Übungsblatt 3 WT:III

Bis zum 23.05.2019 sind Lösungen zu folgenden Aufgaben abzugeben: 1a,b, 2, 4a, 5a,b,d,e, 3, 6.

Hinweis: Auf utilities-online.info/xsdvalidation/ können Sie Instanzdokumente validieren.

Aufgabe 1 : Modellierung in XML Schema (2+2+0+0 Punkte)

Sie haben den Auftrag Abbildungen wie in wissenschaftlichen Arbeiten (zum Beispiel Abbildung 1) zu modellieren, um diese in XML repräsentieren zu können. Gehen Sie für diese Aufgabe davon aus, dass Abbildungen jeweils genau ein Bild enthalten.

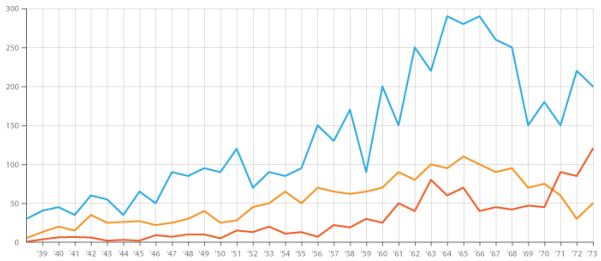

Abbildung 1: Wichtige Ergebnisse.

(a) Erstellen Sie ein XML-Schema das die folgenden Anforderungen an die Modellierung implementiert:

- Eine Abbildung (`<abbildung>`) kann (muss aber nicht) eine Abbildungs-Nummer (`<nummer>`) vom Typ `xsd:integer` haben.
- Wie in HTML soll das eigentliche Bild (`<bild>`, der Graph in Fall von Abbildung 1) als leeres Element im Instanzdokument auftreten und die Quelldatei durch ein Pflichtattribut (hier `quelle`) spezifiziert werden.
- Eine Abbildung enthält genau eine Beschriftung (`<beschriftung>`, im Beispiel der Text nach „Abbildung 1:“).

(b) Welche Inhaltsmodelle verwenden jeweils die Elementtypen `abbildung`, `nummer`, `bild` und `beschriftung` in Ihrem Schema?

(c) Erweitern Sie ihr XML-Schema, so dass es zusätzlich die folgenden Anforderungen an die Modellierung implementiert:

- Ein weiteres Attribut des eigentlichen Bildes soll die Ausrichtung (`ausrichtung`) sein, das die Werte "links", "mittig" oder "rechts" haben kann.
- Der Defaultwert der Ausrichtung soll "links" sein.

(d) Erweitern Sie ihr XML-Schema, so dass es zusätzlich die folgenden Anforderungen an die Modellierung implementiert:

- Die Beschriftung kann vor oder nach dem eigentlichen Bild kommen um Bildüberschriften und Bildunterschriften zu unterscheiden.

Geben Sie das Instanzdokument, dass sie zum testen benutzen, mit ab.

© Stein/Kiesel/Wiegmann/Bevendorff 2019

Aufgabe 2 : XML Namensräume (1+1 Punkte)

(a) Wieso ist das folgende XML-Dokument nicht wohlgeformt?

```
<?xml version="1.0"?>
<myNs1:myElem1 xmlns:myNs1="http://www.uni-weimar.de/myNamespace1"
    xmlns:myNs2="http://www.uni-weimar.de/myNamespace1">
  <myNs2:myElem2 myNs1:myAtt="Test1" myNs2:myAtt="Test2">
    myContent
  </myNs2:myElem2>
</myNs1:myElem1>
```

(b) Gegeben sei das folgende XML-Instanzdokument mit Verwendung verschiedener Namensräume:

```
<?xml version="1.0"?>
<pre1:elem1 xmlns:pre1="http://www.example.org/ns1">
  <elem2 att1="w1"/>
  <pre1:elem3 xmlns:pre2="http://www.example.org/ns2">
    <elem4 xmlns="http://www.example.org/ns3">
      <elem5 att2="w2" pre1:att3="w3" pre2:att4="w4"/>
      <pre2:elem6 att5="w5" pre2:att6="w6"/>
      <pre4:elem7 xmlns:pre4="http://www.example.org/ns2"
          pre4:att7="w7" att8="w8"/>
    </elem4>
  </pre1:elem3>
</pre1:elem1>
```

Geben Sie die verwendeten Namensräume (Bezeichner oder anonym) an und welche Namen aus diesen Namensräumen verwendet werden. Beispiel:

Namensraum: `http://www.example.org/ns1`

Elementnamen: `elem1, elem3`

Attributnamen: `att3`

Aufgabe 3 : XML/DTD (3 Punkte)

Gegeben sei die folgende DTD zur Repräsentation von Katalogen.

```
<!ELEMENT CATALOG (PRODUCT+)>
<!ELEMENT PRODUCT (SPECIFICATIONS+, PRICE)>
<!ELEMENT SPECIFICATIONS (CATEGORY?, TEXT)>
<!ELEMENT PRICE (#PCDATA)>
<!ELEMENT CATEGORY (#PCDATA)>
<!ELEMENT TEXT (#PCDATA)>
```

(a) Erstellen Sie ein valides XML-Instanzdokument mit Wurzelelement `CATALOG` für die obige DTD.

(b) Repräsentieren Sie die Elementstruktur Ihres Dokumentes als Baum und kennzeichnen Sie alle Elementknoten, die nach DTD für die Validität erforderlich waren.

© Stein/Kiesel/Wiegmann/Bevendorff 2019

Aufgabe 4 : Modellierungsvarianten in XML-Schema (2+0+0+0 Punkte)

Gegeben sei ein XML-Fragment, das die Leistungen einer Werkstatt zu einem PKW für einen Kunden zusammenfasst:

```
<ServiceListe FahrgestellNR="1234554356-34" KundenNR="M-1998-023">
  <Service Auftragsnummer="2005-1567">
    <Datum>2005-04-10</Datum>
    <KM>9808</KM>
    <Anlass>Inspektion</Anlass>
    <Arbeiten>Kontrolle und Ölwechsel durchgeführt</Arbeiten>
  </Service>
    <Service Auftragsnummer="2006-2745">
    <Datum>2006-05-06</Datum>
    <KM>25823</KM>
    <Anlass>Inspektion</Anlass>
    <Arbeiten>Kontrolle; defektes Bremslicht (HL)</Arbeiten>
  </Service>
    <Service Auftragsnummer="2008-0641">
    <Datum>2008-02-03</Datum>
    <KM>43076</KM>
    <Anlass>Rückruf</Anlass>
    <Arbeiten>Bremsschläuche ausgetauscht</Arbeiten>
  </Service>
</ServiceListe>
```

(a) Erstellen Sie ein XML-Schema zur Beschreibung von Dokumenten dieser Art. Legen Sie insbesondere einen Zielnamensraum fest, der alle deklarierten Elemente und Attribute enthalten soll. Verwenden Sie zunächst das Russian Doll Prinzip für die Modellierung.

(b) Vervollständigen Sie das obige Fragment zu einem validen Instanz-Dokument für das in der vorherigen Teilaufgabe erstellte Schema. Stellen Sie also mit passenden Angaben die Verwendung dieses Namensraumes und die Verbindung zu Ihrem Schema-Dokument sicher.

(c) Verwenden Sie in einer zweiten Version globale Elementdeklarationen. Welche Auswirkungen hat dies auf die Attribute `elementFormDefault="qualified"` und `attributeFormDefault="qualified"` bzw. die Instanzdokumente?

(d) Verwenden Sie benannte Datentypen.

Validieren Sie die Instanzdokumente gegen Ihre XML-Schema Versionen.

© Stein/Kiesel/Wiegmann/Bevendorff 2019

Aufgabe 5 : DTD (0.5+0.5+0+0.5+0.5 Punkte)

In den nachfolgenden Teilaufgaben wird jeweils eine Elementdeklaration angegeben sowie eine Reihe von Elementinstanzen. Beantworten Sie jeweils die folgenden Fragen für die Elementdeklaration:

1. Welches Inhaltsmodell verwendet die Elementdeklaration?

Beantworten Sie jeweils die folgenden Fragen für die Elementinstanzen:

2. Ist die Elementverwendung in der Instanz valide in Bezug auf die Elementdeklaration?
3. Im Fall "nicht valide": Worin besteht der Fehler?
4. Im Fall "nicht valide": Wie kann die Instanz korrigiert werden?

(a) `<!ELEMENT structure (#PCDATA)>`

(a1) `<structure></structure>`
(a2) `<structure>...</structure>`
(a3) `<structure>...<elem>...</elem></structure>`

(b) `<!ELEMENT structure EMPTY>`

(b1) `<structure></structure>`
(b2) `<structure>...</structure>`
(b3) `<structure>...<elem>...</elem></structure>`

(c) `<!ELEMENT structure ANY>`

(c1) `<structure></structure>`
(c2) `<structure>...</structure>`
(c3) `<structure>...<elem>...</elem></structure>`

(d) `<!ELEMENT structure (#PCDATA|elem|mele)*>`

(d1) `<structure></structure>`
(d2) `<structure>...</structure>`
(d3) `<structure>...<elem>...</elem></structure>`

(e) `<!ELEMENT structure (elem+)+>`

(e1) `<structure></structure>`
(e2) `<structure>...</structure>`
(e3) `<structure>...<elem>...</elem></structure>`

Hinweis: Informationen zur Definition von Elementen finden sie unter http://www.w3.org/TR/2006/REC-xml11-20060816/#elemdecls.

© Stein/Kiesel/Wiegmann/Bevendorff 2019

Aufgabe 6 : Programmiermodul: XML Schema [P] (4+2+1+1 Punkte)

Schreiben Sie ein XML Schema für einen vereinfachten Atom Feed.

(a) Schreiben Sie ein Schema für einen einzelnen Feed-Eintrag, zu dem die Datei `entry.xml` von der Kursseite valide ist. Folgende Punkte müssen durch das Schema gegeben sein:

- Jedes Kindelement von `<entry>` darf höchstens einmal vorkommen.
- Die Kindelemente `<link>` und `<summary>` sind optional, die anderen nicht.
- Die Reihenfolge ist fest: `<id>`, `<title>`, `<updated>`, `<author>`, `<link>`, `<summary>`.
- Jedes `<link>`-Element muss genau ein `href`-Attribut haben.
- Jedes `<author>`-Element muss genau ein `<name>`-Kindelement haben.

(b) Erweitern Sie das obige Schema für ganze Feeds, so dass die Datei `feed.xml` von der Kursseite ebenfalls valide ist. Die Datei `entry.xml` soll weiterhin valide und die obigen Punkte weiterhin gegeben sein. Folgende Punkte müssen zusätzlich durch das Schema gegeben sein:

- Jedes Kindelement von `<feed>` außer `<entry>` darf höchstens einmal vorkommen.
- Das Kindelement `<entry>` darf beliebig oft vorkommen.
- Die Kindelemente `<subtitle>` und `<entry>` sind optional, die anderen nicht.
- Die Reihenfolge ist fest: `<id>`, `<title>`, `<updated>`, `<author>`, `<subtitle>`, `<entry>`.
- Die `<author>`-Kindelemente von `<feed>` und `<entry>` sind von demselben Typ.

(c) Was sind Vor- und Nachteile der Festsetzung der Kindelement-Reihenfolge von `<entry>`?

(d) Wie kann eine chronologische Reihenfolge der Einträge in der Anzeige garantiert werden, auch wenn XML Schema eine solche Reihenfolge nicht erzwingen kann?

Hinweise zur Abgabe

- Erstellen Sie *eine* PDF-Datei, in der Texte und Grafiken zu den Aufgaben enthalten sind.
- Quellcode (Java, JavaScript, PHP, HTML, CSS, XML, XSL, XSD, etc.) innerhalb des PDF-Dokuments wird nicht korrigiert. Quellcode jeder Aufgabe soll, wie im Tutorium vorgestellt, über GitLab verwaltet werden.
- Referenzieren Sie Ihre Quellcode-Dateien in dem PDF-Dokument, so dass sie einer Aufgabe eindeutig zuzuordnen sind. Verlinken sie sie entsprechendern Repositories.
- Abzugeben ist *eine* PDF-Datei die wie folgt benannt ist: `<Nachname>-<MatrikelNr>-webtec-blatt<Übungsblattnummer>.pdf`.
- Gruppenabgaben mit bis zu drei Personen sind erlaubt; pro Gruppe genügt dann eine Abgabe, in der alle Gruppenmitglieder mit Namen, Matrikel, und E-Mail-Adresse verzeichnet sind
- Legen sie die PDF-Datei in das GitLab Repository in dem auch der Code ihrer Gruppe verwaltet wird.

© Stein/Kiesel/Wiegmann/Bevendorff 2019