Grundlagen XML

Andreas Rottmann, Sebastian Riedl

27. März 2002

Inhaltsverzeichnis

1 Überblick

Im folgenden wollen wir Ihnen die Grundlagen von XML, der *Extensible Markup Language* näherbringen.

Dieses Dokument gliedert sich in einen kurzen Einblick über die Entstehungsgeschichte und die Hintergründe von XML, den Aufbau von XML Dokumenten, auch im Vergleich zu anderen Markup-Sprachen, und die Definition ihres Aufbaus (mit der *Document Type Definition* und den *Schematas*) sowie ihre Darstellung mit *XSL/XSLT*.

Für weitergehende Informationen siehe auch [1] und [2].

2 Hintergründe und Geschichte

Nachdem XML sich zum Schlagwort in der IT-Branche gemausert hat, stellt sich natürlich die Frage „Wieso eine neue Markup-Sprache?“.

Die Antwort darauf liegt im wesentlichen in der Schwächen von HTML bei der Beschreibung allgemeiner Daten (HTML eignet sich im wesentlichen nur zur platformübergreifenden Beschreibung des Layouts nicht aber Struktur der dargestellten Daten). Um dies zu illustrieren betrachten wir folgendes einfaches Beispiel einer HTML-Datei:

```
<html>
<head>
  <title>Addresses</title>
</head>
<body>
  <h1>Addresses</h1>
  <b>Frank Muster</b>
  <p>Fakestreet 1
  <p>1010 Silicon Valley
  <p>Telephone: 0111-12345678
</body>
</html>
```

Obwohl für einen menschlichen Betrachter obigen HTML-Dokuments sofort einsichtig ist das es sich um Addressdaten handelt, die aus verschiedenen Elementen (Name, Strasse, ...) aufgebaut sind, kann eine Maschine nur sehr schwer diese einzelnen Elemente extrahieren.

Dies ist zwar der wichtigste, aber nicht der einzige Grund für den Bedarf an einer neuen Markup-Sprache.

Als Vorläufer von XML wenden wir uns kurz SGML, der *Standard Generalized Markup Language*, zu. Mit Hilfe von SGML lassen sich andere Markup-Sprachen erzeugen (z.B. wurde HTML mit SGML definiert). Da SGML sehr komplex und für den praktischen Einsatz zu unhandlich ist, schuf das W3C (*World Wide Web Consortium*) 1998 XML als Teilmenge von SGML. Mit XML lassen sich eigene Tags (z.B. `<address>`) viel einfacher definieren.

Zu den Entwurfszielen von XML nennt das W3C [3] folgende 10 Punkte:

1. XML soll sich im Internet auf einfache Weise nutzen lassen.
2. XML soll ein breites Spektrum von Anwendungen unterstützen.
3. XML soll zu SGML kompatibel sein.
4. Es soll einfach sein, Programme zu schreiben, die XML-Dokumente verarbeiten.
5. Die Zahl optionaler Merkmale in XML soll minimal sein, idealerweise Null.
6. XML-Dokumente sollten für Menschen lesbar und angemessen verständlich sein.
7. Der XML-Entwurf sollte zügig abgefasst sein.
8. Der Entwurf von XML soll formal und präzise sein.
9. XML-Dokumente sollen leicht zu erstellen sein.
10. Knappheit von XML-Markup ist von minimaler Bedeutung.

3 Aufbau von XML-Dokumenten

Ein XML-Dokument besteht aus einzelnen Elementen, jedes Element wiederum aus einem Start-Tag, dem Inhalt und einem Ende-Tag, ähnlich wie in HTML.

3.1 Konzepte

Am besten illustriert die Konzepte von XML an einem kleinen Beispiel; hier eine mögliche XML-Darstellung der obigen Adresse:

```
<?xml version="1.0"?>

<!-- Dies ist ein Kommentar --!>

<addresses>
  <address>
    <name>Frank Muster</name>
    <street>Fakestreet 1</street>
    <zip>1010</zip>
    <city>Silicon Valley</city>
    <telephone number="0111-12345678" />
  </address>
</addresses>
```

Dieses Besipiel dokumentiert bereits viele XML-Konzepte, etwa die Verschachtelung von Tags (`<addresses>` enthält `<address>`), im Beispiel durch Einrückung hervorgehoben. Wichtig ist das ein XML-Dokument ein Wurzelelement enthält, das alle anderen Elemente einschliesst (in diesem Beispiel `<addresses>`). Somit kann das Dokument als Baum angesehen werden. Ausserdem ist XML im Gegensatz zu HTML *case-sensitive*.

Betrachtet man die Elemente im Beispiel, so fällt auf, das es mit der ersten Zeile und dem `telephone` Element zwei Elemente enthält, die sich von allen anderen unterscheiden.

Die erste Zeile in einem XML-Dokument muss immer die Anweisung
`<?xml version="#version" ?>` enthalten, die die festlegt, dass es sich bei dem Dokument um eine XML-Datei nach den Spezifikationen der Version `#version` handelt. So kann ein Anwenderprogramm sofort feststellen, ob es das Dokument überhaupt verarbeiten kann.

Beim `telephone`-Element kommen zwei neue Prinzipen zum Einsatz. Es handelt sich um ein „leeres“ Element, das ein Attribut, nämlich `number` enthält. Mit solchen lassen sich etwa Zustände repräsentieren, etwa bei einer Person `<ist-pensioniert/>`. Das ließe sich allerdings auch als Attribut eines `person`-Tags ausdrücken: `<person beruf="pensioniert">`.

3.2 Details

Nun stellt sich natürlich die Frage, warum wir im obigen Beispiel einfach Tags wie `<address>` oder `<telephone>` verwenden können.

In einem HTML-Dokument werden die zulässigen Tags durch den HTML-Standard festgelegt, in einem XML-Dokument durch die *Document Type Definition* (DTD) oder *Schemata*.

3.2.1 Die Document Type Definition

Die für ein XML-Dokument verwendete DTD gibt man nach der Zeile `<?xml ... ?>` durch eine Zeile der Form:

```
<!DOCTYPE Name-des-Wurzel-Elements SYSTEM "Name-der-DTD-Datei">
```

an, falls die DTD sich in einer externen Datei befindet, oder mit

```
<!DOCTYPE Name-des-Wurzel-Elements [
  <!-- Hier stehen die Definitionen --!>
]>
```

direkt im Dokument.

Eine DTD zum Adress-Beispiel von oben könnte so aussehen:

```
<!DOCTYPE addresses [
  <!ELEMENT addresses (address*)>
```

```
  <!ELEMENT address (name, street, zip, city, telephone?)>
  <!ELEMENT name #PCDATA>
  <!ELEMENT street #PCDATA>
  <!ELEMENT zip #PCDATA>
  <!ELEMENT city #PCDATA>
  <!ELEMENT telephone EMPTY>
    <!ATTLIST telephone number CDATA #REQUIRED>
]>
```

Jedes Element, das im XML-Dokument auftreten darf, wird durch einen `<!ELEMENT ...>` Tag deklariert. Anstatt der drei Punkte wird als erstes der Name des Elements, dann der Inhalt angeführt.

Für den Inhalt gibt es mehrere Möglichkeiten:

- `EMPTY`: Dies definiert ein leeres Element.
- `ANY`: Das Element kann beliebigen Inhalt haben.
- Eine Liste oder Wahlmöglichkeit zwischen den Kindelementen des Elements, wobei diese Struktur geschachtelt sein kann und auch `#PCDATA` (*Parsed Character Data*, Daten, die keine Elemente sind) Einträge enthalten kann.

  Beispielsweise bedeutet `<!ELEMENT div1 (head, (p | list | note)*, div2+)>` „der Tag `div1` enthält ein Element vom Typ `head`, dann eine möglicherweise leere Liste von beliebig vielen Elementen vom Typ `p`, `list` oder `note` (wahlweise), darauf folgt eine Liste von Elementen vom Typ `div2` mit mindestens einem Element“.

Weiters können zu jedem Element Attribute mit `<!ATTLIST ...>` definiert werden. Anstatt der drei Punkte wird als erstes der Name des Elements, dessen Attributliste definiert wird, angeführt, dann eine Liste der Attribute, wobei für jedes sein Name, sein Typ und eine „*default declaration*“ angegeben wird.

Der Typ beschreibt den Zulässigen Wertebereich für das Attribut. Die wichtigsten Typen sind `CDATA` (beliebige Zeichenkette) und die geklammerte Aufzählung von zulässigen Werten.

Die *default declaration* ist entweder `#REQUIRED` (für das Attribut muss ein Wert angegeben werden), `#IMPLIED` (das Attribut hat keinen Default-Wert) oder der Default-Wert, mit optional vorangestelltem `#FIXED` (das Attribut muss als Wert den Default-Wert haben).

3.2.2 Schemata

Bei der Betrachtung der DTD zu obigem Adress-Beispiel fällt auf, dass als Typ für das `number`-Attribut des `telephone`-Elements `CDATA` angegeben ist, also die DTD keine genaueren Angaben zum Aufbau einer Telefonnummer macht. Nach dieser DTD ist also auch `"abcde"` eine gültige Telefonnummer.

Da aber für viele Anwendungen eine strengere Typisierung benötigt wird, existiert eine Alternative zur DTD: Schemata. Dieser Standard wurde vom W3C erst Mitte 2001 verabschiedet.

Da eine Beschreibung der XML-Schemata über den Rahmen dieses Dokuments sprengen würde, verweisen wir für näheres auf [4].

3.2.3 Erweiterungen

Da im Laufe der Zeit etliche Mängel von XML erkannt wurden, hat man, um Abhilfe zu schaffen, unter anderem folgende Erweiterungen zu XML 1.0 entwickelt:

XPath: Erweitert XML um Konzepte zum Verweis auf Elemente oder Mengen von Elementen eines XML-Dokuments.

XPointer: Erweitert *XPath* um die Möglichkeit auf allgemeine Teilbereiche von XML-Dokumenten (etwa Teile von Elementen) zu verweisen.

XLink: Ähnlich wie die Links aus HTML, bietet aber darüber hinausgehende Möglichkeiten.

XSL: XSL (*Extensible Stylesheet Language*) wurde ürsprünglich als XML-basierte Sprache entworfen um folgende zwei Probleme zu lösen:

1. Die Transformation von XML-Dokumenten in eine andere Form.
2. Zur Darstellung von XML-Dokumenten auf *page-oriented devices*, wie Browsern oder Druckern.

Aufgrund von Schwierigkeiten bei der Lösung des zweiten Problems wurde der XSL vom W3C in zwei Sub-Standards aufgeteilt, *XSL Transformations* (XSLT) und *XSL Formatting Objects* (XSL-FO). XSLT wurde bereits als Empfehlung vom W3C verabschiedet, XSL-FO befindet sich noch in Entwicklung.

4 XML APIs

Im folgenden stellen wir kurz die zwei wichtigsten APIs für die Programmierung von XML-Anwendungen vor.

4.1 SAX

SAX (*Simple API for XML*) war anfangs ein rein Java-basiertes API. Als erstes weit verbreitetes XML-API für Java hat es sich zu einem „de facto“ Standard entweickelt. Inzwischen ist SAX auch für etliche andere Sprachen, unter anderem *C++*, *Python* und *Perl* verfügbar.

SAX ist ein ereignisorientiertes API. Der Parser durchläuft das Dokument und ruft für jedes Ereignis (z.B. Beginn oder Ende eines Elements) den dafür von der Anwendung registrierten Callback auf. Damit ist die Programmierung mit SAX ähnlich wie die Programmierung von GUIs, die meist auch ereignisorientert arbeiten.

4.2 DOM

DOM (*Document Object Model*) liest im Gegensatz zu SAX das ganze Dokument ein und überführt es in eine Baumstruktur, die von der Anwendung über die von DOM bereitgestellten Schnittstellen bearbeitet werden kann.

Für sehr grosse Dokumente jedoch ist der Einsatz von SAX gegenüber DOM vorzuziehen, da der Spiecheraufwand von DOM durch den Aufbau einer Baumstruktur zu gross ist.

5 Fazit

XML bietet eine flexible und zeitgemässe Möglichkeit in organisierter Form Daten zu speichern und auszutauschen. Dementsprechend wurd XML als Basis für etliche Anwendungen bzw. Datenformate verwendet.

Im folgenden eine kleine Auswahl:

SOAP (*Simple Object Access Protocol*): Ein XML-basiertes Protokoll, das Anwendungskommunikation über Netzwerke ermöglicht.

XML-RPC (*XML Remote Procedure Calls*): RPCs werden mit mit Hilfe von XML-Dokumenten über HTTP abgewickelt.

SVG (*Scalable Vector Graphics*): Eine Sprache zur Beschreibung von Vektorgraphiken.

SMIL (*Synchronized Multimedia Integration Language*): XML-basierte Sprache für Multimedia-Inhalte.

XHTML: HTML 4.0 in XML redefiniert; hat im Unterschied zu HTML 4.0 etwas strengere Regeln, so ist XHTML *case-sensitive* und lässt keine Start- ohne Ende-Tags zu.

WML: (*Wireless Markup Language*). Das HTML für WAP (*Wireless Application Protocol*).

VoiceXML: Dient zur Entwicklung interaktiver Sprachdienste.

Abbildung 1 bietet nochmals einen abschliessenden Überblick von XML.

A GNUPlot Grafik

Siehe Abb. 2.

Literatur

[1] Tim Schürmann. Sprachen selbst gemixt. *Linux Magazin*, pages 42–46, 1 2002.

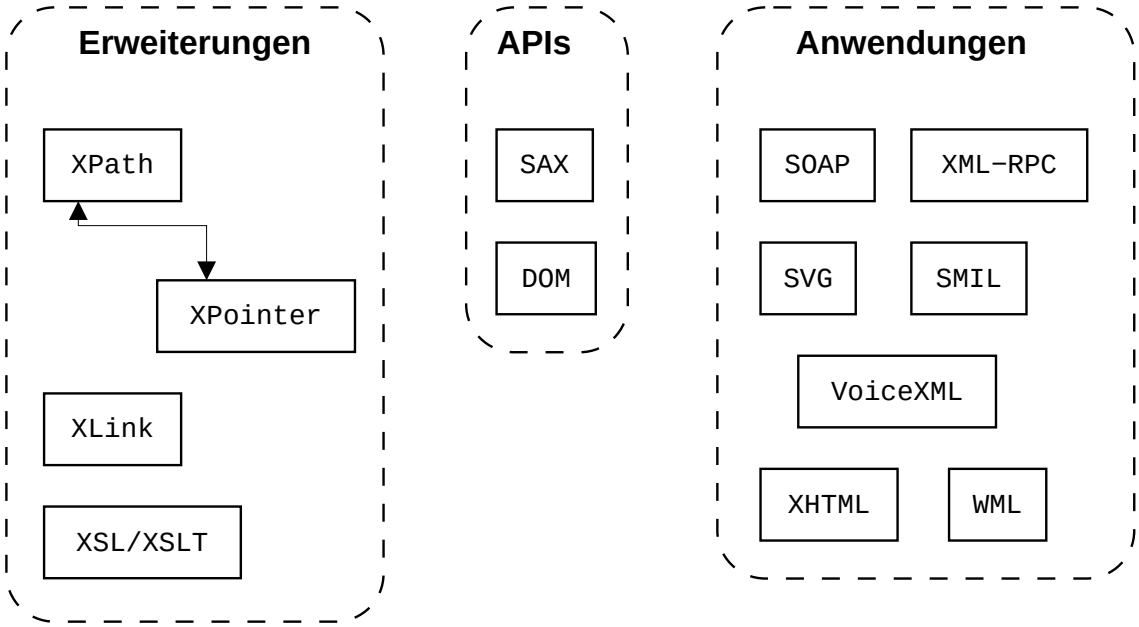

Abbildung 1: XML im Überblick

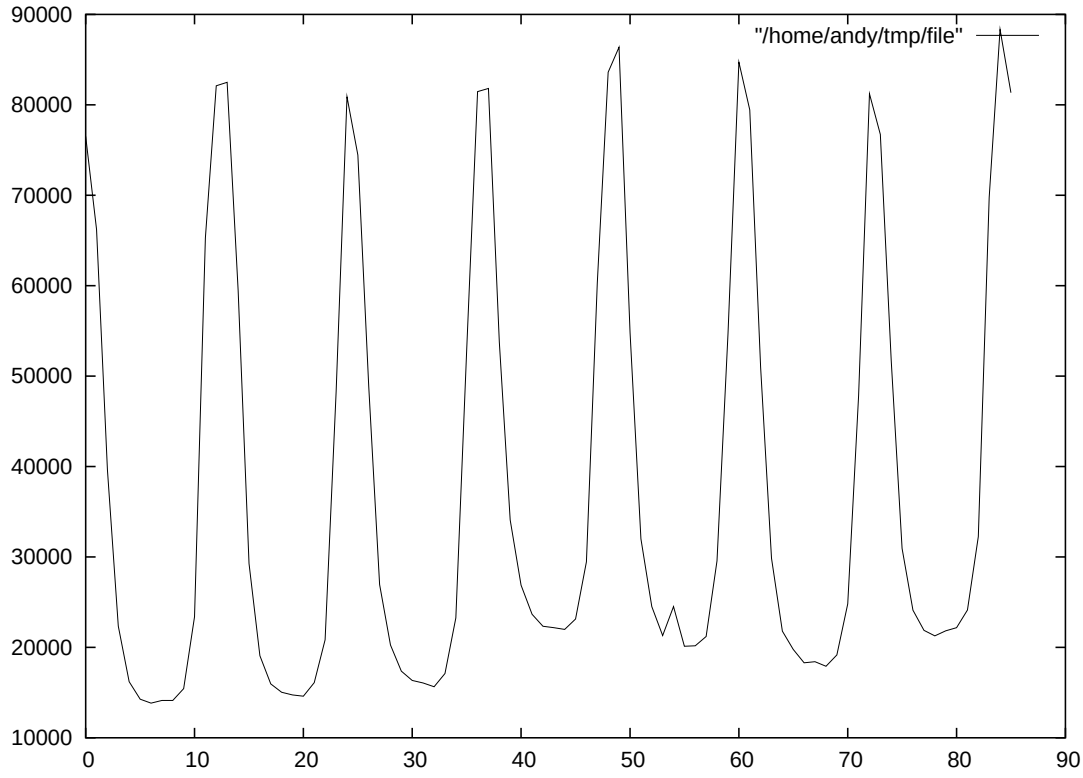

Abbildung 2: GNU-Plot

[2] Tim Schürmann. XML-Grundlagen Teil 2. *Linux Magazin*, 2 2002.

[3] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Second Edition), `http://www.w3.org/TR/REC-xml`.

[4] World Wide Web Consortium. XML Schema, `http://www.w3.org/XML/Schema`.