Lehký úvod do XML

Jiří Kosek
Vysoká škola ekonomická v Praze

E-mail: <jirka@kosek.cz>
Web: http://www.kosek.cz

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).

Jazyk XML (eXtensible Markup Language) je poměrně nový značkovací jazyk. Mezi jeho největší výhody patří naprostá otevřenost a velká flexibilita. Díky tomu se během krátké doby stalo XML velice populární. XML vzniklo zjednodušením jazyka SGML (Standard Generalized Markup Language), který je ISO normou 8879 z roku 1986. Kvůli své složitosti bylo SGML nasazováno jen ve větších aplikacích. XML je oproti tomu jednoduchý jazyk, který vytvořilo konsorcium W3C.

Článek byl připraven ve formátu XML s využitím DTD DocBook. Výsledné formátování bylo provedeno pomocí XSL stylů a programu PassiveTeX.

1. Úvod

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.

Samotný pojem XML se dnes používá ve třech trošku odlišných významech, na které se postupně podíváme. XML je

- formát pro výměnu a ukládání dat;
- metajazyk pro definici dalších jazyků;
- celá sada technologií, které úzce souvisejí s jazykem XML (XSL, XLink, XPointer, ...).

2. XML jako formát pro výměnu dat

Svět se začíná globalizovat, informace nabývají na důležitosti a vzrůstá potřeba jejich efektivního zpracování a vyměňování. Formátů pro výměnu dat existují stovky, ale většina z nich je jen úzce zaměřena a má mnohá omezení. XML oproti většině jiných formátu přináší mnohá vylepšení.

První přínos XML spočívá v usnadnění *vyhledávání informací*. Většina dnes dostupných informací je dnes vytvářena a ukládána v nestrukturované podobě – jako textové soubory, webové stránky apod. Efektivní vyhledávání v takovýchto datech je podmíněno porozuměním uložené informaci. To je bohužel v dnešní době stále nevyřešený problém. Snadné je vyhledávání naopak v databázích – v nich jsou všechny údaje přehledně strukturovány. Problém je však v tom, že databáze obsahují jen nepatrný zlomek informací, které máme k dispozici.

XML přináší možnost strukturování libovolných dokumentů, včetně těch textově orientovaných. XML je značkovací jazyk, což znamená, že jednotlivé části dokumentu označujeme značkami, které přesně specifikují jejich význam. Část webové stránky internetového obchodu s knihami by proto mohla v XML vypadat třeba takto:

```
<nabídka>
  <název>The Art of Computer Programming, Vol. 1</název>
  <autor>Donald E. Knuth</autor>
  <cena>40 GBP</cena>
</nabídka>
```

Vidíme, že v XML dokumentech se podobně jako v HTML používají značky pro označení částí dokumentu. Na rozdíl od HTML si však můžeme volit vlastní názvy značek, a tak co nejpřesněji vyznačit význam jednotlivých informací v textu.

Takto strukturované informace lze velice snadno prohledávat. Průměrně zdatný programátor by dokázal za pár hodin napsat program, který po zadání dotazu typu „Najdi mi stránku, kde se dá nejlevněji koupit kniha The Art of Computer Programming“ skutečně nalezne požadovanou informaci. Stačí nalézt všechny stránky, kde je v tagu `<nabídka>` obsažen tag `<název>`, který obsahuje hledaný text, a z těchto stránek vybrat tu, kde je v tagu `<cena>` uvedena nejnižší hodnota.

První z výhod kterou XML přináší je tedy usnadnění vyhledávání především v rozsáhlých kolekcích dokumentů jako je např. Web. Vyžaduje to samozřejmě, aby všichni autoři stránek označili důležité informace odpovídajícími značkami. Navíc by se měly pro stejné věci používat značky se stejnými názvy. Když si každý vymyslí vlastní názvy značek, bude mít sice pocit svobody, ale situaci tím nijak nepomůže. Dále v přednášce se proto podíváme na to, jak lze formálně specifikovat množinu značek, které lze používat v XML dokumentu.

Mezi další velkou výhodu XML patří jeho *univerzálnost*. V dokumetech lze používat libovolné značky, lze je do sebe zanořovat. Do XML dokumentu tak lze velice přirozeným způsobem uložit téměř libovolnou informaci. XML formát si poradí jak s textově orientovanými daty (kniha, článek, webová stránka), tak i s databázovými údaji (ceník, tabulka zaměstnanců apod.).

Univerzálnost je podpořena i velice dobrou *mezinárodní podporou*. XML již od samého počátku počítá s tím, že existují i jiné jazyky než angličtina. Jako standardní znaková sada se používá 32bitové ISO 10646. V jednom dokumentu tak můžeme míchat dohromady všechny dnes na naší planetě běžně používané znaky. Nic nám zároveň nebrání v použití libovolného kódování, které nám vyhovuje. V Česku se jedná zejména o kódování ISO 8859-2 a windows-1250. V každém XML dokumentu se standardním způsobem zaznamenává informace o použitém kódování.

Velmi důležitou věcí, která by si možná zasloužila první místo v našem výčtu, je *otevřenost*. Formát XML není žádný proprietární formát nějaké komerční firmy. Specifikace XML [XMLSPEC] je poměrně jednoduchá a krátká, a kdokoliv si ji může zdarma stáhnout ze stránek konsorcia W3C [W3C].

Otevřenost a univerzálnost formátu je velice důležitou vlastností, zvláště pokud nám záleží na námi vytvořených datech. Při použití XML nejsme omezeni na používání proprietárních aplikací. Můžeme používat různé nástroje od různých firem na různých platformách, nemusíme se bát, že za pár let si v nové verzi editoru nepřečteme staré dokumenty. Je mnoho oblastí, kde jsou tyto vlastnosti klíčové – například dokumentace k různým zařízením – takové letadlo, raketové silo nebo soustruh mají životnost mnohonásobně delší než verze x.y nějakého dnes běžně používaného textového procesoru. Použitím XML dáme najevo, že důležitá jsou naše data, a ne aplikace, které pro práci s nimi používáme.

Díky bohatému označkování informací můžeme XML dokumenty velice snadno konvertovat do dalších formátů, můžeme opakovaně využívat již jednou existující informace. Tato činnost je v mnoha případech dokonce nezbytná. XML dokumenty obsahují informace, ale nijak nedefinují jejich vzhled. Pro člověka je však nutné údaje nějak přehledně zformátovat. Pro

tyto účely existují speciální *stylové jazyky*, které umožňují přímé zobrazení XML dokumentu, nebo jeho převod do dalších formátu jako je HTML, XHTML, PDF apod.

3. Základy syntaxe XML

XML patří mezi značkovací jazyky (markup languages). Důležité části dokumentu se označují pomocí značek. V terminologii XML se jednotlivým označeným částem dokumentu říká *elementy*. Elementy do sebe mohou být navzájem vnořené a tím dle potřeby zachycovat strukturu informací uložených v dokumentu.

Kdybychom například do XML ukládali elektronickou podobu knihy, skládal by se dokument z elementů kapitola. Každý element kapitola by pak obsahoval element nadpis a několik elementů pro odstavce. Příkladem z odlišné oblasti je uložení databázové tabulky do XML dokumentu. Dokument bude obsahovat několik elementů odpovídajících jednotlivým záznamům (řádkám) tabulky. Každý z těchto elementů pak samozřejmě bude obsahovat další elementy pro jednotlivé položky tabulky. Každý XML dokument se dělí postupně na menší a menší části.

3.1. Elementy

Elementy se v textu vyznačují pomocí tzv. *tagů*. Většině elementů odpovídají dva tagy – počáteční a ukončovací.

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

Naše ukázka obsahuje jeden element `para`. Jeho obsah je vyznačen pomocí tagů `<para>` (počáteční tag) a `</para>` (ukončovací tag). Jen na okraj poznamenejme, že výše uvedená ukázka je asi nejjednodušší XML dokument, který vůbec můžeme vytvořit.

Názvy tagů se zapisují mezi znaky ‘`<`’ a ‘`>`’. Ukončovací 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 ukončovací.

```
<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 uvedeme znak ‘`/`’. Ukončovací 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 ukončovací tag, nebo musí být počáteční tag zapsán jako element s prázdným obsahem. To je velký rozdíl oproti jazyku HTML, kde v mnoha případech můžeme ukončovací tagy vynechat. Při návrhu XML byla jedním z požadavků snadná implementace parserů, které budou XML dokumenty načítat. Tomu odpovídá i větší striktnost syntaxe XML oproti HTML.

3.2. Atributy

Elementy jsou základním stavebním kamenem každého dokumentu. U každého počátečního tagu můžeme použít ještě *atributy*. Atributy se používají k upřesnění významu elementu, k přidání dalších důležitých informací.

```
<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>
```

3.3. Znakové entity

Vzhledem k tomu, že se znak ‘<’ používají k zahájení tagu, není možné ho zapsat do dokumentu jen tak. Pro jeho zápis musíme použít tzv. *znakovou entitu*. Pro zápis znaku ‘<’ je určena entita `<`. Existuje i entita `>` pro zápis ‘>’, ale její používání není nutné.

```
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;`.

```
<monitor úhlopříčka="15&quot;"/>
```

Použití entit `&quot;` a `&apos;` se někdy můžeme vyhnout použitím apostrofů pro oddělení obsahu atributu:

```
<monitor úhlopříčka=’15"’/>
```

3.4. Kořenový element

Každý XML dokument musí být celý obsažen v jednom elementu. Následující ukázka tedy není správný XML dokument, protože se skládá z několika samostatných elementů.

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

Stačí však přidat *kořenový element*, který vše „obalí“, a dokument je rázem v pořádku.

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

3.5. Kódování znaků

Jako znaková sada se v XML dokumentech používá ISO 10646. Tato znaková sada je 32bitová, takže obsahuje dostatek pozic pro všechny znaky všech abeced používaných na Zemi. V současné době je definováno 49 194 znaků, jejichž kódy jsou shodné s Unicode 3.0.

Do dokumentu se znaky musí zapisovat pomocí určitého kódování, které definuje, jak se kód znaku bude reprezentovat nějakou sekvencí bajtů. Standard XML vyžaduje, aby všechny aplikace podporovaly alespoň kódování UTF-8 a UTF-16.

UTF-8 kóduje jeden znak do různého počtu bajtů. Znaky anglické abecedy jsou uloženy do jednoho bajtu a jejich kód odpovídá ASCII kódu. Ostatní znaky jsou kódovány do dvou až šesti bajtů. Konkrétně české znaky s diakritikou jsou kódovány do dvou bajtů. Pokud dokument v UTF-8 otevřeme v editoru, který toto kódování nepodporuje, uvidíme místo českých znaků dost podivné dvojice znaků.

Dalším použitelným kódováním je UTF-16. Je to 16bitové kódování, jeden znak je uložen ve dvou bajtech, které přímo obsahují kód znaku.

Použití UTF-8 a UTF-16 pro české (resp. slovenské) texty není moc pohodlné. Jednak je k dispozici málo editorů, které by umožňovaly bezproblémové použití těchto kódování. Druhý problém, i když už ne tak palčivý, je zbytečné zvětšení velikosti dokumentů. V Česku se dnes používají pro české texty dvě kódování – ISO 8859-2 a windows-1250. Můžeme je použít i v XML dokumentech, ale v tomto případě musíme vždy na začátku dokumentu použít *XML deklaraci* a v ní určit kódování.

```
<?xml version="1.0" encoding="iso-8859-2"?>
```

resp.

```
<?xml version="1.0" encoding="windows-1250"?>
```

Na tuto deklatraci nesmíme zapomínat, její vynechání a použití nestandardního kódování vede často k tomu, že máme problémy vytvořit i jednoduchý korektní XML dokument.

3.6. Zobrazení XML dokumentu a kontrola syntaxe

Splňuje-li dokument všechna výše uvedená pravidla, je syntakticky v pořádku a říkáme o něm, že je *správně strukturovaný (well-formed)*. Takový dokument pak můžeme zpracovat mnoha aplikacemi, které podporují formát XML.

Úplně základní aplikací pro zpracování XML je *parser*. Parser umí číst XML dokument a kontrolovat jeho syntaxi. Parser je obvykle integrální součástí nějaké další aplikace, např. prohlížeče, který pomocí něj čte XML dokument. Asi nejjednodušším způsobem, jak zkontrolovat správnou syntaxi XML dokumentu, je otevřít jej v prohlížeči s podporou XML. Pokud je dokument v pořádku, zobrazí se. Obsahuje-li dokument chyby, prohlížeč nás na ně upozorní. V současné době podporují XML například prohlížeče Mozilla a Internet Explorer 5.

Při zobrazování XML dokumentu prohlížeče neví, jak si přejeme jednotlivé elementy zobrazit. To lze určit pomocí stylu, který definuje způsob zobrazení. Bez něj nám Mozilla dokument zobrazí jako jeden dlouhý odstavec. Internet Explorer zobrazí zdrojový kód XML se zvýrazněnou syntaxí. K možnostem tvorby a použití stylů se v článku ještě vrátíme.

4. XML jako metajazyk pro definici dalších jazyků

XML umožňuje zcela libovolně volit názvy tagů. Na druhou stronu příliš volnosti škodí. Standard XML proto přímo v sobě obsahuje nástroj, který umožňuje definovat elementy přípustné v daném druhu dokumentů, jejich vzájemné vztahy a atributy. Tímto nástrojem je *DTD (Definice Typu Dokumentu)*. Vytvořením vlastního DTD vytvoříme nový jazyk, který základní charakteristiky a syntaxi přebírá z XML, ale má přesně definovanou množinu použitelných elementů.

Dnes existují stovky a možná i tisíce DTD, každé z nich definuje nový jazyk, nový formát, který je založený na XML. Mezi nejznámější jazyky tímto způsobem „odvozené“ od XML patří například:

- XHTML – nástupce jazyka HTML, plně přebírá jeho sémantiku, ale syntaxe je přizpůsobena XML.
- WML (Wireless Markup Language) – jazyk pro tvorbu jednoduchých webových stránek používaných v mobilních telefonech.

- MathML (Mathematical Markup Language) – jazyk pro zápis matematických výrazů.
- SVG (Scalable Vector Graphics) – jazyk pro 2D vektorovou grafiku, navržený speciálně pro potřeby Webu.
- DocBook – de facto standard pro tvorbu dokumentace (používá se například v LDP).

Libovolný dokument můžeme pomocí parseru kontrolovat oproti DTD. Dokument, který splňuje omezení definovaná v DTD, se nazývá *validní*. Parser, který je schopen provádět validaci, nám může ušetřit mnoho práce. Kdybychom měli v XML například uložené faktury, může parser ve spojení s příslušným DTD zcela automaticky zkontrolovat, zda faktura obsahuje údaje o odběrateli, dodavateli a jednotlivé položky. Když budeme mít v XML uložen text knihy, může nám parser zkontrolovat, jestli má každá kapitola název apod.

Kontrolování validity pomocí DTD je výhodné z několika důvodů. Když od někoho naše aplikací obdrží data v XML, může mnoho kontrol provést automaticky parser. Nemusíme ručně psát mnohdy poměrně zdlouhavý a nezáživný kód ošetřující chyby ve vstupních datech.

DTD může využívat i XML editor, který autorovi dokumentu průběžně nabízí vložení jen těch elementů, které jsou v daném kontextu platné.

Možnosti DTD jsou pro velký okruh aplikací zcela dostačující, ale rozhodně neřeší zdaleka všechny problémy spojené s kontrolou syntaxe a obsahu dokumentu. V současné době konsorcium W3C dokončuje standard *XML schémata* [XSDSPEC]. Princip jejich použití je stejný jako u DTD. Oproti DTD přinášejí mnohá vylepšení, která naleznou uplatnění zejména v aplikacích, které používají XML pro ukládání hodně strukturovaných dat databázového typu.

Nejvýraznějším rysem XML schémat je bohatá podpora datových typů. U každého atributu a elementu můžeme určit přesně jeho datový typ (číslo, řetězec, datum apod.) včetně různých integritních omezení. Jsou zde dokonce nástroje pro definici referenční integrity podobně, jak je známe z relačních databází.

Podstatně byly rozšířeny i vyjadřovací schopnosti jazyka. K tomu přispěla i nová syntaxe, která je na rozdíl od DTD založena na XML.

5. X** technologie

Se samotným jazykem XML úzce souvisejí další technologie a jazyky, které jeho možnosti dále rozšiřují. Na ty nejdůležitější z nich se teď stručně podíváme.

5.1. Stylové jazyky

XML dokumenty popisují strukturu dat, ale nijak nedefinují, jak se mají uložené informace prezentovat uživateli, jak se mají formátovat. XML umožňuje důsledně oddělit *obsah* dokumentu od jeho *vzhledu*. Používá se přitom velice jednoduchá myšlenka *stylových jazyků*. Definice vzhledu dokumentu, resp. jeho jednotlivých částí se definuje pomocí nějakého speciálního jazyka v samostatném souboru, kterému se říká *styl*.

Chceme-li XML dokument zobrazit zformátovaný, aplikujeme na něj styl, který provede zformátování dokumentu do výsledné podoby. Tento krok může přitom být v mnoha případech zcela automatický. Například webový prohlížeč si stáhne XML dokument a z metainformace na jeho začátku zjistí, jaký má použít styl pro jeho formátování. Stáhne si tedy i odpovídající styl a dokument zobrazí rovnou zformátovaný.

Největší výhoda oproti klasickému přístupu při zpracování dokumentů je v tom, že k jednomu druhu dokumentů můžeme mít několik různých stylů. Podle potřeby pak můžeme z jednoho zdroje dat generovat mnoho různých podob. To je dnes velmi potřebná vlastnost. Například při tvorbě dokumentace, chceme mít jeden dokument k dispozici v několika formátech – např. HTML, PDF, info apod. Každý formát má přitom specifické vlastnosti. Při použití běžných technologií bychom museli ručně (v lepším případě poloautomaticky) udržovat několik verzí stejného dokumentu. Místo toho však můžeme dokument uložit do XML a vytvořit styl pro každý požadovaný výstupní formát. Například dokumentace k Linuxu (LDP) a mnoho dalších projektů dnes používá DTD DocBook pro tvorbu dokumentace. Z jedné předlohy se pak generuje několik výsledných formátů dokumentace.

Analogický přístup se začíná používat i na Webu. K webovým stránkám se začíná pomalu přistupovat i z jiných koncových zařízení než jsou PC – z mobilních telefonů a různých PDA. Tato zařízení mají omezené možnosti, často používají pro stránky jiný formát než HTML. Poskytování stránek v různých formátech lze přitom vyřešit velice jednoduše – informace uložíme na webový server v XML společně s několika styly pro jednotlivá koncová zařízení. Před odeslání stránky web-server automaticky zkonvertuje informaci pomocí stylu do formátu, který nejlépe vyhovuje klientskému zařízení.

Styly jsou výhodné i v případech, kdy potřebujeme generovat velké množství dokumentů se stejným vzhledem. Vzhled je definován na jednom místě ve stylu. Pokud s ním zpracováváme dokumenty s podobnou strukturou (tj. dokumenty vyhovují stejnému DTD), mají všechny jednotný grafický design. Při požadavku na změnu všech dokumentů (dokumentace ke všem produktům, všechny webové stránky) pak stačí změnit jeden styl a přegenerovat výsledné

podoby dokumentů. Pro jeden dokument se tento přístup nevyplatí, ale pro větší množství dokumentů je již úspora práce zcela patrná.

Pro formátování XML dokumentů se dnes používají převážně dva stylové jazyky – kaskádové styly (CSS) a jazyk XSL.

Kaskádové styly umožňují pro jednotlivé elementy dokumentu definovat základní vizuální vlastnosti, jako výběr písma, jeho velikost, barvu, zarovnání apod. Hodí se jen pro velice jednoduché formátování. Různou úroveň podpory kaskádových stylů společně s XML dnes nabízí například prohlížeče Mozilla, Internet Explorer a Opera.

Speciálně pro potřeby XML byl vyvinut jazyk XSL (eXtensible Stylesheet Language). Ten obsahuje dvě části: transformační jazyk XSLT a jazyk pro abstraktní popis vzhledu dokumentu (tzv. formátovací objekty).

XSLT umožňuje velice jednoduše a efektivně provádět transformaci XML dokumentů do formátů XML, HTML a čistého textu. Při transformaci můžeme vybírat části vstupního dokumentu, třídit je, podmíněně zpracovávat apod. Síla XSLT je obrovská a těžko vyjádřitelná v jednom odstavci. Dnes se ponejvíce používá pro konvertování XML dokumentů do formátů jako je HTML a WML.

Formátovací objekty umožňují z elementárních bloků textu poskládat celý zformátovaný dokument. Vstupní XML dokument nejprve zpracujeme XSLT stylem, který však negeneruje ani HTML, ani WML, ale XML dokument obsahující formátovací objekty. Tento soubor pak zpracujeme procesorem, který provede konečné rozmístění objektů na stránku či obrazovku a zalomení řádků a stránek.

5.2. Tvorba odkazů

Jedním z impulzů pro vznik XML bylo vylepšení možnosti webových stránek. XML proto přináší i řádově lepší možnosti pro tvorbu hypertextových odkazů mezi dokumenty. Pro tvorbu odkazů se používají speciální jazyky XLink a XPointer.

5.3. API pro zpracování XML

Pokud chceme s XML pracovat v našich aplikacích, nemá cenu si psát vlastní parser, který umí načítat XML dokumety. Místo toho je lepší použít již hotové knihovny. Většina z nich přitom podporuje standardizovaná rozhraní pro zpracování XML. Dnes se převážně používají rozhraní DOM a SAX.

DOM (Document Object Model) modeluje XML dokument pomocí stromové hierarchie objektů v paměti. V aplikaci pak můžeme velice snadno přistupovat k libovolným částem dokumentu.

Rozhraní SAX (Simple API for XML) je řízené událostmi. Během čtení dokumentu volá parser námi definované funkce, které obsluhují důležité části dokumentu – počáteční a koncové tagy, obsah elementů apod.

5.4. Dotazovací jazyky

V současné době se velké úsilí věnuje vývoji kvalitních dotazovacích jazyků. Jediný zatím standardizovaný jazyk je XPath (XML Path Language), který umožňuje tvorbu poměrně jednoduchých dotazů, které vybírají části vstupního dokumentu. XPath se používá v několika dalších standardech (XSLT, XML schémata).

XPath je však orientován převážně na textově založené dokumenty. W3C konsorcium pracuje na jazyku XML QL, který se v sobě snaží efektivně a elegantně sloučit možnosti XPath, SQL a části XSLT.

6. Závěr

Před pár lety jeden z tvůrců XML prohlásil, že „XML je ASCII budoucnosti“. Já osobně mám pocit, že tato vize se již začíná naplňovat. Tento příspěvek rozhodně nemohl pokrýt celou problematiku XML a navazujících technologií. Doufám, že vám alespoň poslouží jako dobrý odrazový můstek při studiu podrobnějších zdrojů uvedených v seznamu literatury.

Literatura

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

[XMLSPEC] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen: *Extensible Markup Language (XML) 1.0 [http://www.w3.org/TR/REC-xml]*. W3C. 1998.

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

[XLINK] Steve DeRose, Eve Maler, and David Orchard: *XML Linking Language (XLink) [http://www.w3.org/TR/xlink]*. W3C. 2000.

[XPTR] Steve DeRose, Ron Daniel, Jr., and Eve Maler: *XML Pointer Language (XPointer) [http://www.w3.org/TR/xptr]*. W3C. 2001.

[XSDSPEC] David C. Fallside: *XML Schema Part 0: Primer [http://www.w3.org/TR/xmlschema-0/]*. W3C. 2000.

[DOM] Arnaud Le Hors, Philippe Le Hégaret, Lauren Wood, Gavin Nicol, Jonathan Robie, Mike Champion, and Steve Byrne: *Document Object Model (DOM) Level 2 Core Specification [http://www.w3.org/TR/DOM-Level-2-Core/]*. W3C. 2000.

[XMLCS] Jiří Kosek: *XML pro každého [http://www.kosek.cz/xml/]*. Grada Publishing. 2000.

[SAX] David Megginson: *SAX 2.0: The Simple API for XML [http://www.megginson.com/SAX/]*. 2000.

[W3C] *Stránky konsorica W3C [http://www.w3.org/]*.