XML a Java

Jaroslav Měcháček, ÚVT MU

Značkovací jazyk XML se stal de facto standardem pro popis strukturovaných dat. Spolu s procedurálním, na platformě nezávislým programovacím jazykem Java vytváří skvělou kombinaci pro tvorbu (nejen) internetových aplikací. Java je stále více používána v rámci aplikačních serverů, které jsou kompletně implementovány právě v tomto jazyce a v naprosté většině případů splňují specifikaci J2EE (Java 2 Enterprise Edition). I když firma SUN tak říkajíc trochu zaspala a J2EE v1.2 neobsahovala podporu XML, ve verzi 1.3 vše napravila. Tento článek popisuje možnosti analýzy, transformací a formátování XML dokumentů v Javě.

1 Parsery

Prvním krokem ke zpracování XML dokumentu je jeho analýza, kontrola syntaxe, ověření, že dokument odpovídá danému DTD, popř. doplnění neuvedených hodnot atributů z DTD specifikace. Dokument je potom převeden na interní datovou reprezentaci, kterou je možno dále zpracovat. Veškerou tuto činnost (dále nazývanou parsování) zajistí XML parser.

V zásadě existují dva hlavní přístupy při parsování XML dokumentu - přístup založený na stromové struktuře a přístup založený na událostech.

1.1 API založené na stromové struktuře

Na základě načítaných dat je vytvořena stromová struktura objektů, které reprezentují původní dokument. Tato struktura pak slouží k dalšímu zpracování. Lze se v ní jednoduše pohybovat z uzlu do uzlu, provádět výpočty, přidávat nebo rušit uzly a atributy. Naprostá většina parserů podporuje stromovou strukturu DOM. DOM (Document Object Model) je specifikace programového rozhraní W3C konsorcia, použitelná v mnoha prostředích a aplikacích. Definuje logickou strukturu dokumentu a způsob přístupu a manipulace s dokumentem. Rozhraní je definováno s použitím IDL (Interface Definition Language), pro jednotlivé programovací jazyky pak existují napojení. V případě Javy je to balík `org.w3c.dom`.

1.2 API založené na událostech

Ačkoli by se mohlo zdát, že použití stromové struktury pro reprezentaci a manipulaci s dokumentem je výhodné, existují případy, ve kterých je naprosto nevyhovující. Zásadní nevýhodou jsou vysoké požadavky na systémové zdroje v případě rozsáhlých dokumentů. Je-li například potřeba v dokumentu velkém několik megabytů najít konkrétní data, je převod dokumentu na stromovou strukturu a následné prohledávání plýtváním systémových zdrojů. Řešením je použít API založené na událostech. Parser v tomto případě při zpracování dokumentu negeneruje strom objektů, ale sérii událostí, na které aplikace patřičným způsobem reaguje, aniž by bylo potřeba udržovat celý dokument v paměti. Nejznámějším a nejpoužívanějším API tohoto typu je *The Simple API for XML*, zkráceně SAX. V případě následujícího XML dokumentu

```
<?xml version="1.0"?>
<doc>
  <h1>Ahoj</h1>
</doc>
```

vygeneruje SAX parser tuto sekvenci událostí:

```
start document
start element: doc
start element: h1
characters: Ahoj
end element: h1
end element: doc
end document
```

1.3 Java API for XML Processing

Jak již bylo řečeno na začátku článku, zpočátku SUN v oblasti XML nevyvíjel patřičnou koordinační snahu. Většina dostupných XML parserů v Javě sice používala DOM a SAX, přesto nebylo snadné zaměnit v aplikaci jednu implementaci parseru za jinou. Nakonec však vznikla specifikace *Java API for XML Processing* (JAXP) definující rozhraní pro parsování a manipulaci s XML dokumenty, zahrnující i rozhraní pro XSL transformace.

Rozhraní pro parsování jsou definována v balíku `javax.xml.parsers`:

DocumentBuilder je rozhraní, jímž lze získat DOM strukturu reprezentující původní XML dokument;

DocumentBuilderFactory slouží k vytváření instancí tříd implementujících `DocumentBuilder` rozhraní s předem definovanými vlastnostmi. Vlastností může být například to, že je dokument v průběhu parsování kontrolován oproti DTD specifikaci;

SAXParser a SAXParserFactory obdobně definují rozhraní pro zpracování dokumentů s použitím SAX API.

1.4 Zdarma dostupné XML parsery

Xerces Parser vyvíjený v rámci Apache XML projektu.

XP Autorem je James Clark, důraz je kladen především na rychlost.

Crimson Odvozen od Projekt X parseru firmy SUN, který je součástí referenční implementace JAXP specifikace.

1.5 Projekt JDOM

DOM i SAX jsou obecné specifikace nezávislé na konkrétním programovacím jazyku. Ukazuje se, že implementace DOM v Javě není příliš intuitivní nebo snadno použitelná. Z tohoto důvodu vznikl někdy na začátku roku 2000 projekt JDOM, jehož cílem je vyvinout API pro manipulaci s XML dokumenty, určené pouze pro Javu. JDOM rozhraní, tak jako DOM, reprezentuje XML dokumenty stromovou strukturou objektů, mělo by však být rychlejší, méně paměťově náročné a hlavně mnohem snadněji použitelné. Následující fragment Java kódu vytvoří reprezentaci jednoduchého JDOM dokumentu, který je následně vypsán jako textový XML dokument:

```
Document doc = new Document (
  new Element("doc").addContent(
      new Element("h1").
          setText("Ahoj") )
);
(new XMLOutputter()).
     output(doc,System.out);
```

Konstruovat XML dokument v Javě tímto způsobem je mnohem přirozenější než spojovat textové řetězce. Současně je zajištěno, že výsledný dokument bude korektní.

JDOM je nyní ve stádiu Beta 7 a v rámci JCP (Java Community Process) by mělo být definováno JDOM 1.0 API.

2 XSL transformace

Při zpracování XML dokumentů se dosti často objevuje potřeba transformovat XML dokument na jiný XML dokument, stejného nebo různého DTD. To lze řešit napsáním konkrétního programu v Javě, který modifikuje DOM strom reprezentující původní XML a vygeneruje nový XML dokument. Mnohem elegantnější řešení však nabízí XSL transformace.

The Extensible Stylesheet Language (XSL) je W3C specifikace zahrnující formátování a transformaci XML dokumentů. Ta část, která se zabývá transformacemi XML dokumentů, bývá obvykle označována zkratkou XSLT. XSLT dokument je XML dokument obsahující pravidla, která popisují danou transformaci. Pravidlo sestává ze vzoru, určujícího uzly, na něž je aplikováno, a výstupu, který bude v případě aplikace vygenerován.

Pro ilustraci XSLT uveďme následující jednoduchý příklad.

Vstupní XML dokument obsahuje údaje o prodeji automobilů podle typu:

```
<?xml version="1.0"?>
<prodej>
  <auto typ="skoda" pocet="120"/>
  <auto typ="bmw" pocet="10"/>
  <auto typ="opel" pocet="30"/>
</prodej>
```

XSLT soubor definuje transformaci do HTML formátu:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/
       1999/XSL/Transform">
  <xsl:template match="prodej">
    <html>
      <xsl:apply-templates/>
```

```
    </html>
  </xsl:template>
  <xsl:template match="auto">
    typ: <xsl:value-of select="@typ"/>
    prodáno celkem:
      <xsl:value-of select="@pocet"/>
    <br/>
  </xsl:template>
</xsl:stylesheet>
```

Výsledkem aplikace XSLT souboru je HTML soubor určený k prezentaci na webu:

```
<html>
   typ: skoda
   prodáno  celkem: 120<br>

   typ: bmw
   prodáno  celkem: 10<br>

   typ: opel
   prodáno  celkem: 30<br>
</html>
```

Z uvedeného příkladu je celkem zřejmé, že použití XSLT je mnohem rychlejší a přehlednější než programování transformace v klasickém procedurálním jazyce. Je ovšem třeba mít k dispozici XSLT procesor, který umí XSLT transformace provádět. K nejznámějším XSLT procesorům patří:

Xalan vyvíjený v rámci Apache XML projektu;

Saxon jehož autorem je Michael Kay, k dispozici je v rámci Mozilla licence včetně zdrojového kódu. Ačkoli je vyvíjen jediným člověkem, v mnoha ohledech mívá náskok před konkurencí. Poslední verze podporuje JDOM;

XSLTC umožňující kompilovat XSLT dokumenty přímo do Java bajtkódu, který je pak možno distribuovat jako Java třídy zvané translety;

XT velmi rychlý XSLT procesor, poslední verze je však z roku 1999.

Aplikační rozhraní pro XSL transformace v Javě je součástí JAXP specifikace, konkrétně se jedná o balík `javax.xml.transform`:

DOMSource, SAXSource, StreamSource definují vstupní strukturu dat

DOMResult, SAXResult, StreamResult definují výstupní strukturu dat

Transformer provádí XSLT transformaci

Templates reprezentuje předkompilovanou XSLT šablonu, která pak může být opakovaně aplikována na vstupní data.

3 Formátování dokumentů

XSL-FO (XSL Formatting Objects) je další součást XSL specifikace. S pomocí XSL-FO lze přesně specifikovat vzhled výsledného dokumentu, tak jak bude prezentován čtenáři. Formátování pomocí XSL-FO objektů poskytuje mnohem komplexnější model stránkového rozmístění textu než HTML/CSS a jeho použití je mnohem širší. V budoucnu by mohla tato technologie konkurovat sázecím systémům, jako je například TeX. Většina současných prohlížečů zatím nepodporuje zobrazení dokumentů formátovaných v XSL-FO, takže je obvykle nutné je konvertovat do formátu PDF, PostScript nebo třeba RTF. Postup vzniku dokumentu může být následující: Vstupem je XML soubor popisující logickou strukturu dat (například text knihy, článku), na něj je aplikována XSLT šablona specifikující vzhled jednotlivých prvků (názvy kapitol, poznámky pod čarou) a výsledkem je XML dokument obsahující kompletní informaci o rozmístění a vzhledu textu na stránkách. Na závěr je možno dokument konvertovat použitím aplikace Fop.

Fop je Java aplikace, která umí konvertovat XSL-FO dokumenty do PDF. PDF je primární výstupní formát, lze produkovat i formáty PCL, SVG, Print, AWT nebo TXT. Fop umožňuje vkládat vektorovou grafiku ve formátu SVG. Ke konverzi do formátu RTF je možno použít produkt Jfor.

Literatura

[1] The Simple API for XML, `http://www.megginson.com/SAX/`.

[2] JDOM home page, `http://www.jdom.org/`.

[3] Apache XML Project, `http://xml.apache.org/`.

[4] Elliotte Rusty Harold, The XML Bible `http://www.ibiblio.org/xml/books/bible2/`.

[5] The XSLT Compiler for JavaTM Technology, `http://servlet.java.sun.com/javaone/conf/sessions/2816/0-sf2001.jsp`.

[6] JAXP specifikace, `http://java.sun.com/xml/jaxp/dist/1.1/jaxp-1_1-spec.pdf`.

[7] World Wide Web Consortium, http://www.w3c.org/.
[8] Saxon home page, http://users.iclway.co.uk/mhkay/saxon/index.html. □