XML – Merkblatt 2 – Document Type Definition

Einführung *Document Type Definitions* (DTD) sind eine aus SGML übernommene Syntax zur Beschreibung der Struktur und Aufbau von XML-Dokumenten. DTDs beschreiben XML-Dokumente durch Typdefinitionen. Es gibt elementare Texttypen und komplexe Typen, die ähnlich zu regulären Ausdrücken aufgebaut sind. Typinformationen sind nötig, wenn XML-Dokumente mit einem Programm (XML Prozessor) eingelesen und verarbeitet werden sollen. Der zum Einlesen benötigte XML-Prozessor benötigt die Typinformation um nur XML-Dokumente, die vom Programm sinnvoll verarbeitet werden können, vorab zu erkennen.

Document Type Definitions Mit einer ELEMENT-Deklaration wird jedes Element eines XML-Dialekts definiert. Im dem folgenden leicht geändertem Beispiel aus [2, Seite 77] werden acht Elemente definiert. Die erste fünf enthalten selbst Elemente, die letzten vier haben nur Text als Inhalt (PCDATA, parsed character data).

```
<!ELEMENT populationdata (continent+) >
<!ELEMENT continent (name, country*) >
<!ELEMENT country (name, province*)>
<!ELEMENT province ((name|code), city*) >
<!ELEMENT city (name, pop?) >
<!ELEMENT name (#PCDATA) >
<!ELEMENT code (#PCDATA) >
<!ELEMENT pop (#PCDATA) >
```

Die inhaltliche Struktur (Inhaltsmodell, content model) der ersten fünf Elemente wird mit einem regulären Ausdruck definiert. Dazu können neben den runden Klammern zum Gruppieren von Teilausdrücken, Elementnamen mit den unären Operatoren `+` (mindestens ein Vorkommen), `*` (beliebig viele), `?` (optional), den binären Operatoren `,` (Sequenz) und `|` (Vereinigung) zu neuen Ausdrücken geformt werden. Der Ausdruck `(name, pop?)` definiert den Inhalt eines `city`-Elements aus genau einem `name`-Element gefolgt von einem optionalen `pop`-Element bestehend.

Ein XML-Dokument heißt *gültig* (valid) wenn es wohlgeformt ist und den Regeln gemäß einer angegeben DTD oder einer anderen Typbeschreibung aufgebaut ist.

Damit ein Programm prüfen kann, ob ein XML-Dokument gültig ist oder nicht, muss mit einer DOCTYPE-Deklaration entweder auf eine Resource mit der DTD verwiesen werden oder die DTD im XML-Dokument selbst angegeben werden.

Im folgenden `DOCTYPE`-Beispiel wird das Wurzelelement `populationdata` definiert. Nach dem Bezeichner `PUBLIC` wird eine Zeichenkette mit einer eindeutigen Identifikation des XML-Dialekts angegeben gefolgt von einem optionalen Uniform Resource Locator (URL). Dieser verweist auf eine lokal auf dem System befindliche Version der DTD. `"population.dtd"` ist

ein relativer Dateipfad. Die DTD muss sich deswegen im gleichen Verzeichnis wie das XML-Dokument befinden. Wird nur die Identifikation angegeben, so muss der XML-Prozessor so konfiguriert werden, dass über diese Identifikation die zugehörige DTD gefunden wird.

```
<!DOCTYPE populationdata PUBLIC "Population 1.0" "population.dtd">
<populationdata>
  <continent>
    <name>Europa</name>
  </continent>
</populationdata>
```

Alternativ kann die DTD direkt im XML-Dokument angegeben werden:

```
<!DOCTYPE populationdata [
  <!ELEMENT populationdata (continent+) >
  <!ELEMENT continent (name | country*) >
  <!ELEMENT name (#PCDATA) >
]>
<populationdata>
  <continent>
    <country>Deutschland</country>
    <country>Frankreich</country>
  </continent>
</populationdata>
```

Mit dem Bezeichner `SYSTEM` kann ohne die Identifikation wie bei `PUBLIC` direkt mit der URL die DTD angegeben werden.

```
<!DOCTYPE populationdata SYSTEM "population.dtd">
<populationdata>
  <continent>
    <name>Europa</name>
  </continent>
</populationdata>
```

Bei der Definition des Inhalts von Elementen gibt es noch zwei wichtige Einschränkungen.

1. Die Regeln einer Element-Deklaration müssen deterministisch formuliert sein.
2. Die Operatoren können nur eingeschränkt mit `#PCDATA` – sogenannter *gemischter Inhalt* (mixed content) – verwendet werden.

Bei `((a | b)*, a )` kann ein XML-Prozessor beim Einlesen eines Start-Tags `<a>` nicht deterministisch entscheiden, ob die Regel `(a | b)*` (linke Seite) oder `a` (rechte Seite) verfolgt

werden muss. Um effiziente generische Parser für XML zu ermöglichen, sind nur deterministische Regeln erlaubt. `(b*, a)+` ist ein äquivalenter deterministischer reguläre Ausdruck.

Gemischter Inhalt darf nur in der Form `(#PCDATA | regulaerer-ausdruck )*` definiert werden. In `regulaere-ausdruck` darf kein `#PCDATA` und nur der Operator `|` vorkommen.

```
<!DOCTYPE a [
  <!ELEMENT a (#PCDATA |  b | c)* >
  <!ELEMENT b (#PCDATA) >
  <!ELEMENT c (#PCDATA) >
]>
<a>
  Mixed<b/>Content<c/><c/>
</a>
```

Attribute Alle Attribute eines Elements werden mit genau einer `ATTLIST`-Deklaration definiert. Sie hat folgende Syntax:

```
<!ATTLIST element-name attribute-definition-1 ... attribute-definition-n>
```

Jede der n einzelnen Attribute-Definitionen besteht dabei aus drei Teilen:

1. Den Namen des Attributs.
2. Den Typ des Attributwerts. Neben Typen wie `CDATA`, `ID`, `IDEF` sind auch Aufzählungen von Werten erlaubt in runden Klammern mit `|` möglich. Diese Werte werden nicht in Hochkommas angegeben, sondern als Bezeichner. Sie sind case insensitive.
3. Der Definition, ob das Attribut verpflichtend (`#REQUIRED`), optional (`#IMPLIED`) oder optional mit einem festen Wert (`#FIXED`) ist. Statt `#IMPLIED` kann ein Standardwert, bei `#FIXED` muss dieser angegeben werden.

`CDATA` ist ein Texttyp der auch Verweise wie `&` oder `
` enthalten darf. Mit `IDREF` wird der Wertebereich auf Zeichenketten eingeschränkt, die bei anderen Attributen mit `ID` angegeben wurden.

```
<!ELEMENT a (#PCDATA)>
<!ATTLIST a   rowspawn    CDATA     "1"
              http-equiv  CDATA     #IMPLIED
              id          ID        #REQUIRED
              valign      (top|middle|bottom|baseline) "top"
              available   CDATA     #FIXED  "true">
```

Die folgenden Elemente sind gültig, falls im XML-Dokument der Wert `z5` bei einem Attribute mit Typ `IDREF` angegeben wurde:

```
<a rowspawn="17" id="z5" valign="baseline" available="true"/>
<a id="z5"/>
```

XML-Namensräume Verschiedene XML-Dialekte können in einem XML-Dokument gemischt werden. Um Namenskonflikte gleichnamiger Elemente zu vermeiden, können Namensräume verwendet werden. Ein *XML-Namensraum* wird durch einen Uniform Resource Identifier (URI) definiert. In der Praxis wird ein Uniform Resource Locator (URL) verwendet. Sie werden mit dem Attribute `xmlns` für ein Element und alle enthaltenen Elemente eingeführt (*Standardnamensraum*). Mit `xmlns:prefix` kann ein zusätzlicher Präfix `prefix` definiert werden. `prefix` ist ein frei wählbarer XML-Bezeichner ohne `:` und kann Elementen und Attributen mit `:` vorangestellt werden. Ist für ein Element oder Attribut kein XML-Namensraum definiert, dann heißen diese *nicht qualifiziert*, ansonsten *voll qualifiziert*.

```
<a> <!-- nicht qualifzierter Name -->
  <b xmlns="http://mein.eigener.namensraum"
     xmlns:p="http://dein.eigener.namensraum">
    <p:c/> <!-- explizit voll qualifiziert  -->
    <c/>   <!-- implizit voll qualifiziert über Standardnamensraum -->
  </b>
</a>
```

Das Element `a` ist keinem Namensraum zugeordnet, `b` und `c` sind beide dem Namensraum `http://mein.eigener.namensraum` und `p:c` ist `http://dein.eigener.namensraum` zugeordnet. Es handelt sich deswegen um *unterschiedliche* c-Elemente.

DTDs unterstützen keine Namensräume.

Ergänzende Literatur Die Grundlagen für dieses Merkblatt finden sich in [2, Abschnitt 3.3.1, Seiten 77–79]. XML-Namensräumen sind in [3] spezifiziert. DTDs sind im SGML-Standard definiert. Dieser ist als ISO-Standard nicht frei zugänglich. Eine für HTML konzipierte kurze Einführung gibt es in [1].

Literatur

[1] A brief SGML tutorial. Technical report, W3C. https://www.w3.org/TR/WD-html40-970708/intro/sgmltut.html.

[2] Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, and Pierre Senellart. *Web Data Management*. Cambridge University Press, New York, NY, USA, 2011. http://webdam.inria.fr/Jorge/files/wdm.pdf.

[3] Tim Bray, Dave Hallander, Andrew Layman, Richard Tobin, and Thompson Henry S. Namespaces in XML 1.0 (third edition). Technical report, W3C, December 2009. https://www.w3.org/TR/xml-names/.