
edice
PROVODCE
Česká společnost
pro systémovou
integraci
XML
technologie
Principy a aplikace v praxi
I. Mlýnková, M. Nečaský, J. Pokorný,
K. Richta, K. Toman, V. Toman
GRADA

Upozornění pro čtenáře a uživatele této knihy

Edice Management v informační společnosti

Prof. RNDr. Jaroslav Pokorný, CSc. a kolektiv

XML technologie

Principy a aplikace v praxi

Autoři:
RNDr. Irena Mlýnková, Ph.D. – kap. 2, 4, 7.2
Mgr. Martin Nečaský – kap. 9, 10
Prof. RNDr. Jaroslav Pokorný, CSc. – úvod, kap. 1.5.2, 1.5.3, 3, 7.4
Doc. Ing. Karel Richta, CSc. – kap. 1, 5
Mgr. Kamil Toman – kap. 6, 7.1, 7.3
Mgr. Vojtěch Toman – kap. 8

Vydala Grada Publishing, a.s.
U Průhonu 22, 170 00 Praha 7
tel.: +420 220 386 401, fax: +420 220 386 400
www.grada.cz
jako svou 3420. publikaci

Recenzenti Ing. Jiří Kosek a RNDr. Tomáš Pitner, Ph.D.
Odpovědný redaktor Mgr. Petr Mušálek
Sazba RNDr. Irena Mlýnková, Ph.D. a Mgr. Martin Nečaský
Počet stran 272
První vydání, Praha 2008
Vytiskly Tiskárny Havlíčkův Brod, a.s.
Husova 1881, Havlíčkův Brod

ISBN 978-80-247-2725-7 (tištěná verze)
ISBN 978-80-247-6688-1 (elektronická verze ve formátu PDF)

Obsah

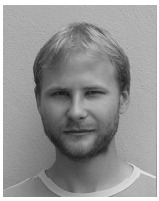
O autorech

RNDr. Irena Mlýnková, Ph.D. absolvovala magisterské i postgraduální studium na Matematicko-fyzikální fakultě UK, kde v současné době působí jako odborná asistentka na katedře softwarového inženýrství. Ve své vědecké činnosti se věnuje problematice správy XML dat v (objektově) relačních databázích, podobnosti XML dat a jejího využití, analýzy reálných XML dat, automatického generování XML dat a XML benchmarkingu. Výsledky své práce publikuje na mezinárodních konferencích (DocEng, DASFAA, DEXA, IDC, ISD, ...) i v odborných knihách a časopisech. Některé z nich získaly významná ocenění. V rámci svého pedagogického působení spoluvytvářela na MFF UK přednášku Technologie XML a Pokročilé technologie XML a na tuto oblast zaměřuje i vedené bakalářské a diplomové práce a studentské projekty.

Mgr. Martin Nečaský vystudoval Matematicko-fyzikální fakultu UK, obor Datové inženýrství. V současnosti působí jako postgraduální student na katedře softwarového inženýrství MFF UK. Tématy jeho výzkumu jsou návrh, integrace a správa XML dat a dále pak webové služby a sémantický web. Výsledky své práce publikuje na mezinárodních konferencích (SAC, EJC, ICDIM, APCCM, ...). Na uvedené oblasti zaměřuje i témata vedených bakalářských a diplomových prací. Na MFF UK vyučuje úvodní kurzy do databázových systémů a programování a předměty Technologie XML a Pokročilé technologie XML. Na FEL ČVUT vyučuje předmět Technologie XML.

Prof. RNDr. Jaroslav Pokorný, CSc. byl jmenován profesorem informatiky v roce 1999. Je autorem více než 250 původních publikací, vědeckých monografií a vysokoškolských učebnic z oblasti zpracování dat. Jako člen organizačního výboru se podílel na uskutečnění osmi mezinárodních konferencí. Je členem vědeckých rad FAV ZÚ v Plzni, FIS VŠE v Praze, rady ÚI AV ČR a MFF UK. Pracuje v komisi pro státní zkoušky na MFF UK a FEL ČVUT, je členem oborové rady pro doktorské studium na MFF UK a dalších školách. Působí jako expert v European Commision, je reprezentantem ČR v organizaci IFIP, stálým recenzentem Computing Reviews a Zentralblatt für Mathematik. Na MFF UK pracuje od roku 1982, v letech 1995–2006 zde zastával funkci vedoucího katedry softwarového inženýrství. Od roku 2008 je proděkanem MFF UK pro vědu a zahraniční styky.

Doc. Ing. Karel Richta, CSc. vystudoval obor Technická kybernetika na FEL ČVUT v Praze, habilitoval se v roce 1992 v oboru Výpočetní technika. V současnosti pracuje na katedře softwarového inženýrství MFF UK v Praze. Zabývá se formálními specifikacemi, softwarovým inženýrstvím, databázovými systémy a programováním. Učí a učil také některé předměty na jiných školách, např. na katedře počítačů FEL ČVUT, katedře informačních technologií VŠE, katedře informačních technologií PEF ČZU, katedře informatiky a matematiky VŠFS a katedře informačních technologií BIVŠ. Je autorem či spoluautorem několika knih, publikoval více než 100 příspěvků na různých konferencích. Je členem ACM a ČSSI.



Mgr. Kamil Toman vystudoval Matematicko-fyzikální fakultu UK, obor Počítačové systémy. V současné době působí jako postgraduální student na katedře softwarového inženýrství. Tématy jeho výzkumné práce jsou zejména problematika adaptivního ukládání XML a vyhledávání v XML datech, analýza reálných XML dat a využití těchto postupů v nativních XML databázích. Výsledky práce publikuje na mezinárodních konferencích. V rámci svého působení na fakultě spoluvytvářel přednášku Technologie XML. Mimo akademickou sféru působí jako softwarový architekt, přičemž se věnuje mimo jiné problematice datových integrací a webových služeb.

Mgr. Vojtěch Toman vystudoval Matematicko-fyzikální fakultu UK, obor Informatika. Po ukončení studií pracoval jako softwarový inženýr ve společnosti Gitus, a.s. Od roku 2006 působí v nizozemské společnosti X-Hive Corporation B.V., která se zabývá vývojem technologií pro nativní správu XML dat a řízení a publikování komplexního XML obsahu. Je aktivním členem pracovní skupiny XML Processing Model konsorcia W3C. Na MFF UK externě přednáší o problematice komprese XML dat.

Předmluva

Jazyk XML standardizovaný konsorciem W3C přispěl k vývoji souvisejících metod, které se dnes souhrnně nazývají XML technologií nebo XML technologiemi. Technologie, jak je známo, je soubor postupů, nástrojů a procesů sloužících nějaké činnosti. V případě XML může jít nejen o transfer dokumentů nebo jejich transformace apod., ale i o jejich ukládání v databázi a zpracování databázovým způsobem.

Značkovací jazyk XML slouží jako výměnný formát dat, tvoří syntaktickou základnu sémantického webu, je na něm založena servisně orientovaná integrace, lze ho dokonce považovat za nový databázový model. Existence XML vedla k rozvoji řady dalších jazyků a podpůrných nástrojů pro použití XML v softwarových systémech.

Konsorcium W3C koordinuje rozvoj XML technologie a snaží se dokonce o vytvoření jejího formálního základu. Četba dokumentů vytvořených W3C je ovšem, podobně jako i jiných standardů, obtížná, díky všem nutným technickým detailům, které takové standardy musí mít. Jedním z cílů knihy je přiblížit technologii informatikovi tak, aby se mohl v případě potřeby orientovat v konkrétních prostředcích souvisejících s XML, které současná praxe nabízí, či dokonce, aby byl schopen takové prostředky vyvíjet. Přiznejme, že zvláště u managementu XML dat, kam patří i současné XML databáze, to není nijak snadný úkol. Seznámit české čtenáře detailněji s databázovým zpracováním XML dat je proto dalším cílem knihy.

Přestože je kniha zacílena na profesionály využívající technologii XML v rámci aplikací, díky jejímu rozdělení do jednotlivých kapitol v ní může i začátečník nalézt potřebný úvodní materiál a vynechat případně detaily speciálních části XML technologie. Nezadatelnou motivací k vydání titulu byl též rychlý vývoj dané problematiky v posledních letech. Připomeňme také v této souvislosti, že od vydání původní české práce o XML [91] Jiřího Koska uplynulo již osm let.

Knihu můžeme neformálně rozdělit do dvou částí. První, tj. kapitoly 1–6, vychází ze standardů. Druhá část, kapitoly 7–10, je zaměřena na problematiku, která se neustále vyvíjí a která spíše mapuje stav v dané oblasti.

Díky rozsahu a orientaci knihy se domníváme, že u nás existuje relativně široká základna potenciálních čtenářů, od studentů informatiky přes vývojáře software až po manažery podnikové informatiky. V textu je použita řada příkladů, které naznačují, jak XML technologii využít a zároveň kudy se ubírá její další vývoj.

Na obsahu knihy se autoři podíleli následujícím způsobem: Jaroslav Pokorný – Úvod, kapitoly 1.5.2, 1.5.3, 3 a 7.4, Karel Richta – kapitoly 1 a 5, Irena Mlýnková – kapitoly 2, 4 a 7.2, Kamil Toman – kapitola 6, úvodní část kapitoly 7 a kapitola 7.3, Vojtěch Toman – kapitola 8, Martin Nečaský kapitoly 9 a 10.

Kniha představuje vyústění aktivit autorů v oblasti XML technologie v posledních sedmi letech. Látka byla nejprve zpracována na úrovni zvaného tutoriálu konference EurOpen v roce 2000, dále pak byla zpracována pro semestrový kurz v informatické sekci MFF UK, který je dnes součástí magisterské výuky. Publikačně byla část materiálu také zpracována ve formě skript. Od roku 2007 je kurz v modifikované formě realizován i na FEL ČVUT. Autoři se v dané oblasti věnují rovněž výzkumu. Jejich výsledky byly prezentovány jak na četných mezinárodních konferencích, tak ve světových časopisech.

Autoři děkují Mgr. Janu Ulrychovi za pečlivé přečtení rukopisu textu, kontrolu příkladů v odpovídajícím SW a za řadu připomínek, které přispěly ke zkvalitnění výsledku. Dále děkují RNDr. Davidu Bednárkovi a Mgr. Jiřímu Dokulilovi za přečtení vybraných částí textu a konzultace k nim. V neposlední řadě pak patří velký dík recenzentům, Ing. Jiřímu Koskovi a RNDr. Tomáši Pitnerovi, Ph.D., za cenné rady, připomínky a postřehy a také sponzorům za významnou pomoc při pokrytí nákladů na vydání knihy.

Kniha obsahuje poměrně velké množství příkladů. Všechny příklady je možné nalézt na webové stránce `http://kocour.ms.mff.cuni.cz/~necasky/bk/technologiexml`. Budou zde také umísťovány opravy případných chyb a další související informace. Připomínky ke knize jsou vítány na e-mailové adrese `technologiexml@ksi.mff.cuni.cz`.

V Praze dne 8. 8. 2008

autoři

Úvod

O XML se můžeme dočíst, že je to značkovací jazyk určený pro značkování textů. Co a k čemu však je to značkování? Stačí si uvědomit, že s nějakým značkováním se setkáváme běžně v životě. K zapsanému textu zapíšeme např. po stranách jednotlivých stránek různá znaménka, jejichž význam známe jen my sami, použijeme podtrhávání, různé barvy pro orámování částí textu apod. Značky tedy slouží k vyznačení jistých částí nebo obecněji jednotek v textu. Zřejmě je možné pomocí značek vtisknout původně i zcela nestrukturovanému textu nějakou strukturu. V XML se taková sada značek nejprve definuje, aby ji mohla využívat nějaká zájmová skupina, a pak se aplikuje na zvolené texty. Značky jsou vlastně jakési závorky vyjádřené opět jako text, který smysluplně vyjadřuje, co obsah textu mezi závorkami znamená. Např. text "`Technologie XML`" označkovaný jako `<název>Technologie XML</název>` můžeme vnímat tak, že text mezi závorkami `<název>` a `</název>` znamená název. O jaký název jde, může být patrné z kontextu výskytu textu. To, že jsou značky volitelné, naznačuje, že XML je vlastně metajazyk. K dispozici jsou pevně daná (jen syntaktická) pravidla, jak značky používat.

Je celkem zřejmé, jak značkovat ručně, lze si však představit i způsob, kdy označkovaný text je podle nějakého předpisu generován z počítačových aplikací. A k čemu je to dobré? Na základě dohodnutých značek pak jedna aplikace může „porozumět" údajům z jiné aplikace.

Prvotním účelem jazyka XML bylo skutečně podpořit výměnu dat. O jaká data vlastně jde? XML, jako zcela univerzální prostředek, samozřejmě může být použit pro jakákoliv data. Typicky může jít o data z relační databáze, která jsou transportována v business-to-business (B2B) úlohách. Dále lze uvést položky z telefonního seznamu, rozvrh odletů letadel nebo data zákaznických smluv. Jejich hlavním rysem je, že jsou pravidelně *strukturovaná* a v době návrhu jejich počítačového zpracování jsou známy jejich typy (v řeči databází – schéma databáze).

Na druhé straně spektra stojí data zcela *nestrukturovaná*, anebo strukturovaná jen velmi málo. Jde obvykle o texty, u kterých sice známe autora a datum vydání apod., ne však již vnitřní strukturu samotného textu. Ta může sice existovat (kapitoly, odstavce), ale nejsou prostředky, jak ji přesně popsat. Obvyklým nástrojem pro manipulaci s tělem takového dokumentu jsou fulltextové systémy. Praxe proto rozlišila XML data na *dokumentově orientovaná* (document-oriented nebo document-centric) a *datově orientovaná* (data-oriented nebo data-centric). Mezi datově orientovanými a dokumentově orientovanými XML daty je však někdy hranice nezřetelná, některé dokumenty mají rysy obou kategorií. Např. objednávka může obsahovat nestrukturovaná data jako např. poznámky nebo komentáře, zatímco článek v časopise obsahuje i strukturovaná data jako např. jméno autora, datum vydání apod. Na tuto kategorii XML dat se hodí termín *hybridní*.

Dále existují data, jako jsou záznamy o vyšetřeních pacienta, soubor kuchařských receptů, webová místa, např. stránka Amazon.com, ale i data ve formátu XHTML (XML následník formátu HTML). Tato data mohou být velmi nepravidelná. Např. záznamy pacientů mohou obsahovat vždy odlišné druhy vyšetření a odpovídající nasazení léků. To znamená, že typy všech položek nejsou známy v době návrhu počítačového uložení (depozitáře, databáze) těchto dat. Jinými slovy řečeno, je obtížné, ne-li nemožné, zkonstruovat jejich schéma. Roli hraje i uspořádání částí dat. Ve známé databázi XML dat obsahující Shakespearovy hry jsou data reprezentována

tak, že lze rozlišit např. všechny repliky Desdemondy, je ovšem vyžadováno, aby pořadí replik v dialozích zůstalo uspořádáno stejně jako v původním textu. Hovoří se pak o *semistrukturovaných datech*, tj. datech, která nemají schéma, mají nepravidelnou nebo implicitní strukturu, jsou hnízděná a heterogenní. O takových datech se předpokládá, že budou spíše zpracovávána strojově než dotazována člověkem. XML text je dobrým příkladem semistrukturovaných dat.

Označkovaná data se tedy vyměňují mezi aplikacemi a dokonce existují požadavky, aby byla vhodným způsobem transformovatelná. Možná do jiných XML dat a nebo k prezentaci v uživatelských výstupech nebo na webových stránkách. V obou případech je občas nutné XML data ukládat do nějakého repozitáře nebo dokonce databáze. To umožňuje jejich sofistikovanější management, a to nejen pro dotazování, ale i pro aktualizaci. Jako databáze může být použit nějaký dostupný systém řízení bází dat (SŘBD) (pro datově orientovaná XML data), ale i speciální databáze, kterým se říká *nativní*. Tyto databáze jsou založeny na implementaci „šité na míru“ XML datům a jejich zpracování. Nepříliš zajímavá je možnost chápat XML data pouze jako text. Pak lze použít pouze prostředky pro práci s textovými daty umožňující indexaci, vyhledávání podle klíčových slov apod.

Podívejme se nyní na XML z hlediska nového databázového modelu. Při bližším zkoumání značkování v XML uvidíme, že správně se značky do sebe zahnizďují hierarchickým způsobem. Tedy XML lze dobře použít pro reprezentaci dat, která jsou přirozeně hierarchická a vyžadují navíc jisté uspořádání. To naplňuje ideu rozšířit stávající možnosti databázového zpracování i tam, kde byly relační databáze nepoužitelné nebo byly použitelné jen velmi obtížně. Připomeňme v této souvislosti obtížně vyjádřitelné dotazy nad stromovými strukturami v SQL, např. součástek a jejich dílů.

Jakmile XML proniklo do světa databází, bylo nutné vytvořit jazyky pro vyjádření schématu XML databáze (i když XML databáze může existovat bez takového schématu). Objevily se dotazovací jazyky a další jazyky pro práci s XML daty. Lze již tedy hovořit o *technologii XML*.

Kniha by měla čtenářům přiblížit základy technologie XML. Jejím cílem není podat podrobný popis formátu XML. V textu jsou zdůrazněny zejména základní myšlenky související s XML, je nastíněn datový model XML a možnosti definice struktury XML dokumentů – jazyky DTD a XML Schema. Kniha se dále zabývá nástroji, jako jsou jazyk XPath, dotazovací jazyk XQuery a transformační nástroj XSLT. Zvláštní důraz je kladen na základy databází orientovaných na XML, smyslem je ukázat, jak ukládat XML data do databáze, jak je zpracovávat databázovým způsobem a jak se k novému trendu staví současné standardy. Ze standardů jmenujme zejména důležitá doporučení konsorcia W3C[1], zvláště pak jeho pracovní skupiny XML Query WG, ale také aktivity týmů rozvíjejících pod ISO a ANSI relační jazyk SQL. Další významnou průmyslovou iniciativou je XML:DB[2], která se zabývá hlavně návrhem API k XML databázím.

Principy jazyka XML jsou uvedeny v kapitole 1. Zahrnují syntaxi jazyka a jeden z možných modelů XML dat – Infoset, dále pak aplikační rozhraní SAX, StAX a DOM umožňující snazší práci s XML daty. V kapitole je také popsán prostředek pro definici struktury XML dokumentů – jazyk DTD. Kapitola 2 je věnována podstatně složitějšímu prostředku, než je DTD, jazyku XML Schema (v aktuální verzi 1.0), který slouží podobnému účelu, ovšem na mnohem vyšší úrovni a s podstatně většími možnostmi. V kapitole 3 vysvětlujeme základy tří důležitých jazyků XML technologie. První z nich XPath (ve verzi 1.0) je příkladem jednoduchého dotazovacího jazyka nad XML daty, XPointer umožňuje odkazování mezi XML dokumenty a XLink zobecňuje vazby mezi webovými místy do prostředí, kde jsou místo HTML stránek XML dokumenty. Výklad pokročilých rysů jazyka XML Schema je obsažen kapitole 4. Jazyk transformací XSLT, další z rodiny XML jazyků, je popsán v kapitole 5. Vyvrcholení výkladu o XML představuje zřejmě kapitola 6, věnovaná jazyku XQuery. Tento jazyk, jehož součástí je mimo jiné jazyk XPath 2.0, představuje zatím největší „oříšek“ v technologii XML, největší výzkumné úsilí dosažené v jejím vývoji, zvláště pak z hlediska efektivní implementace jazyka. Následující rozsáhlá kapitola 7

[1] http://www.w3.org/
[2] http://www.xmldb.org/

se věnuje XML databázím. Rozebírají se v ní tzv. nativní XML databáze a ukládání XML dat v relační databázi. Aktuální je také integrace XML dat a relačních dat podle standardu SQL/XML.

Další kapitoly se zabývají tématy, která v dostupné literatuře zatím nebývají příliš prezentována. Kapitola 8 pojednává o kompresi XML dat. Je zřejmé, že jde o důležitou oblast technologie XML, která přispívá k efektivnosti provozu XML dat, jak při přenosech, tak při dotazování. Konceptuální modelování XML dat je předmětem kapitoly 9. Toto téma je dnes zvlášť aktuální v souvislosti s aplikacemi orientovanými na webové služby. Poslední kapitola 10 je zaměřena na praktické využití XML a jeho podporu v existujících relačních SŘBD. Následuje rejstřík umožňující rychlejší orientaci v textu a seznam obsahující užitečné odkazy na další relevantní literaturu.

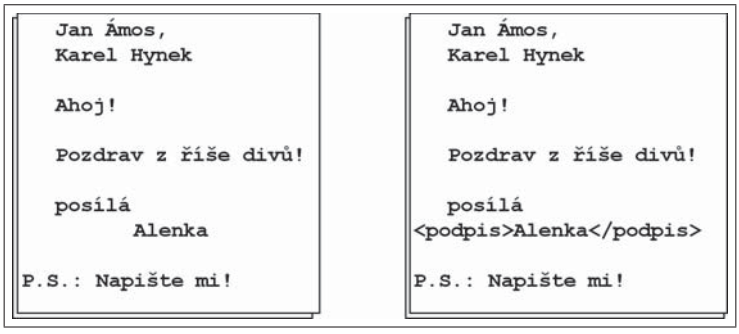
Kapitola 1

Principy formátu XML

Formát XML definovalo konsorcium W3C jako formát pro přenos obecných dokumentů a dat. *XML* je zkratka pro eXtensible Markup Language, tj. rozšiřitelný značkovací jazyk [47]. Návrh XML vychází ze staršího a obecnějšího standardu *SGML* (Standard Generalized Markup Language – ISO 8879:1986 [2]). Poznamenejme, že ze standardu SGML vycházel i formát dokumentů *HTML* (Hyper-Text Markup Language [116]). Sada značek formátu HTML je pevná a slouží k vyjádření prezentační podoby dokumentu. Naproti tomu v XML sada značek pevná není, ale může být definována pro různé sady dokumentů různě. Definice sady značek může být součástí definice XML dokumentu, může být specifikována odkazem, nebo může být dohodnuta předem. Značky mají tvar obecných závorek, např.:

```
<podpis>Josef</podpis>
```

Značky slouží k označení určitých prvků (částí) dokumentu. Značky mají *otevírací závorku* (start-tag), např. `<podpis>`, a *zavírací závorku* (end-tag), např. `</podpis>`. Pokud je text mezi závorkami prázdný, lze dvojici otevírací a zavírací závorky nahradit *prázdným elementem* (empty-element), v našem případě např. `<podpis/>`.



Obr. 1.1 Příklad neoznačkované a označkované zprávy

Uvažme např. hypotetickou zprávu na obrázku 1.1. Vyznačíme-li v této zprávě závorkami podpis odesílatele, bude takový dokument lépe uchopitelný a bude jej možno zpracovat i programem. V XML můžeme navíc stanovit, že dokument typu „zpráva“ musí obsahovat adresu, oslovení, text, podpis a dodatek. Tento požadavek zapíšeme např. pomocí nástroje *DTD* (Document Type Definition) následovně:

```
<!ELEMENT zpráva (adresa, oslovení, text, podpis, dodatek)>
```

Vlastní obsah každé zprávy pak musí mít odpovídající tvar, např.:

```
<zpráva>
  <adresa>Jan Ámos</adresa>
  <oslovení>Ahoj!</oslovení>
  <text>Pozdrav z říše divů!</text>
  <podpis>Alenka</podpis>
  <dodatek>Napište mi!</dodatek>
</zpráva>
```

Pomocí značek XML vyznačíme syntaktickou strukturu dokumentu. Sémantika značek ani význam jejich obsahu nejsou pomocí XML definovány. Např. pro výše zmíněnou zprávu nevíme, co obsah zprávy znamená, nevíme, jak zprávu zpracovat – to je věcí aplikací, které s dokumentem manipulují. Význam XML spočívá v tom, že struktura dokumentu je známa, lze ji kontrolovat a zpracovat obecnými nástroji. Libovolná aplikace si může strukturu dokumentu zjistit a dle této struktury jej zpracovat – např. zjistit, kdo zprávu podepsal.

Potřeba nezávislého formátu pro reprezentaci a přenos obecných dokumentů je nesporná. Tato kapitola představuje úvod, ve kterém bude popsán podrobněji formát XML. V následujících kapitolách se budeme zabývat nástroji, které tvoří *rodinu XML formátů* a představují základ *technologie XML*.

1.1 Formát XML

XML je formát pro reprezentaci a přenos obecných dokumentů. Při návrhu XML se autoři [47] z konsorcia W3C řídili následujícími principy:

- formát XML musí být použitelný v rámci internetu,
- formát XML by měl podporovat širokou škálu aplikací,
- formát XML musí být kompatibilní s formátem SGML,
- musí být snadné vytvářet programy, které manipulují s dokumenty v XML,
- množství variant XML by mělo být minimální – nejlépe žádné a
- XML dokumenty by měly být čitelné a pochopitelné i pro člověka.

Na základě těchto principů navrhli definici XML, která zahrnuje dvě části:

- definici, co to je XML dokument,
- definici programů, které zpracovávají XML dokumenty – XML procesorů.

1.2 XML dokument

XML dokument je určitým způsobem uspořádaná posloupnost znaků jisté abecedy. Implicitně se předpokládá *Unicode* – kód ISO/IEC 10646 [3]. Na rozdíl od formátu HTML rozlišuje XML důsledně malá a velká písmena. Následující konstrukce proto není dobře vytvořený XML dokument, neboť otevírací a zavírací závorka se liší:

```
<Podpis> ... </podpis>
```

Připomeňme, že procesory HTML připouštějí obvykle volnější syntaxi. Koncové závorky lze často vynechat, ve značkách se nerozlišují malá a velká písmena, prohlížeč obvykle zobrazí i ne zcela korektní dokument, např.:

```
<HTML>
  <body>Pozdrav z říše divů!</BODY>
</html>
```

Značky XML mohou využívat i národní abecedu, obecně však tento způsob značení nelze doporučit, neboť určité problémy může způsobit např. různé kódování češtiny apod. Následující ukázka ilustruje použití lokální abecedy ve značkách: podle definice je to správně vytvořený dokument, nemusí však být vždy dobře zpracován.

```
<?xml version="1.0" encoding="UTF-8"?>
<zpráva>
  <podpis>Alenka</podpis>
</zpráva>
```

Fyzicky se XML dokument skládá z posloupnosti prvků nazývaných *entity*. Fyzická entita ještě není logický element dokumentu. Z hlediska procesoru XML může každá fyzická entita obsahovat buď rozpoznatelná data, nebo nerozpoznatelná data. Nerozpoznatelná data mohou být textová nebo binární, která se buď procesoru nepodaří interpretovat jako znaky, nebo se jedná o data pro jinou aplikaci. Rozpoznatelná data jsou sestavena ze znaků zvolené abecedy a představují buď *znaková data* nebo *značky* (markups). Značky vyznačují logickou strukturu dokumentu a tím i jeho *rozložení* (layout). Dokument začíná entitou nazývanou *kořen* (root entity).

Logicky se XML dokument skládá z prologu, deklarací, elementů, komentářů a instrukcí pro zpracování jinými aplikacemi. Logické elementy jsou v dokumentu vyznačeny značkami. Sada značek je obecně libovolná, může být ale předepsána deklaracemi. Deklarace nejsou povinné, pokud ale chceme strukturu dokumentu kontrolovat, musíme ji deklaracemi předepsat. XML dokument je *dobře vytvořen*, pokud všechny rozpoznatelné entity v dokumentu jsou správně vytvořeny a navíc, všechny rozpoznatelné entity, na které existují v dokumentu odkazy, jsou rovněž dobře vytvořeny. Každá dvojice závorek musí být korektně spárována v rámci elementu a tyto dvojice musí být dobře vnořeny do sebe – závorky se nesmí křížit. Z toho plyne, že dobře vytvořený XML dokument má stromovou strukturu. Uvažme jako příklad jednoduchou faxovou zprávu:

```
<?xml version="1.0" encoding="UTF-8"?>
<fax>
  <odesílatel>
    <jméno>Jan Ámos</jméno>
    <číslo>333 333 333</číslo>
  </odesílatel>
  <adresát>
    <jméno>Karel Hynek</jméno>
    <číslo>666 666 666</číslo>
  </adresát>
```

```
  <text> ... </text>
</fax>
```

XML poskytuje možnost definice použitelné sady značek, definici logické struktury a rozložení dokumentu – *XML schéma* dokumentu. Jednoduchým nástrojem na definici je definice typu dokumentu DTD (Document Type Definition). Novějším a mocnějším prostředkem je jazyk XML Schema [122, 38], který bude probrán v kapitolách 2 a 4, kde budou zmíněny i jiné jazyky pro definici XML schématu. Každý dobře vytvořený XML dokument pak může být navíc *validní*, pokud splňuje omezení správnosti (validity constraints) daná specifikací XML schématu. Validita dokumentu proto vyžaduje ověření proti definované specifikaci obsahu.

Rozpoznatelné entity XML dokumentu jsou sestaveny ze znaků. Implicitní znakovou sadou je Unicode, každý XML procesor proto musí být schopen akceptovat na vstupu Unicode zakódovaný pomocí kódování UTF-8, UTF-16, příp. UTF-16LE, či UTF-16BE. Rozlišení mezi těmito dvěma kódy by mělo být možné tak, že dokument v UTF-16 musí začínat `#xFEFF` (znak kódovaný hexadecimální hodnotou Q budeme značit `#xQ`). Speciálním způsobem jsou zpracovány znaky „mezera“ (`#x20`), „tabulátor“ (`#x9`), „nový řádek“ (`#xA` – LINE FEED) a „přechod na začátek řádku“ (`#xD` – CARRIAGE RETURN), tj. *bílé znaky* (white spaces).

V naší lokalitě se pro komunikaci se světem často používá UTF-8, který je kompatibilní s kódem ASCII v tom smyslu, že obsahuje všechny znaky ASCII a další znaky jsou kódovány pomocí 2–4 bytů. Unicode obsahuje všechny znaky všech abeced. Pro češtinu lze také použít kódování ISO-8859-2 nebo Windows-1250, není však jisté, zda toto kódování bude umět každý XML procesor zpracovat.

XML procesor je modul, který umí číst XML dokumenty a zpřístupňuje jejich prvky aplikacím. XML procesor kontroluje, zda je dokument *dobře vytvořen* (well-formed). Porušení těchto pravidel představuje fatální chybu, kterou musí XML procesor detekovat a hlásit aplikaci. Po detekci fatální chyby může XML procesor poskytovat další data aplikaci, ale nesmí pokračovat v normálním zpracování. XML procesor může být *validující* – pak umí kontrolovat, zda je vstupní XML dokument *validní* (valid) vzhledem k zadané specifikaci struktury. Validující XML procesor musí navíc hlásit neshodu s deklarovanou strukturou XML dokumentu.

1.2.1 Elementy a atributy

XML dokument obsahuje jeden nebo více elementů, vymezených závorkami. Každý *element* je identifikován *jménem*, které je uvedeno v závorkách omezujících element, např. element `osoba`. Pokud element není prázdný, tvoří jeho *obsah* entity uzavřené mezi otevírací a uzavírací závorkou.

```
<osoba>
  Text tvořící obsah elementu typu osoba.
</osoba>
```

Obsahem elementu může být libovolná posloupnost znakových dat, jiných vnořených elementů, referencí (odkazů) na jiné entity, sekcí CDATA, instrukcí pro zpracování jinou aplikací nebo komentářů. Jednotlivé složky jsou popsány dále.

Elementy mohou mít *atributy*, kterými jsou blíže charakterizovány. Atributy se vkládají do otevírací závorky elementu ve formě parametrů zadaných pomocí klíčových slov. Každá specifikace atributu má jméno a hodnotu – zapisuje se jako dvojice `jméno="hodnota"`. Hodnota atributu musí být vždy uvedena v uvozovkách nebo apostrofech. Formálně vzato se vždy jedná o řetězec znaků, byť je možné ji interpretovat např. jako celé číslo. Již zmíněnou faxovou zprávu můžeme pomocí atributů vyjádřit např. následovně:

```
<fax>
  <odesílatel jméno="Jan Ámos"    číslo="333 333 333"> ... </odesílatel>
  <adresát    jméno="Karel Hynek" číslo="666 666 666"> ... </adresát>
  <text> ... </text>
</fax>
```

1.2.2 Prázdný element

Prázdné elementy nemají žádný obsah. Slouží pro vyznačení místa v dokumentu, např. vyznačení místa, kam se později doplní nějaký prvek. Prázdný element může mít atributy, které popisují jeho vlastnosti – např. to může být odkaz na objekt, který se má do daného místa vložit. Různé formy zápisu prázdného elementu ukazují následující příklady, kde první element označuje vložený obrázek, zatímco druhý a třetí jsou alternativní zápisy prázdného elementu vyznačujícího místo, kam se má vložit oddělovací čára (break).

```
<IMG align="left" src="http://www.w3.org/Icons/w3c_home"/>
<br></br>
<br/>
```

1.2.3 Komentáře

Komentáře se v XML dokumentu mohou vyskytovat kdekoliv, ale nemohou být umístěny ve značkách. Mají své speciální závorky "`<!--`" a "`-->`", aby se daly odlišit od vlastního obsahu dokumentu. Komentáře se nemohou vnořovat. V komentáři se nesmí vyskytovat řetězec "`--`" (double-hyphen), neboť ten se používá v komentářových závorkách podle syntaxe:

```
<!-- Já jsem příklad komentáře -->
```

XML procesor může umožnit aplikaci číst text komentářů (obsah komentářů). Značky uvedené v těle komentáře se ovšem interpretují jako komentář, nikoliv jako označení elementu. Ve následujícím komentáři se proto text `<příklad>` neinterpretuje jako otevírací závorka elementu `příklad`, ale jako součást textu komentáře.

```
<!-- Zde bude uvedena definice elementu <příklad> -->
```

1.2.4 Instrukce pro zpracování

Definice XML dokumentu umožňuje, aby dokument obsahoval instrukce, které mají být zpracovány jistou aplikací. Myslí se tím instrukce pro zpracování jinou aplikací, než je XML procesor. Takové části se říká *sekce instrukcí* (Processing Instructions – PI) a používají se pro ni speciální závorky se znakem '`?`'. Instrukce obsahuje návěští (PITarget), kterým je identifikována aplikace, pro niž je určena. Z pochopitelných důvodů to nesmí být jakákoliv varianta řetězce "`XML`" (žádná kombinace malých a velkých písmen). Za návěštím může být bílý znak a poté následuje text, který představuje data pro danou aplikaci. Následující instrukce je např. určena pro aplikaci `xql`:

```
<?xql data pro jinou aplikaci?>
```

Obsah sekce instrukcí není chápán jako součást XML dokumentu, je pouze předán ke zpracování dané aplikaci. Např. připojení stylu k dokumentu dosáhneme instrukcí:

```
<?xml-stylesheet href="styl.css" type="text/css"?>
```

1.2.5 Sekce CDATA

Někdy potřebujeme do dokumentu vložit blok textu, který by mohl být považován za značku. Pokud tomu chceme zamezit, použijeme speciální *sekci typu znaková data* – CDATA (Character DATA), která slouží k označení takových míst. Uvedeme-li v dokumentu text:

```
<pozdrav>Ahoj!</pozdrav>
```

bude chápán jako element pozdrav s obsahem "`Ahoj!`". Chceme-li, aby pozdrav v závorkách nebyl chápán jako značka, ale aby byl jako text součástí obsahu elementu, musíme použít konstrukci:

```
&lt;pozdrav&gt;Ahoj!&lt;/pozdrav&gt;
```

Pomocí sekce CDATA to zapíšeme snadněji jako:

```
<![CDATA[ <pozdrav>Ahoj!</pozdrav> ]]>
```

Obecně má sekce CDATA tvar:

```
<![CDATA[ text ]]>
```

Řetězec "`]]>`" se v těle sekce CDATA pochopitelně nesmí vyskytovat.

1.3 Definice struktury dokumentu pomocí DTD

Každý dobře vytvořený XML dokument musí začínat *prologem*, za kterým následuje *kořenový element* dokumentu. Prologem se stanoví verze XML, kódování dokumentu a případně požadavky na strukturu dokumentu. V současnosti jsou přípustné verze 1.0 a 1.1, které se liší zejména tím, že verze 1.1 respektuje změny v kódování Unicode a umožňuje volnější interpretaci jmen.

Každý XML dokument obsahuje jeden či více elementů. Vždy ovšem obsahuje právě jeden kořenový element, jehož žádná část není obsažena v žádném jiném elementu. Součástí dokumentu může být i dříve zmíněná sada komentářů a instrukcí pro jiné aplikace.

Prolog XML dokumentu začíná deklarací verze XML. Deklarace verze dokumentu má tvar, který připomíná instrukci pro aplikaci xml: