tutego™
YOUR TRAINING COMPANY

Dokumenttyp-Definition

Struktur-Beschreibung einer XML-Datei

Aufgabe einer DTD

- Durch eine DTD wird die Form einer XML-Datei festgelegt:
  - Welche Elemente dürfen verwendet werden?
  - Wie werden Elemente miteinander verschachtelt?
  - Welche Attribute haben die Elemente?
  - Welche Werte können Elemente und Attribute annehmen?
  - Welche Elemente und Attribute sind vorgeschrieben oder müssen einen Wert haben?

Wann ist eine DTD notwendig?

- Eine DTD für wenige Dokumente ist oft sinnlos.
- Folgende Situationen erfordern eine DTD:
  - Daten werden von Hand erfasst.
  - Bestimmte Daten sind unbedingt erforderlich.
  - Daten werden ausgetauscht.
  - Die Sprache wird aus Sprachelementen unterschiedlicher Sprachen zusammengesetzt.
- Beispiel: *email.dtd*

Verhalten des Parsers ohne DTD

- Auch ohne DTD können XML-Dokumente sinnvoll verarbeitet werden.
  - Natürlich müssen die Dokumente wohlgeformt sein!
- Das Fehlen einer DTD hat folgende Konsequenzen:
  - Keine Einschränkung der Elemente.
  - Keine Regeln für die Verschachtelung.
  - Keine Regeln für Attribute
  - Alle Attributwerte werden als Zeichenketten interpretiert.

Einbinden einer DTD

- Die DTD kann innerhalb und außerhalb der XML-Datei stehen.
- Innerhalb einer XML-Datei steht die DTD am Anfang der Datei, gleich hinter dem XML-Prolog:

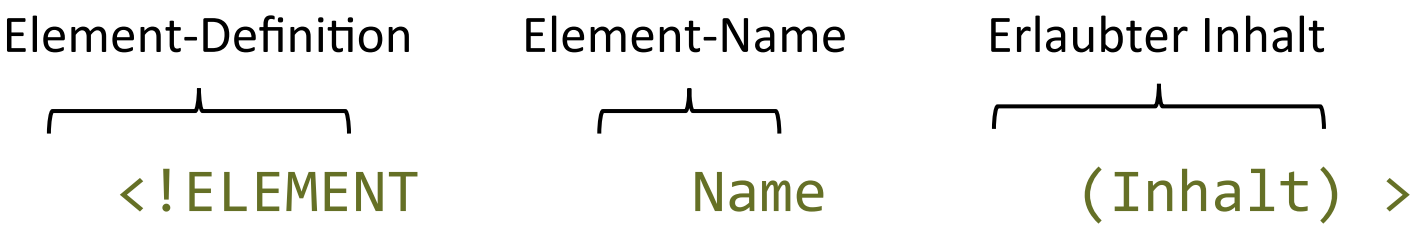
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE email [
  <!-- Definition der einzelnen Elemente -->
]>
```

- Eine extern referenzierte DTD sieht so aus:

```
<!DOCTYPE email SYSTEM "email.dtd">
```

Elementtypen definieren

- Die Element-Deklaration legt Namen und Inhalt des Elements fest.
- Standardschema für Elementtypen:



- Der Inhalt #PCDATA steht für Elementtypen, die Text zum Inhalt haben.

```
<!ELEMENT text (#PCDATA)>
```

PCDATA und CDATA

- PCDATA (Parsed Character Data) und CDATA (Character Data) können beliebigen Text enthalten kann.
- In CDATA sind jedoch auch die Zeichen < und & erlaubt, was für PCDATA nicht gilt!
  - In CDATA wird nur das Endezeichen erkannt]]>.
  - In PCDATA werden dafür Entities expandiert, so dass etwa & zu einem & wird.
- Einschränkung
  - Da Sonderzeichen nicht kodiert werden können, kann man]]> nicht ausmaskieren. Das ist immer Markup!

Regeln für Namen

- Die folgenden Regeln gelten für Namen von Elementen, Attributen und Entities:
  - Buchstaben, Ziffern und Interpunktionszeichen dürfen enthalten sein.
  - Keine Doppelpunkte, Leerzeichen oder spezielle XML-Zeichen
  - Das erste Zeichen darf keine Ziffer sein und nicht die Zeichenkette „xml“ sein.
  - Namen müssen mindestens ein Zeichen lang sein.

Beispiele für Elemente

- Beispiele für Elementdeklarationen
  - `<!ELEMENT absatz (ueberschrift, text) >`
  - `<!ELEMENT html (head?, body) >`
  - ```
    <!ELEMENT email(absender,
                    (kopie|absender)*,
                    text) >
    ```
  - `<!ELEMENT email((kopie|absender)+,text) >`
    ```

Unterelemente

- Falls der Inhalt aus anderen Elementen besteht, werden diese aufgezählt.
- Die Reihenfolge ist wichtig!
- Die Häufigkeit ist immer genau einmal, falls keines der folgenden Häufigkeiten zugeordnet ist:
  - ?: Einmal oder gar nicht.
  - *: beliebig oft, auch gar nicht.
  - +: mindestens einmal.
- Elemente die durch Kommata getrennt sind, müssen alle vorkommen.
- Von den Elementen, die durch den senkrechten Strich | getrennt sind (Oder-Verknüpfung), muss genau ein Element vorkommen.

Elemente mit gemischtem Inhalt

- Text und Unterelemente können gemischt werden. Das nennt sich **mixed content**.

```
<!ELEMENT p (#PCDATA | a | ul | b | i | em)* >
```

- Diese Form, ist die einzige Möglichkeit, Text und Elemente miteinander zu kombinieren.
- Hier geht natürlich auch die Reihenfolge verloren!

EMPTY und ANY

- Element ohne Inhalt werden durch das Schlüsselwort EMPTY deklariert.
  - Sie können weder Unterelemente noch Text beinhalten.
  - Wie etwa
 in HTML, das einen Zeilenvorschub macht.

```
<!ELEMENT br EMPTY>
<!-- forced line break -->
```

- Elemente, denen beliebiger Inhalt zugeordnet werden kann, werden durch das Schlüsselwort ANY deklariert.
- Kommentare in einer DTD haben die gleiche Syntax wie Kommentare in XML.

Attribute in der DTD

- Attributlisten definieren gültige Attribute eines Elements.
- Beispiel

```
<?xml version="1.0"?>
<!DOCTYPE img [
  <!ELEMENT img EMPTY>
  <!ATTLIST img src     CDATA  #REQUIRED>
  <!ATTLIST img height  CDATA  #IMPLIED>
  <!ATTLIST img width   CDATA  "44">
]>
<img src="otto.gif" height="32" width="32"/>
```

Beispiel einer Aufzählung

- Definiton

```
<!ATTLIST datum
monat (Januar | Februar | März | April) #REQUIRED
tag   (1 | 2 | 3 | .. | 31)             #REQUIRED
>
```

- Beispiel

```
<datum monat="März" tag="2" />
<datum monat="Januar" tag="02" />
<datum monat="Dezember" tag="2" />
```

Attributlisten

- Attributlisten definieren gültige Attribute eines Elements.
- Insgesamt gibt es drei Vorgaben für das Vorkommen eines Attributs:
  - #IMPLIED: Das Attribut ist optional.
  - #REQUIRED: Das Attribut muss vorkommen.
  - #FIXED: Der Wert des Attributs ist fest.
- Für Attribute können mögliche Werte vorgegeben werden.

```
<!ATTLIST  form  method  (get|post)  "get">
```

- Es können nur Werte aus diesem Wertebereich genommen werden.

Attributtypen

- Der Inhalt eines Attributs entstammt einer der drei Gruppen
  - CDATA (String Attribute Type): Eine beliebige Zeichenkette
  - Aufzählung (Enumerated Attribute Type): Die möglichen Werte werden aufgezählt. Nur diese Werte sind gültig.
  - Token (Tokenized Attribute Type): Ein bestimmter Wert der durch ein Schlüsselwort beschrieben wird.

- Eine genaue Typisierung (z. B. Ganzzahl) wie in Programmiersprachen ist mit einer DTD nicht möglich.
  - Das ist einer der Gründe für Alternativen wie Schema!

Attributtyp Nametoken (NMTOKEN)

- Attributwerte vom Typ NMTOKEN (Name Token) müssen eine bestimmte Namenskonvention für Attributwerte einhalten.
  - Die Zeichenfolge ist mindestens ein Zeichen lang und besteht aus folgenden Zeichen:

    Letter | Digit | '.' | '-' | '_' | ':' | einige Unicode-Zeichen
  - Bsp. in der DTD:

    ```
    <!ATTLIST journal year NMTOKEN #REQUIRED>
    ```

    Bsp. im XML-Dokument:

    ```
    <journal year=2005 />
    ```
- Achtung: XML-Namen dürfen nicht mit einer Ziffer beginnen, aber NMTOKEN können eine Ziffer als erstes Zeichen enthalten.

Attributtyp NMTOKENS

- Beim Attributtyp NMTOKENS besteht der Attributwert
  - aus einem oder mehreren NMTOKEN,
  - die durch Leerzeichen getrennt sind.

- Beispiel

  DTD:

```
<!ATTLIST Auftritte datum NMTOKENS #REQUIRED>
```

  XML-Dokument:

```
<Auftritte datum="21-08-2002 26-08-2002" />
```

Attributtyp ID

- Ein Attribut vom Typ ID muss einen XML-Namen enthalten, der im Dokument eindeutig ist.
  - Es ist nicht wichtig, wie das Attribut heißt.
  - An dem Beispiel sieht man, dass Zahlen als ID-Werte problematisch sind: Ein XML-Name darf nicht mit einer Ziffer beginnen! Der Wert ist kein NMTOKEN.
  - Es gibt eine zusätzliche Einschränkung: Kein Elementtyp darf mehr als ein ID-Attribut besitzen.

```
<!ATTLIST person personalnummer ID #REQUIRED>
<person personalnummer="_123-45-6789" />
```

Attributtype IDREF und IDREFS

- Der Wert eines Attributs vom Typ IDREF verweist auf ein Element, das ein Attribut vom Typ ID hat.
  - Das ist wie eine Schlüssel-/Fremdschlüssel-Beziehung bei relationalen Datenbanken.
- Die Werte von Attributen vom Typ IDREFS sind eine Liste von IDREF.
  - Die einzelnen Werte werden durch ein Leerzeichen getrennt.
- Der Wert muss in dem XML-Dokument vergeben sein, sonst löst der Parser einen Fehler aus.

Beispiel mit ID und IDREF

```
<!ATTLIST abteilungsleiter personalid ID #REQUIRED>
<!ATTLIST angestellter vorgesetzter IDREF #REQUIRED>

<abteilungsleiter personalid="_123-45-6789">
  Hans Hoppel
</abteilungsleiter>

<angestellter vorgesetzter="_123-45-6789">
  Steffan Stoffel
</angestellter>
```

Datentyp Entität

- Entities werden in HTML verwendet, um Sonderzeichen darzustellen:
  - ä
  - >
  - ©
- In XML ist es möglich, eigene Entities zu definieren und zu verwenden.
- Dadurch können wiederkehrende Teile mit einer Entity verknüpft werden und über die Notation &abkürzung; verwendet werden.

Festgelegte Entitäten

- XML hat folgende festgelegten Entitäten:
  - & für &
  - ' für '
  - > für >
  - < für <
  - " für "
- Für HTML-Dateien werden weitere Entitäten definiert.

Numerische Entitäten

- Spezielle Sonderzeichen können als numerische Entitäten angegeben werden.
- Die Wert können dezimal, hexadezimal oder auch oktal angegeben werden.
- Der entsprechende Wert ergibt sich aus dem gewählten Zeichensatz.
- ç kann durch ç in dezimaler Schreibweise oder auch ç in hexadezimaler Schreibweise angegeben werden.

Definition von Entities

- Entities werden nach folgender Syntax definiert:

```
<!ENTITY [%] Name [SYSTEM | PUBLIC]
                "Wert" [zusätzliche Angaben] >
```

- Entities werden unabhängig von Elementen und Attributen in der DTD definiert:

```
<!ENTITY kuerzel "Text des Entity">
```

  - Diese Entity kann als &kuerzel; verwendet werden. Die Anwendung ersetzt "&kuerzel;" durch "Text des Entity".

Benannte Zeichen-Entitäten

- Da die Nummern für Sonderzeichen nicht leicht zu merken sind, gibt es die wichtigsten Zeichen bereits als benannte Entitäten.

- Für XHMTL gibt es die Entity-Dateien
  - *xhtml-lat1.ent*: spezielle Buchstaben
  - *xhtml-special.ent*: Sonderzeichen wie das Euro-Symbol
  - *xhtml-symbol.ent*: Symbole, wie sie für mathematische Formeln benötigt werden.

Externe Entities

- Es ist auch möglich, Entities in externen Dateien zu pflegen.
- Das ist besonders bei längeren Texten sinnvoll.
- Es können beliebige Texte eingefügt werden. Diese Texte werden vom XML Parser überprüft:

```
<!ENTITY Kuerzel SYSTEM "Datei">
```

- Diese Entity kann wie jede andere verwendet werden. Der Inhalt der Datei wird eingefügt.
- Beispiel: *include.xml*.

Parameter-Entities für DTDs

- Auch für DTDs können Entities festgelegt werden.
- Diese Entities haben die folgende Form:

```
<!ENTITY % kuerzel "Text des Entity">
```

  - Sie werden innerhalb der DTD eingesetzt: % kuerzel;
- Ein Beispiel aus *xhtml1-strict.dtd*:

```
<!ENTITY % Text "CDATA">
```

- Externe Parameter-Entities dienen der Modularisierung von DTDs:

```
<!ENTITY % kuerzel SYSTEM "datei.dtd" >
```

Zusammenfassung Attribute, Entities

- Attributwerte lassen sich in der DTD sehr vielschichtig beschreiben.
- Durch Aufzählungen können Werte vorgegeben werden. Token ermöglichen Beziehungen zwischen den Elementen.
- Interne und externe Entities bieten Möglichkeiten Texte mit Kürzeln zu kodieren.

Öffentliche und lokale DTD

- Durch SYSTEM wird eine lokale DTD verwendet

```
<!DOCTYPE email SYSTEM "email.dtd">
```

- Es lässt sich auch eine öffentliche DTD verwenden.

```
<!DOCTYPE name PUBLIC "Kennung" "URI">
```

- Die Kennung ist gegliedert in
  - Herausgeber der DTD
  - Sprache, in der die DTD verfasst wurde.
  - Beispiel: *xhtml.xml*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
```

- Es kann zusätzlich eine lokale DTD angegeben werden.

Bedingte Abschnitte

- In einer DTD können Teile auskommentiert werden:

```
<![IGNORE[ <!-- DTD-Elemente --> ]]>
```

- Ebenso können Bereiche hinzugenommen werden:

```
<![INCLUDE[ <!-- DTD Elemente --> ]]>
```

- Die kann auch vom Inhalt eines ENTITY abhängig gemacht werden.

```
<!ENTITY % switch "INCLUDE">
<![%switch; [
  <!-- DTD Elemente -->
]]>
```

Eigene oder vorhandene DTD nutzen?

- Generell ist es immer besser, sich an vorhandenen Standards zu orientieren.
- Wenn Sie eigene DTDs entwickeln müssen, können Teile vorhandener DTDs verwendet werden.
- Durch eine DTD werden die Namen der XML-Elemente festgelegt.
- Das Entwickeln einer DTD fördert eine umfangreiche Datenanalyse.
- Beispiel: *xhtml1-strict.dtd*

Zusammenfassung

- DTDs können benutzt werden, um die Namen für die Elemente einer XML-Datei festzulegen.
- Es können Inhalt, Verschachtelung und Häufigkeit festgelegt werden.
- Eine DTD kann in einer externen Datei oder innerhalb der XML-Datei stehen.
- Eine interne DTD kann eine externe DTD nicht überschreiben sondern nur erweitern.