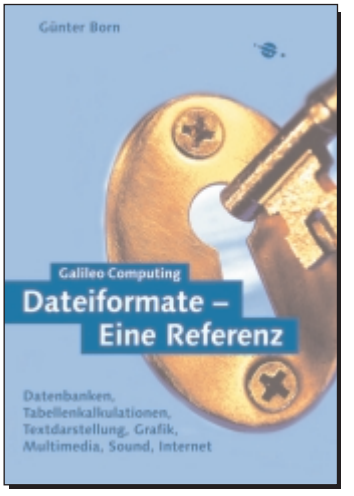
Dieses Kapitel stammt aus dem Buch

›Dateiformate – Eine Referenz‹

von Günter Born.

www.borncity.de



ISBN 3-934358-83-7
119,90 DM

Informationen zum Buch
mit Bestellmöglichkeit

www.galileocomputing.de

Galileo Computing

51 Extended Markup Language (XML)

XML ist eine Anwendung von SGML und stellt eine Sprache zur Beschreibung von Daten dar. Einerseits läßt sich XML nutzen, um HMTL-Dokumente mit Daten anzureichern. Andererseits erlaubt XML die transparente Beschreibung von Daten, die zwischen Anwendungen ausgetauscht werden. XML könnte sich daher als universelles Dateiformat erweisen. So benutzt Microsoft ab Office 2000 neben den nativen Dateiformaten XML zur Speicherung der Dokumente. Sobald Sie ein Dokument als HTML-Datei hinterlegen, werden XML-Elemente zur Abbildung der Daten eingebettet. Anhand dieser XML-Elemente läßt sich der Dokumentinhalt später in der Office-Anwendung lesen.

XML-Dateistruktur

XML-Dokumente werden als reine Textdateien (basierend auf dem Unicode-Zeichensatz) gespeichert, d.h., die Dateien lassen sich mit jedem Texteditor öffnen und bearbeiten. Wichtig ist, daß bei XML-Dokumentdateien Groß-/Kleinschreibung bei den Schlüsselwörtern und Anweisungen zu beachten ist. Die Dokumentdatei besteht dabei aus einer Kopfzeile (Prolog), der Document Type Definition (DTD) sowie aus den eigentlichen Daten (umgangssprachlich: dem Dokumentinhalt), in welche die Anweisungen (als Markup bezeichnet) zur Beschreibung der Dokumentdaten eingebettet sind. Ein einfaches XML-Dokument könnte folgendermaßen aussehen.

```
<?xml version="1.0"?>
<!DOCTYPE meintext system "text.dtd">
<person>
<name>Meier</name>
<vorname>Hans</vorname>
<name>Mueller</name>
<vorname>Armin</vorname>
</person>
```

Abbildung 51.1 Beispiel einer XML-Datei

Die XML-Datei beginnt dabei immer mit der Zeile *<?xml version="1.0"?>*. Diese Zeile weist darauf hin, daß es sich um ein XML-Dokument gemäß der Spezifikation 1.0 handelt.

Der Prolog mit der Processing Instruction

Jedes XML-Dokument muß mit einem einzeiligen Prolog beginnen. Diese *<?xml ...?>*-Anweisung wird auch als XML-Processing Instruction (PI) bezeichnet. Die Anweisung gibt dem lesenden Programm Hinweise, was beim Lesen zu beachten ist. Neben der Ver-

sionsangabe über das Attribut *version* können Sie zusätzlich den in der Dokumentdatei vereinbarten Zeichensatz hinterlegen. Die Kodierung der Zeichen in Dokumenten erfolgt gemäß verschiedenen ISO-Normen. Die ISO-8859-Norm legt in zehn Untergliederungen die Zeichen verschiedener Sprachen fest. Zusätzlich gibt es noch die ISO/IEC-10646-Norm, die weitere Zeichen definiert. Fehlt die Angabe über den Zeichensatz, verwendet der XML-Parser den Standard-Unicode-Zeichensatz. Abweichende Zeichensätze werden über das *encoding*-Attribut vereinbart.

Zeichensatz	Bemerkung
UTF-8	internationaler Zeichensatz (8 Bit)
UTF-16	internationaler Zeichensatz (16 Bit)
ISO-8859-1	Westeuropa (Latin-1)
ISO-8859-2	Osteuropa (Latin-2)
ISO-8859-3	Südeuropa (Latin-3)
ISO-8859-4	Nordeuropa (Latin-4)
ISO-8859-5	Kyrillisch
ISO-8859-6	Arabisch
ISO-8859-7	Griechisch
ISO-8859-8	Hebräisch
ISO-8859-9	Türkisch (Latin-5)
ISO-8859-10	Nordisch (Latin-6)

Tabelle 51.1 Zeichencodierung nach ISO 8859

Die UTF-8-Kodierung basiert dabei auf Unicode und ist zum 7-Bit-ASCII-Code kompatibel, d.h., die ersten 127 Zeichen des ASCII-Codes sind mit dem UTF-8-Unicode identisch. XML-Prozessoren müssen mindestens die Zeichenformate UTF-8 und UTF-16 unterstützen.

Mit der Angabe *<?xml version="1.0« encoding="ISO-8859-1"?>* signalisieren Sie dem XML-Prozessor, daß das Dokument Zeichen aus dem ISO-8859-1-Zeichensatz verwendet. Dies ist beispielsweise erforderlich, wenn im Dokument Texte mit Umlauten vorkommen.

Als weitere Information läßt sich in der *<?xml ..?>*-Processing Instruction angeben, ob das Dokument eine sogenannte Document Type Definition (DTD) besitzt. Dies ist ggf. eine Dokumentdatei, die die im Dokument zulässigen XML-Befehle (Elemente), deren Reihenfolge und deren Eigenschaften (Attribute) definiert. Sie können in der Processing Instruction angeben, ob es eine DTD gibt und wo diese liegt. Mit *<?xml version="1.0" standalone="yes"?>* signalisieren Sie dem XML-Parser, daß keine externe DTD existiert. Der Parser kann dann die Elemente ggf. nach internen Richtlinien verwenden. Mit *<?xml version="1.0" standalone="no"?>* erkennt der Parser, daß eine DTD existiert, es müssen also Hinweise zur DTD im Dokument zu finden sein.

Die Document Type Definition-Angabe

Besitzt das XML-Dokument eine getrennte DTD, wird diese in einer *<!DOCTYPE>*-Anweisung vereinbart:

```
<!DOCTYPE meintext SYSTEM "text.dtd">
```

Nach dem Begriff *!DOCTYPE* folgt ein Name, der die DTD angibt. Das Schlüsselwort SYSTEM bezeichnet eine auf dem System vorhandene DTD, die einfach als Wert angegeben wird. Existiert eine öffentliche Definition, verwenden Sie das Schlüsselwort PUBLIC und geben den URL zur betreffenden DTD-Datei an (*<!DOCTYPE adresse PUBLIC "http://www.borncity.de/text.dtd">*). Als Variante kann die DTD auch direkt im Dokument hinterlegt sein.

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE personen [
<!ELEMENT personen (person)*>
<!ELEMENT person (name, vorname)>
<!ELEMENT name #PCDATA>
<!ELEMENT vorname #PCDATA>
 ]>
<person>
<name>Meier</name>
<vorname>Hans</vorname>
<name>Mueller</name>
<vorname>Armin</vorname>
</person>
```

Abbildung 51.2 Beispiel einer XML-Datei mit interner DTD

XML-Elemente

Der eigentliche Dokumentinhalt mit den Daten (Text, Bilder etc.) wird durch XML-Elemente beschrieben. Diese XML-Elemente erinnern an HTML-Tags, d. h., ein Name wird in spitze Klammern gesetzt (z. B. *<person> ... </person>*). Ein XML-Element besteht dabei immer aus einem einleitenden Start-Tag und einem abschließenden Ende-Tag. End-Tags werden dabei durch ein dem Elementnamen vorangestelltes Slash (/) gekennzeichnet. Das folgende XML-Element bezeichnet also einen Namen:

```
<name>Meier</name>
```

wobei *<name>* der einleitende und *</name>* der abschließende Tag des Elements ist. Innerhalb der beiden Tags folgt dann der Inhalt des Elements. Das Element kann dabei reine Dokumentdaten oder weitere XML-Elemente enthalten. Wichtig ist dabei, daß die

Schreibweise der Elementnamen den Definitionen in der DTD entspricht. Daher ist auf Groß-/Kleinschreibung zu achten, d.h., *<Name>* ist etwas anderes als *<name>.* Die DTD legt auch fest, welche Elemente innerhalb eines Elements untergebracht werden dürfen und ob die Elemente ggf. Attribute besitzen. Und die DTD kann angeben, welche Werte für ein Element zulässig sind (z.B. nur Texte, nur Datumswerte etc.).

Anmerkung: Einige Elemente besitzen selbst keine Daten. Dann wäre es zu aufwendig, die Schreibweise *
</br>* zu verwenden, nur um einen abschließenden Tag zu vereinbaren. In diesem Fall erlaubt die XML-Spezifikation eine verkürzte Schreibweise, bei der das Slash-Zeichen am Ende des einleitenden Tags benutzt wird (z.B. *
*).

Attribute

Ein XML-Element kann bestimmte Daten enthalten, die zwischen einleitendem und abschließendem Tag stehen. Dem Element können aber, abhängig von der DTD, zusätzliche Eigenschaften zugewiesen werden. Diese Eigenschaften sind dabei im einleitenden Tag als Attributwerte in der Form *attributname="wert"* anzugeben. Solche Attribute werden in HTML bei vielen Tags eingesetzt. Ein Beispiel wäre:

```
<img src="bild.gif" />
```

In obiger Zeile stellt *src* das Attribut dar, dessen Wert die Lage der Bilddatei angibt. Gleichzeitig handelt es sich um ein leeres XML-Element, d.h., das XML-Element wird in der verkürzten Schreibweise mit / angegeben.

Anmerkung: Welche Attribute ein Element besitzt, wird in der DTD spezifiziert.

Entities

Ein XML-Dokument besteht aus einer Zeichenfolge im vereinbarten Zeichensatz. Diese Zeichenfolge kann dabei XML-Elemente und die Daten enthalten. Der Datenbereich läßt sich dabei in Textstücke zerlegen. Nun kommt es häufiger vor, daß ein bestimmter Begriff häufiger im Text auftaucht bzw. ersetzbar sein sollte. Denken Sie beispielsweise an einen Namen, der im Text zu ersetzen ist. Hier kommen Entities zum Einsatz. Diese bestehen aus einem Namen, der mit dem &-Zeichen eingeleitet und mit einem Semikolon ; abgeschlossen wird (z.B. <). Der Entität wird dann ein Wert zugewiesen. Immer wenn der XML-Parser auf eine solche Entität stößt, ersetzt er im Datenbereich das Entity durch dessen Wert. XML kennt dabei die in folgender Tabelle vordefinierten Entities.

Entity-Verweis	Zeichen
&amp;	&
&lt;	<
&gt;	>

Entity-Verweis	Zeichen
&apos;	'
&quot;	"

Tabelle 51.2 Vordefinierte Entities

Diese vordefinierten Entities werden benutzt, um Texte mit den betreffenden Zeichen vor dem XML-Parser zu verstecken. Sie können ja nicht einfach das Zeichen < im Text angeben. Der XML-Parser würde dieses Zeichen als Beginn eines Tags interpretieren und bestenfalls einen Fehler auslösen. Wenn Sie dagegen die folgende Konstruktion wählen:

```
<text>Der Wert 60 ist &lt; als 100</text>
```

erkennt der XML-Parser nur gültige Tags. Das vordefinierte Entity < wird dann zur Analysezeit durch das zugeordnete Zeichen < ersetzt.

Kommentare

Innerhalb eines XML-Dokuments dürfen Kommentare auftreten, die durch den XML-Parser überlesen werden. Die XML-Spezifikation erlaubt Kommentare an beliebiger Stelle im Prolog oder in der Dokumentinstanz. Ein Kommentar sieht dann folgendermaßen aus:

```
<!-- beliebiger Text -->
```

Die Zeichenfolge <!-- leitet den Kommentar ein. Dann dürfen beliebige Zeichen folgen. Das Ende des Kommentars wird mit der Zeichenfolge --> markiert.

CDATA-Abschnitte

Manchmal soll ein XML-Element Daten enthalten, die nicht durch den XML-Parser auszuwerten sind. Denken Sie beispielsweise an Bilddaten, an Programmanweisungen etc. Hier könnte es ja vorkommen, daß die Zeichenfolgen, welche die Daten enthalten, ebenfalls Tags aufweisen. Um eine Analyse durch den Parser zu umgehen, werden diese Datenbereiche in einen CDATA-Abschnitt eingebettet.

```
<?xml version="1.0"?>
<job id="T1">
 <comment>
  WSH-Beispielskript by G. Born
 </comment>
 <script language="VBScript">
  <![CDATA[
' zeige ein Meldungsfeld
   name = "Welt"
```

```
   WScript.Echo "Hallo " & name
  ]]>
 </script>
</job>
```

Abbildung 51.3 Beispiel einer XML-Skripdatei mit CDATA-Abschnitt

Das Beispiel aus Abbildung 51.3 demonstriert diesen Sachverhalt. Es handelt sich um eine XML-Datei, die von Microsoft Windows für WSH-Skriptprogramme verwendet wird. Im *<script>*-Element folgt dann der Programmcode. Zufälligerweise enthält dieser Programmcode die Zeichenkombination *& name*. Diese Kombination wird dann vom XML-Parser als Einleitung für eine Entität interpretiert. Da das abschließende Semikolon ; fehlt, löst *& name* einen Fehler aus. Abhilfe schafft die CDATA-Anweisung, die den Inhalt des *<script>*-Elements vor dem Parser versteckt.

Hinweise zur Extended Backus-Naur Form (EBNF)

Die XML-Syntax wird in einem W3C-Dokument nach formalen Kriterien spezifiziert. Wer über die obige Kurzeinführung detailliertere Informationen benötigt, sei an dieser Stelle auf das betreffende im Internet abrufbare Dokument verwiesen (siehe unten). Die XML-Spezifikation baut dabei auf der Extended Backus-Naur Form (EBNF) auf. Bei der Backus-Naur Form handelt es sich um eine Metasprache zur Syntaxbeschreibung formaler Sprachen. Die Sprache setzt sich dabei aus einem Satz von Befehlen zusammen, wobei jeder Befehl ein Sprachelement definiert. Sprachelemente (Metavariablen) werden in spitze Klammern gesetzt. In der XML-Spezifikation kommt eine erweiterte Schreibweise zur Definition der Regeln zum Einsatz. Hierbei werden Bedingungen (productions) zur Darstellung der Syntax aufgelistet. Jede dieser Bedingungen beschreibt einen Teil der XML-Syntax in der Form:

```
[Nummer] Bezeichnung ::= Definition
```

Die Bedingung wird durch eine in eckigen Klammern stehende Nummer eingeleitet. Dann kommt der Bezeichner, der durch zwei Doppelpunkte und ein Gleichheitszeichen von der eigentlichen Definition getrennt ist. In der XML-Spezifikation wird ein »document« folgendermaßen definiert:

```
[1] document ::= prolog element Misc*
```

Das Dokument besteht aus einem Prolog, mindestens einem Element und ggf. weiteren Inhalten *(Misc)* wie beispielsweise Kommentaren. Das Zeichen * gibt dabei an, daß das betreffende Element keinmal, einmal oder mehrfach auftreten darf. Die EBNF für einen Kommentar sieht dann folgendermaßen aus:

```
[15] Comment ::= <!--' ((Char - '-')|
     ('-' (Char - »-«)))* '-->'
```

Die Regel Nummer 15 definiert das XML-Element *Comment*, welches mit <!-- beginnen muß. Daran schließt sich ein beliebiges Zeichen an, welches aber nicht dem Zeichen '-' entsprechen darf. Dies wird durch die Konstruktion (Char – '-') spezifiziert, die besagt, daß das Zeichen nicht mit '-' übereinstimmen darf. Das Zeichen | stellt eine Exklusiv-Oder-Verknüpfung dar (a oder b darf zutreffen, aber nicht beide). Ein Kommentar darf aus einem '-' gefolgt von beliebigen Zeichen, die nicht mehr das Zeichen '-' enthalten, bestehen. Die Zeichenwiederholung wird durch den Stern * symbolisiert. Der Kommentar muß mit den Zeichen '-->' abgeschlossen werden. Der Prolog für ein XML-Dokument ist in der Spezifikation folgendermaßen deklariert:

```
[22] prolog ::= XMLDecl? Misc* (doctypedecl Misc*)?
[23] XMLDecl ::= '<?xml' VersionInfo EncodingDecl?
                 SDDecl? S? '?>'
[24] VersionInfo ::= S 'version' Eq (' VersionNum ' | "
                     VersionNum ")
[25] Eq ::= S? '=' S?
[26] VersionNum ::= ([a-zA-ZO-9_.:] | '-')+
[27] Misc ::= Comment | PI | S
```

Gemäß Definition beginnt der Prolog des XML-Dokuments aus der XML-Deklaration (XMLDecl), die in Regel [23] detaillierter spezifiziert wird. Der Ausdruck *Misc* kann für Kommentare etc. stehen, d.h., im Prolog können kein, ein oder mehrere Kommentare stehen. Der in runden Klammern stehende Ausdruck *(doctypedecl Misc*)?* besagt, daß im Prolog auch eine optionale Dokumenttyp-Deklaration vorkommen darf. Wie diese Dokumenttyp-Deklaration auszusehen hat, wird dann in einer weiteren Regel gemäß der EBNF definiert.

Anmerkung: Details können Sie der XML-Spezifikation 1.0 des W3C *(www.w3c.org)* entnehmen.