XML snadno a rychle

Martin Kuba, ÚVT MU

Ve filmu *Adéla ještě nevečeřela* se americký detektiv naučí plynně česky během cesty vlakem z knihy s názvem *Česky snadno a rychle*. Bohužel, takto efektivní učebnice jsou málokdy k mání, což je zvlášť patrné, pokud se člověk rozhodne naučit dnes takto často zmiňované XML. Při prvním pokusu se totiž na něj vyhrne stádo podivně vyhlížejících pojmů jako DTD, DOM, SAX, Schema, XPath, XPointer, XLink, Namespaces, XSLT a další, z nichž každý je definován mnohastránkovým dokumentem. Slabší povahy mohou pod tímto množstvím standardů propadnout trudnomyslnosti, a odložit učení XML na pozdější dobu.

Tento článek by chtěl pomoci čtenáři se v této džungli výrazů vyznat a ukázat, které rysy XML jsou prakticky použitelné již dnes a které je možno zatím pominout.

1 Čisté XML

XML (eXtensible Markup Language) samotné je relativně jednoduché, intuitivní a praktické. Umožňuje snadno zapsat libovolné informace uspořádané do stromové struktury. Skládá se totiž z tagů (značek, podle slovníku *visačka*, *jmenovka*), které mohou obsahovat vnořené další tagy nebo text, a každý tag může obsahovat tzv. atributy, což jsou dvojice řetězců `název=„hodnota“`. Viz zápis informací o hypotetickém výletu:

```
<?xml version=„1.0“ encoding=„UTF-8“ ?>
<výlet>
 <účastníci>
  <osoba jméno=„Pepa“ />
  <osoba jméno=„Franta“ />
 </účastníci>
 <trasa>
  <start jméno=„Kuřim“ />
  <cíl   jméno=„Tišnov“ />
 </trasa>
 <svačina>
  <věc kolik=„1“>chleba</věc>
  <věc kolik=„2“>řízek
   <poznámka> dobře zabalit! </poznámka>
  </věc>
 </svačina>
</výlet>
```

V tomto příkladu je kořenem struktury tag `výlet`, který obsahuje tři tagy `účastníci`, `trasa`, `svačina`, z nichž každý obsahuje další tagy atd. Zajímavý je druhý tag `věc`, který obsahuje zároveň text, vnořený tag a má atribut. V místech mezi tagy, kde není jiný viditelný text, můžou být mezery, tabulátory a konce řádků, které jsou považovány za nevýznamné (anglicky *ignorable whitespace*). Naopak konec řádku a mezery za textem `řízek` jsou významné, protože bezprostředně sousedí s textem. Pokud tag neobsahuje text ani vnořené tagy, lze jej zapsat dvěma rovnocennými způsoby, buď jako otevírající tag bezprostředně následovaný zavírajícím tagem, nebo jako jediný tag s lomítkem na konci.

Samotná data mohou být uložena jedním ze tří způsobů – v názvech tagů a atributů, jako hodnoty atributů nebo jako text mezi tagy. Při rozhodování, zda je lepší uložit nějaký údaj jako hodnotu atributu nebo jako text, platí jednoduché pravidlo – pokud údaj může obsahovat konce řádků, musí být uložen jako text, jinak je to jedno.

V rámci XML dokumentů je možné používat všechny znaky množiny UNICODE, což zahrnuje všechny živé a pár mrtvých jazyků, žádná národnost tedy nepřijde zkrátka.

Pro svoji stromovou strukturu je XML výhodné pro zápis dat objektového charakteru, zkuste si schválně spočítat, kolik tabulek by bylo potřeba pro zápis stejných informací do relační databáze.

Na rozdíl od jiných druhů textových souborů lze s informacemi zapsanými v XML zacházet pomocí standardizovaných nástrojů, které umožňují např. kontrolovat zda struktura splňuje nějaká syntaktická omezení (tj. odpovídá určitému jazyku), transformovat jeden dokument na jiný, nebo zahrnují programová API pro práci s XML soubory. Standardizované nástroje pak šetří námahu (tj. peníze) při práci s daty.

2 Pomocné konstrukce – DOCTYPE, instrukce, komentáře, entity, CDATA

Kromě tagů, jejich atributů a textu může XML obsahovat ještě pět dalších jevů. První z nich je odkaz na definici DTD, dále komentáře obsažené

mezi znaky <!- ->, procesní instrukce obsažené mezi znaky <? ?>, entity a CDATA sekce, viz příklad:

```
<?xml version=„1.0“ encoding=„UTF-8“ ?>
<!DOCTYPE kořen SYSTEM „jazyk.dtd“>
<!-- komentář -->
<?procesní instrukce ?>
<kořen>
  znaková entita: &#x20AC;
  pojmenovaná entita: &moje;
  <![CDATA[ nebezpečná data s <>&„’ ]]>
</kořen>
```

Co je *DTD* rozebereme v následující sekci. *Komentáře* snad není třeba komentovat. *Procesní instrukce* jsou texty určené programům zpracovávajícím XML, jejich formát je libovolný a záleží jen na programu, co s nimi udělá. (Neměly by však obsahovat data, na to jsou určeny tagy a text.)

Znakové entity umožňují vložit libovolný ze znaků množiny UNICODE pomocí jeho číselné hodnoty zapsané desítkově nebo šestnáctkově.

Pojmenované entity jsou definovány v DTD a umožňují pohodlně najednou vkládat složitější sekvence znaků.

CDATA je mechanismus pro přímé vložení dat obsahujících znaky vyhrazené pro XML syntaxi, tj. <>&“’ které je mimo CDATA sekce nutné zapisovat pomocí entit `< > & &quot;` a `&apos;`.

3 Parsery, DTD a XML Schema – načítání dokumentů

Pro načítání XML dokumentů jsou k dispozici standardní knihovny, tzv. *parsery*, které existují ve dvou provedeních, tzv. *validující* a *nevalidující*. Nevalidující parser provádí minimální kontroly, jako zda jsou tagy správně vnořovány a všechny atributy mají kolem sebe uvozovky, ale nic víc. To je rychlé a v mnoha případech dostačující. Naproti tomu validující parser kontroluje, zda tagy, jejich atributy a texty uvnitř tagů splňují určitá pravidla. Tato pravidla se zapisují pomocí buď historicky staršího DTD, nebo novějšího XML Schema.

3.1 DTD

DTD (Document Type Definition) definuje množinu použitelných tagů a atributů a jejich možné umístění v XML dokumentu, tedy určitý jazyk založený na XML. Například DTD pro jazyk XHTML (což je HTML 4 zapsané jako XML) určuje, že kořenovým tagem je `<html>`, v něm mohou být tagy `<head>` a `<body>`, nikoliv však třeba `<p>`, ten smí být až uvnitř tagu `<body>`. Ukázka kousku možného DTD pro výše uvedený příklad:

```
<!ELEMENT výlet (účastníci,trasa,svačina) >
<!ELEMENT účastníci (osoba*) >
<!ELEMENT osoba EMPTY>
<!ATTLIST osoba jméno CDATA #REQUIRED>
```

DTD může obsahovat definice entit pro zjednodušení psaní, například DTD pro XHTML definuje pojmenované entity pro znaky jako jsou matematické značky (`&forall;`), národní znaky (`&eacute;`) nebo znak eura (`&euro;`). Dále může obsahovat implicitní hodnoty některých atributů, což ušetří místo v dokumentech.

3.2 XML Schema

Bohužel DTD neumí definovat typovost dat, nelze v něm například stanovit, že atribut `kolik` tagu `věc` musí obsahovat přirozené číslo. Proto bylo vytvořeno *XML Schema*, které umožňuje uvalit na data typová omezení. Soubory definující konkrétní Schema jsou (na rozdíl od DTD, které má svůj vlastní definiční jazyk) zapsané opět v XML, a kromě očekávatelných typů jako řetězec, číslo, čas, dokáží popsat i složené objekty nebo typy vzniklé omezením jiných typů, např. číselné a časové intervaly. Ukázka kousku Schema definujícího totéž co poslední dva řádky z ukázky DTD:

```
<xs:element name=„osoba“>
 <xs:complexType>
  <xs:attribute name=„jméno“
      type=„xs:string“ use=„required“ />
 </xs:complexType>
</xs:element>
```

XML Schema by mělo postupem času nahradit DTD, protože kromě definice entit umí všechno co DTD a spoustu věcí navíc.

Trudnomyslný čtenář budiž ubezpečen, že XML se dá prakticky používat i bez jakékoliv znalosti DTD a XML Schema. Stačí když si zapamatuje,

že by se mohly hodit pro kontrolu správného zápisu XML.

4 DOM a SAX – programová API

Pro přebírání XML dat od parseru se vyvinula dvě programová rozhraní s opačnou filozofií a opačnými výhodami a nevýhodami.

DOM (Document Object Model) vytvoří z dat odpovídající strom objektů v paměti. Výhoda je, že s takovou reprezentací dat se snadno pracuje. Nevýhody jsou, že objekty v paměti zabírají až třikrát víc paměti než původní XML soubor a vytvoření objektů stojí určitý čas. DOM je rozhraní definované samotným W3C konzorciem a je nezávislé na programovacím jazyce.

Naproti tomu *SAX (Simple API for XML)* převádí data při načítání XML souboru na posloupnost volání funkcí, tedy určitá funkce je vyvolána vždy na začátku každého tagu, jiná na konci každého tagu, další pro každý souvislý úsek textu, a ještě jiná pro každý komentář atd. Nedefinuje žádný obraz dat v paměti. Výhodou je velká rychlost zpracování a minimální paměťová náročnost, nevýhodou je určité nepohodlí při programování a odpovědnost aplikace za udržování načtených dat v paměti. SAX není standardem W3C, bylo vyvinuto členy emailové konference XML-DEV a stalo se „de facto“ standardem.

5 Namespaces – míchání jazyků

Jmenné prostory (namespaces) byly do XML přidány až dodatečně, například definice DTD vznikla ještě před jejich zavedením, a proto mohou působit jisté obtíže, i když byly zavedeny s ohledem na zpětnou kompatibilitu. Jmenné prostory umožňují kombinovat v jednom XML dokumentu více jazyků, což je potřeba například v XHTML stránce s vloženým SVG obrázkem, protože XHTML i SVG jsou jazyky založené na XML. Nebo v transformacích popsaných pomocí XSLT (viz dále) jsou v jednom souboru přítomny programovací tagy transformačního jazyka i tagy cílového jazyka.

Tagy a atributy z různých jazyků se odlišují prefixy před dvojtečkou, samotné prefixy jsou nevýznamné, pouze zastupují URI (Uniform Resource Identifier) definující jmenný prostor, viz příklad:

```
<j1:značka
 xmlns:j1=„urn:jazyk1“
 xmlns:j2=„http://www.nekde.cz/jazyk2“>
 <j2:jináznačka />
</j1:značka>
```

Vazba mezi prefixem a URI, které zastupuje, se provádí atributem tvaru `xmlns:`*prefix*`=„`URI“ a je platná uvnitř tagu, který tento atribut obsahuje. Pro úspornější zápis lze jeden jmenný prostor definovat jako implicitní atributem tvaru `xmlns=„`*URI*“ a všechny tagy bez prefixu (v oblasti platnosti) pak patří do tohoto prostoru.

Velké zmatení je vyvoláváno tím, že podmnožinou URI jsou stará známá URL[1] (Uniform Resource Locator), jenže URI jsou zde použita pouze pro *jedinečnost* jmenných prostorů, proto URL použitá jako URI na označení jmenného prostoru nemusí odkazovat na existující dokument, dokonce ani stroj uvedený v URL nemusí existovat, pouze reprezentují jedinečnou posloupnost znaků. Pokud tedy URL označující nějaký jmenný prostor zadáte do prohlížeče, s největší pravděpodobností získáte chybu `Not Found`. Toto použití URL je značně kontroverzní, bohužel však zavedené samotným W3C konzorciem.

6 XPath, XLink, XPointer, XQuery – adresy a vyhledávání

6.1 XPath

XPath je jazyk pro zápis výrazů popisujících cestu *uvnitř* XML dokumentu. Lze popsat cestu k tagům, atributům, textům, procesním instrukcím i komentářům, které se souhrně označují jako *uzly* (anglicky *nodes*). Nelze popsat cestu k entitám nebo CDATA sekcím, protože ty jsou nahrazeny už parserem. Výrazu může odpovídat buď právě jeden uzel, pak lze výraz chápat jako adresu v rámci dokumentu, nebo více uzlů, pak lze výraz chápat jako vyhledávání v dokumentu. XPath se nejčastěji používá v XSLT pro výběr množiny zpracovávaných uzlů a v XPointer pro označení adresy uvnitř dokumentu.

Cesta v XPath výrazu se zapisuje jako jeden nebo více *kroků* oddělených lomítky. Začíná v tvz.

[1] Kromě URL patří mezi URI ještě tzv. URN – Uniform Resource Name, označující zdroj bez ohledu na jeho umístění

kontextovém uzlu, který je určen mechanismem mimo XPath, například v XSLT je to právě zpracovávaný uzel. Kroky je možno zapisovat dvěma způsoby, *zkráceným* a *nezkráceným*.

Ve zkráceném zápisu `osoba` určuje tag „osoba“, `@jméno` atribut „jméno“, `..` nadřízený uzel, `/` vrchol dokumentu, `//` kdekoliv v dokumentu, `text()` textový obsah tagu, `comment()` komentář, `processing-instruction()` instrukci. Lze použít obecné popisy, `*` označuje jakýkoliv tag, `node()` jakýkoliv uzel a `@*` jakýkoliv atribut.

V nezkráceném zápisu se kroky zapisují ve tvaru *osa*`::`*test*. Osa (anglicky *axis*) určuje směr kroku (existuje jich 13, např. `child` – přímý potomek, `descendant` – potomek, `parent` – nadřízený uzel, `following-sibling` – uzly na stejné úrovni za kontextovým uzlem). Test vybírá uzel, řetězec *jméno* vybírá uzel jménem „jméno“.

Za zkrácený i nezkrácený zápis lze dále doplnit tzv. *predikáty*, zapisované mezi hranaté závorky. Predikáty jsou pravdivostní výrazy, dále omezující výběr uzlů, např. `[@věk>18]` vybere pouze uzly, obsahující atribut `věk` s číselnou hodnotou vyšší než 18, nebo `[position()=3]` vybere jen uzel, který je třetí v pořadí.

Uveďme tedy konkrétní příklad. Cesta

```
/výlet/child::*/věc[@kolik=1]/text()
```

začíná na vrcholu dokumentu, první krok vybírá tagy `výlet`, druhý krok všechny jejich přímé potomky, třetí krok vybere mezi jejich přímými potomky tagy `věc`, ale jen ty s atributem `kolik` s číselnou hodnotou 1, a poslední krok volí textový obsah tagu. Ve výše uvedeném příkladu této cestě odpovídá jeden uzel, a to text `chleba`.

XPath je díky svému použití v XLST stylesheetech (viz dále) velice dobře zavedený[2] a mnohokrát naimplementovaný standard.

6.2 XLink

XLink umožňuje odkazy mezi *celými* XML dokumenty, oproti hyperlinkům známým z HTML umožňuje i dvousměrné nebo dokonce vícesměrné odkazy. XLink je záležitost značně nová[3] a nepříliš implementovaná, prohlížeč Mozilla implementuje pouze jednosměrné odkazy funkčně odpovídající hyperlinkům známým z HTML:

```
<odkaz
 xmlns:xlink=„http://www.w3.org/1999/xlink“
 xlink:type=„simple“ xlink:href=„doc.xml“
 xlink:actuate=„onRequest“
 xlink:show=„replace“ >
  Click here
</odkaz>
```

6.3 XPointer

XPointer kombinuje XLink a XPath, umožňuje odkázat na část libovolného XML dokumentu. Jako v HTML dokumentech je možné v URL uvést část dokumentu pomocí znaku # a názvu části, např.

```
http://nekde.cz/dokument.html#cast3
```

XPointer umožňuje odkázat na část XML dokumentu určenou pomocí XPath výrazu, např.

```
http://nekde.cz/doc.xml#xpointer(//cast)
```

XPointer je také záležitost nová[4] a neexistuje mnoho implementací.

6.4 XQuery

Horká novinka[5] je XQuery, dotazovací jazyk pro vyhledávání v XML datech, podobně jako SQL je jazyk pro vyhledávání v relačních databázích. Tento standard se teprve vyvíjí a vynucuje si vývoj nové verze jazyka XPath 2.0, v současné době tedy není prakticky použitelný.

Trudnomyslný čtenář může existenci XLink, XPointer a XQuery zatím pominout, avšak znalosti XPath se asi nevyhne.

7 XSLT, XSL, XSL:FO – vzhled a přeměny

XML definuje pouze syntaxi, nikoliv však význam nebo vzhled dat. Pokud je třeba data zobrazit, je nutné přidat informace o tom, jak data převést do vizuální podoby. K tomu se používají tzv. *stylesheety* zapisované pomocí XSL.

XSL (eXtensible Stylesheet Language) je norma složená ze dvou nezávislých částí: XSLT a XSL:FO.

[2] W3C Recommendation z 16. listopadu 1999
[3] W3C Recommendation z 27. června 2001
[4] W3C Proposed Recommendation z 13. listopadu 2002
[5] W3C Working Draft z 15. prosince 2002

7.1 XSL:FO (XSL Formating Objects)

XSL:FO je soubor typografických objektů, jako jsou stránky, bloky textu, poznámky pod čarou atd., a jejich vlastností (anglicky *properties*) jako šířky okrajů, fonty, barvy. Objekty v XSL:FO jsou založeny na objektech používaných v CSS2 (Cascading StyleSheets) pro formátování HTML. XSL:FO objekty jsou zapsatelné jako XML tagy v jistém konkrétním jmenném prostoru, takže je možné je uložit jako XML dokument, nebo mohou být přímo zobrazeny. Ukázka zápisu odstavce a kurzívou zvýrazněného slova:

```
<fo:block
     font-size=„12pt“
     font-family=„Times“
     line-height=„10pt“
     text-align=„justify“>
 Nějaký <fo:inline font-style=„italic“>
 text</fo:inline> odstavce ...
</fo:block>
```

7.2 XSLT (XSL Transformations)

XSLT je programovací jazyk, ve kterém se popisuje transformace XML dokumentu na jiný strom objektů. Původním záměrem XSLT byla transformace XML dokumentu na strom XSL:FO objektů, které jsou následně zobrazeny.

Je však možné popsat transformaci na libovolný jiný XML dokument, na HTML dokument, nebo dokonce na libovolný textový výstup. Při aplikaci různých stylesheetů na jeden XML soubor je tak možné získat různé výstupy (HTML, text, XSL:FO, XML) a pokud je výsledkem transformace opět XML dokument, je možné transformace řetězit.

Ukázka XSLT stylesheetu generujícího seznam věcí na výlet jako HTML stránku:

```
<?xml version=„1.0“ ?>
<xsl:stylesheet version=„1.0“
xmlns:xsl=„http://www.w3.org/1999/XSL/Transform“>
<xsl:output method=„html“/>
<xsl:template match=„/“>
 <html>
  <head><title>Výlet</title></head>
  <body>
  <xsl:for-each select=„//věc“>
   <xsl:value-of select=„@kolik“/>
   <xsl:value-of select=„text()“/> <br/>
  </xsl:for-each>
  </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

7.3 Podpora v prohlížečích

Prohlížeče Mozilla 5 a MSIE 6 v sobě již mají zahrnutu implementaci XSLT. Je tedy možné hned teď zobrazovat XML dokumenty. Pokud XML soubor obsahuje odkaz na XSLT stylesheet provádějící transformaci na HTML (pomocí procesní instrukce, čtenář si to může vyzkoušet doplněním řádku

```
<?xml-stylesheet type=„text/xsl“ href=„vylet.xsl“?>
```

do příkladu s výletem a uložením příkladu XSLT do souboru `vylet.xsl`), stylesheet je aplikován a výsledné HTML pak prohlížeč zobrazí. Pokud XML soubor odkaz neobsahuje, použije se implicitní stylesheet, který u MSIE zobrazí XML s barevně zvýrazněnou syntaxí a JavaScriptem implementovanou možností tagy „rozbalovat“ a „zavírat“. V Mozille se použije implicitní stylesheet definovaný normou XSLT, který vynechá všechny tagy a zobrazí pouze text.

Bohužel, dnešní prohlížeče nemají implementovánu podporu XSL:FO, existuje však nástroj *Apache FOP*, který umí převést strom XSL:FO objektů na PDF soubor. Je tedy možné XML dokument převést pomocí XSLT procesoru na soubor XSL:FO tagů a ty pak pomocí FOPu převést na PDF.

8 Závěr a odkazy

XML samotné je poměrně stabilní standard. Na něj navazující standardy jako XSLT, XPath se bouřlivě vyvíjejí a jsou nahrazovány novějšími verzemi. Další standardy jako XML Signature (digitální podpis) nebo SOAP (vzdálené volání procedur pomocí XML) byly právě dokončeny nebo se teprve dokončují a lze v blízké době čekat jejich další verze. Přesto je možné XML s úspěchem používat již dnes, protože je dostupných mnoho nástrojů pro zpracování XML, jak komerčních tak freewareových, a jeden z nich má pravděpodobně každý z nás již ve svém WWW prohlížeči.

- W3C definice `http://www.w3c.org/xml/`
- kniha E. R. Harold, W. S. Means: „XML in a nutshell“, O'Reilly 2001, ISBN 0-596-00058-8
- parser Apache Xerces `http://xml.apache.org/xerces2-j/`

- XSLT procesor Saxon http://saxon.sf.net
- Apache FOP http://xml.apache.org/fop/ □