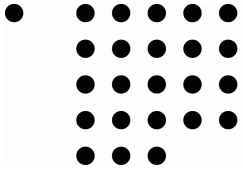

Fachhochschule Köln
Cologne University of Applied Sciences

WBA 2 -
Verteilte Systeme

Prof. Dr. Fischer

Phase 1

Laura-Maria Hermann

11083968

Inhaltsverzeichnis

Laura-Maria Hermann

Einleitung

Das folgende Dokument wurde im Rahmen des Moduls Web-basierte Anwendungen 2: Verteilte Systeme erarbeitet und betreut. Alle Ergebnisse der Lösungen befinden sich im Git sowie im folgenden Dokument.

Aufgabe 1 Begriffserklärungen[1] [2]

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

Wohlgeformtheit:

Die Wohlgeformtheit bedeutet im Allgemeinen die Einhaltung von Regeln. Diese Regeln werden im Folgenden grob beleuchtet und erklärt.

• Existenz eines Wurzelelements

• Öffnende Tags besitzen jeweils ein schließendes Tag <eintrag> Eintrag Nummer Eins </eintrag>

• Leere Elemente können in sich geschlossen sein <eintrag Eintrag Nummer Eins />

• Werte von Attribute werden durch Hochkommata (' ... ') beziehungsweise Anführungszeichen (“ ... “) gekeinnzeichnet

• Alle Zeichen müssen dem Spezifizierten Zeichensatz entsprechen.

• Elemente, die nicht Wurzel sind müssen vollständig von dem umgebenden Element eingeschlossen sein

• XML with correct syntax is "Well Formed" XML.Bearbeiten

Validität:

Beschreibt im Allgemeinen die Gültigkeit von Dokumenten. Ein Dokument ist gültig, wenn

• es wohlgeformt ist

• und zusätzlich den Bedingungen einer DTD genügt.

• XML validated against a DTD is "Valid" XML. Bearbeiten

Namespaces: (Namensraum)

Dienen im Allgemeinen der eindeutigen Identifizierung des Vokabulars (Beispiel: <p> in HTML für Absatz - <p> in XML für bspw. Person) - Vermeidung von Mehrdeutigkeiten.

• Definition einer URI für verschiedene Sprachen wie XHTML (z.B. http://www.w3.org/1999/xhtml)

• Definition der Namensräume mittels Präfix

• Definition durch xmlns=“...“

1 Prof. Dr. K. Fischer: Foliensatz K1, K2 - Web-basierte Anwendungen 2: Verteilte Systeme - Sommersemester 2013

2 RESTundHTTP, Author = Stefan Tilkov, Publisher = dpunkt.verlag, Title = REST und HTTP, Volume = 2, Year = 2011

• <... xmlns:foo="http://www.w3.org/1999/xhtml"> ... <foo:head>...</foo:head> ... </...>

Aufgabe 2 XML Modelle

Aufgabe 2a)

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.

Zu diesem Aufgabenteil wurden mehrere Lösungen erarbeitet und am Ende der sinnigste und beste ausgewählt. Zunächst wurde mit wesentlich mehr Attributen gearbeitet, was sich allerdings als nicht besonders „gut“ erweis.

```
<? xml version="1.0" encoding="UTF-8"?>
<participants>  <!--Root Wurzel-->

   <description>Member-List</description>

   <!--Beginn einer Gruppe-->
   <group id="1">
   <name>Name der Gruppe</name>

   <!--Beginn eines Leaders innerhalb dieser Gruppe-->
   <leader>
      <firstname>David</firstname>
      <lastname>Petersen</lastname>
      <email>bla@muster.com</email>
      <bdate>
        <day>DD</day>
        <month>MM</month>
        <year>YYYY</year>
      </bdate>
      <experience>Profi</experience>
      <drums>vorhanden</drums>
      <notes>Wichtig ist, dass wir einen Stellplatz mit Tageslicht bekommen</notes>
   </leader>

   <!--Beginn eines "normalen Mitglieds" innerhalb dieser Gruppe-->
   <participant>
      <firstname>David</firstname>
      <lastname>Petersen</lastname>
      <email>bla@muster.com</email>
      <bdate>
         <day>DD</day>
         <month>MM</month>
         <year>YYYY</year>
      </bdate>
      <experience>Profi</experience>
      <drums>vorhanden</drums>
      <notes>Wichtig ist, dass wir einen Stellplatz mit Tageslicht bekommen</notes>
   </participant>

   </group>


</participants>
```

Aufgabe 2b)

Erzeugen Sie ein JSON-Dokument, dass zu ihrem XML- Dokument äquivalent ist.


```
{
	"participants":
	{
		"description":"Member-List",
		   "group":
		   {
		      "id":"0",
		      "name":"Bandname",
		      "leader":
		      {
		         "firstname":"Laura",
		         "lastname":"Hermann",
		         "email":"lmh@hermann.com",
		         "bdate":
		         {
		            "day":25,
		            "month":05,
		            "year":1992
		         },
		         "experience":"profi",
		         "drums":"vorhanden",
		         "notes":"Stellplatz mit Tageslicht!"
		      },

		      "member":
		      {
		         "firstname":"Laura",
		         "lastname":"Hermann",
		         "email":"lmh@hermann.com",
		         "bdate":
		         {
		            "day":25,
		            "month":05,
		            "year":1992
		         },
		         "experience":"profi",
		         "drums":"vorhanden",
		         "notes":"Stellplatz mit Tageslicht!"
		      },
		   }
	}
}
```

Aufgabe 3 XML-Dokumente /Schemata

Aufgabe 3a)

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.

Hierzu lässt sich sagen, dass die Kommentare zunächst weggelassen wurden, später dann allerdings zur Lösung der Aufgabe vier eingebunden wurden.

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

<RezeptSammlung xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="Schema.xsd">
   <Beschreibung>Meine Rezeptsammlung</Beschreibung>
   <Rezept id_rezept="1">

      <Rezeptname>Lenchen's Schokoladenkuchen</Rezeptname>
      <Anmerkungen>Arbeitszeit= 1 Stunde, Schwierigkeitsgrad= normal
      </Anmerkungen>

      <Zutaten>
         <Zutat Menge="200" Einheit="Gramm (g)">Butter</Zutat>
         <Zutat Menge="200" Einheit="Gramm (g)">Zucker</Zutat>
         <Zutat Menge="200" Einheit="Gramm (g)">Schokolade</Zutat>
         <Zutat Menge="150" Einheit="Gramm (g)">Mehl</Zutat>
         <Zutat Menge="0.5" Einheit="Teelöffel">Backpulver</Zutat>
         <Zutat Menge="1" Einheit="Päkchen">Vanillezucker</Zutat>
         <Zutat Menge="4" Einheit="Stück"/>
      </Zutaten>

      <Zubereitung>
         <Step id_step="0">Butter und Schokolade im Wasserbad schmelzen.</Step>
         <Step id_step="1">Eier trennen.</Step>
         <Step id_step="2">Eiweiß steif schlagen.</Step>
         <Step id_step="3">Eigelbe, Zucker und Vanillezucker verrühren.</Step>
         <Step id_step="4">Geschmolzene Butter-Schokomasse hinzufügen und mischen.</Step>
         <Step id_step="4">Mehl mit dem Backpulver in die Masse sieben und zum Schluss die steifen
Eiweiße vorsichtig unterheben.</Step>
         <Step id_step="5">In eine gut gefettete Form geben.</Step>
         <Step id_step="6">Bei 180°Grad 40 – 50 Minuten backen.</Step>
      </Zubereitung>

      <Brennwert>295 kcal</Brennwert>
      <Kommentar id_kommentar="0">Das Rezept ist doof! Zu wenig Schokoflocken.</Kommentar>
   </Rezept>

</RezeptSammlung>
```

Aufgabe 3b)

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.

Im Allgemeinen kann man sagen, dass die Struktur bei allen Rezepten die gleiche ist. Es gibt einen Titel, Zutaten, Zubereitungsanweisungen sowie Anmerkungen. Unterschied besteht lediglich in den konkreten Daten der Rezepte, sprich die einzelnen Zutaten deren Mengen und der Zubereitung. Auch unterscheidet sich die Bewertung sowie die Anmerkungen der Rezepte. Ein weitere Unterschied snd die konkreten Datensätze. Beispielsweise unterscheiden sich die Einheiten stets und können teilweise sgar nicht bestimmt werden. Auch die Anzahl der utatten variiert je nach Rezept. (Die Marginalspalte soll an dieser Stelle außer Achrt gelassen werden, dennoch kann man dazu sagen, dass sich die Daten innerhalb dieser entscheidend ändern.) [3]

Aufgabe 3c)

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. 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?

Zunächst zu den Attributen beziehungsweise den Elementen. Im Allgemeinen aber gilt der Tipp Attribute weitestgehend zu vermeiden, da Elemente schlichtweg handlicher sind. Steht man allerdings vor der Wahl sollte man folgende Faustregel beachten: Attribute sollten eher für Metadaten verwendet verwendet, sprich für Datensätze die für den Benutzer nicht sichtbar sind oder sein sollen. Dies wären Datensätze wie Identifikationsnummern. Sie dienen in dem Beispiel des Kuchens lediglich der Identifikation sofern es mehrer Rezepte mit demselben Namen gibt. Für den Benutzer spielt diese Nummer allerdings keine Rolle, er bekommt Sie nocht nicht einmal angezeigt bzw. weiß davon.

Simple Types: Elemente, welche keine weiteren Elemente beinhalten.
Complex Tapes: Elemente, welche Unterelemente beinhalten.

Attribute:
Identifikatoren werden üblicherweise als Attribute verwendet. Zudem habe ich mich auch bei der Einheit und der Menge der Zutaten dafür entschieden, diese als Attribute zu verwenden.

Restriktionen:
Die Angabe von Menge, Einheit und ID's sind für ein Rezept unumgänglich, aus diesem Grund ist die Nutzung (use) also erforderlich (required).
Die Angabe von Zubereitsungsschritten ist bei einem Rezept auch erforderlich. Dabei unterscheiden sich die Rezepte jedoch in der Schwierigkeit, sodass es eventuell nur einen einizgen Schritt gibt oder oftmals unbegrenzt viele (unbounded). Ebenso die Angaben der Zutaten: Mindestens eine und maximal unbegrenzt.

3 http://www.w3schools.com (Aufgerufen am 27.03.13)

Aufgabe 3d)[4] [5]

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.

Simple Types sind Beschreibung, Rezeptname, Anmerkungen sowie Brennwert, da diese lediglich aus Text bestehen und keine Unter(Kind-)Elemente besitzen. An dieser Stelle waren alle Datentypen eindeutig.

Zu den Attributen gehört die Id des Rezepts, die Menge, die Einheit, die ID der Steps sowie die ID der Kommentare.

Die Elemente Zutat(en), Step, Zubereitung, Rezept, Rezeptsammlung, Kommentare und Kommentar sind komplexe Typen, da sie Unterelemente sowie Attribute enthalten.

Einige Elemente enthalten Attribute sowie „normale“ Elemente was beduetet, dass mixed auf true gesetzt wurde.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

   <!--Simple Types-->
   <xs:element name="Beschreibung" type="xs:string"/>
   <xs:element name="Rezeptname" type="xs:string"/>
   <xs:element name="Anmerkungen" type="xs:string"/>
   <xs:element name="Brennwert" type="xs:integer"/>

   <!--Attribute -->
   <xs:attribute name="id_rezept" type="xs:integer"/>
   <xs:attribute name="Menge" type="xs:decimal"/>
   <xs:attribute name="Einheit" type="xs:string"/>
   <xs:attribute name="id_step" type="xs:integer"/>
   <xs:attribute name="id_kommentar" type="xs:integer"/>

   <!--Complex Types-->
   <xs:element name="Zutat">
      <xs:complexType  mixed="true" >              <!--mixed->
         <xs:attribute ref="Einheit" use="required"/>
         <xs:attribute ref="Menge" use="required"/>
      </xs:complexType>
   </xs:element>

   <xs:element name="Step">
      <xs:complexType mixed="true">
            <xs:attribute ref="id_step" use="required"/>
      </xs:complexType>
   </xs:element>

   <xs:element name="Zubereitung">
      <xs:complexType>
         <xs:sequence> <!-- Abarbeitung in bestimmter Reihnefolge -->
            <xs:element ref="Step" minOccurs="1" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
```

4 http://www.w3schools.com (Aufgerufen am 27.03.13)

5 Prof. Dr. K. Fischer: Foliensatz K1, K2 - Web-basierte Anwendungen 2: Verteilte Systeme - Sommersemester 2013

```
    <xs:element name="Zutaten">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="Zutat" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>

    <xs:element name="Rezept">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="Rezeptname"/>
          <xs:element ref="Anmerkungen"/>
          <xs:element ref="Zutaten"/>
          <xs:element ref="Zubereitung"/>
          <xs:element ref="Brennwert"/>
          <xs:element ref="Kommentare"/>
        </xs:sequence>
        <xs:attribute ref="id_rezept" use="required"/>
      </xs:complexType>
    </xs:element>

    <xs:element name="RezeptSammlung">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="Beschreibung"/>
          <xs:element ref="Rezept" maxOccurs="unbounded" minOccurs="1"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>

    <xs:element name="Kommentare">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="Kommentar" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>

    <xs:element name="Kommentar">
      <xs:complexType  mixed="true" >
        <xs:attribute ref="id_kommentar" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

Testdatensatz1:

```
<Rezept id_rezept="1">
      <Rezeptname>Mein Dessert</Rezeptname>
      <Anmerkungen>Ist nicht so einfach wie man denkt - Aber es ist sehr lecker</Anmerkungen>
      <Zutaten>
         <Zutat Einheit="Packung" Menge="1">Mascarpone</Zutat>
         <Zutat Einheit="Packung" Menge="1">Joghurt</Zutat>
         <Zutat Einheit="Brise" Menge="1">Brauner Zucker</Zutat>
         <Zutat Einheit="Packung" Menge="1">Tiefkühl Himbeeren</Zutat>
      </Zutaten>
      <Zubereitung>
         <Step id_step="1">Alles mischen und fertsch</Step>
      </Zubereitung>
      <Brennwert>Will man nicht wissen (MASCARPONE!)</Brennwert>
      <Kommentare>
         <Kommentar id_kommentar="1">Das kann ja jedes Kind</Kommentar>
         <Kommentar id_kommentar="2">Es schmeckt auch mit Erdbeeren sehr gut.</Kommentar>
         <Kommentar id_kommentar="3">Am besten muss es ein paar Stunden in den Kühlschrank,
            dann wird der Zucker oben drauf schön hart.</Kommentar>
      </Kommentare>
   </Rezept>
```

Testdatensatz2:

```
<Rezept id_rezept="2">
      <Rezeptname>Davids Apfelkuchen</Rezeptname>
      <Anmerkungen>Für David. </Anmerkungen>
      <Zutaten>
         <Zutat Einheit="Stück" Menge="5">Äpfel</Zutat>
      </Zutaten>
      <Zubereitung>
         <Step id_step="1">Alles verrühren und schon ist der Kuchen perfekt!</Step>
      </Zubereitung>
      <Brennwert>207</Brennwert>
      <Kommentare>
         <Kommentar id_kommentar="1">Ich find Apfelkuchen an sich nicht sehr lecker.</Kommentar>
         <Kommentar id_kommentar="2">Dies ist mein Kommentar zum testen! Hoffentlich
klappts</Kommentar>
      </Kommentare>
   </Rezept>
```

Aufgabe 4 Java Programmierung

Befehl des Terminals: xjc -d /Users/Laura/Desktop/Studium/4.\
Semester/WBA2/Praktikum/wba2_sose13/Phase1/src /Users/Laura/Desktop/Studium/4.\
Semester/WBA2/Praktikum/XML/Schema.xsd

Automatische Generierung der

Aufgabe 5 JSON vs. XML

Diskutieren Sie, 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.

Der Vorteil die Daten in XML oder JSON (Javascript Object Notation) Formaten zu speichern liegt in der Weiterverarbeitung. In diesen Formaten ist es für die Maschine wesentlich leichter auszulesen und weiterzuverarbeiten. Auch eine einfache Umwandlung, besonders bei JSON Formaten in Java ist ein leichtes.[RESTundHTTP]. Durch die fest vorgeschriebene Struktur und der Aufbau eines solchen Formats, kann auch eine leichtere Umwandlung in ein anderes Format ermöglicht werden. Vergleicht man JSON mit XML lässt sich sagen, dass JSON ausschließlich auf Datenstrukturen fokussiert ist und nicht etwa auf Text. Bei JSON wird im Gegensatz zu XML keine Namespaces oder eine schemabasierte Validierung unterstützt. Andererseits ist JSON von vielen Sprachen schnell und einfach zu verarbeiten.

JSON:
+Kompakt und klein
+Handlich und überschaubar
-nicht erweiterbar

XML:
+Standard
-Komplex