

Het ontwikkelen van XML-tools

Johan Jeuring

Abstract

XML wordt in veel situaties gebruikt, en voor dat gebruik worden talloze tools gebouwd. Er zijn verschillende manieren om XML-tools te ontwikkelen, en dit artikel behandelt de voor- en nadelen van de verschillende alternatieven.

XML en XML-tools

De uitgever Kluwer biedt verschillende soorten informatie aan, zoals wetteksten en belastinggidsen. Deze informatie heeft vaak een vaste structuur: een wet heeft bijvoorbeeld altijd een nummer, en een belastinggids beschrijft componenten die voorkomen op het inkomstenbelastingformulier. De structuur verschilt echter van informatiesoort tot informatiesoort. Voor de beschrijving van een document van dit soort structuren gebruikt Kluwer de Extensible Markup Language (XML) [6], en voor de beschrijving van een structuur zelf een Document Type Definitions (DTD) of een Schema.

Een manier om de populariteit van een onderwerp te bepalen is Google te vragen hoe vaak een term voorkomt op het Web, of eigenlijk, in de databases van Google. Zo komen de Beatles bijna 3 miljoen keer voor, en ABBA honderdduizend keer. XML komt ongeveer 20 miljoen keer voor, en verslaat daarmee alle programmeertalen behalve Java. Een andere populariteitsmaat is het aantal vierkante meters boekenplank dat in beslag wordt genomen door boeken over het onderwerp in een academische boekhandel. In mijn lokale boekhandel verslaat XML menig andere nieuwe technologie. Maar wat is XML?

XML is een eenvoudig, flexibel tekstformaat, dat veel gebruikt wordt voor het uitwisselen van data op het internet, maar ook tussen bedrijven onderling, en tussen bedrijven en hun klanten. Hier is een voorbeeld van een XML document.

```
<bibliography>
  <book>
    <title>      XML in Theory and Practice </title>
    <author>    Chris Bates                </author>
    <year>      2003                       </year>
    <publisher> Wiley                      </publisher>
  </book>
</bibliography>
```

De XML-standaard is vastgesteld door het World Wide Web consortium (W3C). Het W3C ontwikkelt technologieën voor het Web. Zoals de bovenstaande populariteitsmaten al aangeven is de XML-standaard wijd verspreid. Ook in Nederland wordt veel gebruik gemaakt van XML: er bestaat bijvoorbeeld een actieve SGML/XML users group Holland met ongeveer 400 leden.

Waar en hoe wordt XML gebruikt? XML wordt voor talrijke doeleinden gebruikt, naast het beschrijven van wetteksten ook bijvoorbeeld voor het opslaan van genealogische gegevens, het beschrijven van chemische structuren, en het doorgeven van financiële transacties. Om iets te kunnen met dergelijke informatie is software nodig. Zo zijn er XML-tools om te zoeken in (bijvoorbeeld genealogische) databases, om een koopopdracht mee op te stellen, om wetteksten mee te schrijven en te publiceren, en om een chemische structuur beschreven in XML mee te presenteren. Veelgebruikte XML-tools zijn XML databases, XML editors, en XML publishing frameworks.

Een eigenschap van veel XML-tools is dat ze moeten kunnen werken met XML data die gestructureerd is volgens verschillende DTD's of Schema's. Kluwer wil graag dat een wettekst volgens de structuur van wetteksten geschreven wordt, en niet volgens de structuur van een belastinggids. Op wat voor manier wordt een XML-tool ontwikkeld die rekening houdt met verschillende soorten invoerdata?

Dit artikel gaat in op het ontwikkelen van een XML-tool zoals een XML database en een XML editor, die rekening houdt met verschillende soorten invoerdata.

Hoewel de meeste activiteiten met betrekking tot XML plaatsvinden in de VS, zijn er ook een aantal Nederlandse bedrijven die XML-tools ontwikkelen: X-Hive levert een XML database voor het opslaan van en zoeken in XML-documenten, Barbadosoft biedt een raamwerk voor het beheren en programmeren van XML omgevingen, en Genias en Daidalos leveren een serie producten op het gebied van XML publishing. Daarnaast leveren talrijke bedrijven services op het gebied van XML.

Het Ontwikkelen van XML-tools

Een XML-tool verschilt slechts in één opzicht van andere tools: een XML-tool leest, schrijft, zoekt in, en/of publiceert XML-documenten. Voor het ontwikkelen van een XML-tool gelden dus dezelfde principes die gelden voor het ontwikkelen van software in het algemeen. Belangrijke aspecten zijn onder andere modulariteit, abstractie, het encapsuleren van informatie, en sterke typering. Goed gebruik van modulariteit helpt bij het samenstellen van een tool uit verschillende, mogelijk al bestaande, componenten. Abstractie maakt het mogelijk code te hergebruiken. Encapsulatie zorgt ervoor dat wijzigingen in de software lokaal toegepast kunnen worden, en geen effect hebben op het hele systeem. Sterke typering, tenslotte, zorgt ervoor dat veel fouten in de software al in een vroeg stadium ontdekt worden. De rest van dit artikel beschrijft technieken voor het werken met een XML-document in een XML-tool, en analyseert die

technieken aan de hand van bovenstaande principes.

De alternatieven

Er bestaan tenminste vier technieken waarmee een XML-tool een XML-document kan lezen, schrijven, doorzoeken, en/of publiceren.

- De *Simple API for XML* (SAX) is een API waarmee een XML-document ingelezen kan worden. SAX is *event-driven*: een programmeur moet bijvoorbeeld aangeven wat er moet gebeuren als er een *begin document* tag in de invoer staat, of een *begin element* tag. Een Java klasse die gebruik maakt van SAX bevat bijvoorbeeld de volgende code:

```
import javax.xml.parsers.SAXParser;

public class SAXExample {
    ...
    public void startDocument() throws SAXException { ... }
    public void endDocument() throws SAXException { ... }
    public void startElement(String name, AttributeList attributes)
        throws SAXException { ... }
    public void endElement(String name) throws SAXException { ... }
}
```

- De *Document Object Model* (DOM) is een API waarmee een XML-document bewerkt kan worden. Zo kunnen bijvoorbeeld de kinderen van een element opgevraagd worden, en kan een kind aan een element toegevoegd worden. Een DOM-gebaseerde parser leest een XML document in, en stelt een boomstructuur van het document beschikbaar aan de programmeur. Hier is een voorbeeld Java programma dat gebruikt maakt van de DOM:

```
import org.w3c.dom.*;

public class DOMExample {
    ...
    public void processNode(Node currentNode) {
        switch(currentNode.getNodeType())
            ...
            case Node.ELEMENT_NODE:
                NamedNodeMap attributeNodes = currentNode.getAttributes();
                ...
                processChildNodes(currentNode.getChildNodes());
            ...
    } }
}
```

- Er zijn verschillende programmeertalen specifiek ontwikkeld voor het verwerken van een XML-document. Voorbeelden van dit soort programmeertalen zijn W3C's XSLT [7], XDoc, λML, Yatl, en Xtatic [2]. In de meeste

van deze programmeertalen kan een XML-document direct als waarde opgenomen worden, en heeft deze waarde een type, dat gecontroleerd kan worden. Hier is een voorbeeld van een XSLT programma:

```
<xsl:stylesheet version="1.0" ...>
  <xsl:template match="myEmail">
    <html>
      <body><xsl:value-of select="email"/></body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

- Tenslotte bestaan er zogenaamde *Data Bindings* [4] van XML naar een programmeertaal voor verschillende programmeertalen. Een data binding beeldt een DTD of een Schema af op een klassehiërarchie of een datatype, en beeldt een bijbehorend XML-document af op een object van de klassehiërarchie, of een waarde van het datatype. Er bestaan XML data bindings voor Java [3, 5], Python, Prolog, Haskell [8, 1], en veel andere programmeertalen. Bijvoorbeeld, de volgende DTD

```
<!ELEMENT doc      (author*,title,year?)>
<!ELEMENT author  (#PCDATA)>
<!ELEMENT title   (#PCDATA)>
<!ELEMENT year    (#PCDATA)>
```

zou afgebeeld kunnen worden op

```
public class Doc {
  private String[] authorList;
  private String  title;
  private String[] year;

  ...
  public void setTitle(String title) {...}
  ...
}
```

En een XML-document van de doc DTD zou afgebeeld kunnen worden op een object van de klasse Doc.

Een aantal Nederlandse bedrijven die XML-tools ontwikkelen hebben gegevens aangeleverd over de technieken waarmee zij een XML-document lezen, schrijven, of doorzoeken. De DOM wordt door alle aangeschreven bedrijven gebruikt. Verder blijkt dat elke van de bovenstaande alternatieven door tenminste één bedrijf gebruikt wordt.

Een vergelijking

Welke van de bovenstaande technieken kan nu het best gebruikt worden bij de ontwikkeling van een XML-tool? Uiteraard hangt dat van de eisen en grootte van de XML-tool af, maar er zijn een aantal algemene principes die een rol spelen.

Het gebruik van SAX heeft tot gevolg dat code voor het ontleden van een XML-document gemixed moet worden met code voor het gewenste gebruik (de *business logic*). Dit maakt het schrijven van modulaire code zeer moeilijk zo niet onmogelijk. Als ook encapsulatie van het invoerdocument belangrijk is, dan kan beter één van de andere technieken gebruikt worden. Alleen als efficiëntie zeer belangrijk is, en het document dat wordt ingelezen groot is en niet bewerkt hoeft te worden, is SAX een goed alternatief.

Bij het gebruik van de DOM wordt een XML-document ingelezen als een waarde van een universele boomstructuur. Dit maakt zowel modulaire ontwikkeling als encapsulatie mogelijk. Echter, de structuur van een DOM boom is onafhankelijk van de DTD of Schema van het invoerdocument. Dit betekent dat de API toestaat dat we een kind element toevoegen aan een element, terwijl de DTD of Schema van het document dat niet toelaat. Het is dus moeilijk om typecorrectheid te garanderen bij het gebruik van de DOM, en van sterke typering is geen sprake. Overigens lijkt men zich bij het W3C ook zorgen te maken over dit probleem en is er nu een *candidate recommendation* voor *DOM Validation* opgesteld, waarin gespecificeerd wordt hoe een valide document geconstrueerd kan worden met behulp van een uitgebreide DOM.

Een programmeertaal specifiek ontwikkeld voor het verwerken van XML documenten is soms een aantrekkelijk alternatief. Voor het publiceren van een XML document is bijvoorbeeld XSLT vaak zeer geschikt. Voor meer complexe problemen, waar *business logic* geïmplementeerd moet worden, is XSLT echter minder geschikt, en ligt het meer voor de hand om een programmeertaal als Java of Xtatic (een uitbreiding van C# waarin ook een XML-document geconstrueerd en getypeerd kan worden) te gebruiken. Een nadeel van XML-specifieke programmeertalen is dat een nieuwe programmeertaal geleerd moet worden, en dat veel tools die voor veel gebruikte programmeertalen beschikbaar zijn, zoals ontwikkelomgevingen, debuggers, optimaliserende compilers, en geheugenprofilers vaak niet of slechts beperkt aanwezig zijn.

Een data binding biedt de mogelijkheid code te schrijven in een geavanceerde programmeertaal. Afhankelijk van de programmeertaal is het zo mogelijk modulaire programma's te schrijven, en gebruik te maken van abstractie en encapsulatie. Het gebruik van de programmeertaal Haskell, en in mindere mate Java, geeft ook nog eens sterke typering. Een mogelijk nadeel van het gebruik van een XML data binding is dat de gebruiker moet weten hoe XML-documenten van een gegeven DTD of Schema afgebeeld worden. Om de afbeelding te kunnen sturen is het XML binding schema concept ontwikkeld. Een XML binding schema specificeert hoe een element uit een XML document afgebeeld wordt op, bijvoorbeeld, een object van een klasse in Java. Het element `<datum>` kan bijvoorbeeld afgebeeld worden op een object in de Java klasse `Date`. Het schrijven

van een XML binding schema kan echter soms ook weer complex zijn, en het laatste woord over XML data bindings en XML binding schema's is nog niet gezegd.

Conclusies

Het ontwikkelen van een XML-tool verschilt slechts in één opzicht van het ontwikkelen van software in het algemeen: een XML-tool moet een XML-document kunnen lezen, schrijven, en/of bewerken. De belangrijke extra vraag die dus beantwoord moet worden wanneer een XML-tool ontwikkeld wordt is: welke techniek wordt gebruikt voor het bewerken van een XML-document? Dit artikel schetst vier technieken, en analyseert de technieken aan de hand van algemene softwareontwikkelingsprincipes. Uit de analyse blijkt dat SAX slechts in enkele gevallen de voorkeur zal hebben, en dat een XML data binding de beste mogelijkheden biedt voor het ontwikkelen van een goede XML-tool. De XML data binding wereld is echter nog sterk in ontwikkeling, en de verwachting is dat er de komende jaren nog het nodige gezegd zal worden over dit onderwerp.

References

- [1] Frank Atanassow, Dave Clarke, and Johan Jeuring. Scripting XML with Generic Haskell. In *Proceedings of the 7th Brazilian Symposium on Programming Languages, SBLP 2003*, 2003. An extended version of this paper appears as Technical report ICS Utrecht University, UU-CS-2003-023.
- [2] Vladimir Gapeyev and Benjamin C. Pierce. Regular object types. In *European Conference on Object-oriented Programming (ECOOP 2003)*, 2003.
- [3] Brett McLaughlin. *Java & XML data binding*. O'Reilly, 2003.
- [4] Eldon Metz and Allen Brookes. XML data binding. *Dr. Dobb's Journal*, pages 26–36, March 2003.
- [5] Sun Microsystems. Java Architecture for XML Binding (JAXB). <http://java.sun.com/xml/jaxb/>, 2003.
- [6] W3C. XML 1.0. Available from <http://www.w3.org/XML/>, 1998.
- [7] W3C. XSL Transformations 1.0. Available from <http://www.w3.org/TR/xslt>, 1999.
- [8] Malcolm Wallace and Colin Runciman. Haskell and XML: Generic combinators or type-based translation? In *International Conference on Functional Programming*, pages 148–159, 1999.

Prof.dr. Johan Jeuring is hoogleraar en hoofddocent Software Technologie aan de Open Universiteit Nederland en de Universiteit Utrecht.