XML-Schema

Matthias Hansch · Stefan Kuhlins · Martin Schader

XML hat sich als Sprache zur Definition von Dokumenten für den universellen Datenaustausch etabliert. XML-Schema ermöglicht ausgefeilte Definitionen für XML-Dokumente und hat beste Chancen, die bisher eingesetzte DTD abzulösen. Neben einer allgemeinen Einführung geht dieser Beitrag auch auf verfügbare Produkte und neue Möglichkeiten der objektorientierten Softwareentwicklung mit XML-Schema ein.

Das World Wide Web Consortium (W3C) hat die Spezifikation von XML-Schema im Mai 2001 mit dem Status „Recommendation" versehen und somit nach einem dreijährigen Entwicklungsprozess verabschiedet [14]. Ein XML-Schema definiert ähnlich wie die DTD (Document Type Definition) Regeln für die Syntax und die Struktur einer Klasse von XML-Dokumenten [12].

Die DTD ist Teil der XML-Spezifikation. Es hat sich jedoch herausgestellt, dass die DTD Schwächen aufweist, die zur Entwicklung von XML-Schema geführt haben. Die wichtigsten Verbesserungen sind:

- Jedes XML-Schema ist selbst ein XML-Dokument, sodass im Gegensatz zur DTD keine spezielle Syntax verwendet wird, die separate Werkzeuge zur Verarbeitung erfordert. Insbesondere lässt sich jedes XML-Schema wiederum durch ein XML-Schema validieren, ohne dass eine weitere Metaebene benötigt wird.
- Auch komplexe Integritätsbedingungen sind formulierbar.
- XML-Schema enthält eine große Zahl vordefinierter Datentypen und ermöglicht die Definition von eigenen Datentypen, wodurch eine Typprüfung möglich wird.
- Bei Datentypen werden Vererbung und Substitution unterstützt.
- Nullwerte sind, wie bei Referenzen und in Datenbanken üblich, darstellbar.
- Das Modularisieren und Wiederverwenden von XML-Schemata ist möglich.
- Benennungskonflikte können durch Verwendung von XML-Namensräumen vermieden werden.

Beispiel

Einige dieser Besonderheiten sollen anhand des Beispiels in Abb. 1 erläutert werden. In Zeile 2 wird zunächst der Namensraum für alle Schlüsselworte festgelegt, die im Folgenden durch das Präfix `xsd` kenntlich gemacht werden. Der Datentyp `PersonTyp` wird in Zeile 3 als abstrakt definiert, kann also nicht für Element- oder Attributdeklarationen verwendet werden. Das Subelement `Vorname` kann maximal dreimal vorkommen (Zeile 5). `Geburtsdatum` ist mit dem in XML-Schema vordefinierten Datentyp `date` deklariert. In Zeile 12 wird `PersonTyp` als Supertyp für `MitarbeiterTyp` festgelegt, d. h., `MitarbeiterTyp` erbt alle Elemente und Attribute von `PersonTyp` und fügt ihm das Element `Gehalt` hinzu. `Gehalt` besitzt als Datentyp eine Dezimalzahl, die größer oder gleich 10.000 sein muss. Schließlich enthält das Element `Unternehmen` in Zeile 28 eine Referenz auf das Element `Mitarbeiter` des Datentyps `MitarbeiterTyp`, das beliebig oft vorkommen kann.

Ein Beispiel für ein zum Schema passendes XML-Dokument ist in Abb. 2 enthalten.

Matthias Hansch
McKinsey & Company, Birkenwaldstraße 149, 70191 Stuttgart
E-Mail: matthias_hansch@mckinsey.com

Stefan Kuhlins
Universität Regensburg, Lehrstuhl für Wirtschaftsinformatik IV,
Universitätsstraße 31, 93040 Regensburg
E-Mail: stefan.kuhlins@wiwi.uni-regensburg.de

Martin Schader
Universität Mannheim, Lehrstuhl für Wirtschaftsinformatik III,
Schloss, 68131 Mannheim
E-Mail: mscha@wifo3.uni-mannheim.de

* Vorschläge an Prof. Dr. Frank Puppe
<puppe@informatik.uni-wuerzburg.de> oder
Dieter Steinbauer <dieter.steinbauer@schufa.de>
Alle „Aktuellen Schlagwörter" seit 1988 finden Sie unter:
www.ai-wuerzburg.de/as

```
 1  <?xml version="1.0" encoding="UTF-8"?>
 2  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 3    <xsd:complexType name="PersonTyp" abstract="true">
 4      <xsd:sequence>
 5        <xsd:element name="Vorname" type="xsd:string" maxOccurs="3"/>
 6        <xsd:element name="Name" type="xsd:string"/>
 7        <xsd:element name="Geburtsdatum" type="xsd:date"/>
 8      </xsd:sequence>
 9    </xsd:complexType>
10    <xsd:complexType name="MitarbeiterTyp">
11      <xsd:complexContent>
12        <xsd:extension base="PersonTyp">
13          <xsd:sequence>
14            <xsd:element name="Gehalt">
15              <xsd:simpleType>
16                <xsd:restriction base="xsd:float">
17                  <xsd:minInclusive value="10000"/>
18                </xsd:restriction>
19              </xsd:simpleType>
20            </xsd:element>
21          </xsd:sequence>
22        </xsd:extension>
23      </xsd:complexContent>
24    </xsd:complexType>
25    <xsd:element name="Unternehmen">
26      <xsd:complexType>
27        <xsd:sequence>
28          <xsd:element name="Mitarbeiter" type="MitarbeiterTyp"
29            maxOccurs="unbounded"/>
30        </xsd:sequence>
31      </xsd:complexType>
32    </xsd:element>
33  </xsd:schema>
```

Abb. 1. Beispiel für ein XML-Schema

Verfügbare Produkte

Das W3C hat neben der Spezifikation [14] auch eine Einführung in XML-Schema mit zahlreichen Beispielen veröffentlicht [7]. Es gibt bereits ausgereifte XML-Schema-Editoren (beispielsweise *XML Spy Suite* [1] und *Turbo XML* [13]), welche die Erstellung von XML-Schemata durch geeignete Benutzeroberflächen stark vereinfachen und die syntaktische Korrektheit der erstellten XML-Schemata sicherstellen. Die Schemagültigkeit kann aber auch mittels frei verfügbarer Validatoren kontrolliert werden. Soll dies innerhalb selbst entwickelter Programme erfolgen, übernehmen validierende Parser (z. B. *Apache Xerces* [2]) diese Aufgabe. Eine Produktübersicht für XML-Schema ist unter http://www.wifo.uni-mannheim.de/xml-schema/ zu finden.

Die Verbreitung von XML-Schema nimmt ständig zu. Schema-Repositorys, deren wichtigste Vertreter *XML.org* [15] und *Microsoft BizTalk* [8] sind, enthalten wiederverwendbare XML-Schemata für bestimmte Anwendungsbereiche. Auch die Initiative *ebXML* [4] benutzt XML-Schemata zur Definition ihrer Standards.

Ein interessanter Aspekt im Zusammenhang mit der objektorientierten Modellierung von Systemen ist, dass sich jedes XML-Schema (und sogar die komplette Syntax von XML-Schema) als UML-Klassendiagramm darstellen lässt. CASE-Tools, welche die *XML-Metadata-Interchange*-Spezifikation

```
<?xml version="1.0" encoding="UTF-8"?>
<Unternehmen xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Beispiel.xsd">
  <Mitarbeiter>
    <Vorname>Hagen</Vorname>
    <Vorname>Dieter</Vorname>
    <Name>Becker</Name>
    <Geburtsdatum>1967-08-13</Geburtsdatum>
    <Gehalt>39512.34</Gehalt>
  </Mitarbeiter>
  <Mitarbeiter>
    <Vorname>Jutta</Vorname>
    <Name>Stehl</Name>
    <Geburtsdatum>1976-09-11</Geburtsdatum>
    <Gehalt>42500.34</Gehalt>
  </Mitarbeiter>
</Unternehmen>
```

Abb. 2. Beispiel für ein zum Schema in Abb. 1 passendes XML-Dokument

```
Unternehmen unternehmen
    = Unternehmen.unmarshal(new FileReader ("Beispiel.xml"));
Enumeration mitarbeiter
    = unternehmen.enumerateMitarbeiter(); // leider keine Iteratoren
while (mitarbeiter.hasMoreElements()) {
  Mitarbeiter m = (Mitarbeiter) mitarbeiter.nextElement();
  for (int i = 0; i < m.getVornameCount(); ++i)
    out.print(m.getVorname(i) + " ");
  String name = m.getName();
  Date geburtsdatum = m.getGeburtsdatum().toDate();
  float gehalt = m.getGehalt();
  out.println(name + ", "
    + DateFormat.getDateInstance().format(geburtsdatum) + ", "
    + gehalt);
}
// ... Daten modifizieren, z. B.
// unternehmen.getMitarbeiter(0).setGehalt(48000);
if (unternehmen.isValid())
  unternehmen.marshal(new FileWriter("Beispiel2.xml"));
```

Abb. 3. Java-Programmfragment zum XML-Data-Binding mit Castor

(XMI [9]) unterstützen, können automatisch aus einem UML-Klassendiagramm das zugehörige XML-Schema erzeugen. Und auch der umgekehrte Vorgang – das Reverse Engineering – ist möglich.

Objektorientierte Softwareentwicklung

Auch für die objektorientierte Softwareentwicklung bietet XML-Schema interessante Möglichkeiten, wobei speziell die Programmiersprache Java in Betracht kommt. Beispielsweise besitzen viele der in XML-Schema vordefinierten Datentypen direkte Entsprechungen in Java [3]. Darüber hinaus sind sowohl XML als auch Java plattformunabhängig konzipiert.

Der wichtigste Einsatzbereich von XML-Schema bei der objektorientierten Softwareentwicklung

ist derzeit das *XML-Data-Binding*. Dabei werden XML-Schemata durch einen Schema-Compiler automatisch in die Klassen einer objektorientierten Programmiersprache übersetzt. Im Gegensatz dazu muss bei Verwendung einer DTD in einem manuell zu erstellenden XML-Dokument, dem sog. Binding-Schema, die mangelnde Ausdrucksfähigkeit der DTD aufwändig ausgeglichen werden.

Ein XML-Data-Binding-Framework stellt Methoden zum Zugriff und zur Manipulation der Attribut- und Elementwerte eines schemagültigen XML-Dokuments zur Verfügung und nimmt deren Validierung vor. Dieses Verfahren ist effizienter und einfacher als der Einsatz von *SAX* oder *DOM* [11]. Derzeit sind zahlreiche XML-Data-Binding-Frameworks in Entwicklung, dazu gehören insbesondere die Open-Source-Projekte *Castor* [6] und *Zeus* [5] sowie von Sun das *Java API for XML Processing* (JAXP) ab Version 1.2 [10].

In Abb. 3 ist ein Ausschnitt aus einer Methode gezeigt, die alle `Mitarbeiter`-Elemente sowie Subelemente aus einem schemagültigen XML-Dokument ausliest und anzeigt. Mit geeigneten Methoden wird der Inhalt der einzelnen `Mitarbeiter`-Elemente ermittelt. Beim Element `Vorname` ist dabei zu beachten, dass es mehrfach auftreten kann. Auf ebenso einfache Weise können Elemente mittels Methodenaufruf geändert werden.

Ein XML-Data-Binding-Framework übernimmt die Validierungsregeln aus dem XML-Schema und generiert geeigneten Programmcode zur Validierung der zu bearbeitenden Daten. In eigenem Programmcode können entsprechende Prüfungen deshalb entfallen. Dadurch vereinfacht sich die Programmentwicklung, und die Programme sind später leichter zu warten. Im Beispiel wird das Gehalt eines Angestellten modifiziert und anschließend mit `isValid` geprüft, ob die neuen Daten zum Schema passen (s. Abb. 3). Dabei müssen die Daten vor der Validierung nicht erst in einem XML-Dokument gespeichert werden.

Fazit

Aufgrund der Standardisierung durch das W3C nimmt der Einsatz von XML-Schema auf breiter Front zu. Dokumentiert wird dieser Trend durch bereits verfügbare sowie angekündigte Tools. Wie gezeigt, existieren für XML-Schema neben der eigentlichen Anwendung als Beschreibungssprache für XML-Dokumente auch interessante Einsatzgebiete bei der objektorientierten Softwareentwicklung. Dies lässt erwarten, dass XML und XML-Schema auch Eingang in zukünftige Programmierparadigmen finden werden.

Literatur

1. Altova GmbH: XML Spy Suite. http://www.xmlspy.com/
2. Apache XML Project: Xerces – XML Parsers in Java and C++. http://xml.apache.org/
3. Biron, P. V., Malhotra, A.: XML Schema Part 2 – Datatypes. World Wide Web Consortium, Recommendation. http://www.w3.org/TR/xmlschema-2/ (2001)
4. ebXML.org: Electronic Business XML (ebXML). http://www.ebxml.org/
5. Enhydra.org: Zeus. http://zeus.enhydra.org/index.html
6. ExoLab Group: Castor Project. http://castor.exolab.org/
7. Fallside, C.: XML Schema Part 0 – Primer. World Wide Web Consortium, Recommendation. http://www.w3.org/TR/xmlschema-0/ (2001)
8. Microsoft: BizTalk. http://www.biztalk.org/
9. OMG: XMI Production for XML Schema Specification. ftp://ftp.omg.org/pub/docs/ptc/01–12–03.pdf (2001)
10. Sun Microsystems: Java API for XML Processing (JAXP). http://java.sun.com/xml/jaxp/ (2002)
11. Sun Microsystems: Java Architecture for XML Binding (JAXB) – Frequently Asked Questions. http://java.sun.com/xml/jaxb/faq.html (2001)
12. Thompson, H. S., Beech, D., Maloney, M., Mendelsohn, N.: XML Schema Part 1 – Structures. World Wide Web Consortium, Recommendation. http://www.w3.org/TR/xmlschema-1/ (2001)
13. TIBCO Software Inc.: TIBCO TurboXML. http://www.tibco.com/
14. W3C: XML Schema. http://www.w3.org/XML/Schema
15. XML.org: http://www.xml.org/