Freie Universität Berlin

Musterlösung des Übungsblattes 2

ipoDefaultNS.xml: purchaseOrder

Freie Universität Berlin

```
1   <?xml version="1.0"?>
2   <purchaseOrder xmlns:xsi="
    http://www.w3.org/2001/XMLSchema-instance" xmlns="
    http://www.altova.com/IPO" xmlns:ipo="
    http://www.altova.com/IPO" orderDate="1999-12-01">
3     <shipTo export-code="1" xsi:type="ipo:EU-Address">
9     <billTo xsi:type="ipo:US-Address">
16    <Items>
57  </purchaseOrder>
```

Deklaration von purchaseOrder

Freie Universität Berlin

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT purchaseOrder (shipTo, billTo, Items)>
<!ATTLIST  purchaseOrder
   xmlns:xsi CDATA #FIXED "http://www.w3.org/2001/XMLSchema-instance"
   xmlns CDATA #FIXED "http://www.altova.com/IPO"
   xmlns:ipo CDATA #FIXED "http://www.altova.com/IPO"
   orderDate CDATA #REQUIRED >
```

Probleme:

- funktioniert nur mit Präfixen xsi und ipo
- funktioniert nur mit Standard-Namensraum: ipoDefaultNS.xml gültig, ipo.xml nicht
- Struktur von orderDate nicht eingeschränkt

ipoDefaultNS.xml: shipTo

Freie Universität Berlin

```
1   <?xml version="1.0"?>
2   <purchaseOrder xmlns:xsi="
    http://www.w3.org/2001/XMLSchema-instance" xmlns="
    http://www.altova.com/IPO" xmlns:ipo="
    http://www.altova.com/IPO" orderDate="1999-12-01">
3     <shipTo export-code="1" xsi:type="ipo:EU-Address">
4       <name>Helen Zoe</name>
5       <street>47 Eden Street</street>
6       <city>Cambridge</city>
7       <postcode>126</postcode>
8     </shipTo>
9     <billTo xsi:type="ipo:US-Address">
16    <Items>
57  </purchaseOrder>
```

Deklaration von shipTo

Freie Universität Berlin

```
<!ELEMENT shipTo (name, street, city, (postcode | (state, zip)))>
<!ATTLIST shipTo
        export-code CDATA #IMPLIED
        xsi:type (ipo:EU-Address | ipo:US-Address) #REQUIRED
>
```

Probleme:

- kein Zusammenhang zwischen Struktur von shipTo und xsi:type
- kein Zusammenhang zwischen xsi:type und export-code

ipoDefaultNS.xml: billTo

Freie Universität Berlin

```
1   <?xml version="1.0"?>
2   <purchaseOrder xmlns:xsi="
    http://www.w3.org/2001/XMLSchema-instance" xmlns="
    http://www.altova.com/IPO" xmlns:ipo="
    http://www.altova.com/IPO" orderDate="1999-12-01">
3     <shipTo export-code="1" xsi:type="ipo:EU-Address">
9     <billTo xsi:type="ipo:US-Address">
10      <name>Robert Smith</name>
11      <street>8 Oak Avenue</street>
12      <city>Old Town</city>
13      <state>AK</state>
14      <zip>95819</zip>
15    </billTo>
16    <Items>
57  </purchaseOrder>
```

Deklaration von billTo

```
<!ELEMENT billTo (name, street, city, (postcode | (state, zip)))>
<!ATTLIST billTo
        xsi:type (ipo:EU-Address | ipo:US-Address) #REQUIRED
>
```

Problem:

- kein Zusammenhang zwischen Struktur von billTo und xsi:type

ipoDefaultNS.xml: Items und item

```
1   <?xml version="1.0"?>
2   <purchaseOrder xmlns:xsi="
    http://www.w3.org/2001/XMLSchema-instance" xmlns="
    http://www.altova.com/IPO" xmlns:ipo="
    http://www.altova.com/IPO" orderDate="1999-12-01">
3     <shipTo export-code="1" xsi:type="ipo:EU-Address">
9     <billTo xsi:type="ipo:US-Address">
16    <Items>
17      <item partNum="833-AA">
18        <productName>Lapis necklace</productName>
19        <quantity>2</quantity>
20        <price>99.95</price>
21        <comment>Need this for the holidays!</comment>
22        <shipDate>1999-12-05</shipDate>
23      </item>
24      <item partNum="748-OT">
31      <item partNum="783-KL">
37      <item partNum="238-KK">
44      <item partNum="229-OB">
50      <item partNum="128-UL">
56    </Items>
57  </purchaseOrder>
```

Deklarationen von Items und item

Freie Universität Berlin

```
<!ELEMENT Items (item+)>
<!ELEMENT item (productName, quantity, price, comment?, shipDate)>
<!ATTLIST item partNum CDATA #REQUIRED>
```

oder besser

```
<!ATTLIST item partNum ID #REQUIRED>
```

XML sieht eine Möglichkeit vor, dem Parser mitzuteilen, dass der zugewiesene Wert eines bestimmten Attributs dokumentweit nur einmal vorkommen darf. Dies ist ein wichtiges Feature vor allem im Hinblick auf Script-Sprachen. Denn nur bei dokumentweit eindeutigen, identifizierenden Werten ist es möglich, ein Element über den Identifikationswert anzusprechen.

Die Wertzuweisungen an ein Attribut vom Typ ID müssen den Regeln für Namen entsprechen! (Darf also nicht mit einer Ziffer beginnen)

Siehe auch **IDREF**

Die restlichen Deklarationen

Freie Universität Berlin

```
<!ELEMENT name (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT postcode (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT productName (#PCDATA)>
<!ELEMENT quantity (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT shipDate (#PCDATA)>
```

Deklaration von shipTo und billTo

Freie Universität Berlin

```
<!ELEMENT shipTo (name, street, city, (postcode | (state, zip)))>
<!ATTLIST shipTo
        export-code CDATA #IMPLIED
        xsi:type (ipo:EU-Address | ipo:US-Address) #REQUIRED
>
```

```
<!ELEMENT billTo (name, street, city, (postcode | (state, zip)))>
<!ATTLIST billTo
        xsi:type (ipo:EU-Address | ipo:US-Address) #REQUIRED
>
```

- an zwei Stellen identische Strukturen

⇨ nicht veränderungsfreundlich

Verbesserte Version

Freie Universität Berlin

```
<!ENTITY % Address "(name, street, city,(postcodel(state, zip)))">
<!ELEMENT shipTo %Address;>
<!ELEMENT billTo %Address;>
```

⇨ wird Parameter Entity genannt

Parameter-Entitäten enthalten eine benannte Zeichenkette, die mittels %Name; an fast allen Stellen innerhalb einer DTD eingesetzt werden kann. Auf diese Weise lassen sich beispielsweise externe Dateien in eine DTD einbinden und mehrfach vorkommende Bestandteile abkürzen. Parameter-Entitäten werden wie normale Entities deklariert, wobei vor dem Elementnamen ein einzelnes Prozentzeichen steht. Beispiel:

```
<!ENTITY % datei SYSTEM "andere-datei.ent"> %datei;
<!ENTITY % foo.inhalt "(bar|doz)*">
<!ELEMENT foo %foo.inhalt;>
```

In Depth:

XML Syntax, Namensräume

XML-Deklaration

Freie Universität Berlin

<?xml version="1.0" encoding="UTF-8"?>

Attribut version

- verwendete XML-Version: "1.0" oder "1.1"
- obligatorisch

Attribut encoding

- Kodierung der XML-Datei
- optional (default: UTF-8)

Attribut standalone

- Gibt an, ob es eine zugehörige DTD oder ein XML-Schema gibt ("no") oder nicht ("yes").
- optional

Beachte: immer in dieser Reihenfolge!

Wie kann die Bedeutung von XML-Elementen festgelegt werden?

Freie Universität Berlin

- durch Zuordnung des Element-Namens zu einem Namensraum
- Namensraum wird mit einer URI identifiziert:

  z.B. http://www.w3.org/1999/xhtml
- zwei Möglichkeiten:

1. expliziter Namensraum-Präfix
   - zuerst: xmlns:myns="http://www.w3.org/1999/xhtml"
   - dann: z.B. <myns:p>
   - Wahl des Präfixes (ziemlich) egal!
3. Standard-Namensraum
   - zuerst: xmlns="http://www.w3.org/1999/xhtml"
   - dann: z.B. <p>

Welche Element-Namen sind qualified?

Freie Universität Berlin

```
<?xml version="1.0"?>
<BookStore xmlns="http://www.books.org">
    <Book>
        <Title>My Life and Times</Title>
        <Author>Paul McCartney</Author>
        <Date>July, 1998</Date>
        <ISBN>94303-12021-43892</ISBN>
        <Publisher>McMillin Publishing</Publisher>
    </Book>
</BookStore>
```

- alle Element-Namen (einschl. BookStore!) dem Standard-Namensraum zugeordnet
- alle Element-Namen daher namensraumeingeschränkt (qualified)

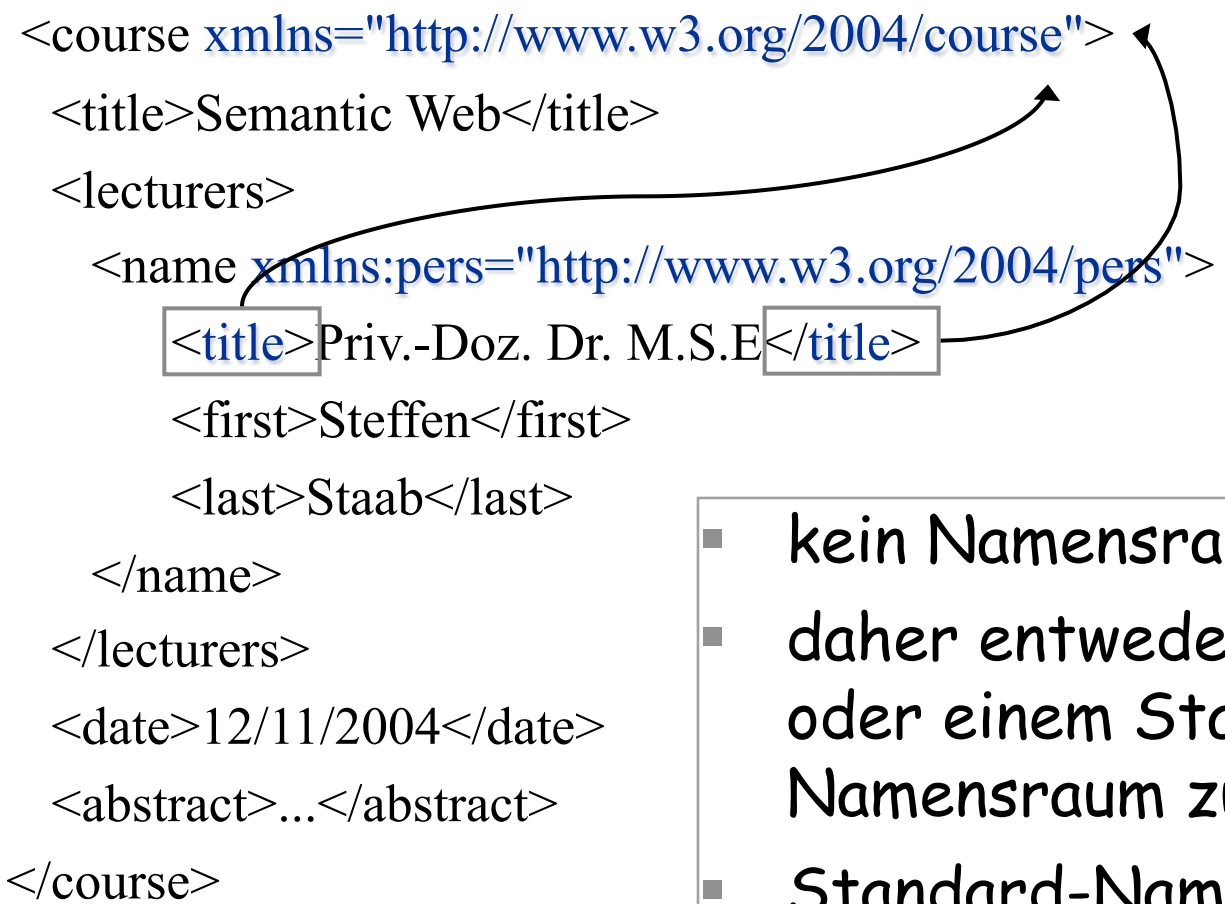
Welche Element-Namen sind qualified?

Freie Universität Berlin

```
<?xml version="1.0"?>
<bk:BookStore xmlns:bk="http://www.books.org">
    <Book>
        <Title>My Life and Times</Title>
        <Author>Paul McCartney</Author>
        <Date>July, 1998</Date>
        <ISBN>94303-12021-43892</ISBN>
        <Publisher>McMillin Publishing</Publisher>
    </Book>
</bk:BookStore>
```

- hier kein Standard-Namensraum festgelegt
- bk:Bookstore: namensraumeingeschränkt (qualified)
- alle anderen Element-Namen: nicht namensraumeingeschränkt (unqualified)

Welchem Namensraum sind die markierten Element-Namen zugeordnet?



- kein Namensraum-Präfix
- daher entweder *unqualified* oder einem Standard-Namensraum zugeordnet
- Standard-Namensraum ist hier "http://…/course"

Zu welchem Namensraum gehören die id-Attribute?

Freie Universität Berlin

```
<course xmlns="http://www.w3.org/2004/course">
  <title id="123">Semantic Web</title>
  <lecturers>
    <name id="999" xmlns="http://www.w3.org/2004/pers">
        <title>Priv.-Doz. Dr. M.S.E</title>
        <first>Steffen</first>
        <last>Staab</last>
    </name>
  </lecturers>
  <date>12/11/2004</date>
  <abstract>...</abstract>
</course>
```

- Attribute gehören *nicht* zum Standard-Namensraum.
- deshalb sind beide id-Attribute *keinem* Namensraum zugeordnet!

Wie kann ein Attribut dem Standard-Namensraum zuordnet werden?

Freie Universität Berlin

```
<course xmlns="http://www.w3.org/2004/course"
        xmlns:course="http://www.w3.org/2004/course">
  <title course:id="123">Semantic Web</title>
  <lecturers>
    <name pers:id="999" xmlns="http://www.w3.org/2004/pers"
                  xmlns:pers="http://www.w3.org/2004/pers">
      <title>Priv.-Doz. Dr. M.S.E</title>
      <first>Steffen</first>
      <last>Staab</last>
    </name>
  </lecturers>
  <date>12/11/2004</date>
  <abstract>...</abstract>
</course>
```

Üben:

Wer hats schon gemacht?

<?xml encoding="UTF-8" version="1.0"?>
<?XML version="1.0" encoding="UTF-8"?>
<?xml version="1.1" encoding="UTF-16"?>
<?xml version="1.0"?>
<?xml version="1.0" standalone="no" ?>
<?xml version="1.1" encoding="UTF-8„ standalone="no" ?>

Was ist korrekt? Unterschiede genau ansehen… und an erste Übung erinnern.

```
<?xml encoding="UTF-8" version="1.0"?>
<?XML version="1.0" encoding="UTF-8"?>
<?xml version="1.1" encoding="UTF-16"?>                    √
<?xml version="1.0"?>                                      √
<?xml version="1.0" standalone="no" ?>                     √
<?xml version="1.1" encoding="UTF-8„ standalone="no" ?>    √
```

„xml" klein schreiben!
Reihenfolge beachten!
Version pflicht!
Encoding optional
Standalone optional

Worum handelt es sich beim folgenden XML-Code?

```
<Inhalt>
    <anfang>Der Anfang</anfang>
    <ende>Das Ende</ende>
</Inhalt>
```

Andere Frage: welche Arten von Inhalten gibt es in XML?

Worum handelt es sich beim folgenden XML-Code?

```
<Inhalt>
    <anfang>Der Anfang</anfang>
    <ende>Das Ende</ende>
</Inhalt>
```

Andere Frage: welche Arten von Inhalten gibt es in XML?

- ☐ unstrukturierter Inhalt
- ☐ strukturierter Inhalt
- ☐ gemischter Inhalt
- ☐ leerer Inhalt

Worum handelt es sich beim folgenden XML-Code?

```
<Inhalt>
    <anfang>Der Anfang</anfang>
    <ende>Das Ende</ende>
</Inhalt>
```

Andere Frage: welche Arten von Inhalten gibt es in XML?

- [ ] unstrukturierter Inhalt
- [x] strukturierter Inhalt
- [ ] gemischter Inhalt
- [ ] leerer Inhalt

XML Element-Definitionen - Gemischter Inhalt

Der Inhalt eines Elements besteht nicht nur aus Text oder weiteren Elementen, sondern er kann auch aus beiden Bestandteilen zugleich bestehen. In diesem Fall spricht man von gemischtem Inhalt. Als Beispiel nehmen wir uns einmal den HTML-Befehl <BODY> heran. Dieser kann sowohl dirkete Texteingaben, als auch eine beliebige Anzahl von Elementen enthalten:

```
<!ELEMENT body (#PCDATA | (table,p,img,br,h1,h2,...usw.))*>
```

Es sollte Ihnen das Prinzip des gemischten Inhaltes verdeutlicht haben:

```
<body>
Hier steht normaler Text (#PCDATA)
<p>...und hier steht der Inhalt des Elementes «p» </p>
</body>
```

» weiterlesen

Leere Elemente

Mit leeren Elementen sind Markup-Befehle gemeint, die für sich alleine stehen können und in die kein Text eingebunden ist. In HTML kennen wir Tags wie z.B. <IMG>, <BR> oder <INPUT>. Alle HTML-Befehle, die kein abschliessendes Tag benötigen, sind solche «leeren Elemente». In XML muss die Schreibweise dieser leeren Markups etwas anders lauten:

```
<BR></BR>
```

Meistens jedoch beendet man ein leeres Element, indem man das Element mit der Zeichenfolge «/>» schliesst.

```
<BR/>
```

Inhalt von Elementen

Freie Universität Berlin

1. unstrukturierter Inhalt:
   - einfacher Text ohne Kind-Elemente

2. strukturierter Inhalt:
   - Sequenz von > Ø Kind-Elementen

3. gemischter Inhalt:
   - enthält Text mit mind. einem Kind-Element

4. leerer Inhalt

PCDATA

Freie Universität Berlin

- Reservierte Symbole < und & in PCDATA nicht erlaubt.
- Symbole wie >, /, (,), {, }, [,], % allerdings erlaubt
- statt < und & Entity References & bzw. < benutzen
- Entity References in XML:

&amp;	⇨	&
&lt;	⇨	<
&gt;	⇨	>
&apos;	⇨	'
&quot;	⇨	"

CDATA

- Unstrukturierten Inhalt mit vielen reservierten Symbolen besser als Character Data (CDATA) darstellen.
- Beispiel:

```
<formula>
  <![CDATA[ X < Y & Y < Z ]]>
</formula>
```

- Inhalt: String zwischen inneren Klammern [ ]
  hier: X < Y & Y < Z
- XML-Parser sucht in CDATA lediglich]]>, analysiert den Inhalt aber ansonsten nicht.
- "]]>" als Inhalt von CDATA nicht erlaubt

DTD…

<!ELEMENT Keksfabrik(Bäcker+, Kunden*, Chef?)>

Was bedeutet das? Beschreibe in einem Satz…

Formuliere eine DTD für folgendes XML

```
<Bäckerei>
        <Bäcker>Jens</Bäcker>
        <Ofen typ=„Steinofen“ />
</Bäckerei>
```

Beachte: Es darf nur einen Bäcker geben. Der Ofentyp ist optional, im Zweifelfall handelt es sich immer um einen Backofen…

Formuliere eine DTD für folgendes XML

```
<Bäckerei>
        <Bäcker>Jens</Bäcker>
        <Ofen typ=„Steinofen“ />
</Bäckerei>
```

Beachte: Es darf nur einen Bäcker geben. Der Ofentyp ist optional, im Zweifelfall handelt es sich immer um einen Backofen… Ein Ofen ist definitiv nötig…

```
<!ELEMENT Bäckerei (Bäcker,Ofen)>
<!ELEMENT Bäcker (#PCDATA)>
<!ELEMENT Ofen EMPTY>
<!ATTLIST Ofen
                typ CDATA #IMPLIED „Backofen“>
```

DTD: Kurzübersicht

Freie Universität Berlin

Der Inhalt eines Elementes kann durch die Angabe anderer Elementnamen und durch einige Schlüsselwörter und Zeichen angegeben werden.
EMPTY für keinen Inhalt
ANY für beliebigen Inhalt
, für Reihenfolgen
| für Alternativen (im Sinne „entweder…oder")
() zum Gruppieren
* für beliebig oft
+ für mindestens einmal
? für keinmal oder genau einmal
Wird kein Stern, Pluszeichen oder Fragezeichen angegeben, so muss das Element genau einmal vorkommen

Ist das Wohlgeformt?

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
    <s:Header>
        <m:RequestID xmlns:m="http://www.lecture-db.de/soap">a3f5c109b</m:RequestID>
    </s:Header>
    <s:Body>
        <m:DbResponse xmlns:m="http://www.lecture-db.de/soap' >
            <m:title value="DOM, SAX und SOAP">
                <m:Choice value="1">Arbeitsbericht Informatik</m:Choice>
                <m:Choice value="2">Seminar XML und Datenbanken
            </m:title>
        </m:DbResponse>
    </s:Body>
</s:Envelope>
```

Lernziel: Verstöße gegen die Regeln der Wohlgeformtheit erkennen können um kryptische Parsermeldung zu verstehen…

<xsd:element name=„Bäcker" type="xsd:string" minOccurs=„1" maxOccurs="unbounded"/>

Wie oft darf das Element "Bäcker" in einem XML-Dokument vorkommen, wenn es dieser Definition entsprechen soll?

Welche Arten von Parsern gibt es?

Welche Arten von Parsern gibt es?

Pull/push

one-step/multi-step

validating/non-validating

Wann benutzt man welche Parser?

Was macht ein SAX-Parser bei Syntax- und Struktur-Fehlern im XML-Dokument?

Wie geht ein SAX-Parser mit Syntax- und Struktur-Fehlern im XML-Dokument um?

- ☐ SAX-Parser können Syntax-Fehler abfangen, jedoch keine Strukturfehler
- ☐ SAX-Parser können Struktur-Fehler abfangen, jedoch keine Syntax-Fehler
- ☐ SAX-Parser können Syntax- und Struktur-Fehler abfangen

Wie geht ein SAX-Parser mit Syntax- und Struktur-Fehlern im XML-Dokument um?

- [ ] SAX-Parser können Syntax-Fehler abfangen, jedoch keine Strukturfehler
- [ ] SAX-Parser können Struktur-Fehler abfangen, jedoch keine Syntax-Fehler
- [x] SAX-Parser können Syntax- und Struktur-Fehler abfangen

Welche Eigenschaften hat ein DOM-Parser?

Welche Eigenschaften hat ein DOM-Parser?

- [ ] Ein DOM-Parser liefert in einem Schritt einen kompletten DOM-Parse-Baum
- [ ] Ein DOM-Parser ist ein Einschritt-Push-Parser
- [ ] Ein direkter Zugriff auf die Elemente des DOM-Parse-Baumes über den Namen ist möglich
- [ ] Ein DOM-Parser erlaubt nicht das Modifizieren und Erstellen von XML-Dokumenten

Welche Eigenschaften hat ein DOM-Parser?

x	Ein DOM-Parser liefert in einem Schritt einen kompletten DOM-Parse-Baum
	Ein DOM-Parser ist ein Einschritt-Push-Parser
x	Ein direkter Zugriff auf die Elemente des DOM-Parse-Baumes über den Namen ist möglich
	Ein DOM-Parser erlaubt nicht das Modifizieren und Erstellen von XML-Dokumenten

Schreibe einen XPath-Ausdruck der alle book Elemente mit einem author als *Kind-Element* im gesamten Dokument auswählt:

Schreibe einen XPath-Ausdruck der alle book Elemente mit einem author als *Kind-Element* im gesamten Dokument auswählt:

book[author]

Schreibe einen XPath-Ausdruck der alle book Elemente mit einem author als *Kind-Element* im gesamten Dokument auswählt:

Eigentlich:

```
//book[author]
```

Wieso das?

Welche Knoten werden mit folgendem Ausdruck beschrieben:

order/item[@item-id]

- [ ] item-Elemente, die Kind von order sind und Attribut item-id haben
- [ ] order-Elemente, die Attribut item-id haben
- [ ] order-Elemente, die item-Elemente als Kinder mit dem Attribut item-id haben
- [ ] die Werte von dem Attribut item-id

Welche Knoten werden mit folgendem Ausdruck beschrieben:

order/item[@item-id]

- [x] item-Elemente, die Kind von order sind und Attribut item-id haben
- [ ] order-Elemente, die Attribut item-id haben
- [ ] order-Elemente, die item-Elemente als Kinder mit dem Attribut item-id haben
- [ ] die Werte von dem Attribut item-id

Wie ist die Beziehung zwischen **SGML**, **HTML**, **XML** und **XHTML (und HTML5)**.

http://www.ib.hu-berlin.de/~wumsta/sgml/xmlsgml.html

http://webdesign.about.com/od/sgml/a/how-are-sgml-html-and-xml-related.htm

In Depth: Entities

Entities

Freie Universität Berlin

Built-in Entities

- e.g. & or >
- predefined

Character Entities

- e.g. ó or ó
- predefined

Entities

Freie Universität Berlin

General Entities

<!ENTITY empty-x-y "<x></x><y></y>">

- Entity name is a valid XML name
- Entity value can be anything as long as it is well formed
- Entity values can contain other entities!
- Entity reference in XML document: &empty-x-y;
- Can be used in DTD where they will eventually be included in the XML document, e.g. attribute default value

External Parsed Entities

Freie Universität Berlin

- Entity values can be external to the entity declaration

<!ENTITY footer SYSTEM „/mypages/footer.xml“>

- As element CDATA but not allowed in attributes
- Validating parsers must retrieve external entities
- The resulting document (after insertion of an external entity) must be well formed, the entity value itself must not be well formed
- There are also external unparsed entities, for non-XML values, but behaviour is unpredictable, i.e. avoid!

Entities

Freie Universität Berlin

Parameter Entities

<!ENTITY % shared „childelement1 childelement2">

- Element and attribute declarations may re-use structure
- Parameter entities allow this shared structure to be placed in one location
- Entity reference in DTD: %shared;

<!ELEMENT parentelement %shared;>

Internal and External DTD Subset

Freie Universität Berlin

- <!DOCTYPE BookStore SYSTEM „book.dtd“ [
  <!ELEMENT BookStore (Book|BookStore)*>
  ]>

- Between [...] is the internal DTD subset
- After the SYSTEM keyword is the external DTD subset

- DTD subsets can not override each others‘ element and attribute declarations
- However, entity declarations can be overriden

Overriding Entities

Freie Universität Berlin

- Parameter entities are more useful with external DTDs
- They can be redefined by the internal DTD subset
- If case of a conflict, the internal DTD subset has precedence
- External parameter entities are also useful to split large DTDs into smaller external subsets
- To include them at validation time, declare an external parameter entity and then use the entity reference

<!ENTITY % names SYSTEM „names.dtd“>

%names;

Entities in DTDs and XML Schema

Freie Universität Berlin

- Shocking but true: there is something in DTDs that can not be done in XML Schema...

declaring entities!