Freie Universität Berlin

XML-Technologien
Tutorium 2

Themen:	von:
Warum validieren? DTD	Kain Kordian Gontarska kainkordian@gmail.com

Warum validieren?

- Wohlgeformtheit reicht nicht aus um konsistente Daten zu erzeugen.

```
<Items>
  <Item id="1" name="Snickers"/>
  <item>
    <Id>two</Id>
    <name>Mars</name>
  </item>
  <ITEM ID="3"><Name>Milky-
Way</Name></ITEM>
</Items>
```

- Wir brauchen weitere Regeln, um fest zu legen wie die Dokumente konkret aussehen sollen.

```
<Items>
  <Item id="1" name="Snickers"/>
  <Item id="2" name="Mars"/>
  <Item id="3" name="Milky-Way"/>
</Items>
```

DTD – Document Type Definition

- Ist ein Regelsatz zur Deklaration eines bestimmten Dokumententyps.
- In einer DTD wird die Struktur eines Dokuments festgelegt.
- Dient unter anderem dem Erstellen einer SGML/XML Anwendung (Bsp. HTML/XHTML)
- DTD ist Bestandteil der XML-Spezifikation, obwohl DTD nicht in XML-Syntax gehalten ist.
- DTD wird in XML zwischen XML-Deklaration und Wurzelelement eingebunden.
- Entweder In XML-Dokument integriert oder Referenz auf externe DTD

DTD - Deklaration

- 3 Möglichkeiten der Deklaration
  - 1. Externe DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE RootName SYSTEM "myDTD.dtd">
<RootName>Hello, World!</RootName>
```

  - 2. Externe publizierte DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE RootName PUBLIC "URI" "myDTD.dtd">
<RootName>Hello, World!</RootName>
```

  - 3. Interne DTD

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE RootName [<!ELEMENT RootName #PCDATA>]>
<RootName>Hello, World!?</Rootname>
```

DTD - Elemente

- Wir deklarieren: Elemente, Attribute und Entitäten (Dabei verwenden wir eine der kontextfreien Grammatik ähnliche Syntax)
- Elemente:
  - Können unterschiedliche Inhalte haben

```
<! ELEMENT myElem EMPTY>
<! ELEMENT myElem ANY>
<! ELEMENT myElem (#PCDATA)>
<! ELEMENT myElem (some, content)>
```

  - Häufigkeit von Elementen lässt sich angeben. (Rekursion möglich, Verschachteln beliebig)

```
<! ELEMENT myElem (child+|myElem)>
```

    - * (n>=0) ,? (n<=1) ,+ (n>0) und „nichts“ (n=1)
  - RegEx „|“

```
<! ELEMENT myElem (child1 |
child2)*>
```

  - Sequenzen (Feste Reihenfolge)

```
<! ELEMENT myElem (child3,child4)?>
```

DTD - Attribute

- Attribute werden Elementgebunden deklariert:

```
<!ATTLIST myElem
  AttrName1 AttrType1
AttrDescription1
  AttrName2 AttrType2
AttrDescription2
  ...
>
```

- Attributtyp:
  - CDATA, String-Aufzählung, NMTOKEN, ID, IDREF
- Attributbeschreibung:
  - #REQUIRED, #IMPLIED, #FIXED

DTD – Interne DTD

- DTD lässt sich direkt in XML-Dokument schreiben:
  - <?xml version="1.0" encoding="UTF-8" standalone=“yes“?>

```
<!DOCTYPE BookStore [
<!ELEMENT BookStore (Book+)>
<!ELEMENT Book (#PCDATA)>
]>
<BookStore>
  <Book/>
</BookStore>
```

DTD – Entitäten

- In DTDs definierbare Abkürzungen für Zeichenfolgen
- Wie & für & oder < für <
- <!ENTITY author "Kain">
- Auch um externe Definitionen in DTD zu importieren (Internal Entity):
  - <!ENTITY % andereDTD SYSTEM "meineAndereDTD.dtd">
- Verwendung:
  - <!ELEMENT someElement %andereDTD;>

DTD - Nachteile bzw. Wieso XML-Schema?

- Nicht in XML-Syntax → eigener Parser notwendig
- Sehr wenige Datentypen (Inhalt lässt sich zB. nicht auf Zahlen beschränken)
- Keine eigenen Datentypen definierbar
- Keine Vererbungshierarchien für Typen
- Keine Namensräume: Es darf keine Namenskonflikte bei mehreren DTDs geben
- Wenn Reihenfolge der Elemente nicht wichtig, müssen alle Permutationen definiert sein
  - <!ELEMENT CD ((Name, Interpret) | (Interpret, Name))>