Tutorial zu Kapitel WT:III

III. Dokumentsprachen

- HTML
- CSS
- XML-Grundlagen
  - XML-Dokumentstruktur
  - Document Type Definition (DTD)
  - Namespaces
- XML-Schema
- XSL-Familie
  - XPath
  - XSLT

Die nach folgenden Erklärungen basieren auf W3C XML 1.0. Vereinfachungen an verschiedenen Stellen sollen das grundsätzliche Verständnis erleichtern.

XML-Grundlagen

XML-Dokumentstruktur

0. Vorbemerkung

   (a) Angabe von URIs

   Wenn in einem XML-Dokument die Angabe einer URI erforderlich ist, kann diese Angabe in Form einer relativen URL erfolgen.

   Eine relative URL wird mit Hilfe der Document Base URL (URL, unter der das XML-Dokument selbst im Web verfügbar ist) zu einer absoluten URL vervollständigt werden.

XML-Grundlagen

XML-Dokumentstruktur (Fortsetzung)

0. Vorbemerkung

(b) Namen

XML erlaubt in Namen neben lateinischen Buchstaben eine Vielzahl von Zeichen, darunter Umlaute und andere sprachspezifische Zeichen, z.B. griechische oder kyrillische Buchstaben mit und ohne Betonungsangaben wie Akzenten.

Einschränkung für Namen

Um Falschinterpretationen von Namensbestandteilen und Zeichensatzprobleme bei Dateien zu vermeiden, schränken wir im Rahmen dieser Vorlesung Namen ein. Erlaubt sind nur lateinische Groß- und Kleinbuchstaben und Unterstrich, zusätzlich Ziffern und Trennzeichen, wenn sie nicht am Anfang stehen.

Zur Kompatibilität mit Namensräumen können zwei Namen (jeweils ohne Doppelpunkt) mit Hilfe des Doppelpunktes zu einem Namen verknüpft werden. Der Teil vor dem Doppelpunkt wird dann als Präfix, der nach dem Doppelpunkt als lokaler Name aufgefasst.

Eingeschränkte Grammatik:

```
Name ::=
        NCName | NCName ":" NCName

NCName ::=
        NameStartChar NameChar*

NameStartChar ::=
        [A-Z] | [a-z] | "_"

NameChar ::=
        [A-Z] | [a-z] | [0-9] | "_" | "-" | "."
```

XML-Grundlagen

XML-Dokumentstruktur: Gesamtdokument

1. XML-Dokument

   Grammatik:

```
document ::=
        prolog element misc*
prolog ::=
        XMLDecl misc* doctypedecl? misc*
misc ::=
        comment | processing-instruction
```

   Eine XML-Dokument besteht aus einem Prolog mit zumindest einer XML-Deklaration. Auf den Prolog folgt genau eine Elementinstanz.
   Vor und nach der Elementinstanz (aber auch darin) sind Kommentare und Verarbeitungsanweisungen möglich.

   Beispiel:

```
<?xml version="1.0"?>
<!-- A very simple example! -->
<myRoot></myRoot>
```

XML-Grundlagen

XML-Dokumentstruktur: Dokumenten-Prolog

2. XML-Deklaration

Grammatik:

```
XMLDecl ::=
        "<?xml" VersionInfo EncodingDecl? StandaloneDecl? "?>"
```

Eine XML-Deklaration kennzeichnet ein Dokument als XML Dokument.

- `VersionInfo`
  Die Versionsinformation hat meist die Form **version="1.0"** oder **version="1.1"**.
- `EncodingDecl`
  Als Zeichensatzkodierung findet sich oft **encoding="UTF-8"** oder **encoding="ISO-8859-1"**.
- `StandaloneDecl`
  Für die Angabe, ob externe Ressourcen verwendet werden, ist als Wert nur **standalone="yes"** oder **standalone="no"** möglich.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

XML-Grundlagen

XML-Dokumentstruktur: Dokumenten-Prolog (Fortsetzung)

3. XML-Dokumenttyp-Deklaration

   Grammatik:

```
doctypedecl ::=
        "<!DOCTYPE" Name ("SYSTEM" URL-External-DTD)?
                         ("[" DTD-intSubset "]")? ">"
```

   Optional legt eine XML-Dokumenttyp-Deklaration eine DTD für ein XML Dokument fest.

   - `URL-External-DTD`
     Die Angabe einer DTD in einer externen Ressource erfolgt durch eine URI, also i.d.R. durch eine URL.
   - `DTD-intSubset`
     Als internes Subset wird eine DTD angegeben, die genauso aufgebaut ist wie in einer externen Datei.
   - Wenn eine externe DTD angegeben wird und ein internes Subset, werden beide Teile zu einer DTD zusammengefasst.

   Beispiel:

```
<!DOCTYPE myRoot SYSTEM "http://www.upb.de/webis2016-example.dtd">
```

   oder

```
<!DOCTYPE myRoot [ <!ELEMENT myRoot (#PCDATA)> ]>
```

XML-Grundlagen

XML-Dokumentstruktur: Dokumenten-Body

4. Elementinstanz

   Grammatik:

```
element ::=
        EmptyElemTag | STag content ETag

EmptyElemTag ::=
        "<" Name ( Attribute )* "/>"

STag ::=
        "<" Name ( Attribute )* ">"

ETag ::=
        "</" Name ">"
```

   Ein XML-Dokument besteht aus nur genau einer Elementinstanz, dem sogenannten Wurzelelement. Eine Elementintanz wird durch öffnendes und schließendes Tag oder durch ein Kombi-Tag gekennzeichnet. Im Inhalt `content` einer Elementinstanz können weitere Elementinstanzen und/oder textueller Inhalt geschachtelt sein.

   Beispiel:

```
<myRoot>
  Some text. <myChild> A little bit more text. </myChild>
</myRoot>
```

XML-Grundlagen

XML-Dokumentstruktur: Dokumenten-Body

4. Elementinstanz

   Inhalt von Elementinstanzen:

   - Text

     Der Text darf keine Markups enthalten, also weder "<" für Anfänge von Element-Tags, noch "&" für Anfänge von Entity-Referenzen, noch "]]>" für das Ende von CDATA-Bereichen.

   - Kindelementinstanzen
   - Entity-Referenzen

     Eine Entity-Referenz &MyEntity; wird währed des Einlesens durch die Zeichenkette ersetzt, für die sie steht.

   - Kommentare <!– My comment. –>
   - Verarbeitungsanweisungen <?php echo "My program." ?>
   - CDATA-Bereiche

     `CDATASection ::= "<![CDATA[" char* "]]>"`

     Der beliebige Text, für den `char*` steht, darf natürlich nicht die Zeichenkette "]]>" für das Ende von CDATA-Bereichen. Markups wie Element-Tags, Entity-Referenzen, Kommentare, Verabeitungsanweisungen werden als Text aufgefasst.

XML-Grundlagen

XML-Dokumentstruktur: Dokumenten-Body (Fortsetzung)

5. Attribute in Elementinstanzen

   Grammatik:

```
Attribute ::=
        Name "=" AttValue
```

   Im öffnenden Tag oder im Kombi-Tag einer Elementinstanz können Attribute enthalten sein. Jedes Attribut besteht aus der Angabe des Attributnamens und einer Wertzuweisung.

   - Der Wert ist in einfache oder doppelte Anführungszeichen eingeschlossen.
   - Enthält der Attributwert doppelte Anführungszeichen, so muss er in der Zuweisung in einfache Anführungszeichen eingeschlossen werden und umgekehrt. Beide Arten von Anführungszeichen darf er nicht enthalten.
   - Attributwerte können Entityreferenzen (z.B. `<`) und Zeichenreferenzen (z.B. `&#x20;`) enthalten.
   - Ein Attributname darf innerhalb eines Tag nur einmal auftreten, mehrfache Verwendung in einem Tag ist nicht erlaubt.

   Beispiel:

```
<myElement1 myAttribute1="value" ... > ... </myElement1>
```

   oder

```
<myElement2 myAttribute1="value" myAttribute2="a second value"/>
```

XML-Grundlagen

XML-Dokumentstruktur: Dokumenten-Body (Fortsetzung)

6. Verarbeitungsanweisungen (Processing Instructions)

   Grammatik:

   ```
   processing-instruction ::=
           "<?" PITarget Char* "?>"
   ```

   Eine XML-Verarbeitungsanweisungen veranlasst die Bearbeitung eines XML-Dokumentes mit einer Applikation.

   - `PITarget`
     Die Applikation wird durch einen Namen angegeben. Dieser Name darf aber nicht `"XML"` sein (oder Varianten davon mit Kleinbuchstaben).
   - `Char*`
     Auf den Namen der Applikation kann eine beliebige Zeichenkette folgen, die z.B. Parameter der Applikation spezifiziert. In dieser Zeichenkette dürfen keine Entities verwendet werden und auch nicht die Zeichenkette `"?>"`.

   Beispiele:

   ```
   <?xml-stylesheet type="text/xsl" href="personen2html.xsl"?>
   ```

   oder

   ```
   <?php echo "Hi, I'm a PHP script!"; ?>
   ```

XML-Grundlagen

XML-Dokumentstruktur: Dokumenten-Body (Fortsetzung)

7. Kommentare

   Grammatik:

   ```
   comment ::=
           "<!--" Char* "-->"
   ```

   Eine XML-Kommentar wird beim Einlesen eines Dokumentes mit verarbeitet und nicht einfach ignoriert. Auf diese Weise können Kommentare bei der Bearbeitung von XML-Dokumenten berücksichtigt werden. Allerdings ist nicht sichergestellt, das der Kommentartext für eine Applikation zugreifbar ist.

   - `Char*`
     Als Kommentartext kann eine beliebige Zeichenkette verwendet werden. In dieser Zeichenkette dürfen keine Entities verwendet werden und auch nicht die Zeichenkette `"--"` (zwei Bindestriche).

   Beispiel:

   ```
   <!-- Ein Beispielkommentar. -->
   ```

XML-Grundlagen

Wohlgeformte XML-Dokumente

1. XML-Prolog im Dokument
2. korrekt gebildete DTD (intern oder extern), falls vorhanden
3. korrekt gebildete Namen für Elementtypen, Attribute, Entities
4. genau ein Wurzelelement
5. balancierte und geschachtelte (unverschränkte) Tags
6. Attributnamen eindeutig pro Element, Wertzuweisungen in Anführungszeichen
7. kein Zeichen "<" in Attributwerten
8. keine Kommentare und Verarbeitungsanweisungen in Tags
9. keine Rekursion in Entity-Definitionen

⋮

Valide XML-Dokumente

1. wohlgeformtes Dokument
2. DTD oder Referenz auf DTD im Dokument
3. Dokumenteninhalt ist konform mit angegebener DTD

DTD

Document Type Definition (DTD)

Eine DTD deklariert die Textauszeichnungen (Markups) für ein XML-Dokument.

1. Namen in Auszeichnungen (Elementnamen und Attributnamen),
2. Modell der Inhalte von Elementen,
3. Datentypen von Attributwerten und Verwendung von Attributen,
4. Entities als Abkürzungen für Zeichenketten (Sonderzeichen oder oft benötigte lange Zeichenketten) im Inhalt von Elementinstanzen oder Attributwerten.

Die Deklaration erfolgt im XML-Dokument (internal subset) oder in einer Datei (Inhalt wie internal subset).

Grammatik:

```
DTD-intSubset ::=
        MarkupDecl*
MarkupDecl ::=
        ElementDecl | AttListDecl | EntityDecl | Comment
```

Kommentare nach XML-Syntax können in der DTD verwendet werden, aber nicht innerhalb von Deklarationen.

XML-Grundlagen

Document Type Definition (DTD): Elementdeklaration

Grammatik:

```
ElementDecl ::=
        "<!ELEMENT" Name ContentSpec ">"

ContentSpec ::=
        "(#PCDATA)" | "EMPTY" | "ANY" | Children | Mixed
```

Eine DTD-Elementdeklaration legt fest, welcher Inhalt zwischen öffnendem und schließendem Tag für den Elementtyp / das Element mit dem angegebenen XML-Namen `Name` erwartet wird. Insgesamt fünf Inhaltsmodelle sind möglich.

- `"(#PCDATA)"` – einfacher Inhalt
  Zwischen öffnendem und schließendem Tag sind Zeichenketten erlaubt (auch Länge 0, aber kein Markup, d.h. Verwendung von `"<"` statt `"<"` und von `"&"` statt `"&"`).
- `"EMPTY"` – leerer Inhalt
  Zwischen öffnendem und schließendem Tag ist kein Inhalt (auch keine Leerzeichen) erlaubt. Es kann das Kombitag `<..../>` verwendet werden.
- `"ANY"` – beliebiger Inhalt
  Zwischen öffnendem und schließendem Tag sind Zeichenketten (wie bei einfachem Inhalt) und beliebige Elementinstanzen erlaubt. Die Kindelemente müssen in der DTD deklariert sein und gemäß ihren jeweiligen Deklarationen aufgebaut sein.

XML-Grundlagen

Document Type Definition (DTD): Elementdeklaration (Fortsetzung)

Grammatik für Inhaltsmodell mit expliziten Kindelementen:

```
ElementDecl ::=
        "<!ELEMENT" Name Children ">"
Children ::= (Choice | Sequence) ("?" | "*" | "+")?
Choice ::= "(" cp ( "|" cp )+ ")"
Sequence ::= "(" cp ( "," cp )* ")"
cp ::= (Name | Choice | Sequence) ("?" | "*" | "+")?
```

Eine DTD-Elementdeklaration legt fest, welcher Inhalt zwischen öffnendem und schließendem Tag für den Elementtyp / das Element mit dem angegebenen XML-Namen `Name` erwartet wird. Insgesamt fünf Inhaltsmodelle sind möglich.

- `Children` – explizite Kindelemente
  Erlaubt sind Listen (Sequence, mindestens ein Element) oder Alternativen (Choice, mindestens zwei Alternativen) von Kindelementen.

  Listen und Alternativen können geschachtelt werden sowie mit Anzahl-Constraints versehen werden(ohne Angabe genau einmal, `"*"` beliebig oft, `"+"` mindestens einmal, `"?"` keinmal oder einmal, d.h. optional).

  (Auch bei Schachtelung von Listen und Alternativen sind die angegebenen Elemente immer direkte Kindelemente.)

XML-Grundlagen

Document Type Definition (DTD): Elementdeklaration (Fortsetzung)

Grammatik für gemischtes Inhaltsmodell:

```
ElementDecl ::=
        "<!ELEMENT" Name Mixed ">"

Mixed ::=
        "(#PCDATA" ("|" Name)+ ")*"
```

Eine DTD-Elementdeklaration legt fest, welcher Inhalt zwischen öffnendem und schließendem Tag für den Elementtyp / das Element mit dem angegebenen XML-Namen `Name` erwartet wird. Insgesamt fünf Inhaltsmodelle sind möglich.

- `Mixed` – gemischter Inhalt
  Zwischen öffnendem und schließendem Tag sind Zeichenketten (wie bei einfachem Inhalt) und Elementinstanzen erlaubt. Die möglichen Kindelemente sind aufgelistet, sie können aber in beliebiger Reihenfolge und Anzahl auftreten.

  Beispiel:
  `<!ELEMENT a ( #PCDATA | b | c | d )* >`
  In eine Elementinstanz von `a` können Zeichenketten und beliebig viele Kindelemente vom Typ `b`, `c` und `d` in beliebiger Reihenfolge und Häufigkeit auftreten.

XML-Grundlagen

Document Type Definition (DTD): Attribut(listen)deklaration

Grammatik:

```
AttListDecl ::=
        "<!ATTLIST" ElementName AttDef* ">"

AttDef ::=
        AttributeName AttType DefaultDecl
```

Eine DTD-Attributlistendeklaration legt fest, welche Attribute für einen Elementtyp / dein Element erlaubt sind. Außer dem Namen der Attribute werden die möglichen Werte (Attributtyp) und die Verwendung (Default-Deklaration) festgelegt.

- `ElementName, AttributeName`
  Für Element und Attribut sind nur XML-Namen zulässig.
- `DefaultDecl`

Default-Deklaration		**Semantik**	
Keyword	Default	Verwendung	Wert, falls Attribut fehlt
`#IMPLIED`		optional	unbekannt
`#REQUIRED`		obligatorisch Attribut darf nicht fehlen.	—
	"*Wert*"	optional	Default-Wert
`#FIXED`	"*Wert*"	optional Attributwert darf nur Default-Wert sein.	Default-Wert

XML-Grundlagen

Document Type Definition (DTD): Attribut(listen)deklaration (Fortsetzung)

Grammatik:

```
AttListDecl ::=
        "<!ATTLIST" ElementName AttDef* ">"

AttDef ::=
        AttributeName AttType DefaultDecl
```

Eine DTD-Attributlistendeklaration legt fest, welche Attribute für einen Elementtyp / dein Element erlaubt sind. Außer dem Namen der Attribute werden die möglichen Werte (Attributtyp) und die Verwendung (Default-Deklaration) festgelegt.

- `AttType` (vereinfacht)

Datentyp für Attribut	mögliche Werte
`CDATA`	beliebige Zeichenketten (kein "<", aber Entity-Referenzen)
`ID, IDREF`	Namen
`NMTOKEN`	Zeichenketten fast wie Namen (Anfangsbuchstabe frei)
`IDREFS`	Listen von Namen, durch Leerzeichen getrennt
`NMTOKENS`	Listen von Zeichenketten fast wie Namen, durch Leerzeichen getrennt
(*Wert1* | *Wert2* | *Wert3*)	nur aufgezählte Werte, Werte selbst jeweils Zeichenketten fast wie Namen

XML-Grundlagen

Document Type Definition (DTD): Entitydeklaration

Grammatik:

```
EntityDecl ::=
        "<!ENTITY" Name EntityValue ">"

EntityRef ::=
        "&" Name";"
```

Eine DTD-Entitydeklaration legt einen Namen für eine Zeichenfolge fest.

Enities werden verwendet, indem eine Entityreferenz angegeben wird. Dies kann überall dort geschehen, wo Zeichenketteninhalt vorgesehen ist (in Elementen zwischen öffendem und schließendem Tag und in Attributwerten).

- Der Wert in `EntityValue` ist in einfache oder doppelte Anführungszeichen eingeschlossen.
- Enthält die Zeichenfolge doppelte Anführungszeichen, so muss sie in der Deklaration in einfache Anführungszeichen eingeschlossen werden und umgekehrt. Beide Arten von Anführungszeichen darf die Zeichenkette nicht enthalten.
- Der Wert in `EntityValue` kann Entityreferenzen (z.B. `<`) und Zeichenreferenzen (z.B. `&#x20;`) enthalten.