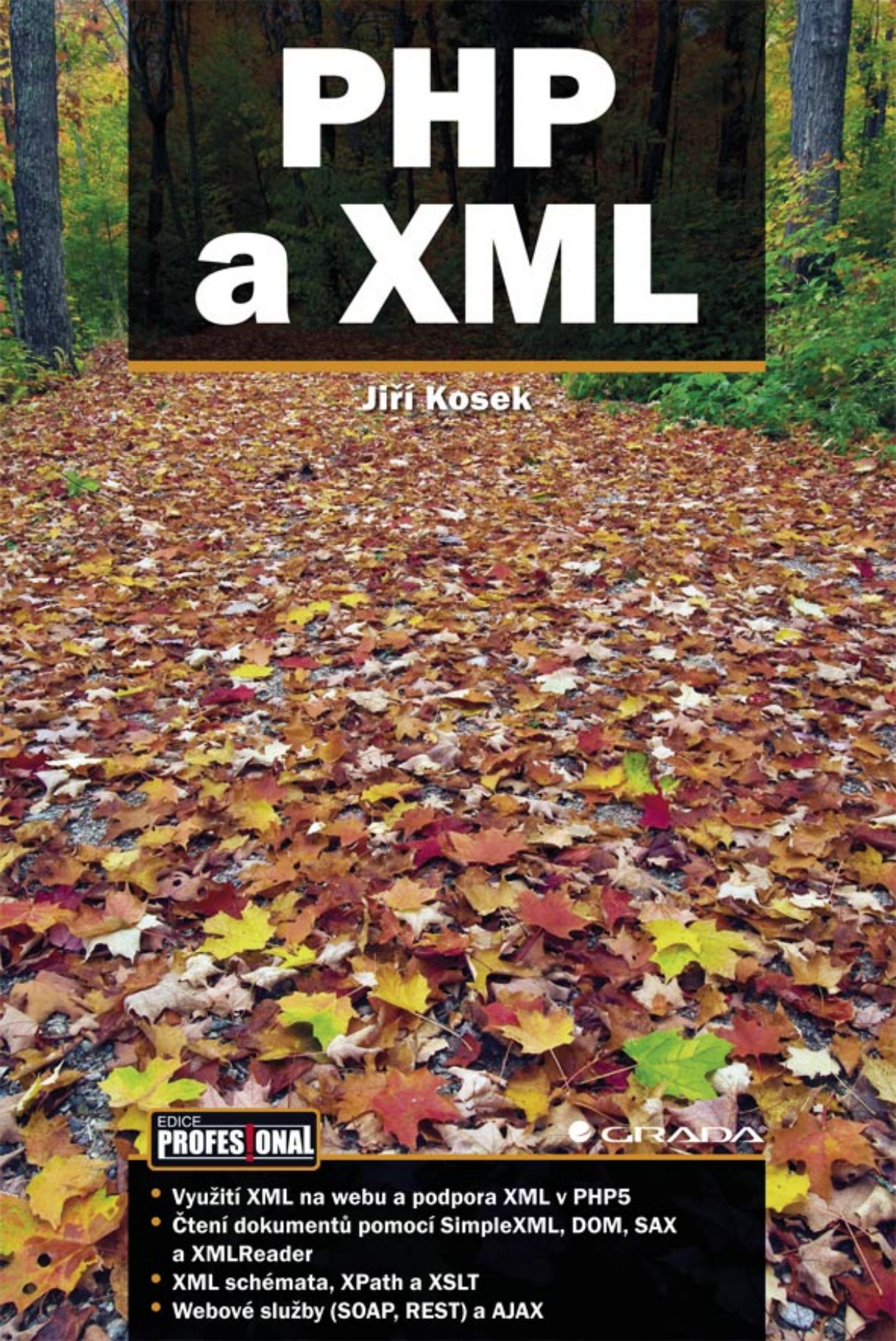
PHP a XML

Jiří Kosek



EDICE PROFESIONAL

GRADA

- Využití XML na webu a podpora XML v PHP5
- Čtení dokumentů pomocí SimpleXML, DOM, SAX a XMLReader
- XML schémata, XPath a XSLT
- Webové služby (SOAP, REST) a AJAX

Upozornění pro čtenáře a uživatele této knihy

Stručný obsah

Obsah

Předmluva

Když jsem před více než deseti lety psal předmluvu k dnes již legendární knize *PHP – tvorba interaktivních internetových aplikací* [11], slíbil jsem v ní, že na doprovodný web knihy umístím informace o práci s formátem XML, které se do knihy již nevešly. Z časových důvodů k naplnění tohoto slibu nikdy nedošlo. Jako satisfakci proto přijměte tuto knihu. Její náplní je pouze XML a jeho použití v PHP.

Deset let je dlouhá doba, ale myslím, že čekání se vyplatilo. Podpora XML byla v PHP až do jeho verze 5.0 poměrně partyzánská. A teprve od verze 5.2 lze PHP považovat za jazyk, ve kterém se dá s XML rozumně pracovat.

O významu XML dnes již není potřeba nikoho přesvědčovat. Ať se nám to líbí nebo ne, XML je zkrátka všude a ve webových aplikacích je potřeba s tímto formátem pracovat. Ať už se jedná o importy a exporty dat, transformaci dat pro prezentační vrstvu nebo backend pro AJAXovou aplikaci. Cílem této knihy je naučit vás používat všechna existující rozhraní pro práci s technologiemi XML, která PHP nabízí. Popsán a vysvětlen je však i samotný jazyk XML, jeho syntaxe a navazující technologie jako XML schémata, dotazovací jazyk XPath a transformační jazyk XSLT. Pro pochopení výkladu tak není nutná žádná velká předchozí zkušenost s XML – vše potřebné je průběžně vysvětleno.

V dnešní době začínají být papír a knihy považovány za příliš konzervativní médium. Asi jsem staromilec a mám tištěné knihy rád. Nicméně na adrese *http://www.kosek.cz/knihy/phpxml/* najdete další informace související s knihou – příklady ke stažení, opravy chyb apod. Máte-li ke knize nějaké připomínky, uvítám je na mojí emailové adrese `<jirka@kosek.cz>`.

Na vzniku knihy má zásluhu mnoho lidí. Nevyčerpatelnou trpělivost prokázali šéfredaktoři počítačové redakce Miroslav Lochman a Daniel Vrba, kteří vydrželi pět let čekat na dokončení knihy. Pokud v knize nebude příliš chyb, je to zásluha korektorky Zuzany Vrbové a redaktorky Evy Steinbachové. Největší dík však patří mé ženě Lence a dětem, kteří trpělivě snášely mé útěky k počítači při psaní knihy.

Přeji vám příjemné čtení knihy.

Jirka Kosek

Liberec, 10. dubna 2009

Úvod

Vývoj moderních webových aplikací klade na vývojáře vysoké nároky, je potřeba znát široké spektrum technologií. Počínaje jazyky HTML a CSS pro definici samotné stránky a jejího vzhledu, přes Javascript pro vytváření vysoce interaktivních aplikací, po nějaký serverový jazyk jako je PHP. Každá větší aplikace navíc potřebuje někam ukládat data, typicky do databáze. K tomu je potřeba znát principy protokolu HTTP a vědět, jak obcházet jeho limity. A aby toho nebylo málo, na mnoha místech se vývojář webové aplikace setká i s formátem XML.

Když XML v polovině 90. let minulého století vznikalo, původní smělé plány byly, že zcela nahradí jazyk HTML při doručování obsahu do prohlížeče. Tato myšlenka se však ukázala jako příliš revoluční a navíc problémy spojené s jazykem XHTML a jeho podporou v prohlížečích v očích mnoha webových vývojářů nevrhly na XML příliš růžové světlo. Nicméně technologie XML jsou dnes pevnou součástí mnoha webových technologií, formátů a protokolů, takže je potřeba práci s tímto formátem ovládat.

Kde se na webu s XML může vývojář setkat? Syntaxi XML využívají mnohé prezentační formáty – počínaje jazykem XHTML, přes stále populárnější vektorový grafický formát SVG, až po jazyky pro definici uživatelského rozhraní v moderních RIA (Rich Internet Application) prostředích, jako je XAML v Silverlightu a MXML ve Flashi. Pokud tedy vaše skripty v PHP v minulosti generovaly převážně HTML kód, časem budou přicházet požadavky na dynamickou tvorbu modernějších, na XML založených, formátů.

XML dnes zcela dominuje na poli publikování metainformací. Jedná se například o formáty pro publikování přehledů nových článků, jako jsou RSS či Atom. Protože začleňování sémantiky ve strojově čitelné podobě přímo do webových stránek je stále v plenkách, mnoho vyhledávačů nabízí vlastní formáty, ve kterých jim můžete předávat informace vylepšující vyhledávání – například Google Sitemap nebo Google Base.

Další využití XML je pro komunikaci a předávání dat. XML se využívá jednak pro výměnu dat mezi backendy jednotlivých aplikací a dále pak v AJAXových aplikacích pro zasílání aktualizací dat do prohlížeče. „Enterprise“ aplikace pak pro samotnou komunikaci nevyužívají prosté XML, ale komplexnější mechanismus webových služeb.

Diverzita koncových zařízení používaných pro přístup k webu se také stále zvyšuje. Webové stránky se už neprohlížejí jen z klasického počítače, ale i z chytrých telefonů nebo různých PDA. Mnohé aplikace potřebují uspokojivě řešit možnost kvalitního tisku. Už nestačí vytvořit aplikaci, která výstupy generuje jen jako HTML. Pro jednotlivá koncová zařízení je potřeba generovat odlišné formáty výstupu a webovou aplikaci je potřeba obohatit o flexibilní prezentační vrstvu. Pro vytvoření takové vrstvy lze využít i jazyk XML a stylové technologie jako XSL.

Výše uvedený výčet toho, kde se na webu můžeme setkat s XML, jistě není úplný. Pouze potvrzuje to, že webový vývojář se dnes neobejde bez znalosti tohoto formátu a práce s ním. Tato kniha vás naučí vše potřebné o formátu XML a navazujících technologiích a o tom, jak lze s tímto formátem pracovat v PHP.

První kapitola je určená zejména pro čtenáře, kteří ještě neznají formát XML. Seznámí se zde se syntaxí jazyka a naučí se ji kontrolovat.

Druhá kapitola pak stručně shrnuje a ukazuje, jaké možnosti pro práci s XML nabízí PHP. Je to ideální místo pro porovnání jednotlivých přístupů pro práci s XML. Nemusíte tak číst celou knihu, ale v této kapitole zjistíte, jaký způsob práce s XML je pro vás nejvhodnější a ten si dále podrobněji nastudujete v odpovídající samostatné kapitole.

Třetí kapitola přímo nesouvisí s XML, ale ukazuje, jak lze v PHP částečně obejít chybějící podporu znakové sady Unicode, kterou využívá i jazyk XML.

Následují čtyři kapitoly, které podrobně popisují jednotlivá rozhraní pro práci s XML – SimpleXML, SAX, DOM a XMLReader. Výběr vhodného rozhraní záleží na povaze dat, která čtete, a na tom, jak je potřebujete zpracovat.

Osmá kapitola seznamuje s dotazovacím jazykem XPath, který nabízí jednoduchou a zároveň mocnou metodu pro vyhledávání a výběr dat v dokumentech XML. Kromě samotného dotazovacího jazyka je zde samozřejmě popsáno, jak jej používat v kombinaci s rozhraními DOM a SimpleXML.

Další dvě kapitoly se věnují kontrole (validaci) dokumentů XML. Zvláště v otevřeném prostředí internetu je potřeba počítat s nejhorším a všechny vstupy do aplikace pečlivě kontrolovat. Pro dokumenty XML takovou kontrolu nabízejí schémata, která dokáží popsat povolenou strukturu a datové typy dokument XML. Devátá kapitola tak popisuje nejpoužívanější schémové jazyky a v desáté kapitole je pak ukázáno, jak pomocí nich prakticky v PHP kontrolovat data uložená v XML.

Jedenáctá kapitola vysvětluje základy jazyka XSLT a jeho použití v PHP. XSLT je nejvhodnější prostředek pro transformace dokumentů XML do dalších formátů, včetně formátu HTML. Nalezne tak uplatnění například v prezentační vrstvě webové aplikace.

Následující dvanáctá kapitola se pak věnuje komunikaci mezi aplikacemi s využitím XML. Popsány jsou jak klasické webové služby, tak jednodušší mechanismy jako REST a AJAX.

Poslední třináctá kapitola pak ukazuje možnosti pro generování dokumentů XML na výstupu skriptu.

Kniha je zaměřena zejména na vysvětlení principů a na ukázky využití jednotlivých technologií a knihoven PHP. Ve většině případů jsou popsány všechny možnosti jednotlivých knihoven PHP. Nicméně kniha primárně neslouží jako referenční příručka, pro tyto účely je vhodné nahlížet i do dokumentace PHP na adrese *http://docs.php.net/manual/en/*.

Všechny příklady byly testovány s PHP ve verzi 5.2 a v současné době nic nenasvědčuje tomu, že by se v blízké době mělo v jazyce PHP měnit něco, co by způsobilo nefunkčnost skriptů. Většina použitých knihoven je standardní součástí PHP. Chcete-li používat XSLT, je potřeba do PHP přidat modul `php_xsl`, pro webové služby je zapotřebí modul `php_soap` a kapitola o Unicode využívá některé funkce z modulu `php_mbstring`.

Dlouhé řádky ve výpisech, které musely být rozděleny, jsou označeny pomocí znaku ‚►‘.

1.

Syntaxe XML

Chceme-li pracovat s dokumenty XML, musíme samozřejmě vědět, jak tyto dokumenty vypadají. V této kapitole se proto seznámíme se syntaxí jazyka XML a dalšími jeho rysy, které bychom měli znát. Znáte-li již XML dobře, a zajímá vás jen, jak se s ním pracuje v PHP, můžete tuto kapitolu směle přeskočit.

1.1 Elementy a struktura dokumentu

Každý XML dokument se skládá z *elementů*. 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>` (ukončovací tag). Jen na okraj poznamenejme, že výše uvedená ukázka je asi nejjednodušším dokumentem XML, který vůbec můžeme vytvořit.

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

Obrázek 1.1: Základní části dokumentu XML

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 také.</para>
```

Není to však příliš pohodlné, a proto můžeme v XML použít ještě jednu variantu tagu, která říká, že element nemá žádný obsah. Počáteční tag ukončíme dvojicí znaků ‚/>' místo pouhého ‚>' a koncový tag vynecháme.

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

Každý dokument XML musí obsahovat pro všechny počáteční tagy odpovídající koncový tag, nebo musí být počáteční tag zapsán jako element s prázdným obsahem. Následující ukázky jsou ukázkami špatných dokumentů, které nevyhovují specifikaci XML.

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

Ukázka je chybná, neboť tag <br> není ukončen.

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

Počáteční tag <para> není ukončen a k ukončovacímu tagu </oara> v dokumentu neexistuje odpovídající počáteční tag. Chybou rovněž je, když se elementy v dokumentu kříží.

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

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

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

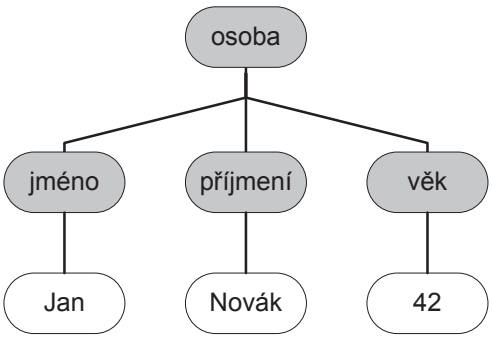
Stačí však přidat jeden element, který vše „obalí“, a vše je v pořádku.

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


1.2 Datový model dokumentu

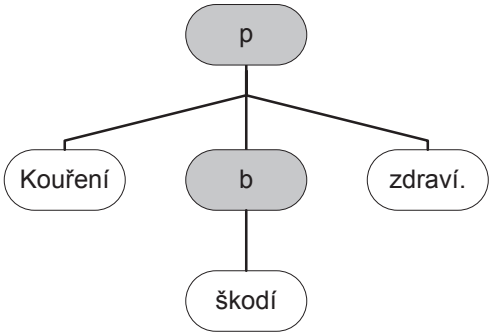
Viděli jsme, že elementy můžeme do sebe zanořovat, takže element může obsahovat další elementy nebo text. Elementy tak vytvářejí hierarchickou stromovou strukturu. Každý dokument XML si proto můžeme představit jako strom, jehož jednotlivé uzly odpovídají jednotlivým elementům (šedé uzly v obrázku), případně textu uvnitř elementů (bílé uzly v obrázku).

```
<osoba>
  <jméno>Jan</jméno>
  <příjmení>Novák</příjmení>
  <věk>42</věk>
</osoba>
```



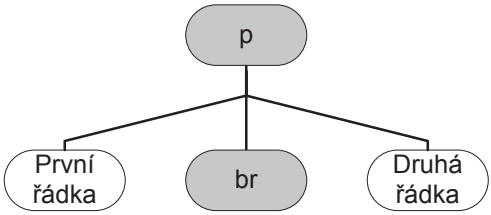
Uzly odpovídající textovému obsahu elementů jsou ve stromu vždy na nejnižší úrovni listů a už na ně nemohou být navěšeny žádné další uzly. V případě, že má element tzv. *smíšený obsah*, jsou jeho dětmi ve stromové reprezentaci jak textové uzly, tak uzly odpovídající elementům.

```
<p>Kouření
<b>škodí</b> zdraví.</p>
```



Prázdné elementy se ve stromu dokumentu objeví jako uzly, které už také nemají žádné děti.

```
<p>První řádka<br/>
Druhá řádka</p>
```



Výše popsanému stromovému modelu dokumentu XML se říká *infoset* [7]. Abstraktní datový model infosetu stručně řečeno říká, že dokument XML je stromová struktura složená z jednotlivých uzlů. Uzlů je přitom několik typů (elementy, textové uzly, atributy, komentáře, instrukce pro zpracování, jmenné prostory a další). U každého uzlu pak infoset definuje několik jeho vlastností jako jméno, rodiče, seznam dětí apod. Na infosetu je tak založena většina jazyků a rozhraní, které jsou vystaveny nad XML. Je to pochopitelné, protože při práci s dokumentem XML nás většinou zajímá jeho struktura a obsah zachycený v elementech, a infoset nabízí právě tento pohled na dokument XML. Většinou nás totiž nezajímá pohled na dokument XML jako na textový soubor, ve kterém se vyskytují speciální značky oddělené pomocí znaků ‚<' a ‚>' od ostatního textu, protože bychom se museli sami starat o syntaktickou analýzu takového zdrojového textu.

1.3 Atributy

Elementy jsou základním prostředkem pro členění informací uvnitř dokumentu XML. Kromě elementů lze pro zachycení informací využít *atributy*. Atributy se vždy zapisují k počátečnímu tagu elementu.

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

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.

```
<odstavec zabezpečení="důvěrné" autor='Jan Novák'>Nějaká tajná informace.</▶
odstavec>
```

U jednoho elementu se přitom nemohou použít dva atributy se shodným názvem. V mnoha případech je jedno, zda nějakou informaci uložíme jako element, nebo atribut. Srovnejte například následující dva fragmenty dokumentu XML:

```
<osoba věk="42">
  <jméno>Pepa</jméno>
</osoba>

<osoba>
  <jméno>Pepa</jméno>
  <věk>42</věk>
</osoba>
```

Nicméně z praxe postupně vyplynulo několik pravidel, která vám pomohou vybrat si, zda je v dané situaci lepší použít element nebo atribut. S pravidly se seznámíme v části 9.6.1.

1.4 Zápis vyhrazených znaků

Vzhledem k tomu, že se znak ‚<' používá pro oddělení tagů od okolního textu, není možné jej zapsat do dokumentu jen tak. Musíme ho opsat jako znakovou entitu `<`.

```
Vyřešte nerovnost 3x &lt; 5
```

Vidíme, že odkaz na znakovou entitu začíná znakem ‚`&`', proto i tento znak musíme do dokumentu vkládat opisem `&`.

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

Pokud potřebujeme uvnitř hodnoty atributu použít zároveň uvozovky i apostrofy, s výhodou využijeme odpovídající opisy `&quot;` a `&apos;`. XML definuje ještě pátou znakovou entitu `>`, která zastupuje znak ‚>'. Tento znak však ve většině případů můžeme zapisovat přímo bez nutnosti opisu.[1]

Tabulka 1.1: Předdefinované znakové entity XML

Entita	Znak
`<`	<
`&`	&
`>`	>
`&apos;`	'
`&quot;`	"

Jazyk XML definuje pouze těchto pět entit. Další znakové entity, které známe z HTML, jako například ` `, `&copy;` nebo `&ndash;`, v XML k dispozici nejsou, nicméně si je můžeme v případě potřeby nadefinovat sami (viz 1.13.1.1).

[1] Opis je nutný pouze v případě, že by se v dokumentu vyskytovala sekvence znaků ‚`]]>`'. Ta se musí přepsat jako `]]>`.

1.5 Názvy elementů a atributů

XML je (na rozdíl například od HTML) citlivé na velikost písmen. Počáteční a koncový tag se proto musí shodovat i ve velikosti písmen. Následující element je chybný, protože si neodpovídá počáteční a koncový tag:

```
<NÁZEV>Tento element je zapsán špatně.</název>
```

Samotná jména elementů a atributů mohou přitom být vytvářena poměrně volně. První znak jména musí být písmeno nebo podtržítko, další znaky mohou navíc obsahovat i čísla, tečku a pomlčku. Písmena přitom mohou být i z jiné než anglické abecedy. Jména elementů a atributů tak můžeme psát klidně česky, nebo třeba rusky v azbuce:

```
<имя>Булгаков</имя>
```

1.6 Deklarace XML

Každý dokument XML by měl začínat deklarací XML, ve které určíme, jakou verzi XML používáme a v jakém kódování je soubor uložen.

```
<?xml version="1.0" encoding="utf-8"?>
<osoba>
  <jméno>Jan</jméno>
  <příjmení>Novák</příjmení>
  <věk>42</věk>
</osoba>
```

Každá aplikace, která podporuje XML, musí umět zpracovat soubor uložený v kódování UTF-8 nebo UTF-16. Proto bychom měli dokumenty XML přednostně ukládat a ostatním posílat v jednom z těchto kódování. V praxi se přitom častěji používá UTF-8 kvůli lepší kompatibilitě se staršími aplikacemi. Dokumenty je možné ukládat i v jiných kódováních, ale pak musíme toto kódování povinně určit v deklaraci XML a nemáme jistotu, že tento dokument zvládnou zpracovat všechny aplikace.

V případě, že potřebujeme do dokumentu vložit nějaký znak, který buď není snadno dostupný na klávesnici nebo nejde reprezentovat v použitém kódování, můžeme do dokumentu vložit odkaz na číselný kód znaku v Unicode (podrobnější vysvětlení problematiky Unicode a kódování naleznete v kapitole 3). Předchozí dokument XML tak můžeme také zapsat jako:

```
<?xml version="1.0" encoding="us-ascii"?>
<osoba>
  <jméno>Jan</jméno>
  <příjmení>Nov&#xE1;k</příjmení>
  <věk>42</věk>
</osoba>
```

Znak „á“ byl nahrazen svým číselným kódem (U+00E1) zapsaným v šestnáctkové soustavě. Kód je možné zapsat i v desítkové soustavě, v číselném odkazu na znak pak chybí ‚x‘.

```
<?xml version="1.0" encoding="us-ascii"?>
<osoba>
  <jméno>Jan</jméno>
  <příjmení>Nov&#225;k</příjmení>
  <věk>42</věk>
</osoba>
```

1.7 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 --. Z toho vyplývá, že komentáře do sebe bohužel nemůžeme zanořovat. 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>
```

1.8 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. V takových případech se používá tzv. sekce CDATA. Oceníme ji 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 type="text/javascript"><![CDATA[
  for (i=0; i < 10; i++)
  {
    document.writeln("<p>Ahoj</p>");
  }]]>
</script>
```

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

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