

Einbindung

In ihrer einfachsten Form können DTD-Strukturen direkt innerhalb eines XML-Dokumentes deklariert werden:

```
<! DOCTYPE {Wurzelement} [
  {Elementdefinitionen}
]>
```

In diesem Fall sollte in der XML-Deklaration wahrscheinlich „standalone="yes"“ stehen.

Die empfohlene Methode ist jedoch, die DTD wie folgt als externe Datei einzubinden:

```
<! DOCTYPE {Wurzelement} SYSTEM "{URI}" >
```

In einer Mischform ist möglich, sowohl externe, als auch interne Definitionen zu deklarieren:

```
<! DOCTYPE {Wurzelement} SYSTEM "{URI}" [
  {Elementdefinitionen}
]>
```

Elemente

Für alle Elemente, die in dem XML-Dokument vorkommen können, muss es einen Eintrag in der DTD geben. Dieser hat die Form:

```
<! ELEMENT {Name} ({Unterelemente}) >
```

Dabei enthält „Unterelemente“ eine Liste aller innerhalb von diesem Element vorkommenden Elemente. Diese müssen freilich auch in der DTD spezifiziert werden.

Die Unterelemente können wie folgt strukturiert werden:

,	Trennt mehrere Elemente, (Komma) die dann <i>in der angegebenen Reihenfolge</i> auftreten müssen.
	Kennzeichnet eine <i>Auswahl von Elementen</i> , (Senkr. Strich) von denen <u>genau eines</u> auftreten muss.
(und)	Bildet eine <i>Untergruppe</i> .
#PCDATA	Kennzeichnet <i>Textinhalt</i> .
EMPTY	Deklariert ein <i>leeres Element</i> .
ANY	Steht für <i>beliebigen Inhalt</i> .

Ist ein Element oder eine Gruppe optional, oder kann es mehrfach vorkommen, wird dies durch nachgestellte Symbole gekennzeichnet:

?	<i>optional</i>	(0 oder 1)
+	<i>mehrfach</i>	(1 bis n)
*	<i>beliebig</i>	(0 bis n)

Beispiel:

```
<! ELEMENT Anrede (Titel?, (Vorname*, Nachname) | Name) >
```

Attribute

Elementattribute werden recht ähnlich wie die Elemente definiert:

```
<!ATTLIST {Elementname} ({Attributname} {Typbeschreibung} {Optionen} ["{Standardwert}"])+>
```

Dabei können – wie der Name „attribute list“ schon suggeriert – beliebig viele Attributbezeichnungen in einer einzigen Deklaration festgelegt werden (siehe Beispiele unten).

Typbeschreibungen

CDATA	Der einfachste Attributtyp – er besagt lediglich, dass der Attributwert „Character Data“, also einen <i>String</i> enthält.
({Werteliste})	Besagt, dass das Attribut einen der Werte in der Liste haben muss. Die Werte werden durch senkrechte Striche (, ') getrennt. Zum Beispiel: <pre><!ATTLIST i m g a l i g n (l e f t c e n t e r r i g h t) ></pre>
NMTOKEN	Enthält einen String, der nur aus Buchstaben, Zahlen, Punkt, Doppelpunkt, Bindestrich und Unterstrich bestehen darf.
NMTOKENS	Enthält mehrere, durch Whitespaces getrennte NMTOKENS.
ID	Bezeichnet, dass dieses Attribut eine eindeutige Kennung enthält. Diese muss für das gesamte Dokument eindeutig sein. Für IDs gelten dieselben Einschränkungen wie für NMTOKENS.
IDREF	Enthält einen Verweis auf eine existierende ID im selben Dokument.
IDREFS	Enthält mehrere durch Whitespaces getrennte ID-Referenzen.
ENTITY	Enthält einen Verweis auf eine zuvor deklarierte Entität (s.u.).
ENTITIES	Enthält mehrere durch Whitespace getrennte Entitäten.
NOTATION	Enthält einen durch eine Notationsdeklaration definierten Wert (wird hier nicht behandelt).
xml :	Enthält einen vordefinierten XML-Wert.

Optionen

#REQUIRED	besagt dass dieses Attribut unbedingt angegeben werden <i>muss</i> . Fehlt es, ist das Dokument nicht gültig. Mit #REQUIRED kann kein <i>Standardwert</i> angegeben werden.
#IMPLIED	Das Attribut ist optional, falls es weggelassen wird, wird der <i>Standardwert</i> angenommen.
#FIXED	Der <i>Standardwert</i> ist festgelegt, er kann nicht geändert werden.

Beispiel:

```
<!ATTLIST i m g   s r c C D A T A # R E Q U I R E D
                 a l t C D A T A # I M P L I E D
                 a l i g n ( l e f t | c e n t e r | r i g h t ) # I M P L I E D " l e f t "
)>
```

Entitäten

Entitäten werden als Platzhalter gebraucht, z.B. um nicht darstellbare Zeichen in den Code einzufügen, oder um einfacher zu gebrauchende Namen für komplizierte Einheiten zu haben.

Die folgenden Entitäten sind vom System vordefiniert:

<code>&l t;</code>	<code><</code>	<i>kleiner-als</i>	<code>&apos;</code>	<code>'</code>	<i>einfache Anführung</i>
<code>&gt;</code>	<code>></code>	<i>größer-als</i>	<code>&quot;</code>	<code>"</code>	<i>doppelte Anführung</i>
<code>&amp;</code>	<code>&</code>	<i>Kaufm. ‚und‘</i>	<code>&#<i>{num}</i>;</code>		<i>Unicode-Zeichen</i>

Dabei können Unicode-Zeichen sowohl als dezimale oder (mit vorgestelltem ‚x‘) als hexadezimale Zahlen angegeben werden. Benannte Unicode-Entitäten (wie sie in [X]HTML vorkommen) sind nicht vordefiniert, sondern müssen explizit deklariert werden.

Weitere Entitäten können wie folgt definiert werden:

```
<! ENTIT Y {Bezeichner} "{Zuweisung}" >
```

Der Bezeichner wird ohne vorgestelltes ‚&‘ und ohne abschließendes Semikolon angegeben. Um beispielsweise die im deutschen üblichen Sonderzeichen als benannte Entitäten zu deklarieren, gibt man an:

```
<! ENTIT Y auml "&#228;" >      <! ENTIT Y Auml "&#196;" >
<! ENTIT Y ouml "&#246;" >      <! ENTIT Y Ouml "&#214;" >
<! ENTIT Y uuml "&#252;" >      <! ENTIT Y Uuml "&#220;" >
<! ENTIT Y szl i g "&#223;" >
```

Wie hier gezeigt, können die Entitätsdefinitionen auch selbst wieder Entitäten enthalten. Und selbstverständlich können auch längere und komplexere Ausdrücke so vordefiniert werden.

Parameterentitäten

Entitäten können auch verwendet werden, um Unterelemente- oder Wertelisten von Element bzw. Attributdefinitionen vorzudefinieren. Damit können dann gleich lautende Definitionen an einer Stelle gemeinsam verwaltet werden.

Diese Form von Entitäten wird durch ein vorangestelltes Prozentzeichen deklariert (und auch referenziert):

```
<! ENTIT Y % al i gnments "(l eft|center|ri ght)" >
<! ATTLI ST H1 al i gn %al i gnments; #I MPLI ED "l eft" >
<! ATTLI ST P al i gn %al i gnments; #I MPLI ED "l eft" >
```

Weiteres

Es ist auch möglich, Entitäten in einer eigenen Datei zu deklarieren. Die dazu erforderlichen Attribute „PUBLIC“ und „SYSTEM“ entsprechen den gleich lautenden Attributen in der o.g. „<!DOCTYPE...“-Spezifikation.

Entitäten können auch auf nicht-textuelle Daten verweisen (v.a. natürlich bei externen Verweisen). Der Datentype muss dann aber entsprechend deklariert worden sein (dies wird hier aber nicht behandelt)