Ngoc-Anh Dang
11082691

Aufgabe 1

Erklären Sie kurz die Begriffe Wohlgeformtheit, Validität und Namespaces im Bezug auf XML und XML-Schema.

XML bedeutet Extensible Markup Language und ist eine Metasprache/ Auszeichungssprache, dient zur hierarchischen Strukturierung von Daten in Form von Textdateien, Besonderheit: ist plattform/-implemantierungsunabhängig

XML-Schema auch XSD genannt „XML Schema Definition“, beschreibt den Aufbau/Syntax eines XML-Dokumentes (Wurzelknoten und die Kinderelemente)

Knotentyp	Verwendung	Repräsentation
Text	Träger des Inhalts (Blätter)	Text, der der Knoten beinhaltet
Element	Definiert hierarchische Struktur	Start- und Endauszeichnungen <bla>…</bla>
Attribut	Elementen zugeordnetes Wertepaar	Name=“value“ innerhalb der Auszeichnung
Wurzel	Genau einmal vorhanden	Implizit
Kommentar	Metadaten zum XML-Dokument	<!--bla-->
Processing Instruction	Anweisungen für verarbeitende Prozesse	<?..Anweisung..?>

Wohlgeformtheit

- XML-Dokument muss eine Baumstruktur aufweisen
- je Dokument nur 1 Wurzelknoten (ausgenommen sind Instructions und Kommentare)
- ein Element mit Inhalt muss zwischen einem öffnenden und schließenden Tag sein außer das Wurzelelement muss jedes Element von Elementen vollständig umschlossen sein
- ein Attributwert muss zwischen , oder „ stehen
- die verwendeten Zeichen entsprechen dem festgelegten Zeichensatz

Validität

- Ausdrücke sind regulär (Folge von Zeichen aus festgelegten Zeichenmenge oder Elementen, …) -> entspricht dem festgelegten XML-Schema
- Ausdrücke basieren auf: Alternative, Konkatenation, Wiederholung

Namespaces

- Namensräume dienen zur Festlegung des Vokabulars eines XML-Dokuments
- Verhindert „clashing names“, unterstützt sublanguages
- wird als URI dargestellt
- dazu wird das Attribut *xmls* verwendet
- wird einmal deklariert, dann werden Präfix genutzt z.B. xmls:*xsd*
- default namespace ist xmls=““;
- weitere Namensräume sind z.B.:
    - http://www.w3.org/2000/svg
    - http://www.w3.org/1998/Math/MathML

Ngoc-Anh Dang
11082691

Aufgabe 2

a) Erzeugen Sie ein XML-Dokument, dass die Daten des folgenden Formulars vollständig erfasst:
http://www.gm.fh-koeln.de/~vsch/anmeldung/gruppenanmeldung.html
Füllen Sie das Dokument mit einem Beispieldatensatz. Achten Sie darauf, dass über das Formular mehrere Personen gleichzeitig erfasst werden können.
Wichtig: Es sollte nicht die HTML-Struktur der Webseite in der XML-Datei abgebildet werden, sondern die zu übertragenden Daten.
b) Erzeugen Sie ein JSON-Dokument, dass zu ihrem XML-Dokument äquivalent ist.

---

a)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Anmeldungen>
    <Gruppe gruppen_id="1" gruppenleiter= "1">

        <Mitglied id="1" vorname="anna" nachname="bunto" email="ab@web.de"
gebdatum="01.01.92" schlagzeug="vorhanden" anmerkungen=""/>

        <Mitglied id="2" vorname="berta" nachname="cilianti" email="bc@web.de"
gebdatum="02.02.92" schlagzeug="nicht vorhanden" anmekungen=""/>

    </Gruppe>
</Anmeldungen>
```

b)

```json
{
      "id":1,
      "Gruppenleiter":1,
      "Mitglieder": [{
            "id": 1,
            "vorname":"anna",
            "name":"bunto",
            "gebdatum":"01.01.92",
            "email":"ab@web.de",
            "schlagzeug":true,
            "anmerkungen":"",
      },
      {
            "id": 2,
             "vorname":"berta",
            "name":"cilanti",
            "gebdatum":"02.02.92",
            "email":"bc@web.de",
            "schlagzeug":false,
             "anmerkungen":""
      }]
}
```

Ngoc-Anh Dang
11082691

Aufgabe 3

a) Gegeben ist folgendes Rezept:
http://www.chefkoch.de/rezepte/24641006006067/Lenchen-s-Schokoladenkuchen.html
Entwickeln Sie ein XML-Dokument, in dem die Daten des Rezeptes abgebildet werden. Achten Sie darauf, dass das Dokument semantisch möglichst reichhaltig ist. Bei dieser und den folgenden Aufgaben lassen sie bitte die Daten in der Marginalspalte auf der rechten Seite weg.
b) Betrachten Sie nun andere Rezepte auf der Webseite http://www.chefkoch.de. Beschreiben Sie welche Gemeinsamkeiten die Rezepte hinsichtlich ihrer Daten haben und worin Sie sich unterscheiden.
c) Arbeiten Sie die Kriterien heraus, die für die Entwicklung einer XML-Schema-Datei beachtet werden müssen. Die Schema-Datei soll die Struktur für eine XML-Datei definieren, in der mehrere unterschiedliche Rezepte gespeichert werden können. Ziel ist es, dass das XML-Schema möglichst restriktiv (einschränken) ist, so dass in der XML-Datei möglichst semantisch sinnvolle Daten bezüglich der Rezepte gespeichert werden können. Ziehen Sie beim Aufstellen der Kriterien u.A. folgende Fragestellungen in Betracht:
Welche Daten müssen in simple und welche in complex-types abgebildet werden?
Für welche Daten ist die Abbildung in Attributen sinnvoller?
Welche Datentypen müssen für die Elemente definiert werden?
Welche Restriktionen müssen definiert werden?
d) Erstellen Sie nun ein XML-Schema auf Basis ihrer zuvor definierten Kriterien. Generieren Sie nun auf Basis des Schemas eine XML-Datei und füllen Sie diese mit zwei unterschiedlichen und validen Datensätzen.

---

a)

```
<?xml version="1.0" encoding="UTF-8"?>
<chefkochbuch>
    <rezept id="1" stunde="1" minuten="0" schwierigkeit="mittel" kcal="295"
portionen="16">
        <zutaten>
           <zutat name="Butter" menge="200" einheit="g" />
           <zutat name="Zucker" menge="200" einheit="g" />
           <zutat name="Schokolade" menge="200" einheit="g" />
           <zutat name="Mehl" menge="120" einheit="g" />
           <zutat name="Backpulver" menge="0.5" einheit="TL" />
           <zutat name="Vanillezucker" menge="1" einheit="Pck." />
           <zutat name="Eier" menge="4" einheit="Stk." />
        </zutaten>
        <schritte>
           <schritt>Butter und Schokolade im Wasserbad schmelzen. </schritt>
           <schritt>Eier trennen. </schritt>
           <schritt>Eiwei steif schlagen. </schritt>
           <schritt>Eigelbe, Zucker und Vanillezucker verrühren.</schritt>
           <schritt>Geschmolzene Butter-Schokomasse hinzufgen und mischen.</schritt>
           <schritt>Mehl mit dem Backpulver in die Masse sieben und zum Schluss die
steifen Eiweiß vorsichtig unterheben.</schritt>
           <schritt>In eine gut gefettete Form geben. </schritt>
           <schritt>Bei 180 Grad 40-50 Minuten backen.</schritt>
        </schritte>
    </rezept>
</chefkochbuch>
```

b)

Gemeinsamkeiten

- Grober Aufbau: Allgemeine Daten zum Rezept (Name, Portionen, …), Zutaten (mit Attributen wie Name und Menge), Zubereitungsschritte

Unterschiede

- Verschiedene und optionale Attribute (Einheit = nach Belieben, Ruhezeit, Koch-/Backzeit,…)
- Manche enthalten Kommentare, manche nicht

c)

XML-Schema XSD (XML Schema Definition), zum Definieren von Strukturen für XML-Dokumente, wird in Form eines XML-Dokuments beschrieben, große Anzahl von Datentypen wird unterstützt, beschreibt in einer komplexen Schemasprache Datentypen, einzelne XML-Schema-Instanzen (Dokumente) und Gruppen solcher Instanzen, Dateiendung „.xsd“

Welche Daten müssen in simple und welche in complex-types abgebildet werden?

- Simple: schritt, alle Attribute, zutat
- Complex: chefkochbuch, rezept, zutaten, schritte

Für welche Daten ist die Abbildung in Attributen sinnvoller?

- z.B. wenn sie einen bestimmten Wertebereich haben..?

Welche Datentypen müssen für die Elemente definiert werden?

- Simple(String, positiveInteger,), complex

Welche Restriktionen müssen definiert werden?

- Keine negative Mengen
- Richtige Einheiten der Mengen
- Bei Schwierigkeiten Wertebereich, nur: leicht, mittel, schwer
- Die Angabe von Minuten dürfen nur zwischen 0 und 59 sein

d)

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified">

      <xsd:element name="chefkochbuch" type="kochbuch_typ"/>

      <xsd:element name="rezept">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element ref="zutaten"/>
                        <xsd:element ref="schritte"/>
                        <xsd:element ref="kommentar" maxOccurs="unbounded"/>
                  </xsd:sequence>
                  <xsd:attribute name="id" type="xsd:positiveInteger" use="required"/>
                  <xsd:attribute name="name" type="xsd:string" use="required"/>
                  <xsd:attribute name="stunde" type="xsd:positiveInteger"
use="optional"/>
```

```
                    <xsd:attribute name="minuten" type="min" use="optional"/>
                    <xsd:attribute name="schwierigkeit" type="stufe" use="optional"/>
                    <xsd:attribute name="kcal" type="xsd:positiveInteger" use="optional"/>
                    <xsd:attribute name="portionen" type="xsd:positiveInteger"
use="required"/>
             </xsd:complexType>
      </xsd:element>

      <xsd:element name="zutaten" type="zutaten_typ"/>
      <xsd:element name="schritte" type="schritte_typ"/>
      <xsd:element name="zutat" type="zutat_typ" />
      <xsd:element name="schritt" type="xsd:string"/>
      <xsd:element name="kommentar" type="xsd:string"/>

      <xsd:complexType name="kochbuch_typ">
            <xsd:sequence>
                  <xsd:element ref="rezept" maxOccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="zutaten_typ">
            <xsd:sequence>
                  <xsd:element ref="zutat" maxOccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="zutat_typ">
            <xsd:attribute name="menge" type="xsd:positiveInteger" use="required"/>
            <xsd:attribute name="einheit" type="unit" use="required"/>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
      </xsd:complexType>

      <xsd:complexType name="schritte_typ">
            <xsd:sequence>
                  <xsd:element ref="schritt" maxOccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complexType>

      <xsd:simpleType name="min">
            <xsd:restriction base="xsd:integer">
                  <xsd:minInclusive value="0"/>
                  <xsd:maxInclusive value="59"/>
            </xsd:restriction>
      </xsd:simpleType>

      <xsd:simpleType name="stufe">
            <xsd:restriction base="xsd:string">
                  <xsd:pattern value="leicht|mittel|schwer"/>
            </xsd:restriction>
      </xsd:simpleType>

      <xsd:simpleType name="unit">
            <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="g"/>
                  <xsd:enumeration value="TL"/>
                  <xsd:enumeration value="EL"/>
                  <xsd:enumeration value="Pck."/>
                  <xsd:enumeration value="Stk."/>
                  <xsd:enumeration value="Tasse"/>
                  <xsd:enumeration value="Scheibe"/>
            </xsd:restriction>
      </xsd:simpleType>

</xsd:schema>
```

Ngoc-Anh Dang
11082691

1. Testdatensatz

```
<?xml version="1.0" encoding="UTF-8"?>
<chefkochbuch xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="file:/C:/Users/Ngoc-
Anh/Dropbox/FH/4.%20Semester/WBA2/Phase1/aufgabe3d_chefkochbuch_schema-kommentare.xsd">
    <rezept id="1" name="Kakao" portionen="1">
        <zutaten>
            <zutat menge="1" einheit="Tasse" name="Milch"/>
            <zutat menge="3" einheit="TL" name="Kakaopulver"/>
        </zutaten>
        <schritte>
            <schritt>Fülle die Tasse mit Milch.</schritt>
            <schritt>Rühre den Kakaopulver ein.</schritt>
        </schritte>
    </rezept>
</chefkochbuch>
```

2. Testdatensatz

```
<?xml version="1.0" encoding="UTF-8"?>
<chefkochbuch xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="file:/C:/Users/Ngoc-
Anh/Dropbox/FH/4.%20Semester/WBA2/Phase1/aufgabe3d_chefkochbuch_schema-kommentare.xsd">
    <rezept id="2" name="Salamibrötchen" portionen="1">
        <zutaten>
            <zutat menge="1" einheit="Stk." name="Brötchen"/>
            <zutat menge="1" einheit="Scheibe" name="Salami"/>
        </zutaten>
        <schritte>
            <schritt>Brötchen aufschneiden.</schritt>
            <schritt>Salami reinlegen.</schritt>
        </schritte>
    </rezept>
</chefkochbuch>
```

Ngoc-Anh Dang
11082691

Aufgabe 4

In dieser Aufgabe entwickeln Sie mit Hilfe des JAXB Frameworks ein Java-Programm, welches die XML-Datei aus der vorigen Aufgabe einlesen, modifizieren und ausgeben kann.
a) Erzeugen Sie zunächst aus der Schema-Datei der vorherigen Aufgabe Java-Objekte. Nutzen Sie dazu den XJC-Befehl über das Terminal und fügen Sie die generierten Klassen ihrem Java-Projekt hinzu. Alternativ zur Terminal-Eingabe existiert ein JAXB Eclipse Plug-In welches hier herunter geladen werden kann: http://sourceforge.net/projects/jaxb-builder.
b) Entwickeln Sie nun das Java-Programm. Es soll die XML-Datei öffnen, einlesen und die enthaltenen Daten über die Konsole wieder ausgeben. Benutzen Sie bitte bei der Bearbeitung der Aufgabe die generierten JAXB-Klassen aus der vorherigen Teilaufgabe.
c) Erweitern Sie ihr Programm so, dass es möglich ist, über die Konsole neue Kommentare zu einem Rezept hinzuzufügen. Benutzen Sie auch hierfür die generierten JAXB-Klassen. Erstellen Sie ein Menü, das in der Konsole angezeigt wird. Über dieses Menü sollen die Auswahl der Funktionen, zum Ausgeben der Daten und Erstellen neuer Kommentare, möglich sein.

a) **done**
b) **done**
c) **done**

Aufgabe 5

Diskutieren Sie, warum es sinnvoll ist Daten in Formaten wie XML oder JSON zu speichern. Stellen Sie außerdem die beiden Formate gegenüber und erläutern Sie kurz deren Vor- und Nachteile.

	XML	JSON
Einfachheit		Syntax ist einfacher gestaltet -> lesbarer, leichter schreibbar, Overhead Reduktion
Komplexität	Werte und Eigenschaften können sowohl als Attribute oder als Kindknoten beschrieben werden -> kann zu Problemen führen	In JSON kann dieses Problem nicht auftreten.
Typisierung	Man kann zusätzlich eigene Typen definieren.	JSON-Daten sind im Gegensatz zu XML-Daten typisiert, wobei nur einige grundlegende Typen unterstützt werden.
		Eine JSON-Definition in JavaScript direkt mit der eval()-Funktion in ein JavaScript-Objekt umsetzen (JSON ist valides JavaScript)
Einsetzbarkeit	XML ist eine Auszeichnungssprache -> vielseitiger einsetzbar	JSON ist ein Datenaustauschformat -> weniger vielseitig einsetzbar
Verbreitung	XML ist weiter verbreitet	JSON verdrängt XML aufgrund seiner Einfachheit dort, wo keine komplizierten Auszeichnungen notwendig sind