nebudu mít hned tak čas ho nějak výrazně vylepšit. Pokud v něm narazíte na nějakou chybu, budu rád, když mne na ní upozorníte zasláním dopisu na <jirka@kosek.cz>. Neposílejte mi však dotazy typu, že vám nejde něco nainstalovat, nebo že něčemu nerozumíte – s největší pravděpodobností nebudu mít čas na ně odpovědět. Lepší je proto dotaz rovnou zaslat do nějaké diskusní skupiny – např. cz.comp.text.docbook[1], resp. <docbook@linux.cz> nebo <docbook-apps@lists.oasis-open.org>. Máte mnohem větší šanci, že vám na dotaz někdo odpoví.

Chcete-li si dále popsané ukázky vyzkoušet, musíte mít nainstalován DocBook a některé podpůrné nástroje. V linuxových distribucích je vše dostupné v několika instalačních balíčcích. Ostatní mohou nejdříve prostudovat kapitolu 9 – „*Instalace*“, kde je stručně popsána instalace nejdůležitějších komponent pro práci s docbookovými dokumenty. Pro snazší začátky si můžete stáhnout nějaký editor, který má podporu DocBooku včetně všech dalších nástrojů již přímo integrovánu – např. oXygen[2] nebo XXE[3].

Pokud byste o DocBooku chtěli vědět více, mohu vám doporučit školení[4].

Tento dokument popisuje DocBook verze 4.5 a možnosti jeho zpracování pomocí XSL a DSSSL stylů. Neočekávám jeho další vývoj a velké změny, časem zveřejním novou příručku, která se bude věnovat výhradně DocBook 5.0 a XSL stylům. Z tohoto důvodu jsem se zároveň rozhodl zveřejnit i části dokumentu, které do této doby byly vyhrazeny jen účastníkům školení.

Ale dost řečí, pusťte se do studia.

— *Jirka Kosek*, 30. října 2001

[1] news:cz.comp.text.docbook
[2] http://www.oxygenxml.com
[3] http://www.xmlmind.com/xmleditor/
[4] http://www.kosek.cz/skoleni.html

Kapitola 1. Úvod

Papírové knihy jsou krásné, ale nehodí se pro všechny oblasti použití. Pro mnoho aplikací je potřeba kromě vytištěné verze nějakého dokumentu získat i jeho elektronickou podobu. Ať už pro publikování na webu, na CD-ROMu nebo v intranetu. Na samotné přípravě tiskovin se počítače dnes podílejí ve velkém měřítku. Velká většina dnes produkovaných knih a časopisů je připravována na počítačích v DTP systémech. Ty nabízejí bohaté možnosti pro práci s grafikou a s formátováním dokumentu, ale obvykle nenabízejí vhodné nástroje pro vytvoření elektronických verzí dokumentů.

Stejné a mnohé další problémy musíme řešit při přípravě dokumentace. Následující seznam shrnuje dnes obvyklé požadavky na dokumentaci:

- generovat dokumentaci v mnoha formátech z jedné předlohy
  - tištěná podoba (PostScript, PDF)
  - on-line nápověda (HTMLHelp, info, JavaHelp)
  - webová podoba dokumentace (HTML stránky)
- plně využít schopnosti jednotlivých výstupních formátů (obsah, rejstřík, odkazy, fulltext apod.)
- podporovat časté změny v dokumentech
  - technologie se rychle vyvíjejí
  - proces generování jednotlivých výstupních formátů by měl být plně automatizován

V současné době existuje na trhu několik proprietárních systémů, které jsou schopné více či méně tyto požadavky splnit. Kromě použití již hotových a do jisté míry neflexibilních nástrojů můžeme použít pro tvorbu dokumentů formát XML. Nebudeme tak svázáni s nějakým proprietárním systémem a získáme větší možnosti, které mohou být zejména do začátku vykoupeny složitější a náročnější konfigurací celého systému pro tvorbu dokumentace.

Použít pro elektronické publikování řešení založené na XML se vyplatí zejména v následujících případech:

- Potřebujeme mít výsledný dokument k dispozici v několika formátech (tištěný, Web, CD-ROM, nápověda v programu apod.). XML popisuje dokumenty na sémantické úrovni, ne na vizuální – konverze je velice snadná.
- Produkujeme rozsáhlé dokumenty – technická dokumentace, vědecké sborníky, dokumentace k programům, slovníky, encyklopedie apod. Proprietární systémy mají často limit na maximální velikost produkovaných dokumentů, nebo začínají být při práci na větších dokumentech nestabilní.
- Produkujeme velké množství dokumentů, i když ty nemusí být nutně velké – redakce časopisů, zpravodajské agentury, legislativa, technická podpora. Dokumenty však mají předem danou strukturu, které se musíme držet.

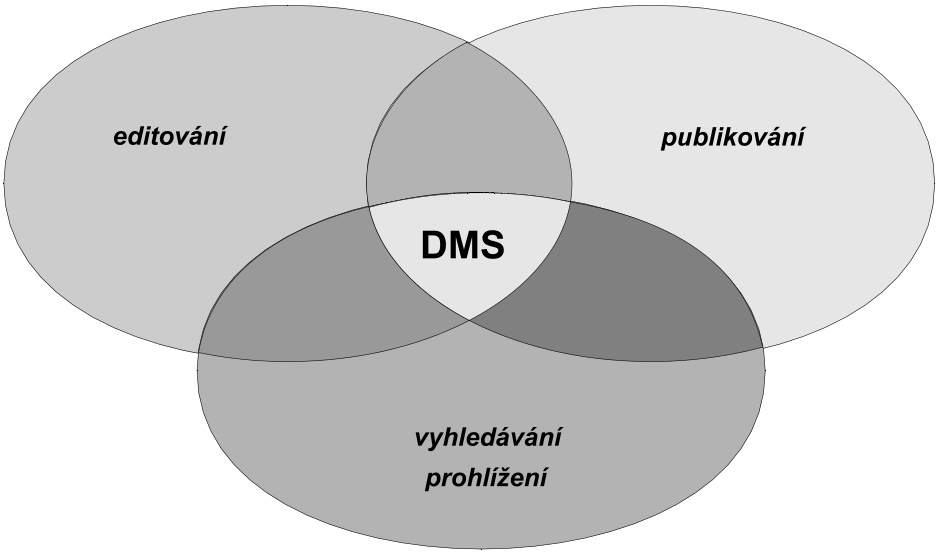
Pokud budeme dokumenty ukládat do XML, získáme tím především možnost je snadno publikovat v několika odlišných formátech, což se dnes stává téměř nezbytností.

Systémy pro elektronické publikování mohou vypadat různě, ale většinou mají společné alespoň základní rysy. Celý proces přípravy dokumentů a jejich zpracování probíhá ve třech poměrně oddělených fázích. První fáze spočívá v *editování a vytváření* samotných dokumentů ve formátu XML. Při této fázi autoři

píši knihy, články, dokumentaci apod. Vytvářené dokumenty jsou upravovány editory a korektory, schvalovány pro publikování nadřízenými pracovníky apod. Hotové dokumenty pak mohou vstoupit do *publikační fáze*. V této fázi se z dokumentů generují potřebné výstupní formáty. Dokumenty obsahující technickou dokumentaci mají většinou přesně daný formát a neobsahují zbytečné umělecké a grafické prvky, lze je tedy většinou formátovat a převádět zcela automaticky. To přináší obrovské úspory peněz a času. Automatizovaný proces generování výstupních formátů umožňuje častou a snadnou aktualizaci dokumentů, což je v době, kdy se software vyvíjí až moc rychle velice důležitá vlastnost.

V některých aplikacích pak může v úvahu připadat ještě *prohledávání databáze dokumentů*. To se uplatní v případech, kdy potřebujeme vyvolat již existující dokumenty, používat jejich části apod, zjišťovat rozdíly mezi jednotlivými verzemi apod.

Obrázek 1.1. Naše dokumenty a data jsou při elektronickém publikování to nejcennější, proto se vše „točí" okolo nich



Jádrem většiny systémů podporujících elektronické publikování je systém pro správu dokumentů. V nejjednodušších verzích může jít o nějaký sdílený disk, kam mají přístup všichni oprávnění uživatelé. Mnohem lepší je však použití specializovaných programů, které dokumenty ukládají do speciální databáze. Přístup k dokumentům pak může být velice dobře kontrolován, systém eviduje změny provedené jednotlivými uživateli a může zajišťovat i funkce pro work-flow dokumentů – před tím, než je dokument považován za finální, jej například musí schválit zodpovědná osoba.

Se systémem pro správu dokumentů (DMS) pak spolupracují editory, ve kterých jednotliví autoři vytvářejí a upravují dokumenty. Při publikování jsou z DMS získány potřebné dokumenty a převedeny do výsledných formátů.

Během školení se žádným DMS nebudeme zabývat, nicméně je dobré vědět, že takové programy existují. Pokud nemáme peněz nazbyt, můžeme plnohodnotný DMS nahradit alespoň nějakým systémem pro správu verzí jako je CVS nebo Subversion.

1.1 DocBook

DocBook je dnes asi druhá nejpoužívanější aplikace SGML/XML, hned za jazykem HTML. DocBook vznikl v roce 1991 jako formát založený na SGML, určený především pro výměnu unixové dokumentace. U jeho zrodu stála firma HaL Computers a nakladatelství O'Reilly. O vývoj a údržbu formátu se staralo

sdružení Davenport. V 90. letech DocBook používalo mnoho velkých firem, které se podílely i na jeho vývoji (např. Novell, Digital, Hewlett-Packard, SCO, Fujitsu).

V roce 1999 se péče o DocBook přesunula pod hlavičku sdružení OASIS. Aktuální verze DocBooku nese číslo označení 4.5 a existuje ve dvou verzích pro SGML i pro XML. My se během školení budeme zabývat XML verzí.

DocBook se vyvinul do podoby systému, který se hodí zejména pro tvorbu počítačové dokumentace. Bez problému ho však lze použít pro zápis libovolných knih a článků. V DocBooku je například vytvořen i tento text, dokumentace k mnoha programům je vytvářena rovněž v DocBooku – např. k operačním systémům Linux a FreeBSD, ke skriptovacímu jazyku PHP, ke grafickým rozhraním KDE a Gnome. DocBook používají i velká počítačová nakladatelství jako O'Reilly nebo velké softwarové firmy (např. Sun pro tvorbu dokumentace používá podmnožinu DocBook pojmenovanou SolBook).

DocBook obsahuje elementy, které umožňují členit dokumenty do kapitol, podkapitol atd. Kromě toho máme k dispozici mnoho elementů, jimiž můžeme označit názvy programů, souborů, parametry příkazů, výpisy programů, obrázky, snímky obrazovky, klávesové zkratky, položky nabídek apod. Získáme tak velice dobře označkovaný dokument, který se dobře převádí do dalších formátů.

Výhodou DocBooku je, že mnoho editorů a nástrojů pro práci s XML v sobě přímo zahrnuje jeho podporu. Stačí program spustit a můžeme v DocBooku začít psát. Volně k dispozici jsou XSL a DSSSL styly, které lze použít pro formátování dokumentů v DocBooku. Z našich dokumentů tak můžeme snadno vytvořit např. HTML stránky, soubor s nápovědou, dokument v RTF, v PDF nebo v PostScriptu.

Samotný DocBook není nic jiného než DTD, které definuje jaké elementy a atributy můžeme v dokumentech používat. Toto DTD se neustále vyvíjí, aby reflektovalo nově vznikající požadavky. Změny v tomto DTD, které nejsou zpětně kompatibilní, se ohlašují alespoň o jednu hlavní verzi dopředu. Další verze DocBooku jako 5.0, 6.0 apod. mohou přinést nekompatibilní změny, pro verzi 5.0 jsou již všechny tyto změny známé. Revize uvnitř hlavních verzí (3.1, 4.2 apod.) jsou vždy zcela zpětně kompatibilní s hlavní verzí. S poměrně velkým předstihem se můžeme připravit na budoucí změny. Jedinou výjimkou v tomto postupu je verze 5.0, která bude místo DTD primárně založena na RELAX NG, ze kterého se budou generovat ostatní formáty schémat (DTD, WXS).

Aktuální informace o DocBooku lze nalézt na stránkách OASIS – http://www.oasis-open.org/docbook/. Kromě aktuálních DTD tu jsou beta-verze budoucích verzí DTD, různá rozšíření a zprávy výboru, který se stará o vývoj DocBooku. Volně dostupné styly pro formátování docbookových dokumentů lze získat na adrese http://docbook.sourceforge.net. Oficiální dokumentace k DocBooku je k dispozici na adrese http://www.docbook.org.

Kapitola 2. První kroky

V této části se podíváme na jednoduchý dokument v DocBooku a ukážeme si, jak ho převést do dalších formátů.

2.1 Lehký úvod do XML

2.1.1 Základy syntaxe

Pro zápis dokumentů v DocBooku se používá jazyk XML. Stručně si zde proto zopakuje základy syntaxe tohoto jazyka.

Každý XML dokument se skládá z *elementů*, které jsou do sebe navzájem vnořené. Elementy se v textu vyznačují pomocí tzv. *tagů*. Většině elementů odpovídají dva tagy – počáteční a koncový.

```
<para>Toto je obsah elementu para.</para>
```

Ukázka obsahuje jeden element `para`. Jeho obsah je vyznačen pomocí tagů `<para>` (počáteční tag) a `</para>` (koncový tag). Jen na okraj poznamenejme, že výše uvedená ukázka je asi nejjednodušším XML dokumentem, který můžeme vytvořit.

Názvy tagů se zapisují mezi znaky `<' a `>'. Koncový tag má před svým názvem ještě znak `/', aby se snadno odlišil od počátečního.

Některé elementy nemusejí mít žádný obsah. Můžeme je samozřejmě zapisovat tak, že za počátečním tagem uvedeme hned ten koncový.

```
<para>Toto je obsah elementu para.<br></br> A tohle taky.</para>
```

Není to však příliš pohodlné, a proto můžeme v XML použít ještě jednu variantu tagu, která říká, že element nemá žádný obsah. Za jméno elementu v počátečním tagu se uvede znak `/'. Koncový tag se pak už nepoužije.

```
<para>Toto je obsah elementu para.<br/> A tohle taky.</para>
```

Každý XML dokument musí obsahovat pro všechny počáteční tagy odpovídající koncový tag, nebo musí být počáteční tag zapsán jako element s prázdným obsahem. Chybou rovněž je, když se elementy v dokumentu kříží.

```
<b>Ukázka <i>překřížení</b> elementů</i>
```

Elementy jsou základním stavebním kamenem každého dokumentu. U každého počátečního tagu můžeme použít ještě *atributy*. Atributy se obvykle používají k upřesnění významu elementu.

```
<para zabezpečení="důvěrné">Nějaká tajná informace.</para>
```

V naší ukázce jsme atributu `zabezpečení` přiřadili hodnotu `důvěrné`. Hodnotu atributu musíme vždy uzavřít do uvozovek nebo do apostrofů. U jednoho tagu lze použít více atributů najednou, stačí je oddělit mezerou.

```
<para zabezpečení="důvěrné" autor="Jan Novák">Nějaká tajná informace.</para>
```

Vzhledem k tomu, že se znaky `<' a `>' používají pro oddělení tagů od okolního textu, není možné tyto znaky zapsat do dokumentu jen tak. Pro jejich zápis musíme použít tzv. *znakové entity*. Pro zápis znaku `<' je určena entita `<` a pro `>' to je `>`.

```
Vyřešte nerovnost 3x &lt; 5
```

Pro samotný zápis ampersandu (`&`) se používá znaková entita `&`.

```
Křupavé rohlíčky vám dodá pekařství Žemlička &amp; syn
```

Pokud potřebujeme uvnitř hodnoty atributu použít zároveň uvozovky i apostrofy, s výhodou využijeme odpovídající entity `&quot;` a `&apos;`.

Každý XML dokument musí být celý obsažen v jednom elementu. Následující ukázka tedy nepředstavuje správný XML dokument.

```
<nadpis>Pokusný nadpis</nadpis>
<odstavec>První odstavec</odstavec>
<odstavec>Druhý odstavec</odstavec>
<odstavec>Třetí odstavec</odstavec>
```

Stačí však přidat jeden element, který vše „obalí”, a vše je v pořádku.

```
<článek>
  <nadpis>Pokusný nadpis</nadpis>
  <odstavec>První odstavec</odstavec>
  <odstavec>Druhý odstavec</odstavec>
  <odstavec>Třetí odstavec</odstavec>
</článek>
```

Splňuje-li dokument všechna výše uvedená pravidla, je syntakticky v pořádku a říkáme o něm, že je *správně strukturovaný (well-formed)*. Takový dokument můžeme směle vypustit do světa, protože si s ním poradí všechny aplikace podporující formát XML.

Většinou však na dokument klademe různá omezení – chceme mít pod kontrolou elementy, které se mohou v dokumentu používat apod. V tomto případě musíme mít pro dokument k dispozici definici typu dokumentu (DTD). Pomocí parseru lze zkontrolovat, zda dokument danému DTD vyhovuje a je tzv. *validní*.

2.1.2 Znakové sady a kódování

Současné počítače pracují vnitřně pouze s čísly. Na obrazovce sice vidíme texty, obrázky nebo třírozměrný model bludiště, ale někde za tím vším jsou již jen čísla. S texty se pracuje také jako s čísly. Každému znaku je přiřazeno číslo. Sada znaků a jím odpovídajících čísel je *znaková sada*.

Mezi nejstarší a nejznámější znakové sady patří ASCII. Tato znaková sada byla 7bitová – obsahovala znaky s kódy 0 až 127. Kromě písmen anglické abecedy, číslic a dalších znaků obsahovalo ASCII i některé řídící znaky. Potřebám angličtiny ASCII zcela vyhovovalo. Pro ostatní jazyky zde však chyběla některá písmena – pro češtinu například znaky s diakritikou. Vzniklo proto několik 8bitových znakových sad, které obsahovaly 256 znaků. Prvních 128 znaků bylo kvůli zpětné kompatibilitě shodných s ASCII. Horní část znakové sady pak obsahovala národní znaky.

Pro češtinu a další středoevropské jazyky existuje znaková sada ISO 8859-2, pro rušinu ISO 8859-5 apod. Microsoft ve Windows používá vlastní znakové sady, které se od ISO mírně liší. Pro češtinu je vhodná znaková sada windows-1250.

Pokud bychom však chtěli v jednom dokumentu používat více různých jazyků, dostaneme se do problémů, protože například znaková sada pro češtinu už neobsahuje azbuku. Postupně proto vznikly ještě další znakové sady, které obsahovaly více znaků. Mezi nejznámější patří 32bitové sady Unicode 3.0 a ISO 10646.

Výhodou těchto znakových sad je, že obsahují znaky všech běžně používaných jazyků – kromě češtiny či ruštiny zde nalezneme i pro nás exotické jazyky jako arabštinu, korejštinu, japonštinu a mnoho dalších.

XML používá jako znakovou sadu ISO 10646, protože je to dnes nejkomplexnější znaková sada a dá se očekávat, že v blízké budoucnosti ji bude přímo podporovat většina operačních systémů. ISO 10646 je vyvíjeno společně s Unicodem, takže definované znaky a jejich kódy se shodují. V následujícím textu se podíváme na to, co pro nás použití znakové sady ISO 10646 znamená v praxi.

Zatímco znaková sada definuje, jaké znaky a pod jakým číslem máme k dispozici, kódování znakové sady určuje, jak jsou jednotlivé kódy znaků převedeny na sekvenci bajtů, které znak reprezentují v paměti počítače, v souboru, při přenosu počítačovou sítí apod.

32bitová znaková sada na první pohled potřebuje pro uložení jednoho znaku 4 bajty. Pokud bychom takto přenášeli například jen anglické texty, bylo by to velice neefektivní. Z tohoto důvodu vzniklo kódování UTF-8 (UCS Transformation Format), které znaky z ASCII kóduje do jednoho bajtu; méně obvyklé znaky jsou pak kódovány v několika bajtech.

Kódování UTF-8 je identické s ASCII. Další znaky nad rámec ASCII jsou kódovány do sekvencí 2 až 6 bajtů. České znaky s diakritikou se v UTF-8 kódují do dvou bajtů.

Existuje ještě kódování UTF-16, které jeden znak ukládá do dvou bajtů. Využívá se přitom faktu, že v současné době je definováno jen něco málo přes 49 tisíc znaků. UTF-16 rovněž obsahuje mechanismus, jak ukládat až 1 milión znaků pomocí „surrogate pairs“ – jeden znak je uložen do čtyř bajtů.

Všechny aplikace, které podporují XML, by měly zvládat práci s kódováními UTF-8 a UTF-16. Pro nás tato kódování nejsou nejvhodnější, jsme zvyklí spíše na windows-1250 a ISO 8859-2. I když jsme dosud o windows-1250 a ISO 8859-2 mluvili jako o znakových sadách, můžeme na ně pohlížet i jako na kódování, která pokrývají pouze část ISO 10646.

U každého XML dokumentu můžeme určit používané kódování. Pokud však budeme používat jiné než UTF-8 nebo UTF-16, musíme si dát pozor, aby toto kódování podporovaly všechny aplikace, s nimiž budeme XML dokument zpracovávat.

Pokud v dokumentu používáme jiné kódování než UTF-8 nebo UTF-16, musíme ho specifikovat pomocí *XML deklarace*, která musí představovat první řádku dokumentu. Nejjednodušší XML deklarace má následující tvar.

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

Deklaraci můžeme používat ve všech dokumentech. Standardně obsahuje pouze určení verze XML, pro zachování zpětné kompatibility v případě dalších verzích XML.

Použité kódování můžeme určit pomocí parametru `encoding`, který je součástí XML deklarace.

```
<?xml version="1.0" encoding="windows-1250"?>
<dokument>
  Dokument si vesele píši ve starém špatném Notepadu.
</dokument>
```

Varování

Pokud dokument nepíšeme v UTF-8 nebo UTF-16, musíme kódování určit v XML deklaraci. Pokud to neuděláme, totálně tím zmateme většinu aplikaci, které s XML dokumentem mají pracovat.

Jeden XML dokument může být fyzicky uložen v několika souborech. Dílčí soubory se načítají jako tzv. externí entity. Pro každou externí entitu můžeme rovněž určit její kódování a to tak, že na prvním řádku použijeme deklaraci kódování. K této problematice se ještě podrobněji vrátíme, až si řekneme, jak rozdělit docbookový dokument do více souborů.

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

Může se stát, že náš editor nepodporuje všechny znaky, které potřebujeme použít. Například píšeme dokument v češtině, ale potřebujeme v něm použít řecké písmeno π. To se ve většině 8bitových kódování (jako např. windows-1250 nebo ISO 8859-2) nevyskytuje. Proto můžeme do XML dokumentu vložit libovolný znak pomocí *znakové entity*. Znaková entita má tvar `&#x`*kód znaku*`;`, kde *kód znaku* je kód znaku ze znakové sady ISO 10646 zapsaný v šestnáctkové soustavě. Můžeme použít i tvar `&#`*kód znaku*`;`, kde je *kód znaku* zapsán v desítkové soustavě.

```
Obsah kruhu se vypočte podle vzorce &#x3c0;r<sup>2</sup>.
Originální&#32;způsob, jak vložit do textu mezeru.
```

Přehled znaků a jejich kódů nalezneme například na serveru sdružení Unicode[1] nebo v mapě znaků ve Windows.

2.1.3 Další syntaktické obraty

Komentáře

Pokud potřebujeme v dokumentu něco vysvětlit nebo část textu dočasně skrýt, s výhodou k tomu použijeme komentář. Komentář je součástí dokumentu, ale parsery jej ignorují a není dále zpracováván. Komentář se zapisuje mezi znaky `<!--` a `-->`.

```
<!-- Vysvětlující text -->
```

Komentář může obsahovat cokoliv, kromě posloupnosti znaků `--`. V komentáři dokonce můžeme používat tagy atd. Jsou však zcela ignorovány. To se hodí pro dočasné vyřazení části dokumentu ze zpracování.

```
<para>První odstavec.</para>
<!-- <para>Šéf mi leze krkem.</para> -->
<para>Třetí odstavec.</para>
```

Sekce CDATA

Pokud potřebujeme do dokumentu vložit větší kus textu, kde se hojně používají znaky se speciálním významem jako `<', `>' a `&', je nepohodlné je zapisovat pomocí znakových entit. Sekce CDATA oceníme zejména v případech, kdy je součástí XML dokumentu kód nějakého programu nebo HTML či XML kód. Použití sekcí CDATA si ukážeme na následujícím dokumentu.

```
<script language="JavaScript">
<![CDATA[
```

[1] http://www.unicode.org

```
  for (i=0; i < 10; i++)
  {
    document.writeln("<p>Ahoj</p>");
  }
]]>
</script>
```

Bez použití sekce CDATA by byl zápis přece jen poněkud krkolomný.

```
<script language="JavaScript">
  for (i=0; i &lt; 10; i++)
  {
    document.writeln("&lt;p&gt;Ahoj&lt;/p&gt;");
  }
</script>
```

Obecně se tedy sekce CDATA zapisují jako `<![CDATA[text]]>`. Text přitom může obsahovat cokoliv, kromě sekvence znaků `]]>`. Konstrukce CDATA existuje v XML pro větší pohodlí autorů, kteří zapisují XML kód ručně.

Instrukce pro zpracování

XML dokumenty mohou být zpracovávány různými programy. Někdy může být užitečné do dokumentu uložit řídící informace, které jsou určeny pouze pro některý program. Můžeme tak do dokumentu zařadit odkaz na styl definující zobrazení v prohlížeči, formátovacímu programu můžeme naznačit, kde má zalomit stránku. Moderní skriptové jazyky pro generování dynamických webových stránek se také zapisují přímo do těla dokumentů. Pro všechny tyto účely má XML k dispozici standardní způsob pro zařazení nestandardních informací. Na libovolné místo dokumentu (kromě značkování – podobně jako u komentářů) můžeme vložit *instrukce pro zpracování (processing instructions)*. Tyto instrukce XML parser ignoruje, předá je nadřazené aplikaci – záleží na ní, zda je nějak využije. Syntaxe instrukcí je velice jednoduchá.

```
<?identifikátor data?>
```

Pomocí *identifikátoru* můžeme rozlišovat jednotlivé druhy instrukcí – do jednoho dokumentu můžeme umístit instrukce pro několik různých programů. Samotná *data* instrukce mohou mít libovolný tvar, ale nesmějí obsahovat sekvenci znaků `?>`.

Například DSSSL a XSL styly rozpoznávají instrukci `dbhtml`, kterou lze ovlivňovat jméno souboru v případě generování výstupu do HTML.

2.2 První dokument v DocBooku

V DocBooku můžeme vytvářet mnoho druhů dokumentů, ale nejobvyklejší jsou knihy. Následující ukázka obsahuje velice jednoduchou knihu zapsanou v DocBooku.

Příklad 2.1. Kniha zapsaná v DocBooku – `prvni.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE book PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
          'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<book lang="cs">
  <bookinfo>
    <title>První pokusná kniha</title>
    <author>
```

```
      <firstname>Jiří</firstname>
      <surname>Kosek</surname>
    </author>
  </bookinfo>
  <preface>
    <title>Úvod</title>
    <para>Odstavec textu.</para>
    <para>...</para>
  </preface>
  <chapter>
    <title>První kapitola</title>
    <para>Text první kapitoly</para>
    <para>...</para>
  </chapter>
  <chapter>
    <title>Druhá kapitola</title>
    <para>Text druhé kapitoly</para>
    <para>...</para>
  </chapter>
  <appendix>
    <title>První příloha</title>
    <para>Text přílohy</para>
    <para>...</para>
  </appendix>
</book>
```

2.3 Formátování prvního dokumentu

Dokument zapsaný v DocBooku sice přesně popisuje to co má – dokument, ale nijak nedefinuje, jak se má dokument zobrazit nebo zformátovat před tiskem. Pokud chceme XML dokument zobrazit musíme mít k dispozici styl, který definuje zobrazení jednotlivých elementů v dokumentu. Styly se zapisují v některém ze *stylových jazyků*. V současné době jsou nejpoužívanější následující stylové jazyky:

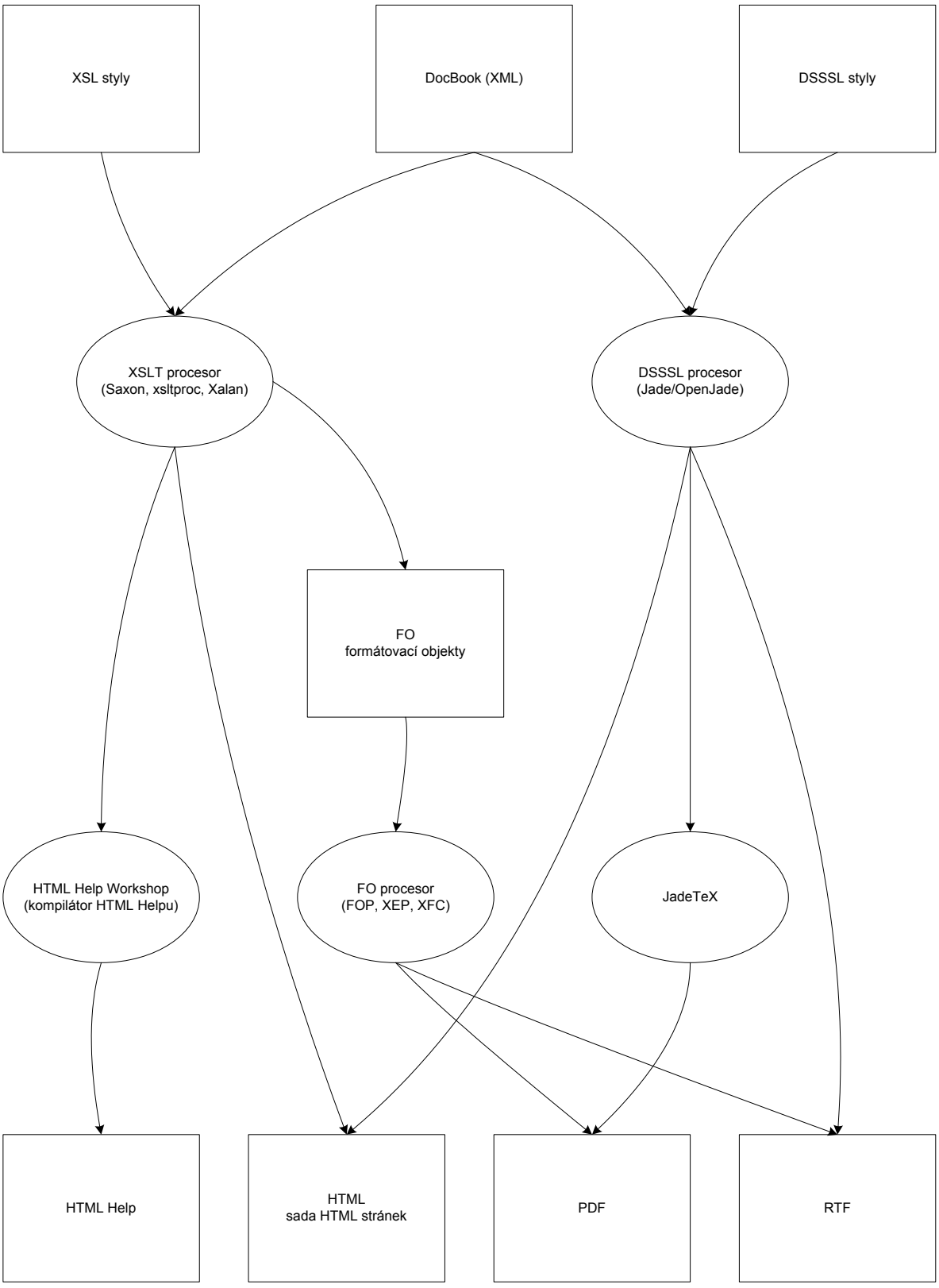
- XSL – stylový jazyk navržený speciálně pro XML. Dnes se již hodně používá a do budoucnosti je nejperspektivnější. Existují volně dostupné XSL styly speciálně určené pro DocBook.
- DSSSL – stylový jazyk původně navržený pro SGML, umí však zpracovat i XML dokumenty. Příliš se nerozšířil a vypadá to, že pozvolna ze světa zmizí. Existují volně dostupné DSSSL styly speciálně určené pro DocBook.
- CSS (kaskádové styly) – pro formátování docbookových dokumentů většinou kvůli své jednoduchosti nepostačují.
- FOSI – stylový jazyk používaný v některých komerčních aplikacích.

O instalaci stylů a nástrojů potřebných pro jejich zpracování se více dozvíte v kapitole 9 – „*Instalace*“. Tabulka 2.1 – „Výstupní formáty podporované DSSSL a XSL styly“ shrnuje nejpoužívanější výstupní formáty, které je možné pomocí jednotlivých stylů a nástrojů získat. Obdobnou informaci přináší i obrázek 2.1 – „Možnosti zpracování dokumentů XSL a DSSSL styly“.

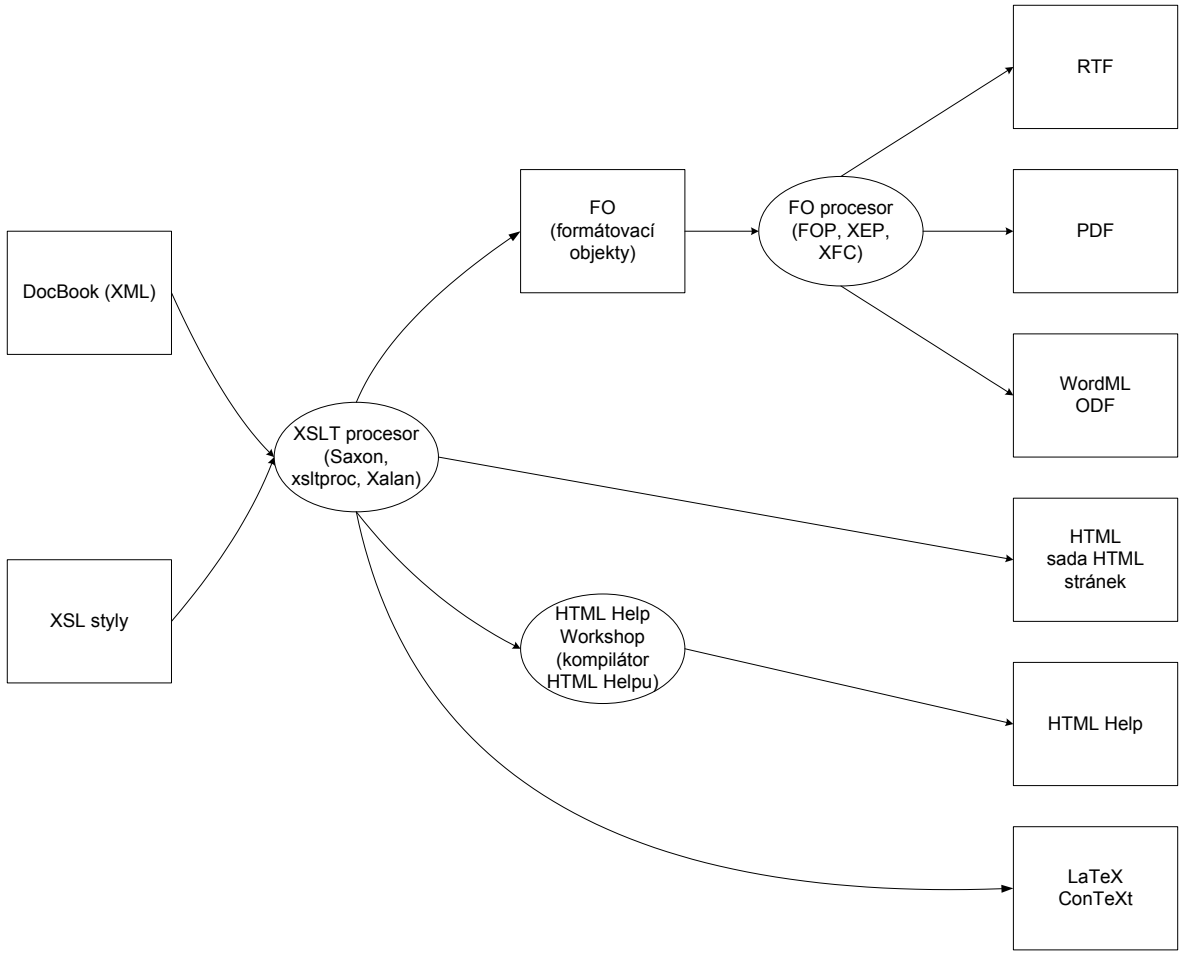
Tabulka 2.1. Výstupní formáty podporované DSSSL a XSL styly

Výstupní formát	Stylový jazyk	Potřebné nástroje
jedna HTML stránka	DSSSL	Jade
jedna HTML stránka	XSL	libovolný XSLT procesor
sada HTML stránek	DSSSL	Jade
sada HTML stránek	XSL	libovolný XSLT procesor
PDF	DSSSL	Jade+JadeTeX nebo Jade+Word+Distiller
PDF	XSL	libovolný XSLT procesor + FO procesor (např. XEP)
RTF	DSSSL	Jade
RTF	XSL	libovolný XSLT procesor + vhodný FO procesor (např. XFC)
WordML	XSL	libovolný XSLT procesor + vhodný FO procesor (např. XFC)
ODF	XSL	libovolný XSLT procesor + vhodný FO procesor (např. XFC)
PostScript	DSSSL	Jade+JadeTeX nebo Jade+Word+ovladač PS tiskárny
PostScript	XSL	libovolný XSLT procesor + FO procesor (např. XEP)
HTML Help	XSL	libovolný XSLT procesor + HTML Help WorkShop
Java Help	XSL	libovolný XSLT procesor + Java Help
Eclipse Help	XSL	libovolný XSLT procesor
manuálové stránky	XSL	libovolný XSLT procesor

Obrázek 2.1. Možnosti zpracování dokumentů XSL a DSSSL styly



Obrázek 2.2. Možnosti zpracování dokumentů pomocí XSL stylů



2.3.1 Použití DSSSL stylů

Pokud chceme použít DSSSL styly musíme mít k dispozici program Jade, který je umí aplikovat na SGML/XML dokumenty. Při spuštění je potřeba Jade zavolat s následujícími parametry:

```
jade -d styl -t výstupní_formát XML_deklarace soubor.xml
```

Pokud například chceme získat RTF, zadáme příkaz:

```
jade -d c:\docbook\dsssl\print\docbook.dsl -t rtf c:\docbook\jade\xml.dcl prvni.xml
```

Po chvilce dostaneme soubor `prvni.rtf`, který můžeme načíst v libovolném editoru, který podporuje formát RTF. Dokument je automaticky editorem zformátován, obvykle jsou pouze chybná čísla stran v obsahu. Při tisku se čísla stran správně přepočítají. Ručně můžeme například v MS Wordu přepočítání vyvolat stiskem kláves **Ctrl**+**A** a **F9**.

Pro získání HTML verze stačí místo stylů určených pro tisk použít HTML styly a jako výstupní formát použít `sgml`:

```
jade -d c:\docbook\dsssl\html\docbook.dsl -t sgml c:\docbook\jade\xml.dcl prvni.xml
```

Získáme několik HTML stránek doplněných o přehledné navigační prvky a automaticky se rovněž vygeneruje obsah dokumentu. Pokud bychom chtěli získat celý dokument jako jednu stránku, při volání Jade předáme stylu parametr ovlivňující jeho chování:

```
jade -d c:\docbook\dsssl\html\docbook.dsl -V nochunks -t sgml c:\docbook\jade\xml.dcl ▶
prvni.xml > prvni.html
```

V tomto případě Jade generovaný HTML kód zapisuje na standardní výstup, a proto jsme ho pomocí > přesměrovali do souboru.

Jade ještě umí generovat výstup pro tisk ve formátech MIF (FrameMaker) a TeX – využívá se přitom balík maker JadeTeX. FrameMaker není v našich krajích moc rozšířen, proto se jím nebudeme zabývat. JadeTeX nepracuje za všech okolností správně a má problémy s delšími dokumenty.

2.3.2 Použití XSL stylů

Pro zpracování dokumentu pomocí XSL stylů musíme mít nějaký XSLT procesor a případně FO procesor. My použijeme Saxon a XEP.

Pro získání HTML verze dokumentu, můžeme použít příkaz:

```
saxon -o prvni.html prvni.xml c:\docbook\xsl\html\docbook.xsl
```

Pokud chceme místo jedné HTML stránky, získat sadu stránek provázaných odkazy, stačí použít styl `chunk.xsl` místo `docbook.xsl`:

```
saxon prvni.xml c:\docbook\xsl\html\chunk.xsl
```

Pokud chceme získat nápovědu ve formátu HTML Help, spustíme speciální styl `htmlhelp.xsl`. Ten se chová velice podobně jako styl `chunk.xsl` a navíc vygeneruje soubor pro kompilátor HTML Helpu. Při spouštění Saxonu ještě musíme nastavit parametr, který určí kódování řídících souborů pro kompilátor HTML Helpu.

```
saxon prvni.xml c:\docbook\xsl\htmlhelp\htmlhelp.xsl "htmlhelp.encoding=windows-1250"
```

Automaticky se vygenerují soubory `toc.hhc` a `htmlhelp.hhp`. Druhý z nich je projektovým souborem pro kompilátor HTML Helpu a můžeme ho proto otevřít v programu HTML Help Workshop. Z menu vybereme příkaz File▻Compile. Do souboru `htmlhelp.chm` vygeneruje se nápověda ve formátu HTML Help. Místo HTML Help Workshopu lze použít i řádkový kompilátor **hhc**.

Poslední formát, který můžeme pomocí XSL stylů snadno získat je PDF. Nejprve si vytvoříme soubor obsahující formátovací objekty:

```
saxon -o prvni.fo prvni.xml c:\docbook\xsl\fo\docbook.xsl
```

z něj pak vygenerujeme PDF například pomocí XEPu:

```
xep -fo prvni.fo -pdf prvni.pdf
```

XEP si také umí domyslet jméno PDF dokumentu sám, takže stačí zadat:

```
xep -fo prvni.fo
```

Proces transformace na formátovací objekty a formátování můžeme spojit do jednoho kroku, protože XEP má v sobě zabudovaný Saxon:

```
xep -xml prvni.xml -xsl c:\docbook\xsl\fo\docbook.xsl
```

2.4 Kontrola syntaxe dokumentu

Správnou syntaxi můžeme zkontrolovat pomocí libovolného validujícího parseru. Většina editorů má parser zabudován přímo v sobě. Mezi nejznámější řádkové parsery patří nsgmls pocházející ještě z dob SGML:

```
nsgmls -wxml -s c:\docbook\jade\xml.dcl prvni.xml
```

Další možností je např. xmllint:

```
xmllint --noout --valid prvni.xml
```

Integrovanou funkci validace obsahuje i většina editorů XML.

Kapitola 3. Přehled struktur a elementů DocBooku

DocBook umožňuje vytvářet několik druhů dokumentů. Mezi ty nejpoužívanější patří knihy (`book`), články (`article`), FAQ (`qandaset`) a referenční stránky (`refentry`). Nyní se podíváme na základní struktury, které se v těchto typech dokumentů používají. Detailní popis naleznete v [1].

3.1 Knihy

Kniha je asi nejpoužívanější typ dokumentu, který se v DocBooku vytváří. Struktura knihy je poměrně volná a vyhoví většině požadavků. Jednak kniha může obsahovat metainformace (`bookinfo`) a různé pomocné části jako věnování (`dedication`) a tiráž (`colophon`). Dále se kniha skládá ze tří druhů elementů:

Navigační komponenty	Sem patří zejména obsah (`ToC`), seznamy obrázků, tabulek, příkladů apod. (`LoT`) a rejstřík (`index`). Tyto elementy se většinou nepoužívají, protože obsahy i rejstřík lze vygenerovat automaticky.
Části	Každá kniha se může skládat z několika částí (`part`) a referencí (`reference`). Části knihy se pak skládají z jednotlivých komponent. U jednodušších knih nemusíme části používat, komponenty můžeme do knihy vložit přímo.
Komponenty	Sem patří elementy, které dělí knihu na úrovni kapitol – předmluva (`preface`), kapitoly (`chapter`), přílohy (`appendix`), slovníček (`glossary`) a seznam literatury (`bibliography`). Pokud chceme sdružit více článků do jednoho celku, můžeme jako komponentu použít i článek (`article`). Jednotlivé komponenty se skládají z elementů, které jsou popsány dále.

Více souvisejících knih můžeme zahrnout do jedné sady (`set`).

3.2 Sekce

Komponenty knih (kapitoly, přílohy apod.) a články můžeme dále strukturovat pomocí následujících elementů:

`sect1`–`sect5`	Pět úrovní sekcí, při vkládání sekcí do sebe nemůžeme přeskočit nějakou sekci v hierarchii.
`section`	Rekurzivní element, který umožňuje vytváření víceúrovňových sekcí rekurzivním zanořováním.
`simplesect`	Sekce, kterou již nelze dále rozdělit na podsekce.
`bridgehead`	Samostatný nadpis sekce, který neslouží jako kontejner pro další elementy nebo sekce.
`refsect1`–`refsect3`	Sekce, které umožňují členit položky reference.

`glossdiv`, `bibliodiv`, `indexdiv`	Elementy pro členění slovníčku, seznamu literatury a rejstříku.

3.3 Metainformace

K většině komponent a vyšších elementů můžeme přidat bibliografické informace jako autor, název, copyright apod. Tyto informace se zapisují do elementu `*info`, kde `*` je název příslušného elementu. Například informace o knize se zapisují do elementu `bookinfo`.

3.4 Blokové elementy

Blokové elementy jsou ty elementy, které text dělí na odstavce a podobné úseky. Bývá u nich obvyklé, že tvoří samostatný odstavec – před i za nimi je zalomená řádka. Oproti tomu existují ještě in-line elementy, které jen vyznačují část textu odstavce a po zformátování se obvykle projeví jen změnou písma.

3.4.1 Odstavce

DocBook obsahuje tři druhy odstavců – `para`, `simpara` a `formalpara`.

3.4.2 Seznamy

K dispozici máme několik druhů seznamů:

`itemizedlist`	Běžný druh seznamu s odrážkami.
`orderedlist`	Číslovaný seznam.
`variablelist`	Seznam s pojmů a jejich definic.

Kromě těchto nejpoužívanějších seznamů jsou k dispozici i některé další – seznam popisů k nějakému výpisu (`calloutlist`), seznam pojmů ze slovníku (`glosslist`) a jednoduché seznamy `simplelist` a `segmentedlist`.

3.4.3 Upozornění

Pro vkládání různých výstrah a upozornění můžeme použít následující elementy – `caution`, `important`, `note`, `tip` a `warning`.

Všechny tyto elementy mohou obsahovat nepovinný nadpis (`title`) a za ním libovolné elementy pro odstavce textu.

3.4.4 Elementy zachovávající řádkování

Následující elementy zachovávají konce řádků tak, jak jsou uvedeny ve zdrojovém XML dokumentu. To je výhodné například u výpisu programů apod. DocBook neobsahuje ekvivalent HTML tagu `br` pro ukončení řádky, proto mají tato prostředí svůj velký význam.

`address`	Element pro zápis poštovních adres. Kromě toho že zachovává konce řádek v něm můžeme použít další elementy, pro vyznačení jména, města, státu apod.

`literallayout`	Element zachovává konce řádků a většinou se zobrazuje normálním písmem.
`programlisting`	Element určený pro zařazování výpisu programů, obvykle se zobrazuje neproporcionálním písmem.
`screen`	Element obvykle zachycuje nějaký výstup z obrazovky apod.
`screenshot`	Speciální element pro zařazování sejmutých obrazovek. Obvykle obaluje obrázek.
`synopsis`	Element obsahující popis syntaxe příkazu nebo funkce.

3.4.5 Příklady, obrázky a tabulky

Pro vkládání příkladů, obrázků a tabulek slouží elementy `example`, `informalexample`, `figure`, `informalfigure`, `table` a `informaltable`. Neformální verze neobsahují nadpis. K praktickému vkládání obrázků a tabulek se ještě vrátíme.

3.4.6 Rovnice

Rovnice lze vkládat pomocí elementů `equation`, `informalequation` a `inlineequation`. Rovnice se vkládají jako obrázek a alternativní text. Pokud chceme do textu vkládat hodně rovnic, je možné DocBook rozšířit o podporu jazyka MathML pro zápis matematických výrazů.

3.4.7 Obrázky

Obrázky se vkládají pomocí elementů `graphics`, `inlinegraphics`, `mediaobject` a `inlinemediaobject`. Pokud chceme mít u obrázků popis, musíme je vložit do elementu `figure`.

3.4.8 Další blokové elementy

`blockquote`	Delší citace.
`classsynopsis`	Element pro zápis definice třídy.
`constructorsynopsis`	Element pro zápis konstruktoru a jeho parametrů.
`cmdsynopsis`	Popis příkazu včetně všech voleb a parametrů.
`destructorsynopsis`	Element pro zápis destruktoru a jeho parametrů.
`epigraph`	Krátká citace na začátku kapitoly.
`funcsynopsis`	Element pro zápis funkce a jejích parametrů.
`highlights`	Souhrn hlavních bodů knihy nebo komponenty.
`methodsynopsis`	Element pro zápis metody a jejích parametrů.
`msgset`	Množina souvisejících chybových hlášení.
`procedure`	Element obsahuje kroky, které tvoří nějaký postup.

`sidebar`	Poznámka na okraji (marginálie).

3.5 Inline elementy

Inline elementy přiřazují význam jednotlivým částem textu. DocBook jich obsahuje opravdu mnoho a záleží na každém autorovi, jak přesně bude dokument značkovat. Jelikož přidávání elementů zdržuje psaní je dobré vždy vytvořit nějaký kompromis. Značkovat jen to, co skutečně potřebujeme – pokud například píšeme dokumentaci k nějakému API, je dobré v textu označovat názvy funkcí – tuto informaci pak můžeme využít při generování rejstříku. Pro snazší orientaci jsou elementy rozděleny do několika kategorií.

3.5.1 Obecné elementy

`abbrev`	Zkrácené slovo.
`acronym`	Zkratka.
`emphasis`	Zvýraznění textu.
`footnote`	Poznámka pod čarou.
`phrase`	Označená část textu.
`quote`	Text v uvozovkách.
`trademark`	Obchodní značka.

3.5.2 Odkazy

`anchor`	Označuje místo v dokumentu.
`citation`	Citace položky ze seznamu literatury.
`citetitle`	Název citovaného díla.
`firstterm`	První výskyt pojmu.
`glossterm`	Pojem ze slovníčku.
`link`	Odkaz na jiné místo dokumentu.
`olink`	Odkaz na jiný dokument zapsaný nepřímo.
`ulink`	Odkaz na URL adresu.
`xref`	Odkaz na jinou část dokumentu.

3.5.3 Značkování

`foreignphrase`	Fráze v cizím jazyce.
`wordasword`	Slovo.
`computeroutput`	Výstup generovaný počítačem.

`literal`	Text, který lze chápat jako literál.
`markup`	Značkování, které má být zobrazeno tak, jak je zapsáno.
`prompt`	Řetězec označující vstupní pole na obrazovce.
`replaceable`	Text, který má být podle potřeby nahrazen uživatelem.
`sgmltag`	Element pro zápis SGML/XML elementů, tagů, atributů apod.
`userinput`	Text zadaný uživatelem.
`code`	Kousek programového kódu.
`uri`	URI adresa.

3.5.4 Matematické výrazy

`inlineequation`	Matematický výraz.
`subscript`	Dolní index.
`superscript`	Horní index.

3.5.5 Uživatelské rozhraní

`accel`	Označuje klávesovou zkratku uvnitř menu.
`guibutton`	Text na tlačítku.
`guiicon`	Text nebo obrázek vystupující jako ikona.
`guilabel`	Text na popisce.
`guimenu`	Název menu.
`guimenuitem`	Koncová položka menu.
`guisubmenu`	Název podmenu.
`keycap`	Text napsaný na klávese.
`keycode`	Kód určité klávesy.
`keycombo`	Kombinace kláves.
`keysym`	Symbolické jméno klávesy.
`menuchoice`	Výběr z menu.
`mousebutton`	Tlačítko myši.
`shortcut`	Klávesová zkratka odpovídající příkazu v menu.

3.5.6 Programování

`action`	Odezva na nějakou událost.
`code`	Kousek programového kódu.
`classname`	Název třídy.
`constant`	Konstanta.
`errorcode`	Chybový kód.
`errorname`	Chybové hlášení.
`errortype`	Druh chyby.
`exceptionname`	Název výjimky.
`function`	Název funkce, rutiny, metody.
`interface`	Část uživatelského rozhraní.
`interfacedefinition`	Název formální specifikace uživatelského rozhraní.
`interfacename`	Název rozhraní (API).
`literal`	Text, který lze chápat jako literál.
`methodname`	Název metody.
`msgtext`	Text hlášení.
`parameter`	Parametr.
`replaceable`	Text, který má být podle potřeby nahrazen uživatelem.
`returnvalue`	Hodnota vrácená funkcí.
`structfield`	Položka struktury/záznamu.
`structname`	Název struktury/záznamu.
`symbol`	Název, který je před zpracováním nahrazen hodnotou.
`token`	Jednotka určená ke zpracování.
`type`	Klasifikace hodnoty.
`varname`	Název proměnné.

3.5.7 Operační systémy

`application`	Název aplikace.
`command`	Příkaz.

`envar`	Proměnná prostředí.
`filename`	Název souboru.
`medialabel`	Název média (diskety, pásky apod.).
`option`	Volba příkazu.
`parameter`	Parametr.
`prompt`	Řetězec označující vstupní pole na obrazovce.
`systemitem`	Položka nebo pojem svázaný se systémem.

3.5.8 Obecné inline elementy

`application`	Název aplikace.
`database`	Databáze.
`email`	E-mailová adresa.
`filename`	Název souboru.
`hardware`	Fyzická součást počítače.
`inlinegraphics`, `inlinemediaobject`	Obrázek vložený přímo do textu.
`optional`	Nepovinná informace.
`remark`	Komentář, který se má zobrazovat v pracovních verzích dokumentu.

3.6 Předdefinované znakové entity

DocBook obsahuje mnoho předdefinovaných entit, které můžeme používat pro zápis neobvyklých znaků. Kompletní přehled lze najít v DTD pro DocBook nebo v [1]. Některé nejpoužívanější entity jsou shrnuty v tabulce 3.1 – „Nejběžnější entity“.

Tabulka 3.1. Nejběžnější entity

Entita	Znak	Popis
`&ndash;`	–	krátká pomlčka
`&mdash;`	—	dlouhá pomlčka
`&hellip;`	…	tři tečky
`&alpha;`	α	řecké písmeno „alfa“
`&half;`	½	1/2
`&copy;`	©	znak copyrightu
`&reg;`	®	registrovaná značka
`&plusmn;`	±	
`&sect;`	§	paragraf
`&ldquor;`	„	české uvozovky (jednodušší je zápis uvozovek pomocí elementu `quote`)
`&ldquo;`	“	

Kapitola 4. Ukázky nejpoužívanějších docbookových konstrukcí

Nyní si na příkladě typických dokumentů ukážeme použití nejběžnějších docbookových elementů.

4.1 Psaní knih a kapitol

Základní členění knihy do kapitol jsme již probrali. Zajímavé možnosti nabízí element `bookinfo`, do kterého lze zapsat mnoho metainformací o knize.

Příklad 4.1. Ukázka knihy a vložených metainformací – `kniha.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE book PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
               'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<book lang="cs">
  <bookinfo>
    <title>První pokusná kniha</title>
    <subtitle>Uživatelská příručka</subtitle>
    <authorgroup>
      <author>
        <firstname>Jiří</firstname>
        <surname>Kosek</surname>
        <affiliation>
          <orgname>nezávislý publicista</orgname>
          <address><email>jirka@kosek.cz</email></address>
        </affiliation>
      </author>
      <author>
        <honorific>Ing</honorific>
        <firstname>Jan</firstname>
        <surname>Novák</surname>
        <affiliation>
          <orgname>Nějaká firma, a.s.</orgname>
          <address><street>Dlouhá 10</street>
<city>Praha 1</city>  <postcode>110 11</postcode></address>
        </affiliation>
      </author>
    </authorgroup>
    <publisher>
      <publishername>První vydavatelská</publishername>
    </publisher>
    <copyright>
      <year>2001</year>
      <holder>První vydavatelská</holder>
    </copyright>
    <releaseinfo>První veřejná verze příručky.</releaseinfo>
  </bookinfo>
  <preface>
    <title>Úvod</title>
    <para>Odstavec textu.</para>
    <para>...</para>
  </preface>
  <chapter>
```

```
    <title>První kapitola</title>
    <para>Text první kapitoly</para>
    <para>...</para>
  </chapter>
  <chapter>
    <title>Druhá kapitola</title>
    <para>Text druhé kapitoly</para>
    <para>...</para>
  </chapter>
  <appendix>
    <title>První příloha</title>
    <para>Text přílohy</para>
    <para>...</para>
  </appendix>
  <appendix>
    <title>Druhá příloha</title>
    <para>Text přílohy</para>
    <para>...</para>
  </appendix>
</book>
```

Podobné metainformace můžeme přidávat i k jednotlivým kapitolám a dalším komponentám dokumentů.

Příklad 4.2. Ukázka kapitoly a vložených metainformací – `kapitola.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE book PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
               'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<book lang="cs">
  <bookinfo>
    <title>První pokusná kniha</title>
    <subtitle>Uživatelská příručka</subtitle>
    <author>
      <firstname>Jiří</firstname>
      <surname>Kosek</surname>
    </author>
    <copyright>
      <year>2001</year>
      <holder>První vydavatelská</holder>
    </copyright>
  </bookinfo>
  <preface>
    <title>Úvod</title>
    <para>Odstavec textu.</para>
    <para>...</para>
  </preface>
  <chapter>
    <title>První kapitola</title>
    <para>Text první kapitoly</para>
    <para>...</para>
  </chapter>
  <chapter>
    <chapterinfo>
      <title>Druhá kapitola</title>
      <author>
        <honorific>Ing</honorific>
        <firstname>Jan</firstname>
        <surname>Novák</surname>
```

```
        <affiliation>
          <orgname>Nějaká firma, a.s.</orgname>
          <address><street>Dlouhá 10</street>
<city>Praha 1</city>  <postcode>110 11</postcode></address>
        </affiliation>
      </author>
    </chapterinfo>
    <title>Druhá kapitola</title>
    <para>Text druhé kapitoly</para>
    <para>...</para>
  </chapter>
  <appendix>
    <title>První příloha</title>
    <para>Text přílohy</para>
    <para>...</para>
  </appendix>
  <appendix>
    <title>Druhá příloha</title>
    <para>Text přílohy</para>
    <para>...</para>
  </appendix>
</book>
```

4.2 Psaní článků

Uvnitř článku můžeme používat v podstatě stejné elementy jako uvnitř kapitol. Navíc může být článek doplněn abstraktem a přílohami. Metainformace o článku mohou obsahovat i takové údaje jako ISSN, ISBN nebo třeba informace o konferenci, kde byl článek prezentován.

Příklad 4.3. Ukázka článku – `clanek.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE article PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
               'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<article lang="cs">
  <articleinfo>
    <title>Lehký úvod do XML</title>
    <author>
      <firstname>Jiří</firstname>
      <surname>Kosek</surname>
      <affiliation>
        <orgname>Vysoká škola ekonomická v Praze</orgname>
        <address>E-mail: <email>jirka@kosek.cz</email>
Web: <otheraddr><ulink url="http://www.kosek.cz">http://www.kosek.cz</ulink></otheraddr></▶
address>
      </affiliation>
    </author>
    <confgroup>
      <confdates>15. až 18. února 2001</confdates>
      <conftitle>SLT 2001</conftitle>
      <confnum>2</confnum>
    </confgroup>
  </articleinfo>
  <abstract>
    <para>Příspěvek posluchače seznámí s jazykem XML, který
přináší mnoho revolučních změn do oblasti elektronického publikování,
výměny a sdílení dat a elektronického obchodu. Kromě základních
```

```
principů XML se příspěvek zmíní i o souvislosti
s dalšími navazujícími technologiemi (stylové jazyky, jazyky pro
definici struktury dokumentu, dotazovací jazyky, jazyky pro tvorbu
odkazů).</para>
  </abstract>
  <section>
    <title>Úvod</title>
    <para>Málokterá technologie se rozšířila tak rychle jako XML. Před
třemi lety o ní skoro nikdo nic nevěděl, a dnes se přitom
používá v mnoha aplikacích. Budeme-li se držet přesné definice
zjistíme, že XML (eXtensible Markup Language) je jednoduchý
rozšiřitelný značkovací jazyk. Co si pod touto definicí představíme
záleží zejména na naší fantazii. V následujícím příspěvku se
proto pokusím vysvětlit, co je to XML a k čemu se dá
použít.</para>
    <para>...</para>
  </section>
</article>
```

Název kořenového elementu se musí shodovat s názvem uvedeným v deklaraci typu dokumentu (`<!DOCTYPE` *název* `...>`).

4.3 Referenční stránky

DocBook byl původně určen jako formát pro výměnu manuálových stránek, a proto není překvapující, že v sobě obsahuje elementy pro popis manuálových stránek a jiných referenčních přehledů. Ukázka použití těchto elementů je na následujícím příkladě.

Příklad 4.4. Ukázka reference – `reference.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE reference PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
                    'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<reference>
  <title>Funkce pro práci s e-mailem</title>
  <titleabbrev>Mail</titleabbrev>
  <partintro>
    <para>Následující funkce umožňují pohodlnou práci s e-mailem.</para>
  </partintro>
  <refentry id="function.mail">
    <refnamediv>
      <refname>mail</refname>
      <refpurpose>Odeslání e-mailu</refpurpose>
    </refnamediv>
    <refsect1>
      <title>Popis</title>
      <funcsynopsis>
        <funcprototype>
          <funcdef>bool <function>mail</function></funcdef>
          <paramdef>string <parameter>to</parameter></paramdef>
          <paramdef>string <parameter>subject</parameter></paramdef>
          <paramdef>string <parameter>message</parameter></paramdef>
          <paramdef>string <parameter>additional_headers</parameter></paramdef>
        </funcprototype>
      </funcsynopsis>
      <para><function>Mail</function> automaticky odmailuje vzkaz
specifikovaný v <parameter>message</parameter> příjemci
```

```
specifikovanému v <parameter>to</parameter>. Přidáním čárky mezi
adresami v <parameter>to</parameter> můžete specifikovat více
příjemců.</para>
    </refsect1>
  </refentry>
  <refentry id="function.ezmlm-hash">
    <refnamediv>
      <refname>ezmlm_hash</refname>
      <refpurpose>Počítá hash hodnotu potřebnou pro EZMLM</refpurpose>
    </refnamediv>
    <refsect1>
      <title>Popis</title>
      <funcsynopsis>
        <funcprototype>
          <funcdef>int <function>ezmlm_hash</function></funcdef>
          <paramdef>string <parameter>addr</parameter></paramdef>
        </funcprototype>
      </funcsynopsis>
    </refsect1>
  </refentry>
</reference>
```

4.4 FAQ

Často potřebujeme do dokumentu zařadit seznam často kladených otázek a odpovědí. DocBook pro to nabízí několik speciálních elementů, celé FAQ se vždy uzavírá do elementu `qandaset`.

Příklad 4.5. Dokument s FAQ – `faqukazka.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE article PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
                  'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<article lang="cs">
  <articleinfo>
    <title>PHP FAQ</title>
    <author>
      <firstname>Jirka</firstname>
      <surname>Kosek</surname>
      <affiliation>
        <address><email>kosek@vse.cz</email></address>
      </affiliation>
    </author>
  </articleinfo>
  <para><emphasis>V následujících odstavcích se pokusím
zodpovědět na nejčastější dotazy a problémy, se kterými se setkávají
především začínající uživatelé systému PHP. Vycházím přitom ze
skutečných dotazů, se kterými jsem setkal během výuky PHP a
v různých diskusních skupinách.</emphasis></para>
  <qandaset>
    <qandaentry>
      <question>
        <para>Nainstaloval jsem si úspěšně PHP42, ale skriptům se
nepředávají proměnné z formulářů.</para>
      </question>
      <answer>
        <para>Tento problém jste si způsobili sami snahou o co
nejvýkonnější server. Distribuce PHP4 pro Windows obsahuje dva
```

```
předpřipravené konfigurační soubory <filename>php.ini</filename>
&ndash; <filename>php.ini-dist</filename> a
<filename>php.ini-optimized</filename>. Většina uživatelů slepě sáhne
po druhém konfiguračním souboru. Ten však vypíná mnoho funkcí
interpretu PHP, aby maximálně urychlil jeho běh. Vypnuté je v něm
i načtení parametrů z formulářů do proměnných. Pro jeho zapnutí
stačí v <filename>php.ini</filename> správně nastavit hodnotu
direktivy <literal>register_globals</literal>:</para>
        <programlisting>register_globals = On</programlisting>
      </answer>
    </qandaentry>
    <qandaentry>
      <question>
        <label>Záludná otázka</label>
        <para>Ani předchozí krok nepomohl, stále se mi některé
proměnné z formulářů nepředávají.</para>
      </question>
      <answer>
        <label>Šalamounská odpověď</label>
        <para>Nejpravděpodobnější je varianta, že někdy
v minulosti jste si v prohlížeči uložili cookie se stejným
názvem jaký má proměnná. Při předávání proměnných skriptům mají
cookies standardně vyšší prioritu než proměnné z formulářů.
Řešením je buď přejmenovat proměnné, nebo si nepotřebné cookies
z prohlížeče vymazat.</para>
        <para>Pokud vaše aplikace používá cookies, vyplatí se používat
prohlížeč, který umožňuje snadné prohlížení a odstraňování
cookies. Takovým prohlížečem je například Mozilla, která získala
ocenění produkt roku. Zkuste v ní příkaz
<menuchoice><guimenu>Tasks</guimenu><guisubmenu>Privacy and Security</guisubmenu>
<guisubmenu>Cookie Manager</guisubmenu><guimenuitem>View Stored
Cookies</guimenuitem></menuchoice>.</para>
      </answer>
    </qandaentry>
  </qandaset>
</article>
```

Pokud máme delší seznam otázek a odpovědí, můžeme je pomocí `qandadiv` rozdělit na několik částí. Jedna položka FAQ může obsahovat libovolný počet odpovědí (i žádnou) na jednu otázku.

4.5 Vkládání obrázků

Pro vkládání obrázků je nejlepší používat elementy `mediaobject` a `inlinemediaobject`. Ve starších verzích DocBooku se používal element `graphics` a `inlinegraphics`. Výhodou `mediaobject`u je možnost vložení obrázku do dokumentu v několika formátech – například ve vektorové a bitmapové podobě. Při formátování se pak automaticky použije ten obrázek, který nejvíce vyhovuje požadavkům generovaného výstupního formátu.

Příklad 4.6. Dokument s obrázky – `obrazky.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE chapter PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
                  'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<chapter lang="cs">
  <title>Kapitola s obrázky</title>

  <mediaobject>
```

```
  <imageobject>
    <imagedata fileref="schema.eps" format="EPS"
               srccredit="Jiří Kosek, 2000"/>
  </imageobject>
  <imageobject>
    <imagedata fileref="schema.pdf"/>
  </imageobject>
  <imageobject>
    <imagedata fileref="schema.wmf" format="WMF"/>
  </imageobject>
  <imageobject>
    <imagedata fileref="schema.png" format="PNG"/>
  </imageobject>
  <textobject>
    <para>Schéma WAPu &ndash; delší text, pro který se generuje
    textový popis</para>
  </textobject>
  <textobject>
    <phrase>Schéma WAPu &ndash; krátké do atributu ALT</phrase>
  </textobject>
</mediaobject>

<figure>
  <title>Obrázek s popisem zvětšený na 200 % volného místa</title>
  <mediaobject>
    <imageobject>
      <imagedata fileref="schema.png" format="PNG" scale="200"/>
    </imageobject>
  </mediaobject>
</figure>

<figure>
  <title>Obrázek s popisem zmenšený na 25 % volného místa</title>
  <mediaobject>
    <imageobject>
      <imagedata fileref="schema.png" format="PNG" scale="25"/>
    </imageobject>
  </mediaobject>
</figure>

<figure>
  <title>Obrázek s popisem roztažený na šířku stránky</title>
  <mediaobject>
    <imageobject>
      <imagedata fileref="schema.png" format="PNG" width="100%"/>
    </imageobject>
  </mediaobject>
</figure>

<figure>
  <title>Obrázek s šířkou 4cm zarovnaný doprava</title>
  <mediaobject>
    <imageobject>
      <imagedata fileref="schema.png" format="PNG"
                 width="4cm" align="right"/>
    </imageobject>
  </mediaobject>
</figure>
```

```
<figure>
  <title>Obrázek je podle potřeby proporcionálně zvětšen do plochy
  2 &times; 5 cm</title>
  <mediaobject>
    <imageobject>
      <imagedata width="2cm" depth="5cm" fileref="schema.png"
      format="PNG"/>
    </imageobject>
  </mediaobject>
</figure>

<figure>
  <title>Obrázek zmenšený na 50 % skutečné velikosti</title>
  <mediaobject>
    <imageobject>
      <imagedata contentwidth="50%" fileref="schema.png"
      format="PNG"/>
    </imageobject>
  </mediaobject>
</figure>

<figure>
  <title>Obrázek zobrazený jako 2 &times; 10 cm</title>
  <mediaobject>
    <imageobject>
      <imagedata contentwidth="2cm" contentdepth="10cm"
      fileref="schema.png" format="PNG"/>
    </imageobject>
  </mediaobject>
</figure>

<figure>
  <title>Obrázek se umístí do prostoru 8 &times; 8 cm
  a zmenší se na 1/2 původní velikosti</title>
  <mediaobject>
    <imageobject>
      <imagedata width="8cm" depth="8cm" scale="50"
      fileref="schema.png" format="PNG"/>
    </imageobject>
  </mediaobject>
</figure>

<figure>
  <title>Obrázek zobrazený ve své originální velikosti</title>
  <mediaobject>
    <imageobject>
      <imagedata contentwidth="100%" contentdepth="100%"
      fileref="schema.png" format="PNG"/>
    </imageobject>
  </mediaobject>
</figure>

<figure>
  <title>Obrázek postaru</title>
  <graphic fileref="schema.png" format="PNG"/>
</figure>
```

```
  <para>Zařazení obrázku do odstavce <inlinemediaobject>
      <imageobject>
        <imagedata fileref="sipka.png"/>
      </imageobject>
      <textobject>
        <phrase>&rArr;</phrase>
      </textobject>
    </inlinemediaobject> a text pokračuje dál.</para>

</chapter>
```

Umístění a zpracování obrázků lze ovládat pomocí několika atributů. Jediný problém je v tom, že interpretování těchto atributů ve velké míře závisí na konkrétní aplikaci, která provádí formátování. Je dobré vyzkoušet, jaké formáty a jak podporují programy použité v našem systému.

V části věnované úpravám stylů se ještě podíváme na to, jak lze ovlivňovat automatický výběr jedné z variant obrázku, případně automaticky doplňovat příponu obrázku.

4.6 Tabulky

Tabulky se do docbookových dokumentů zapisují podobným způsobem jako v HTML. Tabulka se zapisuje po řádkách a jednotlivé řádky pak po buňkách zleva doprava. V současné době obsahuje DocBook pro zápis tabulek takzvaný CALS model, který je na poli SGML/XML de facto standard. DocBook 5.0 bude používat zjednodušenou verzi CALS [31]. Je proto lepší již dnes nepoužívat věci, které budou časem z DocBooku odstraněny. Následující příklady se omezují na použití elementů a atributů, které budou dostupné i v DocBooku 5.0.

Příklad 4.7. Ukázky tabulek – `tabulky.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE chapter PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
                  'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<chapter lang="cs">
  <title>Kapitola s tabulkami</title>

  <table>
    <title>Pokusná tabulka</title>
    <tgroup cols="3">
      <colspec colwidth="4cm"/>
      <colspec colwidth="2cm"/>
      <colspec colwidth="10cm"/>
      <thead>
        <row>
          <entry>Výstupní formát</entry>
          <entry>Stylový jazyk</entry>
          <entry>Potřebné nástroje</entry>
        </row>
      </thead>
      <tbody>
        <row>
          <entry>jedna HTML stránka</entry>
          <entry>DSSSL</entry>
          <entry>Jade</entry>
        </row>
        <row>
          <entry>jedna HTML stránka</entry>
```

```
        <entry>XSL</entry>
        <entry>libovolný XSLT procesor</entry>
      </row>
      <row>
        <entry>sada HTML stránek</entry>
        <entry>DSSSL</entry>
        <entry>Jade</entry>
      </row>
      <row>
        <entry>sada HTML stránek</entry>
        <entry>XSL</entry>
        <entry>libovolný XSLT procesor</entry>
      </row>
      <row>
        <entry>RTF</entry>
        <entry>DSSSL</entry>
        <entry>Jade</entry>
      </row>
      <row>
        <entry>PDF</entry>
        <entry>DSSSL</entry>
        <entry>Jade+JadeTeX nebo Jade+Word+Distiller</entry>
      </row>
      <row>
        <entry>PDF</entry>
        <entry>XSL</entry>
        <entry>libovolný XSLT procesor + FO procesor (např. PassiveTeX)</entry>
      </row>
    </tbody>
  </tgroup>
</table>

<table>
  <title>Tabulka se sloučenými buňkami</title>
  <tgroup cols="3">
    <colspec colname="c1" align="left"/>
    <colspec colname="c2" align="char" char=","/>
    <colspec colname="c3" align="char" char=","/>
    <thead>
      <row>
        <entry morerows="1" align="center" valign="middle">Měsíc</entry>
        <entry namest="c2" nameend="c3" align="center">Prodej zboží</entry>
      </row>
      <row>
        <entry align="center">A</entry>
        <entry align="center">B</entry>
      </row>
    </thead>
    <tbody>
      <row>
        <entry>Leden</entry>
        <entry>865,54</entry>
        <entry>16,3</entry>
      </row>
      <row>
        <entry>Únor</entry>
        <entry>917,7</entry>
        <entry>8,1</entry>
```

```
      </row>
      <row>
        <entry>Březen</entry>
        <entry>1036,8</entry>
        <entry>18,9</entry>
      </row>
    </tbody>
  </tgroup>
</table>

<table>
  <title>Tabulka s rámečkem, ale bez linek mezi buňkami</title>
  <tgroup cols="3" colsep="0" rowsep="0">
    <colspec align="left"/>
    <colspec align="char" char=","/>
    <colspec align="char" char=","/>
    <tbody>
      <row>
        <entry>Leden</entry>
        <entry>865,54</entry>
        <entry>16,3</entry>
      </row>
      <row>
        <entry>Únor</entry>
        <entry>917,7</entry>
        <entry>8,1</entry>
      </row>
      <row>
        <entry>Březen</entry>
        <entry>1036,8</entry>
        <entry>18,9</entry>
      </row>
    </tbody>
  </tgroup>
</table>

<table frame="topbot">
  <title>Tabulka s linkou nahoře a dole</title>
  <tgroup cols="3" colsep="0" rowsep="0">
    <colspec align="left"/>
    <colspec align="char" char=","/>
    <colspec align="char" char=","/>
    <tbody>
      <row>
        <entry>Leden</entry>
        <entry>865,54</entry>
        <entry>16,3</entry>
      </row>
      <row>
        <entry>Únor</entry>
        <entry>917,7</entry>
        <entry>8,1</entry>
      </row>
      <row>
        <entry>Březen</entry>
        <entry>1036,8</entry>
        <entry>18,9</entry>
      </row>
```

```
      </tbody>
    </tgroup>
  </table>

  <table frame="topbot">
    <title>Tabulka s linkou nahoře, dole a mezi sloupci</title>
    <tgroup cols="3" colsep="1">
      <colspec align="left"/>
      <colspec align="char" char=","/>
      <colspec align="char" char=","/>
      <tbody>
        <row>
          <entry>Leden</entry>
          <entry>865,54</entry>
          <entry>16,3</entry>
        </row>
        <row>
          <entry>Únor</entry>
          <entry>917,7</entry>
          <entry>8,1</entry>
        </row>
        <row>
          <entry>Březen</entry>
          <entry>1036,8</entry>
          <entry>18,9</entry>
        </row>
      </tbody>
    </tgroup>
  </table>

  <para>Odstavec textu, za kterým bude tabulka bez nadpisu.</para>

  <informaltable>
    <tgroup cols="3">
      <colspec align="left"/>
      <colspec align="char" char=","/>
      <colspec align="char" char=","/>
      <tbody>
        <row>
          <entry>Leden</entry>
          <entry>865,54</entry>
          <entry>16,3</entry>
        </row>
        <row>
          <entry>Únor</entry>
          <entry>917,7</entry>
          <entry>8,1</entry>
        </row>
        <row>
          <entry>Březen</entry>
          <entry>1036,8</entry>
          <entry>18,9</entry>
        </row>
      </tbody>
    </tgroup>
  </informaltable>

</chapter>
```

Každá tabulka se musí skládat alespoň z jednoho elementu `tgroup`. Narozdíl od HTML musíme vždy povinně určit počet sloupců v atributu `cols`. Dovnitř `tgroup` se pak vkládají definice jednotlivých sloupců (`colspec`), záhlaví tabulky (`thead`) a tělo tabulky (`tbody`). Řádky se zapisují do elementu `row` a buňky do `entry`.

Od verze 4.3 umožňuje DocBook i alternativní zápis tabulek používající HTML syntaxi.

4.7 Seznamy

Seznamy asi nepotřebují příliš velký komentář.

Příklad 4.8. Ukázka seznamů – `seznamy.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE chapter PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
                  'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<chapter lang="cs">
  <title>Kapitola se seznamy</title>

  <itemizedlist>
    <listitem>
      <para>jablka</para>
    </listitem>
    <listitem>
      <para>hrušky</para>
    </listitem>
    <listitem>
      <para>švestky</para>
    </listitem>
  </itemizedlist>

  <orderedlist>
    <listitem>
      <para>Praha</para>
    </listitem>
    <listitem>
      <para>Brno</para>
    </listitem>
    <listitem>
      <para>Bratislava</para>
    </listitem>
  </orderedlist>

  <itemizedlist>
    <title>Ovoce:</title>
    <listitem>
      <para>jablka</para>
    </listitem>
    <listitem>
      <para>hrušky</para>
    </listitem>
    <listitem>
      <para>švestky</para>
    </listitem>
  </itemizedlist>

  <orderedlist continuation="continues">
```

```
    <title>Číslovaný seznam může pokračovat v přerušeném číslování</title>
    <listitem>
      <para>Ostrava</para>
    </listitem>
    <listitem>
      <para>Plzeň</para>
    </listitem>
  </orderedlist>

  <para>Následující seznam bude číslován římskými číslicemi</para>

  <orderedlist numeration="upperroman">
    <listitem>
      <para>SGML</para>
    </listitem>
    <listitem>
      <para>HTML</para>
    </listitem>
    <listitem>
      <para>XML</para>
    </listitem>
  </orderedlist>

  <variablelist>
    <varlistentry>
      <term>XML</term>
      <listitem>
        <para>eXtensible Markup Language</para>
      </listitem>
    </varlistentry>
    <varlistentry>
      <term>SGML</term>
      <listitem>
        <para>Standard Generalized Markup Language</para>
      </listitem>
    </varlistentry>
    <varlistentry>
      <term>HTML</term>
      <term>XHTML</term>
      <listitem>
        <para>Jazyky pro tvorbu webových stránek odvozené od SGML,
        resp. XML.</para>
      </listitem>
    </varlistentry>
    <varlistentry>
      <term><emphasis>XSL</emphasis></term>
      <listitem>
        <para>XSL = eXtensible Stylesheet Language. Tento popis
        uděláme delší ať je vidět, že se zalomí do několiak řádek.</para>
        <para>Můžeme samozřejmě použít i více odstavců.</para>
      </listitem>
    </varlistentry>
  </variablelist>

</chapter>
```

Seznamy do sebe můžeme samozřejmě navzájem zanořovat.

4.8 Odkazy

Ve větších dokumentech se bez odkazů neobejdeme. Pokud chceme vytvořit odkaz na určitou část dokumentu (kapitolu, obrázek apod.) musíme cílovému elementu přiřadit jednoznačný identifikátor v atributu `id`.

Na takto označené místo pak můžeme vytvořit odkaz pomocí elementů `xref` a `link`. Rozdíl je v tom, že při použití prvního elementu se automaticky generuje text odkazu, u `link`u text určujeme sami.

Internetové odkazy se zapisují pomocí elementu `ulink`.

Příklad 4.9. Ukázka odkazů – `odkazy.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE book PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
              'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<book lang="cs">
  <bookinfo>
    <title>První pokusná kniha</title>
    <subtitle>Uživatelská příručka</subtitle>
  </bookinfo>
  <preface>
    <title>Úvod</title>
    <para>Odstavec textu. Více informací naleznete na mé <ulink
url="http://www.kosek.cz">domovské stránce</ulink>. Seznam najdete na
adrese <ulink
url="http://www.seznam.cz">http://www.seznam.cz</ulink>.</para>
  </preface>
  <chapter>
    <title>První kapitola</title>
    <para>Text první kapitoly</para>
    <para>Popis instalace programů používaných v této kapitole najdete
    v příloze <xref linkend="apb"/>.</para>
    <para>Popis instalace programů používaných v této kapitole najdete
    v <link linkend="apb">samostatné příloze B</link>.</para>
    <para>Popis instalace programů používaných v této kapitole najdete
    v <xref endterm="apb-short" linkend="apb"/>.</para>
    <para>Popis instalace programů nepoužívaných v této kapitole najdete
    v <xref linkend="apa"/>.</para>
  </chapter>
  <chapter>
    <title>Druhá kapitola</title>
    <para>Text druhé kapitoly</para>
    <para>Informace o prodeji naleznete v tabulce <xref
    linkend="tab.prodeje"/>.</para>
    <table id="tab.prodeje">
      <title>Prodeje za Q1</title>
      <tgroup cols="3">
        <colspec align="left"/>
        <colspec align="char" char=","/>
        <colspec align="char" char=","/>
        <tbody>
          <row>
            <entry>Leden</entry>
            <entry>865,54</entry>
          <entry>16,3</entry>
          </row>
```

```
          <row>
            <entry>Únor</entry>
            <entry>917,7</entry>
            <entry>8,1</entry>
          </row>
          <row>
            <entry>Březen</entry>
            <entry>1036,8</entry>
            <entry>18,9</entry>
          </row>
        </tbody>
      </tgroup>
    </table>
    <para>Obrázek <xref linkend="pic.wap-schema"/> znázorňuje schéma
WAP sítě.</para>
    <para><xref xrefstyle="template:Obrázek %t (%n)" linkend="pic.wap-schema"/>
 znázorňuje schéma WAP sítě.</para>
    <para>Obrázek <xref xrefstyle="select: labelnumber" linkend="pic.wap-schema"/>
 znázorňuje schéma WAP sítě.</para>
  </chapter>
  <appendix id="apa" xreflabel="Příloha A - Obrázková příloha">
    <title>Obrázková příloha</title>
    <figure float="0" id="pic.wap-schema">
      <title>Schéma WAPu</title>
      <mediaobject>
        <imageobject>
          <imagedata fileref="schema.png" format="PNG"/>
        </imageobject>
      </mediaobject>
    </figure>
  </appendix>
  <appendix id="apb">
    <title>Druhá příloha</title>
    <titleabbrev id="apb-short">2. příloha</titleabbrev>
    <para>Text přílohy</para>
    <para>...</para>
  </appendix>
</book>
```

Automaticky generovaný text odkazu je ovlivněn několika atributy. Jednak můžeme u odkazu říci, z kterého elementu se má vytáhnout text použitý pro odkaz (atribut `endterm`). U každého elementu také můžeme v atributu `xreflabel` nastavit text, který se použije pro vytvoření textu odkazu místo samotného obsahu elementu.

V češtině je trošku problém s automaticky generovanými názvy jako „Obrázek“, „Tabulka“ apod. V textu se mohou vyskytovat v několika různých pádech, s tím však styl nemůže počítat. Pro jazyky jako čeština jsou styly upraveny tak, že vkládají pouze číslo obrázku či tabulky a text dopíše autor sám ve správném pádu. V části věnované úpravě stylů si ukážeme, jak modifikovat tvar křížového odkazu.

Tip

Ve větších dokumentech se člověk může snadno ztratit v identifikátorech, které používá. Je proto dobré zvolit si nějaký systém. Například můžeme na začátek identifikátoru vkládat prefix, který určí typ objektu, který označuje. Například `pic.` pro obrázky a `tab.` pro tabulky. Obvyklé bývá také vložení čísla kapitoly nebo jejího identifikátoru jako součást identifikátoru vnořeného elementu.

4.9 Popis třídy, rozhraní, funkce apod.

V DocBooku 4.0 bylo přidáno mnoho nových elementů pro popis tříd a rozhraní. Pomocí atributů můžeme určit pro jaký jazyk je definice určena. Pro některé jazyky umí styly upravit formátování dle konvencí daného jazyka.

Příklad 4.10. Ukázka definice třídy a rozhraní – `trida.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE chapter PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
               'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<chapter lang="cs">
  <title>Ukázky definice tříd a rozhraní</title>
  <!-- Příklady jsou převzaty z DocBook 4.0 Update Reference
       http://www.docbook.org/tdg/40update/classsynopsis.html
       a drobně upraveny -->

  <section>
    <title>Třída v Javě</title>

    <classsynopsis language="java">
      <ooclass>
        <modifier>public</modifier>
        <classname>TextFileWriter</classname>
      </ooclass>
      <ooclass>
        <classname>HandlerBase</classname></ooclass>
      <fieldsynopsis>
        <modifier>private</modifier>
        <type>Writer</type>
        <varname>writer</varname>
      </fieldsynopsis>

      <fieldsynopsis>
        <modifier>public</modifier>
        <type>String</type>
        <varname>writerName</varname>
        <initializer>"MyWriter"</initializer>
      </fieldsynopsis>

      <methodsynopsis>
        <modifier>static</modifier>
        <modifier>public</modifier>
        <void/>
        <methodname>write</methodname>
        <methodparam>
            <type>ResultTreeFragment</type>
          <parameter>frag</parameter>
        </methodparam>
        <methodparam>
          <type>String</type>
          <parameter>file</parameter>
        </methodparam>
        <exceptionname>SAXException</exceptionname>
      </methodsynopsis>

    </classsynopsis>
```

```
  </section>

  <section>
    <title>Definice rozhraní v IDL</title>

    <classsynopsis class="interface" language="idl">
      <oointerface>
        <interfacename>Element</interfacename></oointerface>
      <oointerface>
        <interfacename>Node</interfacename></oointerface>

      <fieldsynopsis>
        <modifier>readonly</modifier>
        <modifier>attribute</modifier>
        <type>DOMString</type>
        <varname>tagName</varname>
      </fieldsynopsis>

      <methodsynopsis>
        <type>DOMString</type>
        <methodname>getAttribute</methodname>
        <methodparam>
          <modifier>in</modifier>
          <type>DOMString</type>
          <parameter>name</parameter>
        </methodparam>
      </methodsynopsis>

      <methodsynopsis>
        <void/>
        <methodname>setAttribute</methodname>
        <methodparam>
          <modifier>in</modifier>
          <type>DOMString</type>
          <parameter>name</parameter>
        </methodparam>
        <methodparam>
          <modifier>in</modifier>
          <type>DOMString</type>
          <parameter>value</parameter>
        </methodparam>
        <exceptionname>DOMException</exceptionname>
      </methodsynopsis>
    </classsynopsis>

  </section>

</chapter>
```

4.10 Komentované výpisy kódu

DocBook nabízí několik elementů pro snadné komentování výpisů zdrojového kódu.

Příklad 4.11. Komentované výpisy – `callouts.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE chapter PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
                  'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<chapter lang="cs">
  <title>Ukázka <quote>callouts</quote></title>
  <example>
    <title>Malá exkurze do SGML DTD pro jazyk HTML</title>
    <programlisting>&lt;!ELEMENT UL - - (LI)+ &gt;                    <co id="co.element"/>
&lt;!ATTLIST UL                                <co id="co.atributy"/>
  type       (disc|square|circle) #IMPLIED  <co id="co.atribut"/>
  &gt;</programlisting>
    <calloutlist>
      <callout arearefs="co.element">
        <para>Deklarace elementu <sgmltag class="element">ul</sgmltag>
(nečíslovaný seznam). Element musí mít počáteční i ukončovací tag a
může obsahovat pouze elementy <sgmltag
class="element">li</sgmltag>.</para>
      </callout>
      <callout arearefs="co.atributy">
        <para>Deklarace atributů použitelných u <sgmltag
          class="element">ul</sgmltag>.</para>
      </callout>
      <callout arearefs="co.atribut">
        <para>Atribut <sgmltag class="attribute">type</sgmltag> může
mít jednu z hodnot <symbol>disc</symbol>, <symbol>square</symbol>
a <symbol>circle</symbol>.</para>
      </callout>
    </calloutlist>
  </example>

  <para>Další ukázka ukazuje možnost označení oblastí pomocí
souřadnic. Je převzata z testovacích dokumentů pro DSSSL styly.</para>

  <programlistingco lang="en">
    <areaspec>
      <area id="prologue" coords="1"/>
      <area id="skipeof" coords="4"/>
      <areaset id="xreq" coords="">
        <area id="require1" coords="9"/>
        <area id="require2" coords="10"/>
      </areaset>
      <area id="use" coords="11 12"/>
      <area id="funccall" coords="27"/>
    </areaspec>
    <programlisting>@rem = '--*-Perl-*--
@echo off
perl.exe %_batchname %$
goto endofperl
@rem ';

# Compress mail...

require 'n:/home/nwalsh/lib/cygnus.pl';
require 'timelocal.pl';
use Cwd;
```

```
select (STDERR); $| = 1;
select (STDOUT); $| = 1;

@DIRS = ("/home/nwalsh/Mail");
while (@DIRS) {
    $dir = shift @DIRS;
    opendir (DIR, $dir);
    while ($fname = readdir(DIR)) {
        $file = "$dir/$fname";
        next if ! -d $file;
        next if $fname =~ /^\.\.?$/;

        print "$file\n";
        push (@DIRS, $file);
        &amp;compress ($file);
    }
}

exit;</programlisting>
    <calloutlist>
      <callout arearefs="prologue">
        <para>The prologue handles embedding a Perl script in a DOS
batch file.</para>
      </callout>
      <callout arearefs="skipeof">
        <para>The <literal>goto</literal> statement, interpreted by
the DOS batch file interpreter, skips over the body of the Perl
script.</para>
      </callout>
      <callout arearefs="require1">
        <para>The <literal>require</literal> statement sources in
external program fragments.</para>
      </callout>
      <callout arearefs="use">
        <para>The <literal>use</literal> statement is similar, but has
additional utility.  It is a Perl5 function.  (Note that this callout
area specifies both a line and a column.)</para>
      </callout>
      <callout arearefs="funccall">
        <para>This is a user subroutine call.</para>
      </callout>
    </calloutlist>
  </programlistingco>

</chapter>
```

4.11 Seznamy literatury

DocBook nabízí poměrně bohaté možnosti pro zápis bibliografických záznamů.

Příklad 4.12. Ukázka seznamu literatury – `literatura.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE bibliography PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
          'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<bibliography id="literatura">
<title>Literatura a další zajímavé odkazy</title>
```

```
<biblioentry>
<abbrev>REC-XSLT</abbrev>
<editor>
<firstname>James</firstname>
<surname>Clark</surname>
</editor>
<title><ulink url="http://www.w3.org/TR/xslt">XSL Transformations
(XSLT) Version 1.0</ulink></title>
<publishername>W3C</publishername>
<pubdate>1999</pubdate>
</biblioentry>

<biblioentry id="bib.ptux">
<title><ulink url="http://www.cranesoftwrights.com">Practical
Transformation Using XSLT and XPath</ulink></title>
<publisher>
<publishername>Crane Softwrights</publishername>
</publisher>
<pubdate>2000</pubdate>
<isbn>1-894049-04-7</isbn>
</biblioentry>

<biblioentry>
<abbrev>SGML-NT</abbrev>
<author>
<firstname>Markus</firstname>
<surname>Hoenicka</surname>
</author>
<title>SGML for NT</title>
<subtitle>A brief tutorial how to set up a free SGML editing and
publishing system for Windows NT</subtitle>
<releaseinfo>URL: <ulink
url="http://ourworld.compuserve.com/homepages/hoenicka_markus/ntsgml.html">http://▶
ourworld.compuserve.com/homepages/hoenicka_markus/ntsgml.html</ulink></releaseinfo>
</biblioentry>

<biblioentry>
<abbrev>DSSSL</abbrev>
<title>Information technology &ndash; Processing languages &ndash; Document Style
Semantics and Specification Language (DSSSL)</title>
<subtitle>ISO/IEC 10179:1996(E)</subtitle>
<releaseinfo>URL: <ulink
url="ftp://ftp.ornl.gov/pub/sgml/WG8/DSSSL/dsssl96b.pdf">ftp://ftp.ornl.gov/pub/sgml/WG8/▶
DSSSL/dsssl96b.pdf</ulink></releaseinfo>
<releaseinfo>URL: <ulink
url="ftp://ftp.ornl.gov/pub/sgml/WG8/DSSSL/readme.htm">ftp://ftp.ornl.gov/pub/sgml/WG8/DSSSL/▶
readme.htm</ulink></releaseinfo>
</biblioentry>

<biblioentry>
<abbrev>XSLTQUICK</abbrev>
<title><ulink
url="http://www.mulberrytech.com/quickref/XSLTquickref.pdf">XSLT and
XPath Quick Reference</ulink></title>
<publishername>Mulberry Technologies</publishername>
<pubdate>2000</pubdate>
</biblioentry>
```

```
<biblioentry>
<abbrev>XPATHTUT</abbrev>
<author>
<firstname>Miloslav</firstname>
<surname>Nič</surname>
</author>
<title><ulink
url="http://www.zvon.org/xxl/XPathTutorial/General/examples.html">XPath
Tutorial</ulink></title>
</biblioentry>

<biblioentry>
<abbrev>XSLTUT</abbrev>
<author>
<firstname>Miloslav</firstname>
<surname>Nič</surname>
</author>
<title><ulink
url="http://www.zvon.org/xxl/XSLTutorial/Books/Book1/index.html">XSLT Tutorial</ulink></title>
</biblioentry>

<biblioentry>
<abbrev>GNU-WWW</abbrev>
<author>
<firstname>Lenka</firstname>
<surname>Třísková</surname>
</author>
<title>GNU nástroje pro tvorbu WWW stránek</title>
<publisher>
<publishername>Grada Publishing</publishername>
</publisher>
<pubdate>2000</pubdate>
<isbn>ISBN 80-7169-861-X</isbn>
<pagenums>244</pagenums>
</biblioentry>

<biblioentry>
<abbrev>TDG</abbrev>
<authorgroup>
<author>
<firstname>Norman</firstname>
<surname>Walsh</surname>
</author>
<author>
<firstname>Leonard</firstname>
<surname>Muellner</surname>
</author>
</authorgroup>
<title>DocBook</title>
<subtitle>The Definitive Guide</subtitle>
<pubdate>1999</pubdate>
<edition>1</edition>
<isbn>ISBN: 156592-580-7</isbn>
<pagenums>648</pagenums>
<releaseinfo>URL: <ulink
url="http://www.docbook.org/tdg/html/docbook.html">http://www.docbook.org/tdg/html/▶
docbook.html</ulink></releaseinfo>
```

```
</biblioentry>

<biblioentry>
<abbrev>DocBook4</abbrev>
<authorgroup>
<author>
<firstname>Norman</firstname>
<surname>Walsh</surname>
</author>
<author>
<firstname>Leonard</firstname>
<surname>Muellner</surname>
</author>
</authorgroup>
<title>DocBook 4.0</title>
<subtitle>Update Reference</subtitle>
<releaseinfo>URL: <ulink url="http://www.docbook.org/tdg/40update/">http://www.docbook.org/▶
tdg/40update/</ulink></releaseinfo>
</biblioentry>

<biblioentry>
<abbrev>XSL-DOC</abbrev>
<authorgroup>
<author>
<firstname>Norman</firstname>
<surname>Walsh</surname>
</author>
<author>
<firstname>Bob</firstname>
<surname>Stayton</surname>
</author>
</authorgroup>
<title>DocBook XSL Stylesheet Documentation</title>
<releaseinfo>URL: <ulink url="http://www.nwalsh.com/docbook/xsl/doc/">http://www.nwalsh.com/▶
docbook/xsl/doc/</ulink></releaseinfo>
</biblioentry>

<biblioentry>
<abbrev>TableExchange</abbrev>
<author>
<firstname>Norman</firstname>
<surname>Walsh</surname>
</author>
<title>Organization for the Advancement of Structured Information
Standards (OASIS) Technical Memorandum TR 9901:1999</title>
<subtitle>XML Exchange Table Model Document Type Definition</subtitle>
<pubdate>1999</pubdate>
<publisher>
<publishername>OASIS</publishername>
</publisher>
<releaseinfo>URL: <ulink
url="http://www.oasis-open.org/html/tm9901.htm">http://www.oasis-open.org/html/tm9901.htm</▶
ulink></releaseinfo>
</biblioentry>

<biblioentry>
<abbrev>DSSSL-DOC</abbrev>
<author>
```

```
<firstname>Norman</firstname>
<surname>Walsh</surname>
</author>
<title>The Modular DocBook Stylesheets</title>
<releaseinfo>URL: <ulink
url="http://www.nwalsh.com/docbook/dsssl/doc/">http://www.nwalsh.com/docbook/dsssl/doc/</▶
ulink></releaseinfo>
</biblioentry>

<biblioentry>
<abbrev>DB-L</abbrev>
<title>Archiv mailing listu
<email>docbook@lists.oasis-open.org</email></title>
<releaseinfo>URL: <ulink
url="http://lists.oasis-open.org/archives/docbook/">http://lists.oasis-open.org/archives/▶
docbook/</ulink></releaseinfo>
</biblioentry>

<biblioentry>
<abbrev>DBAPPS-L</abbrev>
<title>Archiv mailing listu
<email>docbook-apps@lists.oasis-open.org</email></title>
<releaseinfo>URL: <ulink
url="http://lists.oasis-open.org/archives/docbook-apps/">http://lists.oasis-open.org/archives/▶
docbook-apps/</ulink></releaseinfo>
</biblioentry>

<biblioentry>
<abbrev>DSSSL-Pages</abbrev>
<title>Stránky věnované jazyku DSSSL</title>
<releaseinfo>URL: <ulink url="http://www.netfolder.com/DSSSL/">http://www.netfolder.com/DSSSL/▶
</ulink></releaseinfo>
</biblioentry>

</bibliography>
```

Odkaz na zdroj se vytváří buď pomocí elementu `xref`, který ukazuje na `id` zdroje, nebo pomocí elementu `citation`, který obsahuje zkratku zdroje uvedenou v seznamu literatury v elementu `abbrev`.

Nelíbí-li se nám formátování seznamu literatury, můžeme si upravit styly. To je však velice náročné, proto se obvykle volí jiná cesta. Nejjednodušší je jednotlivé zdroje zapisovat pomocí elementu `bibliomixed`, který dovoluje ručně zadat veškerý text včetně různých interpunkčních znamének.

```
<bibliomixed>
<abbrev>Ko99</abbrev>
Kosek, J.: <citetitle>PHP &ndash; tvorba interaktivních internetových
aplikací</citetitle>. Grada Publishing. Praha 1999.
</bibliomixed>
```

Citujeme-li hodně, můžeme použít některý z nástrojů pro práci s citačními databázemi, který podporuje DocBook. Například RefDB[1], BibTeXML[2] nebo JReferences[3].

[1] http://refdb.sourceforge.net/
[2] http://bibtexml.sourceforge.net/
[3] http://jreferences.sourceforge.net/

4.12 Nejběžnější inline elementy

Inline elementů existuje v DocBooku několik desítek. Následující příklad ukazuje použití těch elementů, jejichž zpracování závisí na hodnotě atributu nebo na kontextu, ve kterém jsou použity.

Příklad 4.13. Ukázka inline elementů – `inline.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE chapter PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
                  'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<chapter lang="cs">
<title>Ukázka vybraných inline elementů</title>

<section>
<title>Zápis SGML/XML značkování</title>

<literallayout>
<sgmltag>element</sgmltag>
<sgmltag class="attribute">atribut</sgmltag>
<sgmltag class="attvalue">hodnota atributu</sgmltag>
<sgmltag class="element">element</sgmltag>
<sgmltag class="emptytag">empty</sgmltag> &ndash; prázdný element
<sgmltag class="starttag">tag</sgmltag> &ndash; počáteční tag
<sgmltag class="endtag">tag</sgmltag> &ndash; ukončovací tag
<sgmltag class="genentity">mdash</sgmltag> &ndash; odkaz na entitu
<sgmltag class="numcharref">x201C</sgmltag> &ndash; odkaz na číselnou znakovou entitu
<sgmltag class="paramentity">iso-pub</sgmltag> &ndash; parametrická entita
<sgmltag class="pi">Pub caret</sgmltag> &ndash; SGML instrukce pro zpracování
<sgmltag class="xmlpi">xml-stylesheet href="styl.css" type="text/css"</sgmltag> &ndash; XML ▶
instrukce pro zpracování
<sgmltag class="sgmlcomment">nějaký komentář</sgmltag>
</literallayout>

</section>

<section>
<title>GUI elementy &ndash; menu, klávesové zkratky apod.</title>

<para>Náš úžasně geniální program se spouští pomocí kliknutím na ikonu
<guiicon>GENPROG</guiicon>. Program lze ukončit stiskem <keycombo>
<keycap>Alt</keycap><keycap>F4</keycap></keycombo> nebo pomocí příkazu
z menu <menuchoice><guimenu><accel>S</accel>oubor</guimenu>
<guimenuitem><accel>K</accel>onec</guimenuitem></menuchoice>.</para>

<para>Soubor lze otevřít pomocí
<menuchoice>
<shortcut>
<keycombo><keycap>Ctrl</keycap><keycap>O</keycap></keycombo>
</shortcut>
<guimenu>Soubor</guimenu>
<guimenuitem>Otevřít</guimenuitem>
</menuchoice>.</para>

</section>

<section>
<title>Pár dalších ukázek</title>
```

```
<para>Moje e-mailová adresa je
<email>jirka@kosek.cz</email>. Neposílejte mi nic
<emphasis>zbytečně</emphasis>. Myslím to <emphasis
role="bold">vážně</emphasis>. Na následujícím <phrase
role="important">kusu textu</phrase> si ukážeme úpravy stylů.</para>

</section>

</chapter>
```

Kapitola 5. Úprava DSSSL stylů

Průběh formátování pomocí DSSSL stylů lze řídit mnoha parametry. Kompletní přehled parametrů je popsán v dokumentaci k DSSSL stylům [10]. Složitější úpravy je nutné naprogramovat přímo v DSSSL [25].

5.1 Úprava chování stylu pomocí parametrů

Pokud chceme upravit chování stylu, musíme si vytvořit nový styl. Ten přitom naimportuje původní styl a předefinuje potřebné parametry. DSSSL styly jsou vlastně SGML dokumenty, takže existující styl se importuje pomocí mechanismu entit. Kostra našeho stylu může vypadat zhruba takto:

```
<!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN" [
<!ENTITY docbook.dsl SYSTEM "c:/docbook/dsssl/print/docbook.dsl" CDATA DSSSL>
]>

<style-specification id="my-docbook-print" use="docbook">

;; naše úpravy

</style-specification>

<external-specification id="docbook" document="docbook.dsl">
```

Musíme samozřejmě na druhé řádce upravit cestu tak, aby ukazovala na místo, kde se na našem systému nachází styl, ze kterého vycházíme. Z podstaty DSSSL existují pro DocBook dva druhy stylů – pro tisk (adresář `print`) a pro HTML (adresář `html`) – které jsou zcela samostatné.

Když chceme pomocí upraveného stylu zformátovat nějaký dokument, předáme Jade jako parametr náš styl:

```
jade -d tisk.dsl -t rtf c:\docbook\jade\xml.dcl prvni.xml
```

5.1.1 Běžné úpravy pro tisk

Následující příklad ukazuje, jak se dají změnit nejběžnější parametry. Pro zápis hodnot true a false se používají sekvence `#t` a `#f`. Některé parametry mají jako svoji hodnotu seznam hodnot.

Příklad 5.1. Úprava DSSSL stylu pro tisk – `tisk.dsl`

```
<!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN" [
<!ENTITY docbook.dsl SYSTEM "c:/docbook/dsssl/print/docbook.dsl" CDATA DSSSL>
]>

<style-specification id="my-docbook-print" use="docbook">

;; naše úpravy

;; definice velikosti papíru
(define %paper-type% "A4")

;; text dokumentu nebude odsazen
(define %body-start-indent% 0pi)
```

```
;; text výpisů bude používat menší písmo (95 %)
(define %verbatim-size-factor% 0.95)

;; definice vlastní velikosti písma v dokumentu
(define %visual-acuity% "eleven")
(define %bf-size%
  ;; Defines the body font size
  (case %visual-acuity%
    (("tiny") 8pt)
    (("normal") 10pt)
    (("eleven") 11pt)
    (("presbyopic") 12pt)
    (("large-type") 24pt)))

;; zarovnání do bloku a dělení slov
(define %default-quadding%  'justify)
(define %hyphenation% #t)

;; definice písma (funguje pouze pro RTF backend)
;;(define %body-font-family% "Palatino Linotype")
;;(define %title-font-family% "Arial Black")

;; definice vlastní velikosti okrajů písma
(define %left-margin%  2in)
(define %right-margin%  2in)
(define %top-margin%  2.5in)
(define %bottom-margin%  2.5in)
(define %header-margin%  1in)
(define %footer-margin%  1in)

;; oboustranný tisk
(define %two-side% #t)

;; počet sloupců v dokumentu
(define %page-n-columns% 1)

;; má se generovat obsah knihy
(define %generate-book-toc% #t)

;; seznamy čeho se mají automaticky generovat
(define ($generate-book-lot-list$)
  (list (normalize "table")
        (normalize "equation")))

;; mají se číslovat sekce (normálně se číslují jen kapitoly)
(define %section-autolabel% #t)

;; mají se kreslit horizontální čáry před a za obrázkem
(define %figure-rules% #t)

;; seznam elementů, jejichž obsah se objeví na titulní straně
(define (book-titlepage-recto-elements)
  (list (normalize "title")
 (normalize "subtitle")
 (normalize "graphic")
 (normalize "corpauthor")
 (normalize "authorgroup")
 (normalize "author")
```

```
 (normalize "editor")
 (normalize "corpname")
        (normalize "date")))

</style-specification>

<external-specification id="docbook" document="docbook.dsl">
```

5.1.2 Běžné úpravy pro generování HTML

Princip úprav je stejný jako u stylu pro tisk. Stačí změnit načítaný styl a podívat se do dokumentace, které parametry máme k dispozici.

Příklad 5.2. Úprava DSSSL stylu pro generování HTML – `html.dsl`

```
<!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN" [
<!ENTITY docbook.dsl SYSTEM "c:/docbook/dsssl/html/docbook.dsl" CDATA DSSSL>
]>

<style-specification id="my-docbook-html" use="docbook">

;; naše úpravy

;; přípona pro HTML soubory
(define %html-ext% ".html")

;; adresář, kam se mají ukládat vygenerované HTML soubory
;; (define %output-dir% "html")

;; má se používat výše uvedený adresář
;; (define use-output-dir #t)

;; jméno hlavní stránky
(define %root-filename% "docbook")

;; mají se jména odvozovat z hodnoty atributu ID
(define %use-id-as-filename% #t)

;; má se generovat obsah knihy
(define %generate-book-toc% #t)

;; seznamy čeho se mají automaticky generovat
(define ($generate-book-lot-list$)
  (list (normalize "table")
        (normalize "equation")))

;; mají se číslovat sekce (normálně se číslují jen kapitoly)
(define %section-autolabel% #t)

;; mají se kreslit horizontální čáry před a za obrázkem
(define %figure-rules% #t)

;; seznam elementů, jejichž obsah se objeví na titulní straně
(define (book-titlepage-recto-elements)
  (list (normalize "title")
 (normalize "subtitle")
 (normalize "graphic")
```

```
 (normalize "corpauthor")
 (normalize "authorgroup")
 (normalize "author")
 (normalize "editor")
 (normalize "corpname")))

;; do hlavičky stránky přidáme informaci o použitém kódování
(define %html-header-tags%
  '(("META" ("HTTP-EQUIV" "Content-Type") ("CONTENT" "text/html; charset=utf-8"))))

;; kde se mají geenrovat navigační odkazy
(define %header-navigation% #t)
(define %footer-navigation% #t)

;; má se zobrazovat obsah elementu comment/remark
(define %show-comments% #t)

</style-specification>

<external-specification id="docbook" document="docbook.dsl">
```

5.2 Předefinování pravidel pro zpracování elementů

Pří úpravách stylů se využívá toho, že naše deklarace mají přednost před těmi naimportovanými. Pokud se nám nelíbí, jak se zpracovává nějaký element, stačí si zkopírovat odpovídající kód z originálních DSSSL stylů a upravit ho. To používáme v následující úpravě, která je schopná text uzavřený mezi `<phrase role="important">` a `</phrase>` zobrazit tučně.

Příklad 5.3. Úprava pravidla v DSSSL stylu – `tiskml.dsl`

```
<!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN" [
<!ENTITY docbook.dsl SYSTEM "c:/docbook/dsssl/print/docbook.dsl" CDATA DSSSL>
]>

<style-specification id="my-docbook-print" use="docbook">

;; naše úpravy

;; pokud použijeme <![CDATA[<phrase role="important">...</phrase>]]>,
;; zobrazí se text tučně
(element phrase
  (if (equal? (normalize "important") (attribute-string (normalize "role")))
      ($bold-seq$)
      ($charseq$)))

</style-specification>

<external-specification id="docbook" document="docbook.dsl">
```

5.3 Úpravy automaticky generovaných textů

Při zařazování odkazů na různé objekty se text odkazu automaticky generuje. Pokud nám tvar automaticky generovaných odkazů nevyhovuje, můžeme jej upravit. Pro češtinu ovlivňují generování textů dva soubory `dbl1cs.dsl` a `dbl1cs.ent` uložené v adresáři `common`. Druhý soubor obsahuje definice českých překladů jednotlivých textů, v prvním je pak definován seznam `cs-xref-strings`, který definuje, z čeho se skládají automaticky generované texty odkazů.

Například prvek seznamu

```
(list (normalize "figure")      "&Figure; %n")
```

říká, že při generování odkazu na obrázek (element `figure`) se doplní obsah entity `&Figure;` (ta obsahuje text Obrázek) následovaný jeho číslem (`%n`). Pokud nechceme, aby se před číslo obrázku vkládal text, stačí tuto řádku nahradit za

```
(list (normalize "figure")      "%n")
```

Podobně můžeme postupovat pro další elementy. Kromě `%n` ještě můžeme použít další zástupné znaky uvedené v tabulce 5.1 – „Zástupné znaky pro definici vlastního textu křížových odkazů“.

Tabulka 5.1. Zástupné znaky pro definici vlastního textu křížových odkazů

Znak	Popis
`%p`	Číslo stránky, na které se vyskytuje cíl odkazu.
`%g`	Lidsky srozumitelný název elementu, na který odkaz ukazuje (např. „Kapitola“ pro `chapter`).
`%n`	Číslo objektu (obrázku, kapitoly apod.).
`%t`	Název elementu, na který odkazujeme (např. název kapitoly, obrázku apod.).

Poznámka

Současná verze stylů již pro češtinu nezahrnuje do automaticky generovaného odkazu názvy v prvním pádu jako *Obrázek* nebo *Tabulka*. I přesto se občas hodí upravit tvar generovaného odkazu, například do něj zahrnout číslo strany.

Jednou z možností je naše úpravy provést přímo v DSSSL stylech. Pak se nám ale snadno stane, že si stáhneme novější verzi a jednou pracně provedené úpravy si přemažeme. Lepší je proto provést změny podobně jako v předchozích případech. Nakopírujeme si do našeho adresáře soubor `dbl1cs.ent` a v našem stylu s úpravami předefinujeme prvky seznamu `cs-xref-strings`.

Příklad 5.4. Úprava automaticky generovaných textů – `tiskcs.dsl`

```
<!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN" [
<!ENTITY docbook.dsl SYSTEM "c:/docbook/dsssl/print/docbook.dsl" CDATA DSSSL>
<!ENTITY % cs.words
  PUBLIC "-//Norman Walsh//ENTITIES DocBook Stylesheet Localization//CS"
         "dbl1cs.ent">
%cs.words;
]>

<style-specification id="my-docbook-print" use="docbook">

;; upravený seznam se šablonami generovaných textů
(define (cs-xref-strings)
  (list (list (normalize "appendix")    (if %chapter-autolabel%
         "%n"
         "%n \U-2013; \U-201E;%t\U-201C;"))
 (list (normalize "article")     (string-append %gentext-cs-start-quote%
             "%t"
             %gentext-cs-end-quote%))
 (list (normalize "bibliography") "%t")
 (list (normalize "book")        "%t")
 (list (normalize "chapter")     (if %chapter-autolabel%
```

```
        "%n"
        "%n \U-2013; \U-201E;%t\U-201C;"))
 (list (normalize "equation")    "%n")
 (list (normalize "example")     "%n")
 (list (normalize "figure")      "%n na stran\U-011B; %p")
 (list (normalize "glossary")    "%t")
 (list (normalize "index")       "%t")
 (list (normalize "listitem")    "%n")
 (list (normalize "part")        "%n \U-2013; \U-201E;%t\U-201C;")
 (list (normalize "preface")     "%t")
 (list (normalize "procedure")   "%n \U-2013; \U-201E;%t\U-201C;")
 (list (normalize "reference")   "%t")
 (list (normalize "section")     (if %section-autolabel%
        "%n"
        "\U-201E;%t\U-201C;"))
 (list (normalize "sect1")       (if %section-autolabel%
        "%n"
        "\U-201E;%t\U-201C;"))
 (list (normalize "sect2")       (if %section-autolabel%
        "%n"
        "\U-201E;%t\U-201C;"))
 (list (normalize "sect3")       (if %section-autolabel%
        "%n"
        "\U-201E;%t\U-201C;"))
 (list (normalize "sect4")       (if %section-autolabel%
        "%n"
        "\U-201E;%t\U-201C;"))
 (list (normalize "sect5")       (if %section-autolabel%
        "%n"
        "\U-201E;%t\U-201C;"))
 (list (normalize "simplesect")  (if %section-autolabel%
        "%n"
        "\U-201E;%t\U-201C;"))
 (list (normalize "sidebar")     "%t")
 (list (normalize "step")        "%n")
 (list (normalize "table")       "%n")))

</style-specification>

<external-specification id="docbook" document="docbook.dsl">
```

Kapitola 6. Úprava XSL stylů

Průběh formátování pomocí XSL stylů lze řídit mnoha parametry. Kompletní přehled parametrů je v dokumentaci ke stylům [8]. Složitější úpravy je nutné naprogramovat přímo v XSL [17], [16].

6.1 Úprava chování stylu pomocí parametrů

Pokud chceme upravit chování stylu, musíme si vytvořit nový styl. Ten přitom naimportuje původní styl a předefinuje potřebné parametry. XSLT styly jsou XML dokumenty, které obsahují instrukce pro XSLT procesor a tagy, které se mají generovat (HTML, FO apod.).

```
<?xml version="1.0" encoding="windows-1250"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">

<xsl:import href="file:///c:/docbook/xsl/html/docbook.xsl"/>

<!-- Úpravy parametrů -->

</xsl:stylesheet>
```

Musíme samozřejmě u `xsl:import` upravit cestu tak, aby ukazovala na místo, kde se na našem systému nachází styl, ze kterého vycházíme. Cestu ke stylu musíme zapisovat jako URL adresu, proto je na začátku poněkud nezvyklé `file:///`. Pokud máme správně nastavené katalogové soubory, je vhodné styly identifikovat URL adresou, na které jsou dostupné odkudkoliv z Internetu. Naše úpravy stylů pak budou snáze přenositelné mezi systémy.

```
<?xml version="1.0" encoding="windows-1250"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">

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

<!-- Úpravy parametrů -->

</xsl:stylesheet>
```

I v XSL máme dva druhy stylů – pro tisk (formátovací objekty – adresář `fo`) a pro HTML (adresář `html`) – které jsou zcela samostatné.

Když chceme pomocí upraveného stylu zformátovat nějaký dokument, předáme XSLT procesoru jako parametr náš styl:

```
saxon -o prvni.html prvni.xml html.xsl
```

6.1.1 Běžné úpravy pro tisk

Následující příklad ukazuje, jak se dají změnit nejběžnější parametry. Výběr parametrů zpočátku nebyl tak bohatý jako u DSSSL, ale vývoj XSL FO stylů pokračuje velice rychle a již dnes jsou mnohem lépe parametrizovatelné než DSSSL styly.

Příklad 6.1. Úprava XSL stylu pro tisk – `tisk.xsl`

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

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

<!-- Úpravy parametrů -->

<!-- Velikost papíru -->
<xsl:param name="paper.type" select="'A4'"/>

<!-- XSLT procesor může používat rozšíření pro callouts apod. -->
<xsl:param name="use.extensions" select="1"/>

<!-- Rozšíření specifická pro daný FO procesor -->
<!-- <xsl:param name="passivetex.extensions" select="1"/> -->

<xsl:param name="xep.extensions" select="1"/>

<!-- Velikost písma textu -->
<xsl:param name="body.font.master">11</xsl:param>

<!-- Velikost okrajů -->
<xsl:param name="page.margin.inner" select="'1in'"/>
<xsl:param name="page.margin.outer" select="'1in'"/>

<!-- Číslování sekcí a kapitol -->
<xsl:param name="section.autolabel" select="1"/>
<xsl:param name="section.label.includes.component.label" select="1"/>
<xsl:param name="chapter.autolabel" select="1"/>
<xsl:param name="appendix.autolabel" select="1"/>
<xsl:param name="part.autolabel" select="1"/>
<xsl:param name="preface.autolabel" select="0"/>

<!-- Nechceme obrázek -->
<xsl:param name="draft.watermark.image" select="''"/>

<!-- Nadpisy jsou zarovnány s textem, jak je zvykem v evropské typografii -->
<xsl:param name="body.start.indent" select="'0pt'"/>

</xsl:stylesheet>
```

6.1.2 Běžné úpravy pro generování HTML

Princip úprav je stejný jako u stylu pro tisk. Stačí změnit načítaný styl a podívat se do dokumentace, které parametry máme k dispozici.

Příklad 6.2. Úprava XSLT stylu pro generování HTML – `html.xsl`

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

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

<!-- Úpravy parametrů -->

<!-- Změna výstupního kódování -->
<xsl:output method="html" encoding="iso-8859-2"
                          saxon:character-representation="native"/>

<!-- Mají se používat rozšíření -->
<xsl:param name="use.extensions" select="1"/>

<!-- Mají se sekce automaticky číslovat -->
<xsl:param name="section.autolabel" select="1"/>

<!-- Mají čísla sekcí obsahovat i čísla kapitol -->
<xsl:param name="section.label.includes.component.label" select="1"/>

<!-- Mají se číslovat kapitoly -->
<xsl:param name="chapter.autolabel" select="1"/>

</xsl:stylesheet>
```

XSLT styly používají různé styly pro generování jedné HTML stránky a celé sady HTML stránek. Pokud chceme sdílet nastavení parametrů pro oba dva režimy, můžeme nastavení uložit do samostatného souboru.

Příklad 6.3. Styl se sdílenými úpravami parametrů – `html-common.xsl`

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

<!-- Úpravy parametrů -->

<!-- Změna výstupního kódování -->
<xsl:output method="html" encoding="iso-8859-2"
                          saxon:character-representation="native"/>

<!-- Mají se používat rozšíření -->
<xsl:param name="use.extensions" select="1"/>

<!-- Mají se sekce automaticky číslovat -->
<xsl:param name="section.autolabel" select="1"/>

<!-- Mají čísla sekcí obsahovat i čísla kapitol -->
<xsl:param name="section.label.includes.component.label" select="1"/>

<!-- Mají se číslovat kapitoly -->
<xsl:param name="chapter.autolabel" select="1"/>

<!-- Mají se vypnout navigační lišty -->
<xsl:param name="suppress.navigation">0</xsl:param>

</xsl:stylesheet>
```

Tento styl pak můžeme načítat pomocí `xsl:include`.

Příklad 6.4. Styl pro HTML využívající společná nastavení – `html-normal.xsl`

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

<xsl:import href="http://docbook.sourceforge.net/release/xsl/current/html/docbook.xsl"/>
<xsl:include href="html-common.xsl"/>

</xsl:stylesheet>
```

Příklad 6.5. Styl pro sadu HTML stránek využívající společná nastavení a přidávající pár dalších nastavení – `html-chunk.xsl`

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

<xsl:import href="http://docbook.sourceforge.net/release/xsl/current/html/chunk.xsl"/>
<xsl:include href="html-common.xsl"/>

<!-- Kódování výstupních HTML stránek -->
<xsl:param name="chunker.output.encoding" select="'windows-1250'"/>
<xsl:param name="saxon.character.representation" select="'native'"/>

<!-- Přípona pro generované soubory -->
<xsl:param name="html.ext" select="'.html'"/>

<!-- Název hlavního souboru -->
<xsl:param name="root.filename" select="'index'"/>

<!-- Adresář, kam se mají ukládat vygenerované stránky -->
<xsl:param name="base.dir" select="''"/>

<!-- Do jaké hloubky se mají vytvářet chunky pro sekce
     (0 = chunky se vytvářejí pouze na úrovni kapitol) -->
<xsl:param name="chunk.section.depth" select="1"/>

<!-- Má první sekce kapitoly tvořit samostatný chunk -->
<xsl:param name="chunk.first.sections" select="'0'"/>

</xsl:stylesheet>
```

6.1.3 Úpravy pro generování HTML Helpu

HTML Help je založen na výstupu do HTML a můžeme proto použít všechny parametry, které známe ze stylů pro výstup do HTML a pro chunkované HTML. Navíc lze použít několik dalších parametrů, které ilustruje následující příklad. Pro české texty je důležité zejména správné nastavení parametrů *`chunker.output.encoding`*, *`htmlhelp.encoding`* a *`saxon.character.representation`*.

Příklad 6.6. Parametry pro výstup do HTML Helpu – `htmlhelp.xsl`

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

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

```
<!-- Kódování řídících souborů -->
<xsl:param name="htmlhelp.encoding" select="'windows-1250'"/>

<!-- Kódování výstupních HTML stránek -->
<xsl:param name="chunker.output.encoding" select="'windows-1250'"/>
<xsl:param name="saxon.character.representation" select="'native'"/>

<!-- Mají se číslovat položky v obsahu -->
<xsl:param name="htmlhelp.autolabel" select="0"/>

<!-- Jméno výsledného CHM souboru -->
<xsl:param name="htmlhelp.chm" select="'napoveda.chm'"/>

<!-- Jméno projektového souboru -->
<xsl:param name="htmlhelp.hhp" select="'htmlhelp.hhp'"/>

<!-- Jméno souboru s obsahem -->
<xsl:param name="htmlhelp.hhc" select="'toc.hhc'"/>

<!-- Tlačítko pro skok na domovskou stránku -->
<xsl:param name="htmlhelp.button.home">1</xsl:param>
<xsl:param name="htmlhelp.button.home.url">http://www.kosek.cz</xsl:param>

</xsl:stylesheet>
```

Chceme-li sdílet nastavení běžných parametrů pro výstup do HTML i s výstupem do HTML Helpu, můžeme použít stejný mechanismus jako dříve (využití instrukce `xsl:include`).

6.2 Předefinování pravidel pro zpracování elementů

Pří úpravách stylů se využívá toho, že naše šablony mají přednost před těmi naimportovanými. Pokud se nám nelíbí, jak se zpracovává nějaký element, stačí si zkopírovat odpovídající šablonu z originálních XSL stylů a upravit ji. To používáme v následující úpravě, která je schopná text uzavřený mezi `<phrase role="important">` a `</phrase>` zobrazit tučně.

Příklad 6.7. Úprava složitějšího pravidla z XSL stylu – `tiskml.xsl`

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

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

<!-- Úpravy parametrů -->

<!-- Velikost papíru -->
<xsl:param name="paper.type" select="'A4'"/>

<!-- XSLT procesor může používat rozšíření pro callouts apod. -->
<xsl:param name="use.extensions" select="1"/>

<!-- Nechceme obrázek -->
<xsl:param name="draft.watermark.image" select="''"/>

<!-- Rozšíření specifická pro daný FO procesor -->
<xsl:param name="xep.extensions" select="1"/>
```

```
<!-- Upravená šablona pro <phrase role="important"> -->
<xsl:template match="phrase">
  <xsl:choose>
    <xsl:when test="@role='important'">
      <xsl:call-template name="inline.boldseq"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:call-template name="inline.charseq"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

</xsl:stylesheet>
```

6.3 Úpravy automaticky generovaných textů

Automaticky generované texty pro češtinu jsou uloženy v souboru `common/cs.xml`. Upravovat přímo lokalizační soubor není dobrý nápad, protože při aktualizaci stylů o naše úpravy přijdeme. XSL styly proto nabízejí mnohem elegantnější způsob předefinování některých textů. Ve stylu stačí nastavit parametr *`local.l10n.xml`* tak, aby ukazoval na XML dokument s lokalizačními úpravami. Soubor s úpravami pak má stejnou strukturu jako originální lokalizace. Díky mechanismu jmenných prostorů můžeme tento soubor zařadit přímo do stylu nastavujícího ostatní parametry.

Příklad 6.8. Doplnění čísla obrázku v odkazu o název obrázku – `tiskcs.xsl`

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

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

<!-- Úpravy parametrů -->

<!-- Velikost papíru -->
<xsl:param name="paper.type" select="'A4'"/>

<!-- XSLT procesor může používat rozšíření pro callouts apod. -->
<xsl:param name="use.extensions" select="1"/>

<!-- Z jakého dokumentu se bude číst uživatelská změna lokalizace -->
<xsl:param name="local.l10n.xml" select="document('')"/>

<!-- Lokalizační dokument může být přímo součástí stylu -->
<i18n xmlns="http://docbook.sourceforge.net/xmlns/l10n/1.0">

  <!-- Jazyk lokalizace -->
  <l10n language="cs">

    <!-- Úprava šablon pro křížové odkazy -->
    <context name="xref-number-and-title">
      <template name="figure" text="%n (%t)"/>
    </context>

  </l10n>
</i18n>
```

```
<!-- Nechceme obrázek -->
<xsl:param name="draft.watermark.image" select="''"/>

</xsl:stylesheet>
```

Lokalizační soubory obsahují několik dalších kontextů. Mezi nejzajímavější patří asi `title-numbered` a `title-unnumbered`, které se používají pro nadpisy s číslem, resp. bez čísla.

6.4 Úprava vzhledu generovaných HTML stránek pomocí CSS

HTML stránky generované pomocí stylů používají pouze základní HTML značkování, nijak nemění barvy, použitá písma apod. Implicitní vzhled HTML stránek v prohlížečích není zrovna dvakrát oslňující, navíc pro čtení na obrazovce jsou vhodnější bezpatková písma. Styly umožňují pomocí parametru zadat jméno souboru s kaskádovým stylem (CSS). Odkaz na něj se pak automaticky vloží do každé generované stránky. Navíc styly většinu kontejnerových elementů uzavírají do elementu `div` a nastavují u něj třídu na jméno shodné s názvem původního docbookového elementu. Například každá kapitola je uzavřena v následující obálce:

```
<div class="chapter">
  ...
</div>
```

Toho pak můžeme využít v kaskádovém stylu. Na příkladě 6.9 – „Ukázkový kaskádový styl – `docbook.css`“ je ukázáno použití této vlastnosti.

Příklad 6.9. Ukázkový kaskádový styl – `docbook.css`

```
body { font-family: Verdana, Helvetica CE, Arial CE, Arial, Helvetica, sans-serif;
       font-size: 9pt;
       background-color: white;
       color: black;
       margin: 0px; }

table { font-size: 9pt }

.title, .subtitle { color: navy }

.programlisting,
.programlistingco,
.screen { background-color: #d0d0d0;
          padding: 5pt;
          margin: 5pt}

:hover { color: red;
         text-decoration: underline; }

a { text-decoration: none;
    color: blue; }

.navfooter, .navheader { background-color: #EEDDFF; }

.chapter, .refentry,
.book, .reference,
.preface, .colophon { margin-left: 10pt;
```

```
                    margin-right: 10pt; }
```

Pro automatické zařazení odkazu na styl do všech stránek stačí do parametru *html.stylesheet* uložit cestu k CSS souboru. Např.:

```
<xsl:param name="html.stylesheet">docbook.css</xsl:param>
```

6.5 Úprava vzhledu tištěného výstupu pomocí vlastností FO

U nejpoužívanějších elementů lze jejich vzhled měnit i jednodušeji než změnou celé šablony. V našem stylu s úpravami můžeme předefinovat množinu atributů, která reprezentuje vlastnosti FO aplikované na daný element. Kdybychom například chtěli mít všechny nadpisy sekcí první úrovně vycentrované a navíc červené, použijeme:

```
<xsl:attribute-set name="section.title.level1.properties">
  <xsl:attribute name="color">red</xsl:attribute>
  <xsl:attribute name="text-align">center</xsl:attribute>
</xsl:attribute-set>
```

Příklad 6.10. Změna tištěného výstupu pomocí vlastností FO – `tisk-vlastnosti.xsl`

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

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

<!-- Úpravy parametrů -->

<!-- Velikost papíru -->
<xsl:param name="paper.type" select="'A4'"/>

<!-- XSLT procesor může používat rozšíření pro callouts apod. -->
<xsl:param name="use.extensions" select="1"/>

<!-- Nechceme obrázek -->
<xsl:param name="draft.watermark.image" select="''"/>

<!-- Rozšíření specifická pro daný FO procesor -->
<xsl:param name="xep.extensions" select="1"/>

<!-- Snadná změna vzhledu vybraných elementů pomocí množin atributů -->
<xsl:attribute-set name="section.title.level1.properties">
  <xsl:attribute name="color">red</xsl:attribute>
  <xsl:attribute name="text-align">center</xsl:attribute>
</xsl:attribute-set>

</xsl:stylesheet>
```

6.6 Změna vzhledu titulní strany

DocBook umožňuje pomocí elementů pro metainformace jako `bookinfo` a dalších vložit do dokumentu bohaté informace jako název, autor, vydavatel, datum vydání apod. Ne všechny tyto informace se impli-

citně zobrazují na začátku generovaných dokumentů. V XSL stylech je proto vzhled titulní stránky řízen speciální šablonou, kterou lze samozřejmě předefinovat.

Výchozí vzhled titulních stran je definován v souboru `fo/titlepage.templates.xml` resp. `html/titlepage.templates.xml`. Pokud chceme změnit vzhled formátování titulní strany pro určitý element *x* musíme změnit šablonu `<t:titlepage t:element="`*x*`"...>`. Vytvoříme si nový soubor a šablonu v něm změníme. Na ukázce 6.11 – „Šablona pro změnu vzhledu titulní strany knihy – `ts-sablona.xml`“ jsme na titulní stranu přidali informaci o vydavateli a držiteli copyrightu a změnili barvu titulku na modrou.

Příklad 6.11. Šablona pro změnu vzhledu titulní strany knihy – `ts-sablona.xml`

```
<t:templates xmlns:t="http://nwalsh.com/docbook/xsl/template/1.0"
             xmlns:param="http://nwalsh.com/docbook/xsl/template/1.0/param"
             xmlns:fo="http://www.w3.org/1999/XSL/Format"
             xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<t:titlepage t:element="book" t:wrapper="fo:block">
    <t:titlepage-content t:side="recto">
      <title
             t:named-template="division.title"
             param:node="ancestor-or-self::book[1]"
             text-align="center"
             color="blue"
             font-size="24pt"
             space-before="18pt"
             font-weight="bold"
             font-family="{$title.font.family}"/>
      <subtitle
                text-align="center"
                color="blue"
                font-size="20pt"
                space-before="16pt"
                font-family="{$title.font.family}"/>
      <corpauthor font-size="17pt"
                  keep-with-next="always"
                  space-before="2in"/>
      <authorgroup space-before="2in"/>
      <author font-size="17pt"
              space-before="10pt"
              keep-with-next="always"/>
      <publisher space-before="12cm"
                 text-align="start"
                 font-size="10pt"
                 font-family="{$body.font.family}"/>
      <copyright text-align="start"
                 color="red"
                 font-size="8pt"
                 font-family="{$body.font.family}"/>
    </t:titlepage-content>

  <t:titlepage-content t:side="verso">
  </t:titlepage-content>

  <t:titlepage-separator>
      <fo:block break-after="page"/>
  </t:titlepage-separator>
```

```
  <t:titlepage-before t:side="recto">
  </t:titlepage-before>

  <t:titlepage-before t:side="verso">
  </t:titlepage-before>
</t:titlepage>

</t:templates>
```

Šablona je jen jakýsi prefabrikát, který musíme upravit pro použití jako součást běžných XSL stylů. K tomu slouží styl `template/titlepage.xsl`, který z naší šablony vygeneruje normální XSL styl. Vygenerování stylu provedeme příkazem:

```
saxon -o ts-sablona.xsl ts-sablona.xml c:\docbook\xsl\template\titlepage.xsl
```

Vygenerovaný styl `ts-sablona.xsl` pak sloučíme se standardním stylem (viz ukázka 6.12 – „Sloučení nové titulní strany se standardním stylem – `jinats.xsl`“), který použijeme pro generování výstupu.

Příklad 6.12. Sloučení nové titulní strany se standardním stylem – `jinats.xsl`

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

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

<xsl:param name="paper.type" select="'A4'"/>
<xsl:param name="draft.watermark.image" select="''"/>

</xsl:stylesheet>
```

Zcela obdobný mechanismus funguje i pro šablony titulních stran při výstupu do HTML. Tyto šablony však slouží pouze k určení elementů, které se mají dostat do výsledného HTML. Obvykle se u nich nastavuje pouze atribut `class` a pro definici vzhledu se pak používají kaskádové styly.

Kapitola 7. Pokročilé techniky

V této kapitole se podíváme, jak řešit některé problémy, se kterými se setkáme při tvorbě dokumentace. Půjde jak o jednoduché tipy tak i o postupy vedoucí k výsledkům, které jsou mnohdy dostupné pouze v poměrně drahých komerčních nástrojích.

7.1 Rozdělení jednoho dokumentu do více souborů

U větších dokumentů je celkem logické jejich rozdělení do několika souborů. Jednak se s menšími soubory lépe pracuje, a druhak může na jednom dokumentu pracovat více lidí najednou – každý edituje jen jednu jeho část. V DocBooku lze jeden dokument rozdělit na více části velice snadno pomocí mechanismu externích textových entit.

Příklad 7.1. Rozložení dokumentu do několika souborů – `velkakniha.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE book PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
               'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd' [
<!ENTITY kap1 SYSTEM "xinline.xml">
<!ENTITY kap2 SYSTEM "xcallouts.xml">
<!ENTITY kap3 SYSTEM "xobrazky.xml">
<!ENTITY kap4 SYSTEM "xseznamy.xml">
<!ENTITY kap5 SYSTEM "xtabulky.xml">
<!ENTITY kap6 SYSTEM "xtrida.xml">
<!ENTITY ref  SYSTEM "xreference.xml">
]>
<book lang="cs">
  <bookinfo>
    <title>Velká kniha</title>
    <subtitle>Složená z několika entit</subtitle>
  </bookinfo>
  <preface>
    <title>Úvod</title>
    <para>Následuje několik sice nesouvisejících kapitol, ale na
ukázku to stačí ne?</para>
  </preface>
  &kap1;
  &kap2;
  &kap3;
  &kap4;
  &kap5;
  &kap6;
  &ref;
</book>
```

Jediné, na co si musíme dát pozor, je to, že v XML musí jednotlivé entity začínat jen deklarací kódování a nesmějí obsahovat deklaraci typu dokumentu (`<!DOCTYPE...>`).

Příklad 7.2. Ukázka načítané entity

```
<?xml version="1.0" encoding='utf-8'?>
<chapter lang="cs">
  <title>Kapitola se seznamy</title>
    ...
</chapter>
```

```
<!-- Keep this comment at the end of the file
Local variables:
sgml-doctype: velkakniha.xml
sgml-parent-document: ("velkakniha.xml" "book" "chapter")
End:
-->
```

Komentář na konci je používán PSGML módem v Emacsu. Pokud tyto parametry nastavíme, dokáže Emacs bez problémů pracovat s dokumenty rozdělenými do souborů. V parametrech se udává jméno souboru, do kterého je entita vložená, element, do kterého je entita vnořena, a nakonec element, který je kořenovým elementem entity.

Podobně lze nastavit parametry pro editor jEdit:

Příklad 7.3. Ukázka načítané entity v jEditu

```
<?xml version="1.0" encoding="utf-8"?>
<chapter lang="cs">
  <title>Kapitola se seznamy</title>
    ...
</chapter>
<!-- jEdit buffer-local properties: -->
<!-- :xml.root=velkakniha.xml: -->
```

Práce s externími entitami není úplně pohodlná – musíme je předem deklarovat a jednotlivé entity nemohou obsahovat vlastní deklaraci typu dokumentu. Tento problém lze obejít používáním standardu XInclude pro vkládání XML dokumentů. Tento standard zatím není podporován všemi aplikacemi, nicméně jej podporuje například XSLT procesor **xsltproc** nebo parser Xerces, který může být použit společně se Saxonem, takže je možné XInclude použít již dnes.

Jednotlivé části dokumentu uložíme do samostatných souborů, které jsou samostatné XML dokumenty včetně vlastní deklarace typu dokumentu. To nám umožňuje jejich bezproblémové samostatné zpracování pomocí klasických nástrojů, XML editory nevyjímaje. Pokud pak chceme části dokumentu (např. kapitoly) použít v nějakém větším dokumentu, vložíme je pomocí speciálního elementu XInclude:

```
<xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
            href="kapitola.xml"/>
```

Příklad 7.4. Složení dokumentu z několika částí pomocí XInclude – `velkakniha2.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE book PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
               'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<book lang="cs" xmlns:xi="http://www.w3.org/2001/XInclude">
  <bookinfo>
    <title>Velká kniha</title>
    <subtitle>Složená z několika entit</subtitle>
  </bookinfo>
  <preface>
    <title>Úvod</title>
    <para>Následuje několik sice nesouvisejících kapitol, ale na
ukázku to stačí ne?</para>
  </preface>
  <xi:include href="inline.xml"/>
  <xi:include href="callouts.xml"/>
  <xi:include href="obrazky.xml"/>
  <xi:include href="seznamy.xml"/>
```

```
  <xi:include href="tabulky.xml"/>
  <xi:include href="trida.xml"/>
  <xi:include href="reference.xml"/>
</book>
```

Pokud chceme takto složený dokument validovat, musíme parseru říci, aby validaci provedl až po složení celého dokumentu (volbou `--postvalid` místo `--valid`):

```
xmllint --xinclude --postvalid --noout dokument.xml
```

Při zpracování dokumentu XSL styly musíme zapnout podporu XInclude. Procesor **xsltproc** k tomu používá parametr `--xinclude`. Např. pro převod do jedné HTML stránky použijeme:

```
xsltproc --xinclude -o dokument.html c:/docbook/xsl/html/docbook.xsl dokument.xml
```

V Saxonu se podpora XInclude aktivuje rekonfigurací použitého parseru XML (viz „Saxon s podporou XML katalogů a XInclude“).

7.2 Automatický výběr obrázků

Pokud připravujeme dokumentaci pro tištěný i on-line výstup, potřebujeme mít obrázky většinou ve dvou formátech – ve vektorovém a bitmapovém. Styl by pak měl vybrat formát, který je nejvhodnější pro dané médium. Existují v zásadě dva způsoby, jak toho jednoduše dosáhnout.

První metoda využívá toho, že do dokumentu vkládáme obrázky v několika (alespoň ve dvou formátech) do elementu `mediaobject`. Styl pak podle formátu (atribut `format`) nebo přípony souboru vybere nejvhodnějšího kandidáta. Vybere se přitom první obrázek, který je v podporovaném formátu. Obrázky by proto měly být uvedeny v pořadí od nejkvalitnějšího (např. vektorová verze v EPS, PDF nebo WMF) k méně kvalitním (např. bitmapové verze v GIF nebo PNG).

```
<mediaobject>
<imageobject>
<imagedata fileref="pic/uvod-ep.eps" format="EPS"/>
</imageobject>
<imageobject>
<imagedata fileref="pic/uvod-ep.png" format="PNG"/>
</imageobject>
</mediaobject>
</figure>
```

V DSSSL stylech můžeme seznam podporovaných formátů změnit nastavením následující proměnných:

```
(define preferred-mediaobject-extensions
  (list "eps" "ps" "jpg" "jpeg" "png"))

(define preferred-mediaobject-notations
  (list "EPS" "PS" "JPG" "JPEG" "PNG" "linespecific"))
```

V XSL stylech lze podobného efektu dosáhnout změnou následujících šablon:

```
<xsl:template name="is.graphic.format">
  <xsl:param name="format"></xsl:param>
  <xsl:if test="$format = 'PNG'
                or $format = 'JPG'
                or $format = 'JPEG'
                or $format = 'linespecific'
                or $format = 'GIF'
```

```
                or $format = 'GIF87a'
                or $format = 'GIF89a'
                or $format = 'BMP'">1</xsl:if>
</xsl:template>

<xsl:template name="is.graphic.extension">
  <xsl:param name="ext"></xsl:param>
  <xsl:if test="$ext = 'png'
                or $ext = 'jpeg'
                or $ext = 'jpg'
                or $ext = 'avi'
                or $ext = 'mpg'
                or $ext = 'mpeg'
                or $ext = 'qt'
                or $ext = 'gif'
                or $ext = 'bmp'">1</xsl:if>
</xsl:template>
```

Druhý způsob předpokládá, že stejné obrázky v různých formátech máme uložené v souborech se stejným názvem, ale odlišnou příponou. Do dokumentu pak odkaz vkládáme bez přípony a styl automaticky doplní odpovídající příponu. Obrázek tedy do dokumentu vložíme pomocí následujícího kódu.

```
<mediaobject>
  <imageobject>
    <imagedata fileref="schema"/>
  </imageobject>
</mediaobject>
```

Ve skutečnosti přitom máme k dispozici např. obrázky `schema.eps` a `schema.png`. Při generování tištěného výstupu, kdy chceme použít obrázky v EPS, přidáme do stylu následující parametr:

```
(define %graphic-default-extension% "eps")
```

Pro generování HTML podoby budeme mít jiný styl a v něm jako standardní příponu nastavíme PNG.

Obdobný parametr lze nastavit i v XSL stylech:

```
<xsl:param name="graphic.default.extension">png</xsl:param>
```

Novější verze XSL stylů podporují ještě jednu metodu. U elementu `imageobject` můžeme v atributu `role` určit výstupní formát, pro který je obrázek určen. Jako hodnota se typicky uvádí `html` nebo `fo`. Styly pak vyberou odpovídající obrázek, bez ohledu na pořadí elementů `imageobject`. Dokonce můžeme mít i několik různých variant obrázků pro různé procesory FO:

```
<mediaobject>
  <imageobject  role="html">
    <imagedata  format="PNG"  fileref="pic/uvod-ep.png"/>
  </imageobject>
  <imageobject  role="fop">
    <imagedata  format="SVG"  fileref="pic/uvod-ep.svg"/>
  </imageobject>
  <imageobject  role="xep">
    <imagedata  format="PDF"  fileref="pic/uvod-ep.pdf"/>
  </imageobject>
</mediaobject>
```

Stylům pak v parametru *`preffered.mediaobject.role`* předáme údaj o tom (`html`, `fop`, `xep`), jaké obrázky chceme použít.

7.3 Generování rejstříku

Kvalita rejstříku generovaná různými druhy stylů se liší, nicméně poslední verze XSL stylů jsou schopné generovat rejstřík se všemi obvyklými náležitostmi.

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

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

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

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

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

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

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

V rejstříku se pak takto definovaná hesla objeví například jako:

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

Rejstříková hesla v DocBooku mohou mít až čtyři úrovně.

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

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

Ve vygenerovaném rejstříku pak dostaneme interval:

XML
  historie, 27–42

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

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

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

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

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

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

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

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

Ve výsledném rejstříku bychom dostali:

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

- X -
XML schéma, 81, viz též DTD

Tímto jsme se seznámili skoro se všemi možnostmi zápisu rejstříkových hesel v DocBooku. Nezmínili jsme pouze možnost uložit definici rejstříkových hesel zcela mimo místo jejich výskytu s využitím atributu `zone`. Tato metoda není dle mého názoru příliš praktická, více se o ní můžete dočíst v [1].

7.3.1 XSL styly

V XSL stylech je situace velice jednoduchá. Pokud máme v dokumentu označené nějaké termíny jako rejstříková hesla, rejstřík se automaticky vygeneruje. Vyzkoušet si to můžete na souboru `uvod.xml`, který obsahuje několik rejstříkových hesel.

```
saxon -o uvod.html uvod.xml c:\docbook\xsl\html\docbook.xsl
```

Rejstřík by měl v tištěné verzi fungovat úplně stejně, ale prakticky můžete zkusit, že např. v PassiveTeXu se zatím správně nedopočítají čísla stran. Oproti tomu komerční XEP si s čísly stran poradí. Podpora

tvorby rejstříků v XSL stylech a procesorech FO se neustále zlepšuje. Při použití komerčních procesorů FO jako XEP a XSL Formatter fungují i takové věci jako slučování duplicitních čísel stran apod.

Standardní mechanismus pro generování rejstříku bohužel neumí správně obsloužit takové případy jako slovo začínající písmenem „ch“ nebo znakem s diakritikou. V takových případech je nutné použít rejstříkový mechanismus nové generace, který funguje pouze s procesorem Saxon a s některými verzemi xsltproc. Pro jeho potřebí si musíme připravit vlastní styl s úpravami, který načítá soubor `autoidx-ng.xsl` odpovídající použité verzi stylů.

Příklad 7.5. Generování českého rejstříku – `html-rejstrik.xsl`

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

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

<!-- Úpravy parametrů -->

<!-- Změna výstupního kódování -->
<xsl:output method="html" encoding="iso-8859-2"
                          saxon:character-representation="native"/>

<!-- Mají se používat rozšíření -->
<xsl:param name="use.extensions" select="1"/>

<!-- Mají se sekce automaticky číslovat -->
<xsl:param name="section.autolabel" select="1"/>

<!-- Mají čísla sekcí obsahovat i čísla kapitol -->
<xsl:param name="section.label.includes.component.label" select="1"/>

<!-- Mají se číslovat kapitoly -->
<xsl:param name="chapter.autolabel" select="1"/>

</xsl:stylesheet>
```

7.3.2 DSSSL styly

Nejprve si musíme vytvořit nový XML soubor pro rejstřík. Toho dosáhneme spuštěním perlového skriptu, který je součástí distribuce stylů.

```
perl c:\docbook\dsssl\bin\collateindex.pl -N -o index.xml
```

Vznikne nám soubor `index.xml`, do kterého se bude ukládat rejstřík. Tento soubor bychom měli do našeho dokumentu načíst pomocí mechanismu externích entit (viz `uvod2.xml`).

Nyní musíme vygenerovat podklady pro skript `collateindex.pl`. Vždy musíme použít HTML verzi stylu (i tehdy, když chceme rejstřík získat v tištěné verzi dokumentu):

```
jade -d c:\docbook\dsssl\html\docbook.dsl -t sgml -V html-index c:\docbook\jade\xml.dcl ▶
uvod2.xml
```

Jade nám vygeneruje soubor `HTML.index`, ze kterého lze vytvořit rejstřík. Rejstřík v souboru `index.xml` vytvoříme pomocí příkazu.

```
perl c:\docbook\dsssl\bin\collateindex.pl -o index.xml HTML.index
```

Jediný problém je v tom, že vygenerovaný soubor na svém začátku nemá správnou deklaraci kódování. To můžeme změnit ručně, případně opravit skript. Nemusíme také dělat nic, protože tento dokument budeme zpracovávat jen pomocí Jade, který deklaraci kódování nevyžaduje.

Nyní můžeme zcela běžným způsobem zpracovat dokument a budeme v něm mít rejstřík jak pro tištěnou tak pro HTML verzi.

7.4 Vkládání obsahu externích souborů

Pokud potřebujeme do dokumentu vložit obsah externího souboru, můžeme k tomu použít element `inlinegraphic` nebo `inlinemediaobject`. DSSSL styly tuto vlastnost podporují automaticky, v XSL stylech musíme zapnout podporu rozšíření nastavením parametru `use.extensions=1`.

```
<programlisting><inlinegraphic fileref="seznamy.xml" format="linespecific"/></programlisting>
```

S výhledem do budoucna je lepší použít konstrukci využívající `inlinemediaobject`.

```
<programlisting><inlinemediaobject>
<imageobject>
<imagedata fileref="seznamy.xml" format="linespecific"/>
</imageobject>
</inlinemediaobject></programlisting>
```

DocBook 4.2 nabízí poněkud přehlednější možnost pro dosažení stejného efektu.

```
<programlisting><textobject>
<textdata fileref="seznamy.xml"/>
</textobject></programlisting>
```

Kromě atributu `fileref` můžeme použít i atribut `encoding` pro určení kódování načítaného dokumentu. DSSSL styly tuto novou sémantiku zatím nepodporují, XSL už ano, ale informaci o kódování souboru podporují až od verze stylů 1.65.2 (resp. 1.66.0).

7.5 Generování sady HTML stránek

Pokud nám nevyhovují jména generovaných stránek, můžeme pomocí parametru stylu říci, že se jména mají brát z identifikátorů odpovídajících elementů (parametr *`use.id.as.filename`*).

```
<chapter id="jazyk">
<title>Drobná vylepšení jazyka</title>
  ...
```

Další možností je použití speciální instrukce pro zpracování `<?dbhtml filename="`*`jméno souboru`*`"?>`.

```
<chapter>
<title>Drobná vylepšení jazyka</title>
<?dbhtml filename="jazyk.html"?>
  ...
```

7.6 Profilace dokumentů (podmíněné dokumenty)

V mnoha případech potřebujeme z jedné předlohy generovat několik obsahově drobně odlišných verzí dokumentů. Uživatelská příručka k programu, který je k dispozici ve verzích pro Windows a pro Linux, se bude patrně lišit jen v pár bodech týkajících se instalace a konfigurace. Je kvůli tomu zbytečné vytvářet a udržovat sladěné dva dokumenty. Stejně tak můžeme mít kromě standardní dokumentace, také příručku doplňovanou o aktuální postřehy z pracoviště technické podpory. Opět asi vhodnější tyto dokumenty uchovávat v jednom, aby byly naprosto synchronizované.

Mnohem vhodnější je vytvářet dokument jeden a v něm označit části určené pro konkrétní platformu nebo skupinu uživatelů. Při generování výsledné podoby dokumentu pak provedeme profilaci, či chceteli přizpůsobení cílově skupině. Z dokumentu se vyberou pouze jeho odpovídající části. DocBook k těmto účelům nabízí např. atributy `os` a `userlevel`. Do nich se ukládá identifikátor operačního systému, resp. skupiny uživatelů, pro které je daná část dokumentu určena. Část příručky tak může vypadat takto.

Příklad 7.6. Dokument s dvěma verzemi postupu pro různé operační systémy

```
<para>Pro správný chod programu je potřeba nastavit proměnnou
<envar>APPHOME</envar>.</para>

<para os="Unix">Proměnnou nastavíme příkazem <command
moreinfo="none">APPHOME=/usr/local/app; export APPHOME</command>.</para>

<para os="Win">Proměnnou nastavíme v <application
moreinfo="none">Ovládacím Panelu</application> tak, že v položce
<guimenuitem moreinfo="none">Systém</guimenuitem> vybereme volbu
<guibutton moreinfo="none">Prostředí</guibutton>...</para>
```

Když budeme generovat příručku pro Windows, budeme chtít vynechat všechny elementy, které mají atribut `os` a nemají v něm uloženou hodnotu `Win`. Totéž bude analogicky platit pro unixovou verzi dokumentace.

Tento problém můžeme velice jednoduše vyřešit tím, že při formátování pomocí stylu z dokumenty momentálně nepotřebné části vyřadíme. V XSLT toho lze dosáhnout velice jednoduše použitím importu stávajícího stylu a přidáním jednoduché šablony. Tento přístup sice vygeneruje správný výstup, ale některé části dokumentu jako obsah budou obsahovat i položky pro elementy, které jsme nechtěli zpracovat. Úpravy XSL stylů by proto musely být poměrně komplexní.

Schůdnější cestou je vytvoření jednoduchého stylu, který z původního dokumentu pouze vynechá nepotřebné části. Dostaneme tak nový zcela legální docbookový dokument, který můžeme zpracovat libovolným nástrojem včetně XSL a DSSSL stylů. Postup se sice o jeden krok prodlouží, ale v případě opakovaného použití tohoto postupu si můžeme vytvořit dávkový soubor, který vše provede za nás. Styl pro profilaci dokumentů naleznete v souboru `profiling/profile.xsl` ve standardní distribuci XSL stylů.

K vytvoření profilovaného dokumentu stačí na náš dokument aplikovat tento styl a XSLT procesoru v parametrech předat informace o operačním systému a/nebo skupině uživatelů, pro které chceme dokument připravit. Informace se předávají v parametrech *`profile.os`* a *`profile.userlevel`*.

```
saxon -o výstup dokument c:\docbook\xsl\profiling\profile.xsl "profile.os=kód OS" ▶
"profile.userlevel=kód úrovně uživatele"
```

Parametry mohou obsahovat i více hodnot oddělených středníkem. V současné době styl umožňuje profilaci na základě následujících parametrů a jim odpovídajích atributů:

profile.os, *profile.userlevel*, *profile.arch*, *profile.condition*, *profile.conformance*, *profile.revision*, *profile.revisionflag*, *profile.security*, *profile.vendor*, *profile.role*, *profile.lang*	Profilace je provedena na základě obsahu odpovídajícího atributu.
profile.attribute	Jméno atributu, ve kterém jsou informace pro profilaci. Tento parametr použijeme v případě, kdy námi použitý parametr nemá přímo svůj parametr.
profile.value	Hodnota pro uživatelsky definovaný atribut.
profile.separator	Oddělovač identifikátorů profilů. Standardně se používá středník (`;`).

Od verze 1.50 umožňují XSL styly provádět profilaci a transformaci během jednoho běhu stylu. Ke všem stylům jako `docbook.xsl`, `chunk.xsl` a `htmlhelp.xsl`. Např. pro profilaci pro Windows a následný převod do HTML můžeme použít:

```
saxon dokument.xml c:\docbook\xsl\html\profile-chunk.xsl "profile.os=Win"
```

7.7 Odkazy mezi dokumenty

DocBook nabízí bohaté možnosti pro tvorbu odkazů v rámci jednoho dokumentu pomocí elementů `xref` a `link`. Při zpracování rozsíhlejších dokumentačních projektů však potřebujeme vytvářet odkazy i mezi jednotlivými dokumenty. Jedním z řešení je všechny dokumenty spojit do jedné obrovské sady knih s využitím elementu `set`. Má to však své nevýhody – dokument je hodně velký, pro jeho skládání je proto potřeba využít XInclude, chceme-li pracovat s jednotlivými knihami současně. Navíc je mnohem náročnější dodržet jedinečnost identifikátorů v takto rozsáhlém dokumentu.

Přijatelnějším řešením je využití elementu `olink`, který umožňuje vytváření odkazů mezi nezávislými dokumenty. Odkaz je určen dvěma parametry – identifikátorem dokumentu a identifikátorem místa v dokumentu. Identifikátor dokumentu si můžeme volit libovolně, identifikátor místa v dokumentu musí být nějaké `id` existující v dokumentu.

```
<olink targetdoc="InstalačníPříručka" targetptr="uvod"/>

<olink targetdoc="InstalačníPříručka" targetptr="uvod">úvod
instalační příručky</olink>
```

První varianta odkazu se přitom chová podobně jako `xref` a automaticky vygeneruje text odkazu. Druhá varianta je obdobou elementu `link`, kdy text odkazu určujeme ručně.

Práce s takto vytvořenými odkazy samozřejmě vyžaduje, aby existovalo nějaké mapování, které z identifikátoru dokumentu dokáže určit jeho umístění a skutečný název souboru. Pro tyto účely je potřeba vytvořit si databázi cílů v dokumentech. Tuto databázi umí z velké části automaticky generovat XSL styly. Přesný postup si popíšeme dále.

První co musíme vymyslet, jsou identifikátory pro naše dokumenty. Je dobré vymyslet jména, která zůstanou unikátní i po případném rozšiřování kolekce dokumentů a budou dostatečně výstižná. V našem

příkladu takovým identifikátorem byl řetězec `InstalačníPříručka`. Zvolené identifikátory pak používáme pro tvorbu odkazů mezi dokumenty v elementu `olink`.

Další rozhodnutí spočívá ve vymyšlení adresářové struktury, pro dokumentaci v podobě HTML stránek. Tuto hierarchii adresářů musíme znát dopředu, aby se mohly správně dopočítat relativní odkazy, které vzniknou z `olink`u. Výstupní hierarchii musíme uložit v podobě dokumentu XML, jak ukazuje následující příklad.

Příklad 7.7. Databáze cílů v dokumentech – `olinkdb.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE targetset SYSTEM "http://docbook.sourceforge.net/release/xsl/current/common/▶
targetdatabase.dtd" [
<!ENTITY ol1targets SYSTEM "ol1target.db">
<!ENTITY ol2targets SYSTEM "ol2target.db">
]>
<targetset>
  <targetsetinfo>
    Popis databáze cílů odkazů. Tato databáze je pouze jednoduchá
    testovací pro účely školení.
   </targetsetinfo>

  <!-- Site map for generating relative paths between documents -->
  <sitemap>
    <dir>
      <dir name="book1">
        <document targetdoc="OLink1">
          &ol1targets;
        </document>
      </dir>
      <dir name="book2">
        <document targetdoc="OLink2">
          &ol2targets;
        </document>
      </dir>
    </dir>
  </sitemap>
</targetset>
```

Elementy `dir` (kromě toho na nejvyšší úrovni) odpovídají výstupní adresářové struktuře. V našem případě tedy výstupní HTML stránky očekáváme v adresářích `book1` a `book2`. Elementy `dir` do sebe můžeme podle potřeby vnořovat a vytvářet tak libovolně hluboké adresářové struktury.

V místě, kde se pak nachází nějaký dokument převedený do HTML použijeme element `document` a v atributu `targetdoc` určíme identifikátor dokumentu, který se pak používá v elementech `olink`. V našem případě máme dokumenty dva s názvy `OLink1` a `OLink2`. První z nich je přitom v adresáři `book1` a druhý v adresáři `book2`. Nepoužíváme-li výstup do sady HTML stránek, ale jen do jedné velké HTML stránky, musíme u elementu `document` použít ještě atribut `baseuri` a určit jméno HTML stránky, která obsahuje převedený dokument.

Obsah elementu `document` obsahuje odkaz na externí entitu, která zastupuje celý malý dokument XML. V tomto dokumentu jsou pak obsaženy všechny informace potřebné pro vyhodnocení odkazů na daný dokument. Tento dokument je poměrně obsáhlý, ale nemusí nás to trápit, protože jej lze vygenerovat automaticky. Stačí spustit běžnou XSL transformaci a pomocí parametrů říci, že se má soubor vygenerovat. Pro naše dva dokumenty bychom patřičné části databáze vytvořili pomocí příkazů:

```
saxon olink1.xml chunk.xsl "collect.xref.targets=only" "targets.filename=ol1target.db"

saxon olink2.xml chunk.xsl "collect.xref.targets=only" "targets.filename=ol2target.db"
```

Tento příkaz bychom měli spustit vždy, když se daný dokument (v našem případě `olink1.xml` nebo `olink2.xml`) změní, aby se odkazy vyhodnocovaly správně.

Při běžném převodu dokumentů do HTML nebo formátovacích objektů musíme nyní stylům předat další parametry, které nesou informaci o aktuálním dokumentu, databázi cílů odkazů apod. Naše dva ukázkové dokumenty bychom mohli převést do HTML například pomocí následujících příkazů:

```
saxon olink1.xml chunk.xsl "target.database.document=olinkdb.xml" "current.docid=OLink1" ▶
"base.dir=book1/"

saxon olink2.xml chunk.xsl "target.database.document=olinkdb.xml" "current.docid=OLink2" ▶
"base.dir=book2/"
```

Pokud zároveň generujeme výstup do jedné HTML stránky a do sady HTML stránek, musíme si vytvořit dvě samostatné databáze cílů odkazů. Pro odkazy při výstupu do formátovacích objektů lze využívat databáze určené pro HTML, protože se využívají pouze texty odkazů, nevytváří se skutečné hypertextové odkazy. Podpora `olink`u pro HTML Help bude dostupná v blízké budoucnosti.

7.8 Automatické vyznačování změn v XML dokumentech

Při editorských a korektorských úpravách dokumentů je užitečná možnost přehledného zobrazení všech změn mezi dvěma odlišnými verzemi téhož dokumentu. Pro XML dokumenty nelze úspěšně použít běžné nástroje pro porovnávání textů jako **diff**, některé hi-end XML editory proto nabízejí pohodlné nástroje. Další možností je využití utility **diffmk**[1] od Normana Walshe, resp. její novější verze napsané v Javě[2].

Tyto utility umí porovnat obsah dvou souborů a do třetího uložit obě dvě verze s vyznačenými změnami.

```
perl diffmk -doctype docbook stary.xml novy.xml zmeny.xml
```

resp.

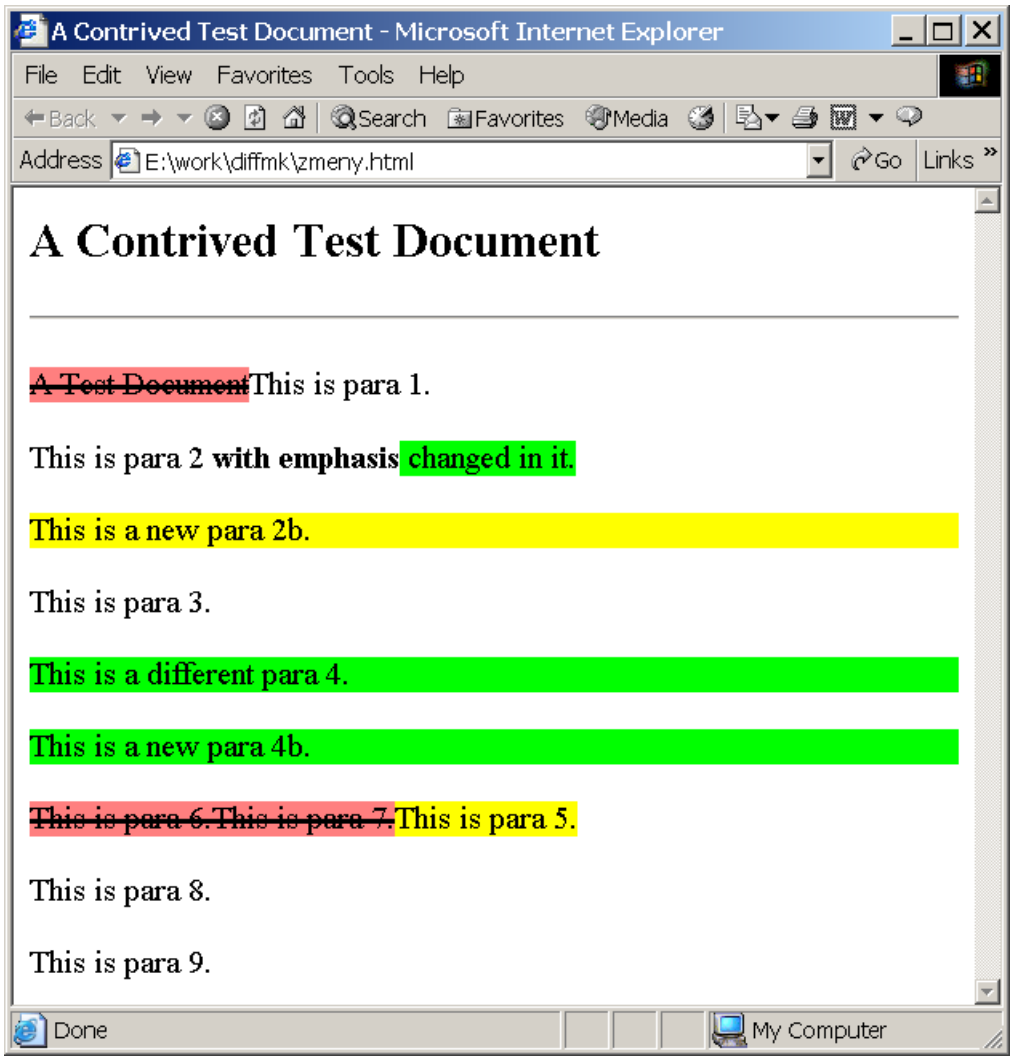
```
java com.sun.xtc.diffmk.DiffMk -doctype docbook stary.xml novy.xml zmeny.xml
```

Nově získaný dokument `zmeny.xml` obsahuje ve speciálních docbookových atributech popis rozdílů mezi verzemi. XSL styly mají speciální styl pro interpretaci těchto informací:

```
saxon -o zmeny.html zmeny.xml c:\docbook\xsl\html\changebars.xsl
```

[1] http://www.sun.com/xml/developers/diffmk/
[2] http://sourceforge.net/projects/diffmk/

Obrázek 7.1. Automatické vyznačení změn v dokumentu



Kapitola 8. Výstupní formáty generovatelné z DocBooku

V úvodní kapitole 2 – „*První kroky*“ jsme se seznámili s tím, jak z DocBooku generovat nejběžnější výstupní formáty pomocí DSSSL a XSL stylů. V této kapitole se podíváme na další metody a výstupní formáty, které lze společně s DocBookem použít. Celý text přitom bude rozčleněn podle výstupního formátu. Snadno tak zjistíte, jaké jsou alternativní možnosti pro získání požadovaného výstupu. Už z principu nemůže být seznam kompletní, a přidání podpory dalšího výstupního formátu je celkem snadné, protože XML dokumenty lze velmi snadno zpracovávat a dále konvertovat.

8.1 HTML

8.1.1 Pomocí DSSSL stylů

```
jade -d c:\docbook\dsssl\html\docbook.dsl -t sgml c:\docbook\jade\xml.dcl dokument.xml
```

```
jade -d c:\docbook\dsssl\html\docbook.dsl -V nochunks -t sgml ▶
c:\docbook\jade\xml.dcl dokument.xml > dokument.html
```

8.1.2 Pomocí XSL stylů

```
saxon dokument.xml c:\docbook\xsl\html\chunk.xsl
```

```
saxon -o dokument.html dokument.xml c:\docbook\xsl\html\docbook.xsl
```

```
xsltproc c:/docbook/xsl/html/chunk.xsl dokument.xml
```

```
xsltproc -o dokument.html c:/docbook/xsl/html/docbook.xsl dokument.xml
```

8.2 Tištěný výstup

8.2.1 Pomocí DSSSL stylů

Výstup do RTF:

```
jade -d c:\docbook\dsssl\print\docbook.dsl -t rtf c:\docbook\jade\xml.dcl dokument.xml
```

RTF lze přímo načíst do většiny textových editorů. Chceme-li získat PDF, můžeme RTF dokument zpracovat pomocí textového editoru (např. MS Word) a Adobe Distilleru (nebo jeho funkční obdoby).

PDF a PS výstup lze snadno získat i pomocí výstupu do TeXu. Musíme mít nainstalovanou poměrně novou verzi TeXu s formátem JadeTeX. Poslední verze JadeTeXu je k dispozici na adrese http://jadetex.sourceforge.net/.

```
jade -d c:\docbook\dsssl\print\docbook.dsl -t tex -V tex-backend ▶
c:\docbook\jade\xml.dcl dokument.xml
pdfjadetex dokument
pdfjadetex dokument
pdfjadetex dokument
```

Pro vygenerování PostScriptové verze můžeme použít příkazy:

```
jade -d c:\docbook\dsssl\print\docbook.dsl -t tex -V tex-backend ▶
c:\docbook\jade\xml.dcl dokument.xml
jadetex dokument
jadetex dokument
jadetex dokument
dvips dokument
```

JadeTeX musíme spustit třikrát, aby se správně spočítala čísla stran v obsahu a v křížových odkazech.

8.2.2 Pomocí XSL stylů (přes formátovací objekty)

V první fázi musíme získat formátovací objekty:

```
saxon -o dokument.fo dokument.xml c:\docbook\xsl\fo\docbook.xsl
```

nebo

```
xsltproc -o dokument.fo c:/docbook/xsl/fo/docbook.xsl dokument.xml
```

Výsledné formátovací objekty pak musíme zpracovat některým z procesorů FO.

XEP

```
xep -fo dokument.fo -pdf dokument.pdf
```

```
xep -fo dokument.fo
```

Nebo jednorázově včetně transformace:

```
xep -xml dokument.xml -xsl c:/docbook/xsl/fo/docbook.xsl
```

PassiveTeX

PassiveTeX je procesor FO postavený nad TeXem. Pro jeho činnost je potřeba moderní instalace TeXu. Aktuální verze PassiveTeXu je dostupná na adrese http://www.tei-c.org/Software/passivetex/.

Pro získání PDF z FO pak stačí použít příkaz:

```
pdfxmltex dokument.fo
```

Pokud chceme, aby nám seděla čísla stran v obsahu, musíme poslední příkaz spustit dvakrát po sobě.

PostScript získáme obdobně. Nejprve si vygenerujeme formátovací objekty a pak spustíme příkazy:

```
xmltex dokument.fo
xmltex dokument.fo
dvips dokument
```

FOP

```
fop -fo dokument.fo -pdf dokument.pdf
```

XFC

Program XFC[1] umožňuje formátovací objekty konvertovat do formátů RTF, WordML a ODF.

[1] http://www.xmlmind.com/foconverter/

```
fo2rtf dokument.fo dokument.rtf

fo2wml dokument.fo dokument.xml

fo2odt dokument.fo dokument.odt
```

Kromě toho obsahuje příjemné grafické prostředí XSL Utility, pro snadné spouštění transformací DocBooku do dalších formátů.

8.2.3 Přes LaTeX

Na adrese http://db2latex.sourceforge.net/ je k dispozici sada XSL stylů, které jsou schopné převést docbookový dokument na LaTeX. Umíte-li LaTeX je to celkem dobrá cesta, jak získat typograficky mnohem dokonalejší dokument než z klasických DSSSL nebo XSL stylů. Styly potřebují většinou drobně upravit – jednak proto, abyste s nimi mohli zpracovat české dokumenty, a také proto, aby výstup odpovídal přesně vašim potřebám.

8.2.4 Přes HTML

V nouzi nejvyšší můžeme z DocBooku vygenerovat HTML a pomocí prohlížeče nebo textového procesoru jej vytisknout. Není to zdaleka nejlepší řešení, ale někdy se může hodit.

8.3 HTML Help

Pro generování HTML Helpu musíme mít HTML Help Workshop – http://msdn.microsoft.com/library/tools/htmlhelp/chm/HH1Start.htm.

```
saxon dokument.xml c:\docbook\xsl\htmlhelp\htmlhelp.xsl "htmlhelp.encoding=windows-1250" ▶
"chunker.output.encoding=windows-1250" "saxon.character.representation=native"
hhc htmlhelp.hhp
```

8.4 JavaHelp

Pro prohlížení JavaHelpu musíte mít k dispozici JavaHelp – http://java.sun.com/products/javahelp/.

```
saxon dokument.xml c:\docbook\xsl\javahelp\javahelp.xsl "javahelp.encoding=windows-1250" ▶
"chunker.output.encoding=windows-1250" "saxon.character.representation=native"
jhindexer *.html
jar -cvf help.jar *
hsviewer jhelpset.hs help.jar
```

8.5 Nápověda pro Eclipse

Eclipse[2] je open-source vývojové prostředí.

```
saxon dokument.xml c:\docbook\xsl\eclipse\eclipse.xsl "eclipse.plugin.id=com.example.help"
```

Vygenerované soubory pak stačí nahrát do adresáře `com.example.help` (nebo jiného v závislosti na nastavení parametru) v adresáři `plugins` a restartovat Eclipse.

[2] http://eclipse.org

8.6 TeXInfo, manuálové stránky

Projekt docbook2X – http://docbook2x.sourceforge.net/.

Manuálové stránky umí generovat i novější verze standardních XSL stylů.

Kapitola 9. Instalace

V následujících odstavcích se podíváme na základní instalaci DocBooku – na instalaci DTD a DSSSL a XSL stylů pro formátování, včetně příslušných DSSSL a XSLT procesorů.

9.1 DocBook DTD

DocBook sám o sobě není skutečně nic jiného než DTD pro XML (případně pro SGML). Aktuální verzi DTD lze stáhnout ze stránek OASIS – http://www.oasis-open.org/docbook/. Pro případ výpadku serveru, je DTD zrcadleno i na stránkách http://www.docbook.org. DTD je rozděleno do několika souborů a proto je nejlepší si stáhnout celý ZIP archiv. Následující postup předpokládá práci s XML verzí DTD 4.5.

Získaný archiv můžeme rozbalit na libovolné místo na disku. V dalším budeme předpokládat, že jsme DTD rozbalili do adresáře `c:\docbook\dtd`.

Pokud chcete pro zpracování DocBooku používat i nástroje určené pro SGML – např. Jade nebo PSGML mód v Emacsu, je ještě potřeba nastavit katalogové soubory.

Součástí DTD je i katalogový soubor `docbook.cat`, který zařadíme mezi načítané katalogy uložené v proměnné `SGML_CATALOG_FILES`.

```
SGML_CATALOG_FILES=c:\docbook\dtd\docbook.cat
```

Konkrétní provedení nastavení proměnné závisí na použité platformě. Ve Windows 95/98 musíme upravit soubor `autoexec.bat` a restartovat počítač, ve Windows NT/2000 se nastavení provádí pomocí Ovládací panel▻System▻Pokročilé▻Proměnné prostředí. V Unixu nejspíše nastavení proměnné provedeme v nějakém startovacím skriptu jako `.profile` nebo `.bash.rc`.

Jelikož některé aplikace dávají přednost systémovým identifikátorům před veřejnými, je užitečné do katalogového souboru `docbook.cat` přidat řádku

```
SYSTEM "http://www.oasis-open.org/docbook/xml/4.3/docbookx.dtd" "docbookx.dtd"
```

Správnou instalaci DTD můžeme otestovat například tak, že pomocí parseru provedeme validaci již existujícího docbookového dokumentu.

9.2 DSSSL styly a Jade

DSSSL styly původně vytvořil Norman Walsh. Nyní styly vyvíjí skupina vývojářů pod jeho vedením. Styly je možné získat na adrese http://docbook.sourceforge.net. Styly jsou distribuovány jako ZIP archiv, který můžeme rozbalit například do adresáře `c:\docbook\dsssl`. Na stejné stránce jako styly můžeme získat i jejich dokumentaci.

Pro zpracování stylů budeme potřebovat nějaký DSSSL procesor. V současné době je možné používat dva – Jade (http://www.jclark.com/jade/) a jeho dále vyvíjeného následníka OpenJade (http://openjade.sourceforge.net/). Dále budeme předpokládat, že jsme si Jade stáhli a nainstalovali jej do adresáře `c:\docbook\jade`. Je vhodné tento adresář přidat do cesty, aby se nám Jade snadno spouštěl.

Pro bezproblémovou práci DSSSL stylů s XML dokumenty budeme ještě potřebovat soubor http://www.oasis-open.org/cover/ISOEnts.zip. Ten obsahuje deklarace standardních ISO entit v SGML

formátu. Soubor rozbalíme například do adresáře `c:\docbook\isoent`. Ve stejném adresáři vytvoříme soubor `isoent.cat`, který slouží jako katalog.

```
PUBLIC "ISO 8879:1986//ENTITIES Diacritical Marks//EN" "isodia"
PUBLIC "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN" "isonum"
PUBLIC "ISO 8879:1986//ENTITIES Publishing//EN" "isopub"
PUBLIC "ISO 8879:1986//ENTITIES General Technical//EN" "isotech"
PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN" "isolat1"
PUBLIC "ISO 8879:1986//ENTITIES Added Latin 2//EN" "isolat2"
PUBLIC "ISO 8879:1986//ENTITIES Greek Letters//EN" "isogrk1"
PUBLIC "ISO 8879:1986//ENTITIES Monotoniko Greek//EN" "isogrk2"
PUBLIC "ISO 8879:1986//ENTITIES Greek Symbols//EN" "isogrk3"
PUBLIC "ISO 8879:1986//ENTITIES Alternative Greek Symbols//EN" "isogrk4"
PUBLIC "ISO 8879:1986//ENTITIES Added Math Symbols: Arrow Relations//EN" "isoamsa"
PUBLIC "ISO 8879:1986//ENTITIES Added Math Symbols: Binary Operators//EN" "isoamsb"
PUBLIC "ISO 8879:1986//ENTITIES Added Math Symbols: Delimiters//EN" "isoamsc"
PUBLIC "ISO 8879:1986//ENTITIES Added Math Symbols: Negated Relations//EN" "isoamsn"
PUBLIC "ISO 8879:1986//ENTITIES Added Math Symbols: Ordinary//EN" "isoamso"
PUBLIC "ISO 8879:1986//ENTITIES Added Math Symbols: Relations//EN" "isoamsr"
PUBLIC "ISO 8879:1986//ENTITIES Box and Line Drawing//EN" "isobox"
PUBLIC "ISO 8879:1986//ENTITIES Russian Cyrillic//EN" "isocyr1"
PUBLIC "ISO 8879:1986//ENTITIES Non-Russian Cyrillic//EN" "isocyr2"
```

DSSSL styly jsou ve skutečnosti SGML dokumenty, pro jejich správné zpracování potřebujeme použít další katalogové soubory. Do `SGML_CATALOG_FILES` musíme ještě před katalog pro DocBook přidat katalog pro DSSSL styly (ten se distribuuje s Jade) a pro ISO entity.

```
SGML_CATALOG_FILES=c:\docbook\jade\catalog;c:\docbook\isoent\isoent.cat;c:\docbook\dtd\docbook.cat
```

Pro bezproblémový provoz je potřeba zavést katalogové soubory v tomto pořadí.

Jade je program původně určený pro SGML. Výběr správného kódování pro dokument nemusí být vždy úplně automatický. Kódování, které bude Jade na vstupu očekávat, se získává z proměnné prostředí `SP_ENCODING`. Pokud máme dokumenty v XML a používáme různá kódování, je vhodné nastavit proměnnou na hodnotu `XML`. Jade pak pro výběr kódování používá pravidla definovaná ve specifikaci XML. Jediný problém je v tom, že Jade pro kódování windows-1250 používá nestandardní identifikátor `windows`. Pokud máme dokumenty v tomto kódování, musíme proměnnou `SP_ENCODING` nastavit na hodnotu `windows` a všechny dokumenty pak musíme mít v kódování windows-1250. Kódování iso-8859-n jsou rozpoznávána správně, takže si můžeme vybrat mezi nastavením proměnné na hodnotu `XML` nebo `ISO-8859-2`.

9.3 XSL styly, XSLT a FO procesor

XSL styly opět vytvořil Norman Walsh a nyní se o jejich další vývoj stará širší komunita vývojářů. Je možné je stáhnout z adresy http://docbook.sourceforge.net. Styly jsou distribuovány jako ZIP archiv, který můžeme rozbalit například do adresáře `c:\docbook\xsl`. Dokumentace je ke stylům he šířena jako samostatný archiv.

Pro zpracování stylů budeme potřebovat nějaký XSLT a případně i FO procesor.

9.3.1 XSLT procesor

Mezi nejlepší dnešní XSLT procesory patří Saxon. Pro pohodlnou práci se Saxonem je velice vhodné v něm aktivovat podporu katalogů a XInclude.

Saxon s podporou XML katalogů a XInclude

Všechny komponenty Saxonu a podpora katalogových souborů jsou dostupné jako javové archivy. Pro lepší orientaci můžeme všechny potřebné soubory nahrát do společného adresáře, např. `c:\docbook\batch`. Jedná se přitom o následující soubory:

`saxon.jar`	Samotný XSLT procesor Saxon. Tento javový archiv je obsažen přímo v jeho distribuci – http://saxon.sourceforge.net. Je potřeba použít Saxon z řady 6.x (např. 6.5.5).
`resolver.jar`	Samotná podpora katalogových souborů. Je součástí balíku XML Commons dostupného na adrese http://mirror.styx.cz/apache/xml/commons/, konkrétně v souboru `xml-commons-resolver-1.x.zip`.
`xercesImpl.jar`	Parser Xerces. Umožňuje spolupráci Saxonu s katalogovými soubory a přidává implementaci XInclude. Je dostupný na adrese http://xml.apache.org/dist/xerces-j/. Novější verze Javy (od verze 1.4 výše) obsahují již Xerces přímo v sobě, takže jej není nutné instalovat.
`saxon643.jar`	Rozšiřující funkce pro lepší zpracování DocBooku. Soubor je přímo součástí standardní distribuce XSL stylů (v podadresáři `extensions`).

Pro snadné spouštění Saxonu si můžeme vytvořit dávkový soubor `saxon.bat`.

```
@java -cp ▶
c:\docbook\batch\;c:\docbook\batch\resolver.jar;c:\docbook\batch\xercesImpl.jar;c:\docbook\batch\saxon.jar;c:\docbook\batch\saxon65.jar▶
-Dorg.apache.xerces.xni.parser.XMLParserConfiguration=org.apache.xerces.parsers.XIncludeParserConfiguration ▶
-Djavax.xml.parsers.DocumentBuilderFactory=org.apache.xerces.jaxp.DocumentBuilderFactoryImpl ▶
-Djavax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl ▶
com.icl.saxon.StyleSheet -x org.apache.xml.resolver.tools.ResolvingXMLReader -y ▶
org.apache.xml.resolver.tools.ResolvingXMLReader -r ▶
org.apache.xml.resolver.tools.CatalogResolver %*
```

Javová cesta ke třídám je nastavena i přímo do adresáře `c:\docbook\batch`, protože zde je potřeba umístit soubor `CatalogManager.properties`. Pokud chceme, aby se použil katalogový soubor pro DocBook, můžeme do souboru uložit následující:

```
#CatalogManager.properties

verbosity=1

# Always use semicolons in this list
catalogs=file:///c:/docbook/dtd/catalog.xml

prefer=public

static-catalog=yes

allow-oasis-xml-catalog-pi=yes

catalog-class-name=org.apache.xml.resolver.Resolver
```

Od verze 4.2 je součástí distribuce DTD (`c:\docbook\dtd`) i odpovídající soubor `catalog.xml` s XML katalogovým souborem, který používáme. Jen pro ilustraci se můžeme podívat na to, jak zhruba vypadá.

```
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
         prefer="public">

    <public publicId="-//OASIS//DTD DocBook XML V4.2//EN"
            uri="docbookx.dtd"/>
    <public publicId="-//OASIS//ENTITIES DocBook XML Notations V4.2//EN"
            uri="dbnotnx.mod"/>
    <public publicId="-//OASIS//ENTITIES DocBook XML Character Entities V4.2//EN"
            uri="dbcentx.mod"/>
    <public publicId="-//OASIS//ELEMENTS DocBook XML Information Pool V4.2//EN"
            uri="dbpoolx.mod"/>
    <public publicId="-//OASIS//ELEMENTS DocBook XML Document Hierarchy V4.2//EN"
            uri="dbhierx.mod"/>
    <public publicId="-//OASIS//ENTITIES DocBook XML Additional General Entities V4.2//EN"
            uri="dbgenent.mod"/>
    <public publicId="-//OASIS//DTD DocBook XML CALS Table Model V4.2//EN"
            uri="calstblx.dtd"/>

</catalog>
```

V souboru `CatalogManager.properties` můžete do vlastnosti catalogs zadat i více souborů, stačí je oddělit středníkem.

Saxon s XML katalogy je schopný mapovat i URL adresy stylů tak, aby byly načítány z lokálního disku a ne ze sítě. Pro aktivování této funkce stačí do katalogového souboru přidat instrukci:

```
<rewriteURI
  uriStartString="http://docbook.sourceforge.net/release/xsl/current/"
  rewritePrefix="file:///c:/docbook/xsl/"/>
```

Saxon s podporou SGML catalogů

Katalogové soubory původně vznikly pro jazyk SGML a jsou definovány v Technickém prohlášení OASIS TR9401[1]. Této syntaxi katalogových souborů rozumí i Jade a proto můžeme používat jeden katalog pro různé programy.

Poznámka

Doporučuji vám používat Saxon s podporou XML katalogů, protože nabízí více funkcí, např. mapování stylů na jejich lokální kopie. Následující postup je zde zachován spíše z historických důvodů.

Všechny komponenty Saxonu a podpora katalogových souborů jsou dostupné jako javové archivy. Pro lepší orientaci můžeme všechny potřebné soubory nahrát do společného adresáře, např. `c:\docbook\batch`. Jedná se přitom o následující soubory:

`saxon.jar`	Samotný XSLT procesor Saxon. Tento javový archiv je obsažen přímo v jeho distribuci – http://saxon.sourceforge.net.
`crimson.jar`	Parser, který podporuje mnoho kódování včetně iso-8859-2 a windows-1250. Parser můžeme získat například na adrese http://xml.apache.org/dist/crimson/.

[1] http://www.oasis-open.org/html/a401.htm

`saxoncatalog.jar`	Soubor s podporou katalogových souborů pro Crimson. Lze stáhnout z http://www.kosek.cz/xml/saxon/saxoncatalog.jar.
`saxon65.jar`	Rozšiřující funkce. Soubor je přímo součástí standardní distribuce XSL stylů.

Pro snadné spouštění Saxonu si můžeme vytvořit dávkový soubor `saxon.bat`.

```
@java -cp ▶
c:\docbook\batch\crimson.jar;c:\docbook\batch\saxoncatalog.jar;c:\docbook\batch\saxon.jar;c:\docbook\batch\saxon65.jar ▶
-Dxml.catalog.files=%SGML_CATALOG_FILES% com.icl.saxon.StyleSheet -w0 -x ▶
cz.kosek.CatalogXMLReader -y cz.kosek.CatalogXMLReader %1 %2 %3 %4 %5 %6 %7 %8 %9
```

xsltproc

Místo Saxonu můžete použít libovolný jiný XSLT procesor, i když těžko najdete nějaký jiný, který nabízí srovnatelný výkon a funkčnost. V některých případech je ještě rychlejší než Saxon xsltproc[2]. Zatím pro něj však nejsou hotová všechna rozšíření potřebná pro některé pokročilé funkce stylů a občas se v xsltproc objeví nějaká chyba.

Windows verzi xsltproc můžete získat na adrese http://www.zlatkovic.com/projects/libxml/binaries.html. Stáhněte si binární distribuci knihoven libxml, libxslt, iconv a zlib. Ze stažených archivů stačí nakopírovat následující soubory do nějakého adresáře, která je v cestě (např. `c:\docbook\batch`).

```
libxml2.dll                libexslt.dll               xsltproc.exe
iconv.dll                  zlib1.dll
libxslt.dll                xmllint.exe
```

Aby xsltproc našel katalogový soubor (použije se stejný XML katalog jako v předchozí sekci pro Saxon), musíme jeho URI adresu nastavit v proměnné prostředí `XML_CATALOG_FILES`. V našem případě bychom ji nastavili na hodnotu `file:///c:/docbook/dtd/catalog.xml`.

9.3.2 FO procesor

Instalace FO procesoru závisí na konkrétním FO procesoru, který používáte.

XEP

FO procesor od firmy RenderX[3]. Kromě komeční placené verze lze pro osobní použití zdarma získat verzi „personal“, která na každou stránku dolů vloží malé logo. Počeštění XEPu (české vzory dělení slov a podpora českých znaků), je dostupné na adrese http://www.kosek.cz/sw/xep/.

XEP je schopný před samotným formátováním provést rovnou i transformaci. V tom případě je dobré si dávku pro spouštění XEPu upravit tak, aby XEP rovnou podporoval XML katalogy a XInclude, tak jak je popsáno v „Saxon s podporou XML katalogů a XInclude“. Dávka `xep.bat` pak může vypadat například takto:

```
@echo off
rem   This batch file encapsulates a standard XEP call.

set ▶
CP=%~dp0\xercesImpl.jar;%~dp0\;%~dp0\resolver.jar;%~dp0\saxon65.jar;%~dp0\..\XEP\lib\xep.jar;%~dp0\..\XEP\lib\saxon.jar;%~dp0\..\XEP\lib\xt.jar
```

[2] http://xmlsoft.org/XSLT/
[3] http://www.renderx.com

```
java -Xmx512M -classpath "%CP%" ▶
-Dorg.apache.xerces.xni.parser.XMLParserConfiguration=org.apache.xerces.parsers.XIncludeParserConfiguration ▶
-Dcom.renderx.sax.entityresolver=org.apache.xml.resolver.tools.CatalogResolver ▶
-Dcom.renderx.jaxp.uriresolver=org.apache.xml.resolver.tools.CatalogResolver ▶
com.renderx.xep.XSLDriver "-DCONFIG=%~dp0\..\XEP\xep.xml" %*

set CP=
```

XFC

XFC[4] je komerční procesor FO, který je však dostupný zdarma (ve verzi „standard“). Umí formátovací objekty převádět do formátů RTF, WordML (formát dokumentů podporovaný v MS Word 2003 a vyšší) a ODF (podporovaný např. v OpenOffice).

FOP

FOP je open-source procesor od sdružení Apache – http://xml.apache.org/dist/fop/. Standardní verze neobsahuje podporu českých znaků a dělení slov, je však k dispozici samostatně na adrese http://www.kosek.cz/sw/fop/index.html.

Antenna House

Opět komerční FO procesor, který ve své zkušební verzi doplní na každou stránku reklamní odkaz. Umožňuje prohlížení výsledného dokumentu přímo na obrazovce. Podporuje češtinu. K dispozici je na adrese http://www.antennahouse.com/. České vzory pro dělení jsou k dispozici na adrese http://www.kosek.cz/sw/axf/index.html.

[4] http://www.xmlmind.com/foconverter/

Kapitola 10. Podpora DocBooku v editorech

Pro efektivní vytváření dokumentů v DocBooku samozřejmě potřebujeme editor, který nám maximálně usnadní práci. Nyní se podíváme na XML editory, které se ve spojení s DocBookem nejčastěji používají. Ukážeme si, jak je připravit pro editování dokumentů v DocBooku. Mnoho editorů se přitom dnes dodává již se zabudovanou podporou DocBooku – obsahují DTD i styly pro následnou konverzi DocBooku do výstupních formátů.

10.1 XMLmind XML Editor

XMLmind XML Editor[1] je WYSIWYG editor XML, který je pro osobní použití zdarma. Uživatelské rozhraní není sice tak příjemné jako u editorů Epic a XMetaL, ale jeho nedostatky v mnoha případech vyváží cena (je zdarma i pro komerční využití). Navíc se dodává s předkonfigurovanou podporou DocBooku a zvládá editování dokumentů rozdělených pomocí entit nebo XInclude do několika souborů.

Placená verze umožňuje i snadné spouštění konverze DocBooku do různých výstupních formátů.

10.2 oXygen

oXygen[2] je v dnešní době asi jeden z nejlepších editorů XML na úrovni zdrojového kódu. Ihned po instalaci obsahuje podporu DocBooku – DTD i styly.

10.3 Emacs+PSGML

Nastavení Emacs a PSGML je podrobně popsáno v několika publikacích, proto se jím nebudeme detailně zabývat [13], [14].

10.4 Emacs+nXml

http://www.thaiopensource.com/nxml-mode/

10.5 jEdit

jEdit[3] je editor napsaný v Javě a nabízí pomocí plug-inů poměrně komfortní prostředí pro editování dokumentů XML, včetně doplňování elementů a atributů na základě DTD. Pro pohodlnou práci s XML je užitečné nastavení, kdy kromě dokumentu vidíme vlevo jeho stromovou strukturu, vpravo se nám nabízejí podle DTD elementy ke vložení a v dolní části vidíme seznam chyb. Toto rozložení můžeme nastavit pomocí příkazu Utilities▻Global Options.... Nyní vybereme jEdit▻Docking. Pro okno Structure Browser nastavíme dokovací pozici left, pro XML Insert right a pro Error List bottom.

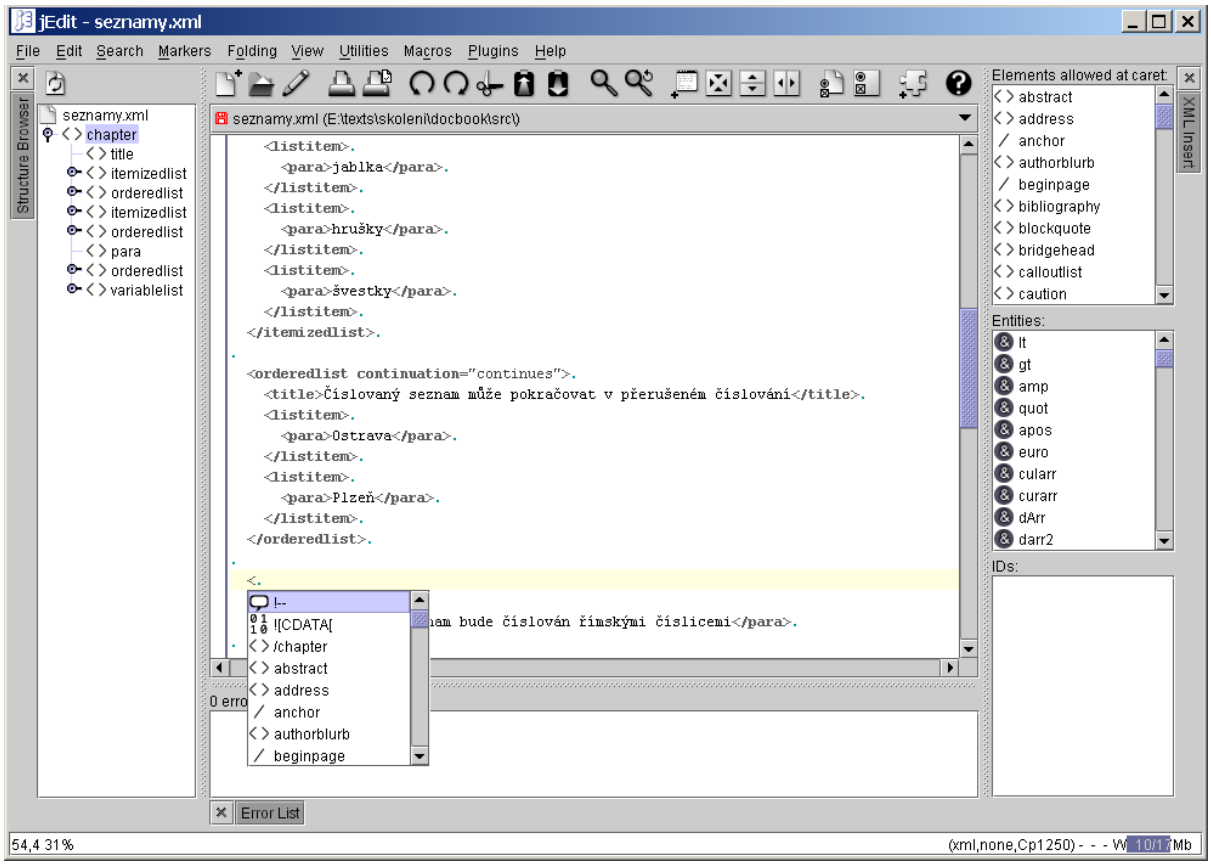
Dokument si můžeme nechat kdykoliv zvalidovat nebo zkontrolovat alespoň well-formedness. Stačí jej uložit, při ukládání se automaticky spustí parser a v okně Error List si můžeme prohlédnout jednotlivé chyby.

[1] http://www.xmlmind.com/xmleditor
[2] http://www.oxygenxml.com/
[3] http://www.jedit.org

Pro editování dokumentů v DocBooku je vhodné přidat si do jEditu katalogový soubor, který zamezí načítání DTD ze sítě. Zvolíme opět Utilities▻Global Options... a nyní Plugins▻XML▻Catalogs. A přidáme katalogový soubor pro DocBook – v našem případě `c:\docbook\dtd\docbook.cat`.

Pro automatické doplnění ` ` za neslabičné předložky lze použít plug-in tildify[4].

Obrázek 10.1. jEdit při editování DocBookového dokumentu



10.6 Epic

Epic je dodáván již s několika předinstalovanými DTD. Nejlépe nakonfigurované je DTD, které vychází z DocBooku 4 a přidává do něj podporu pro vkládání matematických vzorců. Editor je pro toto DTD velice dobře nakonfigurován a to se netýká jen stylů. Podle kontextu v dokumentu se například mění význam klávesy **Enter** – v odstavci její stisk způsobí vytvoření nového odstavce, v nadpisu pak vložení prvního odstavce za nadpis, ve výpisu programu přechod na nový řádek. V liště programu přibudou tlačítka pro snadnu změnu formátování či pro vytváření seznamů. Podobné chování lze samozřejmě vytvořit pro libovolné DTD, ale přece jen to jen to určitý čas zabere.

Pokud si k Epicu dokoupíte publikační moduly, umí editor sám docbookové dokumenty formátovat pro tisk, převádět do HTML, HTML Helpu a dalších formátů. Standardně je podporována profilace dokumentu pro různé skupiny čtenářů – v dokumentu stačí pomocí příznaků označit vybrané elementy, při tisku si pak můžeme vybrat profil, pro který se bude výstup generovat.

S Epicem je standardně dodávána i zcela běžná SGML verze DocBooku 4.0. My si v následujícím textu popíšeme, jak do Epicu přidat podporu standardní XML verze DocBooku.

[4] http://home.tiscali.cz/ca895221/tildify/index.html

10.6.1 Instalace poslední verze DocBooku do Epicu

Pokud pro zpracování DocBooku budeme používat i jiné aplikace než Epic, patrně nám nebude vyhovovat použití upravené verze DocBooku. Nezbude nám nic jiného, než do Epicu přidat standardní XML verzi DocBooku.

Požadavky na instalaci

Pro úspěšnou instalaci potřebujeme Epic Editor a Epic Architect a samozřejmě DTD pro DocBook. Následující postup předpokládá, že Epic je nainstalován v adresáři `f:\program files\epic` a že DTD pro DocBook máme v adresáři `e:\sgml\dtd\dbx412`. Pokud máte soubory v jiných adresářích, musíte v postupu odpovídajícím způsobem zaměnit adresáře.

Postup instalace

Celá instalace se skládá z několika kroků.

1. **Vytvoření adresářů pro naše vlastní DTD**

   Uživatelem definovaná DTD je dobré nahrát do jiného adresáře, než do toho, kde je Epic nainstalován. Ušetříme si tak problémy při případném reinstalaci nebo upgradu programu.

   Na libovolném místě na disku si vytvoříme adresář (např. `f:\epic`) a v něm vytvoříme čtyři podadresáře se jmény `doctypes`, `entities`, `graphics` a `lib`.

2. **Nastavení cest v Epic Editoru a Epic Architectu**

   Spustíme Epic Editor nebo Epic Architect a z menu vybereme příkaz Options▻Preferences. Přepneme se na záložku File a do polí Catalog Path, Graphics Path a Entities Path doplníme cestu k námi vytvořeným adresářům (viz obrázek 10.2 – „Nastavení cest v Epicu“).

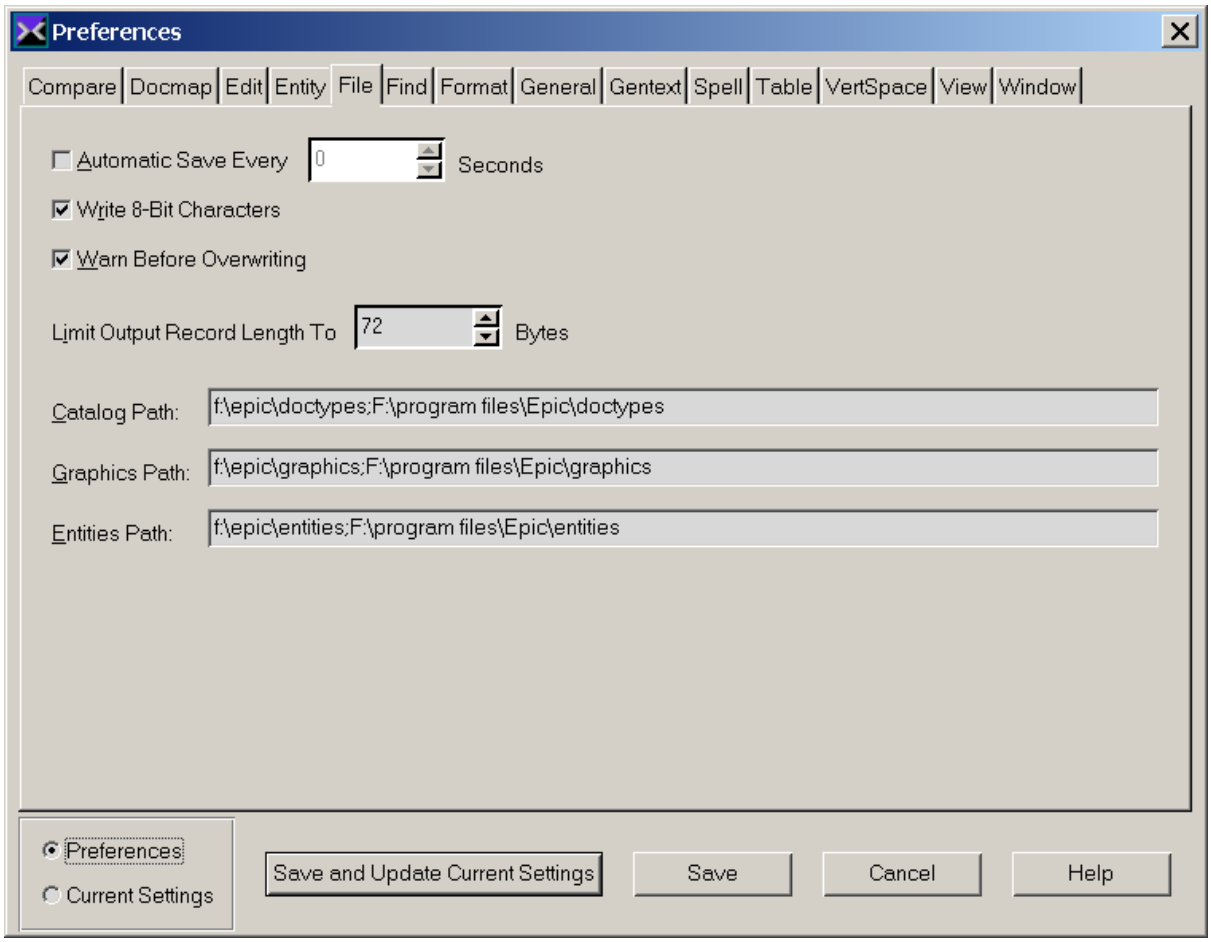
   Změny uložíme stiskem tlačítka Save and Update Current Settings.

3. **Zkompilování DTD DocBook do interního formátu Epicu**

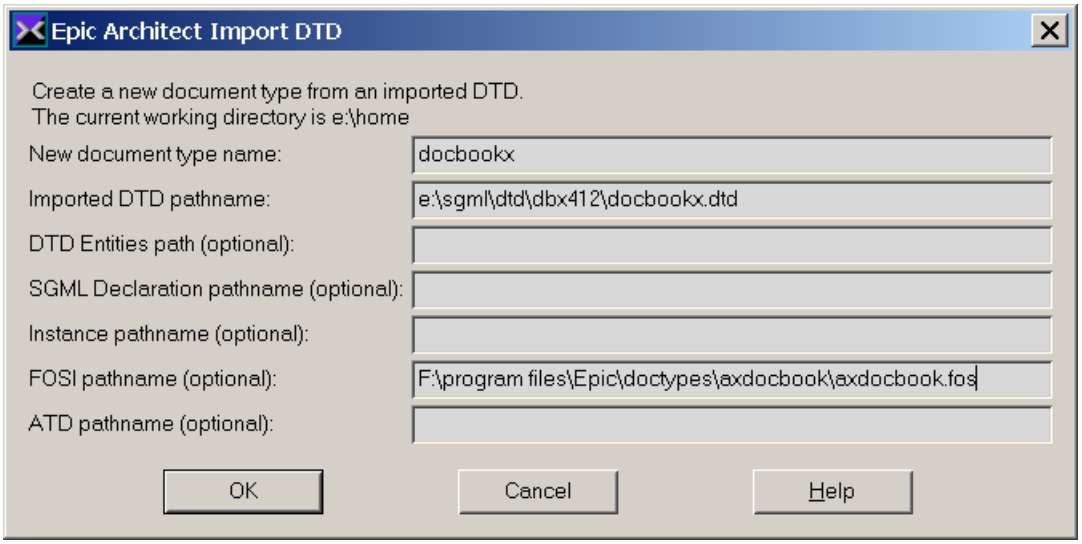
   DTD bychom měli zkompilovat do pracovního adresáře, odkud se pak bude instalovat. Epic Architect si pracovní adresář načítá z proměnné prostředí `HOME`. V našem ukázkovém postupu jsem proměnnou nastavil na hodnotu `e:\home`.

   Spustíme Epic Architect. V menu Compile aktivujeme volbu XML Application. Vybereme příkaz File▻Import. Do dialogového okna vyplníme údaje jako na obrázku 10.3 – „Import DTD pro DocBook“. Stačí vyplnit námi zvolený název, cestu k DTD a cestu ke stylům pro upravený DocBook od ArborTextu – tím vyřešíme problém se zobrazováním dokumentů.

Obrázek 10.2. Nastavení cest v Epicu



Obrázek 10.3. Import DTD pro DocBook

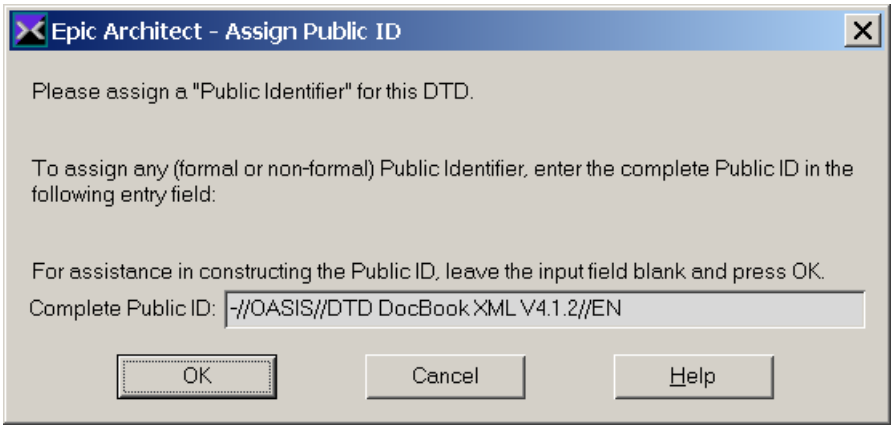


Tlačítkem OK pak spustíme import.

4. **Průběh importu**

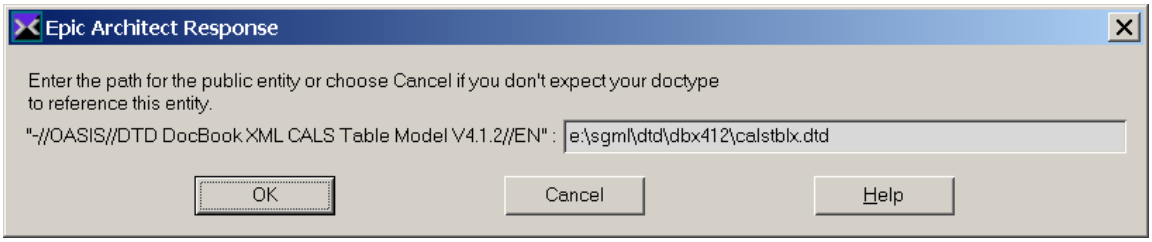
Na začátku importu se kompilátor ptá, jaký veřejný identifikátor má používat pro dané DTD (viz obrázek 10.4 – „Nastavení veřejného identifikátoru“).

Obrázek 10.4. Nastavení veřejného identifikátoru



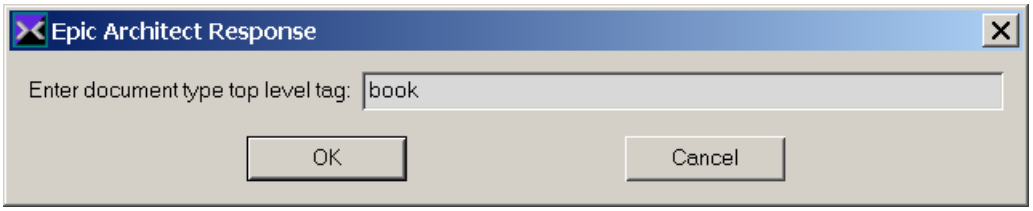
Během importu Epic Architect nenajde entitu, která obsahuje fragment DTD pro tvorbu tabulek. Cestu k tomuto souboru musíme zadat ručně (viz obrázek 10.5 – „Ruční zadání cesty k entitě, kterou nelze nalézt“).

Obrázek 10.5. Ruční zadání cesty k entitě, kterou nelze nalézt



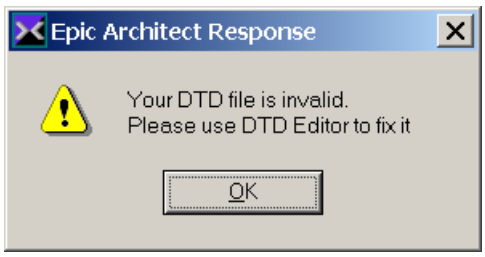
Nakonec se nás program zeptá na kořenový element. Většinou budeme v DocBooku asi vytvářet knihy, vybereme proto `book` (viz obrázek 10.6 – „Volba kořenového elementu“).

Obrázek 10.6. Volba kořenového elementu



Poznámka

Někdy instalace skončí neúspěšně s chybovým hlášením o špatném DTD.

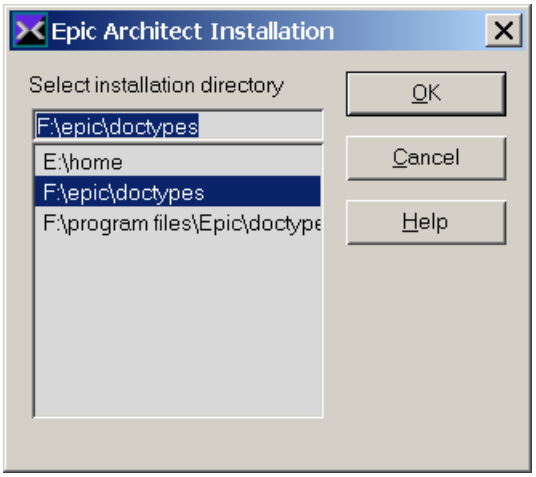


Epic Architect se v tomto případě chybně přepnul do SGML režimu. Stačí v menu znovu vybrat XML režim (Compile▻XML Application).

5. **Instalace nového DTD do editoru**

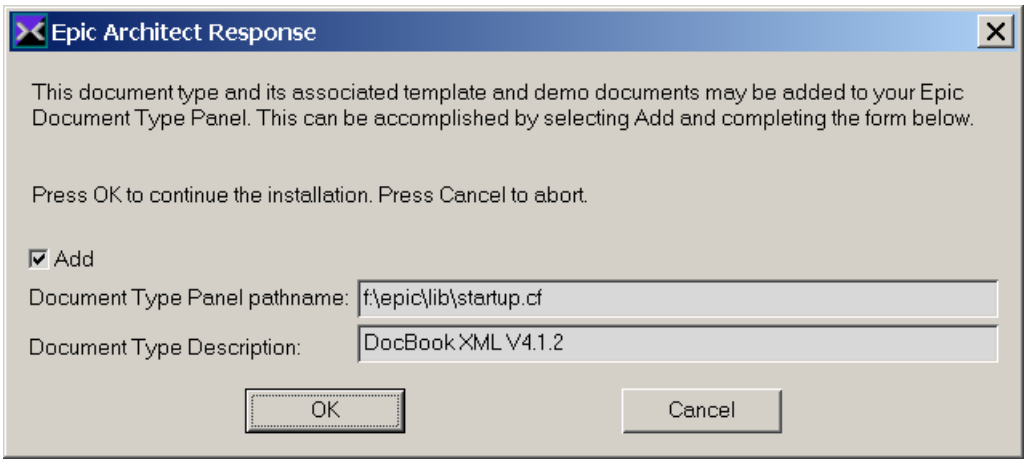
Po kompilaci ještě musíme DTD nainstalovat. Vybereme příkaz File▻Install. Zvolíme místo, kam se má DTD instalovat (`f:\epic\doctypes`) – obrázek 10.7 – „Výběr místa, kam se má DTD nainstalovat“.

Obrázek 10.7. Výběr místa, kam se má DTD nainstalovat



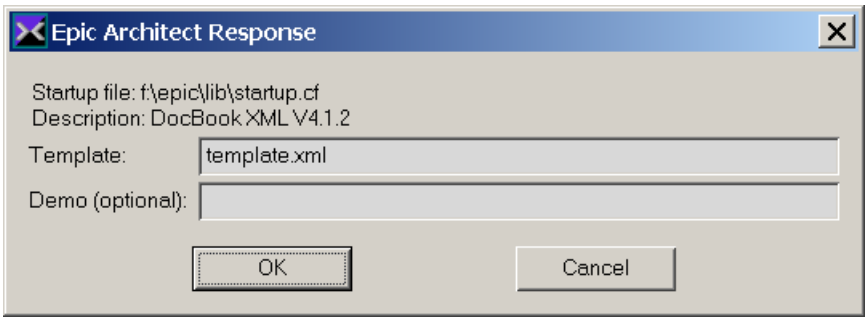
Dále si musíme zvolit jméno, pod kterým nám bude Epic Editor nabízet naše DTD při vytváření nových dokumentů (viz obrázek 10.8 – „Výběr názvu pro naše DTD“). Pokud chceme vytvořit šablonu pro nové dokumenty, musíme zaškrtnout volbu Add.

Obrázek 10.8. Výběr názvu pro naše DTD



Na výzvu pro zadání souborů se šablonou, zadáme jako jméno souboru `template.xml` (obrázek 10.9 – „Výběr souborů se šablonou“).

Obrázek 10.9. Výběr souborů se šablonou



6. **Upravení šablony pro nové dokumenty**

   V šabloně pro nové dokumenty, která se nám přidá do Epicu, je špatně doplněný (nebo chybí) systémový identifikátor. Pro lepší přenositelnost dokumentů je vhodné nahradit ho standardní hodnotou. Otevřeme soubor `e:\epic\doctypes\docbookx\template.xml` a opravíme ho na:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN" "http://www.oasis-open.org/▶
docbook/xml/4.0/docbookx.dtd">
<book><?Pub Caret1?></book>
```

   V případě potřeby můžeme změnit použité kódování nebo doplnit poněkud větší kostru dokumentu. Instrukce pro zpracování určuje místo, kam se umístí kurzor po otevření nového dokumentu.

10.6.2 Epic a české znaky

Epic se zpracováním českých znaků nemá žádné problémy. Standardně pracuje v kódování UTF-8, ale zvládá i ISO-8859-2. Pokud chceme, aby Epic používal jiné kódování než UTF-8, stačí ve stávajících dokumentech zaměnit informaci o kódování v XML deklaraci, pro nové dokumenty se kódování přebírá ze šablony (více viz předchozí sekce).

Standardně se české znaky ukládají jako odkazy na odpovídající entity (`&eacute;` apod.). Pokud chceme, aby se znaky ukládaly přímo, stačí aktivovat volbu Options▻Preferences▻File▻Write 8-Bit Characters.

10.7 XMetaL 1.x a 2.0

10.7.1 Příprava DTD

XMetaL 1.x a 2.0 používá vlastní katalogové soubory, proto je dobré pro bezproblémovou práci přidat položky pro DocBook do souboru `extid.map`, který je v adresáři s instalací XMetaLu. Do souboru přidáme řádky:

```
#
# DocBook
#
"-//OASIS//DTD DocBook XML V4.3//EN" ! "c:\docbook\dtd\docbookx.dtd"
! "http://www.oasis-open.org/docbook/xml/4.3/docbookx.dtd" "c:\docbook\dtd\docbookx.dtd"
```

Při prvním otevření dokumentu, který používá toto DTD si ho XMetaL zkompiluje. Při první kompilaci mu občas musíme pomoci nalézt nějaké entity, pak už však vše pracuje tak, jak má.

10.7.2 Přizpůsobení editoru pro dané DTD

Pro pohodlnou práci v editoru musíme mít odpovídající CSS styl definující zobrazení dokumentu a je dobré upravit pravidla pro DocBook – editor pak rozpozná, co je obrázek, co seznam apod.

Kromě toho lze editor plně přizpůsobit pomocí zabudovaného skriptovacího jazyka (JScript, VBScript a další).

10.7.3 Psání českých znaků v XMetaLu

XMetaL 1.x a 2.0 bohužel nepodporuje Unicode, takže v něm nelze rovnou psát dokumenty v češtině. Nicméně lze editor „znásilnit“ tak, aby v něm šlo psát dokumenty v kódování windows-1250.

Nejprve musíme upravit hodnoty několika parametrů v souboru `xmetal.ini` v adresáři s XMetaLem:

```
export_convert_to_ISOlat1 = false
export_convert_to_ISOnum = false
export_convert_remainder_to_char_ref = false
import_convert_ISOlat1 = false
import_convert_ISOnum = false
import_convert_char_ref = false
```

V kaskádovém stylu pak musíme nastavit font, který má ve svém názvu na konci CE.

10.8 XMetaL 2.1 a 3.0

Nová verze XMetaLu přináší mnohá vylepšení. Konečně jsou podporovány standardní katalogové soubory. Pro bezproblémové editování docbookových dokumentů stačí do souboru `c:\Program Files\SoftQuad\XMetaL 2\Rules\catalog` přidat řádku, která odkazuje na katalog pro DocBook DTD.

```
CATALOG "c:\docbook\dtd\docbook.cat"
```

Provozujete-li XMetaL 2.1 ve Windows 2000/NT je k dispozici podpora Unicode. Lze tak bez problémů psát české texty. Editor podporuje pouze kódování UTF-8 a UTF-16.

Chceme-li, aby editor lépe zobrazoval dokumenty a usnadňoval editaci, musíme mít k dispozici kaskádový styl a soubor s úpravami. Ukázka těchto souborů je dostupná na adrese http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/docbook/contrib/tools/xmetal/.

10.9 XMetaL 4.x

Postup úprav nejnovější verze XMetaLu je podobný jako u předchozích verzí, ale je potřeba mít k dispozici vývojové prostředí XMetaL Developer. Úpravy XMetaLu pro komfortní použití s DocBookem je možné získat na komerční bázi[5].

[5] http://docbook.cz/konzultace.html

Literatura a další zajímavé odkazy

DocBook obecně

[1] Norman Walsh a Leonard Muellner. *DocBook*. The Definitive Guide. 1999. 1. ISBN 156592-580-7. 648. URL: http://www.docbook.org/tdg/en/html/docbook.html.

[2] Nik Clayton. *FreeBSD Documentation Project Primer for New Contributors*. URL: http://www.freebsd.org/doc/en_US.ISO8859-1/books/fdp-primer/index.html.

[3] David Mason, Daniel Mueth a Alexander Kirillov. *The GNOME Handbook of Writing Software Documentation*. URL: http://developer.gnome.org/projects/gdp/handbook/gdp-handbook/.

[4] Mark F. Komarinski, Jorge Godoy a David C. Merrill. *LDP Author Guide*. URL: http://www.linuxdoc.org/LDP/LDP-Author-Guide/.

[5] David Rugge, Mark Galassi a Eric Bischoff. *Writing Documentation Using DocBook*. A Crash Course. URL: http://opensource.bureau-cornavin.com/crash-course/.

[6] Dave Pawson. *Docbook Frequently Asked Questions*. URL: http://www.dpawson.co.uk/docbook/index.html.

[7] *DocBook WiKi*. URL: http://wiki.docbook.org/topic/.

Styly pro DocBook

[8] Norman Walsh, Bob Stayton a Jiří Kosek. *DocBook XSL Stylesheet Documentation*. URL: http://docbook.sourceforge.net/release/xsl/current/doc/.

[9] Robert Stayton. *DocBook XSL: The Complete Guide*. URL: http://www.sagehill.net/docbookxsl/index.html.

[10] Norman Walsh. *The Modular DocBook Stylesheets*. URL: http://docbook.sourceforge.net/release/dsssl/current/doc/.

[11] Norman Walsh. *The Design of the DocBook XSL Stylesheets*. URL: http://nwalsh.com/docs/articles/dbdesign/.

[12] Michael Wiedmann. *DocBook DSSSL Stylesheet FAQ*. URL: http://www.miwie.org/docbook-dsssl-faq.html.

Konfigurace nástrojů

[13] Markus Hoenicka. *SGML for NT*. A brief tutorial how to set up a free SGML editing and publishing system for Windows NT. URL: http://ourworld.compuserve.com/homepages/hoenicka_markus/ntsgml.html.

[14] Lenka Třísková. *GNU nástroje pro tvorbu WWW stránek*. Grada Publishing. 2000. ISBN ISBN 80-7169-861-X. 244.

[15] Pavel Žampach. *Použití DocBooku v českém prostředí*. URL: http://www.volny.cz/zampach/dbk/.

XSL

[16] Sharon Adler, Anders Berglund, Jeff Caruso, Stephen Deach, Paul Grosso, Eduardo Gutentag, Alex Milowski, Scott Pernell, Jeremy Richman a Steve Zilles. *Extensible Stylesheet Language (XSL) – Version 1.0*. W3C. 2000. URL: http://www.w3.org/TR/xsl.

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

[18] *Practical Transformation Using XSLT and XPath*. Crane Softwrights. 2000. ISBN 1-894049-04-7. URL: http://www.cranesoftwrights.com.

[19] Michael Kay. *XSLT Programmer's Reference*. Wrox Press. 2000. ISBN 1-861003-12-9.

[20] Jiří Kosek. *XSLT v příkladech*. URL: http://www.kosek.cz/xml/xslt/.

[21] Sal Mangano. *XSLT Cookbook*. O'Reilly. 2003. ISBN 0-596-00373-2.

[22] *XSLT and XPath Quick Reference*. Mulberry Technologies. 2000. URL: http://www.mulberry-tech.com/quickref/XSLTquickref.pdf.

[23] Miloslav Nič. *XPath Tutorial*. URL: http://www.zvon.org/xxl/XPathTutorial/General/examples.html.

[24] Miloslav Nič. *XSLT Tutorial*. URL: http://www.zvon.org/xxl/XSLTutorial/Books/Book1/index.html.

DSSSL

[25] *Information technology – Processing languages – Document Style Semantics and Specification Language (DSSSL)*. ISO/IEC 10179:1996(E). URL: ftp://ftp.ornl.gov/pub/sgml/WG8/DSSSL/dsssl96b.pdf. URL: ftp://ftp.ornl.gov/pub/sgml/WG8/DSSSL/readme.htm.

[26] Javier Farreres. *The DSSSL Book*. An XML/SGML Programming Language. Kluwer Academic Publishing. 2004. ISBN 1-4020-7592-8.

[27] *Stránky věnované jazyku DSSSL*. URL: http://www.netfolder.com/DSSSL/.

Diskusní skupiny

[28] *Archiv listu* `<docbook@linux.cz>`. URL: http://www.linux.cz/lists/archive/docbook/.

[29] *Archiv mailing listu* `<docbook@lists.oasis-open.org>`. URL: http://lists.oasis-open.org/archives/docbook/.

[30] *Archiv mailing listu* `<docbook-apps@lists.oasis-open.org>`. URL: http://lists.oasis-open.org/archives/docbook-apps/.

Další

[31] Norman Walsh. *Organization for the Advancement of Structured Information Standards (OASIS) Technical Memorandum TR 9901:1999*. XML Exchange Table Model Document Type Definition. 1999. OASIS. URL: http://www.oasis-open.org/html/tm9901.htm.

[32] Mark Johnson. *DocBook Bookmarks*. URL: http://www.dulug.duke.edu/~mark/docbookmarks/.

[33] Jirka Kosek. *DocBook a generování rejstříků*. URL: http://docbook.cz/clanky/dbindex.html.

[34] *České stránky o DocBooku*. URL: http://docbook.cz.