XML

Michael Dienert

20. Oktober 2010

Inhaltsverzeichnis

1 Auszeichnungssprachen

"Auszeichnungssprache" ist die Übersetzung von "Markup Language". Der Begriff *markup* stammt aus dem Handel und bedeutet dort einfach *Preiserhöhung*.

Dieser Begriff wird nun auch auf *Textateien* angewendet. Ursprünglich enthielt der *markup* in einer Textdatei spezielle Codes, mit denen der Ausdruck des Texts auf einem Drucker feingesteuert wurde (Hoch-, Tiefstellen, fett, kursiv, ...).

Die dabei verwendeten Markups waren natürlich sehr speziell und verschiedene Unternehmen und Institutionen konnten damit strukturierte Dokumente nicht untereinander austauschen.

Die Idee, den markup zu Standardisieren kam deshalb schon sehr früh in den 60er (!) Jahren auf. Der damals entwickelte Standard hiess *GenCode*. Damit erstellte Dateien konnten auf unterschiedlichen Druckereimaschinen gesetzt werden. GenCode wurde von der *Graphic Communications Association* entwickelt.

Bei grossen Unternehmen bereitete der fehlende markup-Standard auch den Abteilungen innerhalb einer Firma Probleme, so dass IBM in den 70er Jahren den Haus-Standard *GML*, *Generalized Markup Language*, entwickelte.

GML verwendet bereits die bekannte **Tag**-Syntax mit den spitzen Klammern und wurde speziell für die Arbeit von Schreibkräften an einer Konsole optimiert, d.h. es besitzt ein einfaches Eingabeformat.

Dabei bedeutet *tag* etwa so viel wie "Preisschild", womit wir wieder bei der Preisauszeichnung (einer Ware) wären.

Nun hatte man zwei Standards und in den 80er Jahren wurde GML und GenCode zu **SGML** (Standardized GML) verschmolzen und 1986 als ISO-Standard eingeführt.

SGML ist ein sehr komplexer Standard, aber dennoch wichtig, da HTML eine SGML Anwendung ist.

1.1 HTML, XML und XHTML

html wurde ursprünnglich von Tim Berners-Lee eingeführt und war bewusst einfach gehalten, damit es jeder und jede an einem Nachmittag erlernen kann.

Diese Einfachheit macht html natürlich auch begrenzt und es ist damit nicht in der Lage, alle denkbaren Dokumente zu beschreiben.

Aus diesem Grund hat Jon Bosak 1997 die Sprache **XML** vorgestellt: e**X**tensible **M**arkup **L**anguage. Man kann XML als Kompromiss zwischen dem unmöglich komplexen SGML und dem zu beschränkten HTML betrachten. Da die Entwickler von XML mit SGML sehr vertraut waren, ist XML eine *reine Untermenge* von SGML.

Da XML erweiterbar ist, entfällt die Beschränktheit von html. Da XML-Dokumente nur wenige Regeln erfüllen müssen, ist es auch nicht unnötig komplex und leicht erlernbar.

1.2 Der Unterschied zwischen html und xml

XML und html sehen auf den ersten Blick sehr ähnlich aus. Es gibt jedoch einen entscheidenden Unterschied:

XML dient dazu, Daten zu **beschreiben**. html wurde von Sir Tim Berners-Lee entwickelt, um Daten **darzustellen**.

2 Ein erstes XML-Beispiel

Hier ein erstes XML-Beispiel:

```
<? xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!-- +++++++++++++++++ hier beginnt der Prolog +++++++++++++++ -->
<!DOCTYPE spielplan [
   <!ELEMENT spielplan (eintrag)*>
   <!ELEMENT eintrag (titel, regie, darsteller*, jahr)>
   <!ELEMENT titel (#PCDATA)>
   <!ELEMENT regie  (#PCDATA)>
   <!ELEMENT darsteller (#PCDATA)>
   <!ELEMENT jahr (#PCDATA)>
]>
<!-- -->
<!-- ++++++++++ hier beginnt der wohlgeformte haupteil ++++++++ -->
<spielplan>
   <eintrag>
      <titel>Müllers Büro</titel>
      <regie>Niki List</regie>
      <darsteller>Barbara Rudnik</darsteller>
      <darsteller>Christian Schmidt</darsteller>
      <darsteller>Andreas Vitasek</darsteller>
      <jahr>1987</jahr>
   </eintrag>
</spielplan>
```

3 Die XML-Regeln

- Im Unterschied zu html wird in XML-Dokumenten *streng* zwischen Gross- und Kleinschreibung unterschieden. Zum Beispiel wären also die Tags `<passwd>` und `<Passwd>` in einem XML-Dokument verschieden. Damit das nicht zu einem völligen Chaos führt, gilt die Empfehlung: **Alles kleinschreiben!**

- Ein XML-Dokument besteht aus *Elementen*. Ein gültiges XML-Dokument muss mindestens 1 Element besitzen. Ein Element besteht aus folgenden Teilen:

  - Der Elementname. Dieser **muss** angegeben werden und er steht zwischen einem Paar spitzer Klammern. Diese Einheit aus Klammern und Elementnamen bezeichnet man als **Tag**. Zu jedem Start-Tag muss es wiederum ein End-Tag geben, bei dem vor dem Elementnamen ein Schrägstrich steht. Ein Beispiel hierzu:

    ```
    <br></br>
    ```

    Es gibt auch eine abgekürzte Schreibweise, bei der das Start- und Endtag verschmelzen:

    ```
    <br/>
    ```

  - Das Element kann, muss aber keine Attribute enthalten. Diese stehen nach dem Elementnamen im Start-Tag. Ein Attribut besteht aus Attributsnamen und Attributswert. Zwischen Namen und Wert muss ein = Zeichen stehen, der Attributswert **muss** in einem Paar einfacher oder doppelter Anführungszeichen stehen. Beispiel:

    ```
    <appendComment author="micha"/>
    ```

  - Natürlich kann das Element einen Inhalt haben. Der Inhalt kann Text oder ein weiteres Element sein:

    ```
    <attribute>
       <name>path</name>
       <required>true</required>
       <rtexprvalue>true</rtexprvalue>
    </attribute>
    ```

    `name`,`required` und `rtexprvalue` werden in diesem Fall als **Kinder** des Elements `attribute` bezeichnet.

  - Schliesslich kann es einen Namensraum geben. Dieser steht vor dem Elementnamen im Start-Tag:

    ```
    <core:redirect url="./tabelle.jsp"/>
    ```

    Der Namensraumpräfix muss aber zuvor deklariert werden:

    ```
    <order xmlns:xi="http://www.w3.org/2001/XInclude">
      <xi:include href="order_details.xml"/>
    </order>
    ```

    Der Namensraum gilt nun nur innerhalb des Elements, in dem er definiert wurde. Damit gilt er natürlich auch für die Kinder dieses Elements. Namensräume gibt es erst in der XLM-Version 1.1.

Ganz wichtig:

Ein Element ist die ganze Stulle , bestehend aus Brot, Butter, Wurst, Gurken usw. Die Tags sind dabei nur die Brotscheiben!

- Neben Elementen darf das Dokument natürlich auch Kommentare enthalten:

```
<!-- der Kommentartext -->
```

- Dann gibt es noch die XML-Deklaration mit Version und Kodierung:

```
<? xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
```

  Lässt man das Encoding weg, wird UTF-8 als Standardwert angenommen. `standalone="yes"` heisst, dass dieses Dokument seine eigene DTD enthält. Was das ist, wird weiter unten beschrieben.

- Streng genommen ist die XML-Deklaration ein Spezialfall der sog. *Verabeitungsbefehle* (Processing Instructions) die in das Paar `<?  ?>` eingeschlossen werden. Hier wieder ein Beispiel:

```
<?PRINT clear page?>
```

- Zum Abschluss der XML-Bestandteile muss man hier noch die Dokumententyp-Deklaration (Document Type Declaration)erwähnen. Diese Deklaration enthält die *Grammatik*, also die Beschreibung wie das Dokument aufgebaut sein muss. Die Grammatik wird *Document Type Definition*, **DTD**, genannt. Die DTD kann im Dokument oder in einer externen Quelle stehen. Ist die gesamtte DTD intern vorhanden, spricht man von einem *eigenständigen Dokument* (`standalone="yes"`, siehe oben). Eine genauere Beschreibung von DTDs folgt im nächsten Kapitel.

- Nun die wichtigste XML-Regel: XML-Dokumente müssen *wohlgeformt* sein. Das heisst:
  - Alle Tags müssen *ausgegelichen* sein.
  - Die Tags müssen korrekt verschachtelt sein. D.h. das End-Tag eines Kindselements muss vor dem End-Tag des Elternelements stehen.
  - Es muss ein sog. *Wurzelelement* geben. Alle anderen Elemente müssen Kinder dieses Wurzelelements sei.

4 Die Dokumenttyp-Definition DTD

Ein wohlgeformtes Dokument wird zu einem **gültigen Dokument**, wenn es einer Dokumenttyp-Definition (Document Type Definition) DTD entspricht.

Unabhängig vom Speicherort - also bei eigenständigen Dokumenten innerhalb der XML-Datei oder als externe Datei - wird mit der DTD der exakte Aufbau und die Struktur des Dokuments festgelegt.

Hier nocheinmal die DTD aus dem Beispiel aus Kap. 2, diesmal als externe Datei:

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
   <!ELEMENT spielplan (eintrag)*>
   <!ELEMENT eintrag (titel, regie, darsteller*, jahr)>
   <!ELEMENT titel (#PCDATA)>
   <!ELEMENT regie  (#PCDATA)>
   <!ELEMENT darsteller (#PCDATA)>
   <!ELEMENT jahr (#PCDATA)>
```

Diese DTD kann im eigentlichen Dokument mit folgender Zeile eingebunden werden:

```
<!DOCTYPE spielplan SYSTEM "spielplan.dtd" [ ... ] >
```

Hier die wichtigsten Teile einer DTD:

ELEMENT :

```
<!ELEMENT eintrag (titel, regie, darsteller*,jahr)>
```

hiermit wird festgelegt, dass auf das Element `eintrag` die Elemente `titel, regie, darsteller` und `jahr` folgen **müssen**. Dabei darf man Kardinalitäten angeben:

? : das Element kann keinmal oder einmal auftreten (0,1)

+ : das Element kann einmal oder mehrmals auftreten (1,n)

***** : das Element kann keinmal, einmal oder mehrmals auftreten (0,1,n)

| : damit werden innerhalb einer Liste Alternativen angegeben

, : die Elemente müssen in der angegebenen Reihenfolge erscheinen

PCDATA: Parsed Character Data, also Text.

ATTLIST : hiermit werden die Atrribute eines Elements aufgelistet. Ein Beispiel:

```
<!ELEMENT passwd (#PCDATA)>
<!ATTLIST passwd
 encrypttype CDATA #REQUIRED
 expdate     CDATA #IMPLIED>
```