Phase 1 – Dokumentation

Boris Cosic 11068185

WBA 2 – Verteilte Systeme
Betreut durch
Prof. Dr. Fischer

Inhalt

1.1 Einleitung

Innerhalb des Praktikums des Faches WBA 2 – Verteilte Systeme gilt es diverse Aufgaben zu erfüllen
Innerhalb dieser Dokumentation finden sie die verschiedenen, gestellten Aufgaben und Erläuterungen zu deren Lösungen. Dabei ist es meine Intention entworfene Lösungen reichhaltig zu begründen. Um die XML-Dateien besser lesen zu können sind diese auch noch als .xml beigefügt, genauso wie JSON (.js) und XML-Schema(.xsd).

1.2 Aufgabe 1

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

Namensräume
Ein XML-Dokument besteht im Wesentlichen aus Elementen und Attributen. Bei der Zuordnung zu einer Document Type Definition (DTD) gehören die Elemente und Attribute dem Namensraum dieser DTD an. Eine solche eindeutige Zuordnung besteht nicht, wenn die DTD fehlt. Das Dokument kann dann zwar wohlgeformt sein aber es ist nicht klar in welchem Namensraum sich die Element- und Attributnamen befinden.

Eine Bezeichnung des Namensraums ist auch dann wichtig, wenn es Elemente und Attribute aus unterschiedlichen Namensräumen gibt (z.B. bei der Verwendung von HTML-Elementen in einem XML Dokument, wo der Element-Name doppelt vorkommt).

Zu diesem Zweck hat das W3-Konsortium den Begriff der qualifizierten Namen (qualified names) eingeführt. Qualifizierte Namen bestehen immer aus einem Präfix, der den Namensraum bezeichnet, und einem lokalen Namensteil, der den Namen des Elements oder Attributs innerhalb des Namensraums bezeichnet. Beim Arbeiten mit mehreren Namensräumen gleichzeitig ist es wichtig, qualifizierte Namen zu notieren.

Wohlgeformtheit
Der Begriff "Wohlgeformtheit" bedeutet, dass eine Datei die Regeln von XML korrekt einhält.
Beispiel:

```
<?xml version="1.0"?>
<Dialog>
        <Othello rolle="Feldherr">Heute ist ein schöner Tag!</Othello>
        <Cassio rolle="Leutnant von Othello">So ist es.</Cassio>
</Dialog>
```

Die XML-Datei ist wohlgeformt weil:

- am Beginn die XML-Deklaration steht (stellt Bezug zu XML her).
- Es gibt mindestens ein Datenelement (hier drei: <Dialog>,<Othello>, <Cassio>
- Es gibt bei den Datenelementen ein äußerstes Element, das alle anderen Datenelemente enthält (<Dialog></Dialog>).

Würde es in dem Beispiel noch einen Bezug zu einer DTD geben, dann wäre das Dokument nicht nur wohlgeformt, sondern auch noch gültig (valide).

Validität

Ein gültiges (valides) Dokument beinhaltet am Anfang eine XML-Deklaration und anschließend eine Document Type Definition (DTD). Die Regeln der DTD können intern im Dokument angegeben werden oder aus einer externen Datei abgerufen werden. Ein gültiges XML-Dokument ist also wohlgeformt und validierbar gegen die DTD-Regeln.

Beispiel:

```
<?xml version="1.0"?>
<!DOCTYPE rezept [
  <!ELEMENT rezept (zutaten, zubereitung)>
  <!ELEMENT zutaten (#PCDATA)>
  <!ELEMENT zubereitung (#PCDATA)>
]>
<rezept>
  <zutaten>1/2 Liter Milch.</zutaten>
  <zubereitung>Alles miteinader verrühren.</zubereitung>
</rezept>
```

Erläuterung:

Zunächst erfolgt die XML-Deklaration. Danach werden mit der DTD die internen Regeln festgelegt (alternativ Verweis auf externe DTD). Im Beispiel wird ein Typ "rezept" mit dem gleichnamigen Dokument-Element "rezept" und zwei abhängigen Elementen ("zutaten", "zubereitung") definiert.

Im Anschluss folgen die Daten. Der Name des Dokumenttyps ("rezept") muss gleich dem Namen des Dokument-Elements sein. Das Dokument-Element ist das äußerste Element in der Hierarchie, dem alle anderen Elemente untergeordnet sind.

Innerhalb des Dokument-Elements werden die Daten nach den Regeln notiert, die von der DTD vorgegeben werden. Im Beispiel wurden im DTD-Bereich die beiden Elemente "zutaten" und "zubereitung" als innere Elemente des Dokument-Elements "rezept" definiert. Der DTD-Bereich im Beispiel erlaubt nichts anderes als die einmalige Verwendung dieser beiden Elemente.

Das Beispiel erfüllt damit alle Kriterien einer gültigen und vollständigen XML-Datei.

Quellen

http://de.selfhtml.org/xml/regeln/namensraeume.htm
http://de.selfhtml.org/xml/regeln/begriffe.htm#wohlgeformt
http://de.selfhtml.org/xml/regeln/begriffe.htm#gueltig_vollstaendig

1.3 Aufgabe 2

Aufgabenstellung

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 Lösung

```
<?xml version="1.0" encoding="UTF-8"?>
<formular>
<description>Formular Gruppenanmeldung</description>
	<gruppe id="1">
	<gruppenleiter>		<!-- einmalig -->
		<first_name>
			<text>Hans</text>
		</first_name>
		<last_name>
			<text>Werner</text>
		</last_name>
		<email>
			<text>Hans.werner@gmx.de</text>
		</email>
		<b_day>
			<value>19.08.1977</value>
		</b_day>
			<erfahrung option="Amateur"/>
			<schlagzeug option="vorhanden" />
		<anmerkungen>
			<text>alles super hier</text>
		</anmerkungen>
	</gruppenleiter>

	<mitglied>		<!-- mehrmalig -->
	<first_name>
			<text>Sven</text>
		</first_name>
		<last_name>
			<text>Olm</text>
		</last_name>
		<email>
			<text>Sven.Olm@aol.com</text>
		</email>
		<b_day>
```

```
                <value>18.03.2003</value>
            </b_day>
                <erfahrung option="Profi"/>
                <schlagzeug option="vorhanden" />
        </mitglied>
        </gruppe>
        <agb></agb>
</formular>
```

Erläuterung

Anhand der Folien und der Website http://www.w3schools.com/xml/ wurde zunächst der grundsätzliche Aufbau einer XML Datei betrachtet. Als nächstes habe ich mir das Formular angesehen und überlegt welche XML Elemente und Attribute benötigt werden.

Elemente:

```
<formular>

<description></description>
        <gruppe>
        <gruppenleiter>
            <first_name>
            <last_name>
            <email>
            <b_day>
                <erfahrung>
                <schlagzeug>
            <anmerkungen>

        <mitglied>
        <first_name>
            <last_name>
            <email>
            <b_day>
                <erfahrung>
                <schlagzeug>
        <agb></agb>
```

Um alle Auswahlmöglichkeiten des Formulars in XML widerzuspiegeln habe ich noch verschiedene Attribute eingesetzt, an Stellen an denen es Sinn macht und auch so wenig wie möglich da von der w3school geraten wird Attribute so wenig wie möglich zu benutzen. Zunächst habe ich bei den Elementen *<gruppe>* und *<person>* ein ID-Attribut hinzugefügt.

Da es mehrere Gruppen und Personen gibt, dient die ID der eindeutigen Identifikation. Die Dropdown Menü-Auswahlmöglichkeiten aus dem Formular werden ebenfalls als Attribut den Elementen zugeordnet:

<erfahrung option="Amateur"/>
<schlagzeug option="vorhanden" />

Dies dient einer besseren Übersicht und klareren Strukturierung. Abschließend wurde das XML Dokument über den W3C Validator erfolgreich auf korrekte Syntax überprüft.

b)

JSON Lösung

```
var gruppenleiter= {

              "first_name": "Hans",
              "last_name": "Werner",
              "email": "Hans.werner@gmx.de",
              /*var wernerBday = {"tag":19, "monat":08, "jahr":1977};
              "bday":wernerBday.tag+"."+wernerBday.monat+"."+wernerBday.jahr,*/
              "b_day": "19.08.1977" ,
              "erfahrung": "Amateur",
              "schlagzeug": "vorhanden",
              "anmerkung": "alles super hier"
              };
var mitglied= {

              "first_name": "Sven",
              "last_name": "Olm",
              "email": "Sven.Olm@aol.com",
              "b_day": "18.03.2003" ,
              "erfahrung": "Profi",
              "schlagzeug": "vorhanden"
              };
var agb= {

}
```

Erläuterung

Anhand der Folien und der Website http://www.w3schools.com/json/ wurde zunächst der grundsätzliche Aufbau einer JSON Datei betrachtet. Als nächstes habe ich die zuvor erstellte XML-Dateien angesehen und versucht eine äquivalent JSON Datei zu erzeugen.

1.4 Aufgabe 3

Aufgabenstellung

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 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.

Xml Lösung

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

<rezept id="77">
	<title>Lenchen's Schokoladenkuchen</title>
	<bild name="Schokoladenkuchen"/>
	<bilder>
		<pic></pic>
		<zurück></zurück>
		<stop></stop>
		<nächstes></nächstes>
	</bilder>
	<drucken></drucken>
	<speichern></speichern>
	<speichern_unter></speichern_unter>
	<empfehlen></empfehlen>
	<fotos_hochladen></fotos_hochladen>
	<zutaten>
		<umrechnen>
			<zutat menge="200" einheit="g" name="Butter"/>
			<zutat menge="200" einheit="g" name="Zucker"/>
			<zutat menge="200" einheit="g" name="Schokolade, Blockschokolade"/>
			<zutat menge="120" einheit="g" name="Mehl"/>
			<zutat menge="0,5" einheit="TL" name="Backpulver"/>
			<zutat menge="1" einheit=Ppkt." name="Vanillezucker"/>
			<zutat menge="4" einheit="" name="Eier"/>
			<portionen wert="16"/>
		</umrechnen>
	</zutaten>

	<zubereitung>
		<arbeitszeit zeit= "1" einheit= "Std"/>
		<schwierigkeitsgrad option= "normal"/>
		<brennwert_p_P menge="295" einheit="kcal"/>
		<anleitung>Butter und Schokolade im Wasserbad schmelzen. Eier trennen.
Eiweiß steif schlagen. Eigelbe, Zucker und Vanillezucker verrühren. Geschmolzene
Butter-Schokomasse hinzufügen und mischen. Mehl mit dem Backpulver in die Masse sieben
und zum Schluss die steifen Eiweiße vorsichtig unterheben. In eine gut gefettete Form
geben. Bei 180°Grad 40 – 50 Minuten backen.</anleitung>
	</zubereitung>
		<share_buttons>
			<fb_like></fb_like>
			<google_plus></google_plus>
			<twitter></twitter>
			<email></email>
		</share_buttons>
	<kommentare>
		<kommentar id="1" author="swieselchen" date="07.02.2002">Habe Deinen
Kuchen gestern gebacken (kleine Abwandlung: statt Blockschokolade hatte ich nur
Kuvertüre, zartbitter und ich habe noch etwas Rumaroma und eine Prise Salz dazugegeben)
```

```
mein LAG war total begeistert. Ich übrigends auch, super Rezept.</kommentar>
            <neuer_kommentar></neuer_kommentar>
            <kommentar_beantworten></kommentar_beantworten>
            <kommentar_hilfreich>
                  <gut></gut>
                  <schlecht></schlecht>
            </kommentar_hilfreich>
      </kommentare>
</rezept>
```

Erläuterung

Für das konkrete Rezept „**Lenchen's Schokoladenkuchen**„ habe ich mir die Seite angeschaut und habe die einzelnen Elemente erst einmal aufgeschrieben, dann habe ich wo es Sinn machte Attribute hinzugefügt.

b)

Alle Rezepte bestehen aus Zutaten diese wiederum mit ihrem Namen und einer menge in Einheiten angegeben werden, manche Zutaten bestehen aus zwei beschreibenden Namen wie zum Beispiel "Käse, gerieben"oder bieten eine alternative "Petersilie oder TK", in gewissen Rezepten kommen Zutaten häufiger vor z.b. „Wasser" für Körnerbrötchen Teig
Die Menge an Beispiel Bildern variiert von Rezept zu Rezept, manche haben auch ein Video in dem vor gekocht wird.
Des weiteren hat jedes Rezept eine Zubereitungsrubrik in der Rand Informationen angegeben werden wie zum Beispiel die Arbeitszeit Schwierigkeit usw. und ein Text der meist eine Anleitung für das jeweilige Rezept ist.
Kommentare von anderen Nutzern.

http://www.chefkoch.de/rezepte/745721177147257/Lasagne.html
http://www.chefkoch.de/rezepte/1830281296676145/Koernerbroetchen-nach-Paniniart.html

d)

Xml-Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="rezept">
<xs:complexType>
<xs:sequence>
<xs:element name="title" type="xs:string"/>
<xs:element name="autor" type="xs:string"/>
<xs:element name="personenAnzahl" type="xs:float" minOccurs="0" />
<xs:element name="bild" minOccurs="0" maxOccurs="5">
<xs:complexType>
<xs:attribute name="name" type="xs:string" />
<xs:attribute name="description" type="xs:string" />
<xs:attribute name="url" type="xs:string" />
</xs:complexType>
</xs:element>
<xs:element name="zutaten" minOccurs="1" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>

<xs:element name="zutat" minOccurs="1" maxOccurs="unbounded">
<xs:complexType>
```

```
<xs:attribute name="menge" type="xs:float" />
<xs:attribute name="einheit" type="xs:string" />
<xs:attribute name="name" type="xs:string" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="zubereitung">
<xs:complexType>
<xs:sequence>
<xs:element name="step" minOccurs="1" maxOccurs="unbounded">
<xs:complexType>
<xs:attribute name="id" type="xs:integer" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="kommentare" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="kommentar" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:attribute name="id" type="xs:integer" />
<xs:attribute name="author" type="xs:string" />
<xs:attribute name="date" type="xs:date" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="related" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="relRezept" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:attribute name="ref" type="xs:integer" />
<xs:attribute name="url" type="xs:string" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Erläuterung

Auf Basis der Informationen des W3C habe ich den Aufbau meines XML Schemas hergeführt. Angesetzt habe ich bei der XML Datei aus Aufgabe 3, die den Grundaufbau weitestgehend vorgegeben hat. Folgendes zeigt den wesentlichen Umgang mit Elementen und Attributen.

```
<xs:complexType>
<xs:sequence>
<xs:element name="title" type="xs:string"/>
<xs:element name="autor" type="xs:string"/>
<xs:element name="personenAnzahl" type="xs:float" minOccurs="0" />
<xs:element name="bild" minOccurs="0" maxOccurs="5">
<xs:complexType>
```

```
<xs:attribute name="name" type="xs:string" />
<xs:attribute name="description" type="xs:string" />
<xs:attribute name="url" type="xs:string" />
</xs:complexType>
</xs:element> </xs:sequence>
</xs:complexType>
```

Besonderheiten sind in diesem Aufbau Attribute wie „minOccurs“ die dem Element eine bestimmte Häufigkeit vorschreiben, mit der sie im Dokument auftreten müssen. In diesem Beispiel wurde die Begrenzung der Bilder auf 5 gesetzt, um performant arbeiten zu können. Sind diese Werte nicht gesetzt, kann das Element genau einmal auftreten. Da das Schema allgemein gehalten ist, lassen sich damit auch andere Rezepte abbilden.

1.5 Aufgabe 4

Fehlt aufgrund von problemen

1.6 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.

Mit XML oder JSON lässt sich sehr gut beschreiben bzw. veranschaulichen, was für Daten genutzt werden. XML und JSON sind plattformunabhängig und werden von fast jeder Programmiersprache akzeptiert, aus diesem Grund werden sie häufig im Zusammenhang mit APIs (Programmierschnittstellen) genutzt

XML

Vorteile
- Es existiert ein Datums bzw. Zeitformat
- Durch seine nahe Anlegung an die HTML Syntax ist es gut von Menschen lesbar

Nachteile
- Gefahr von Code-Overhead
- benötigt separate Software zur Interpretation
- Öffnende und Schließende Tags

JSON

Vorteile
- Wie XML von Menschen sehr gut Lesbar
- Selbstbeschreibend und es kommt ohne Schema oder eine Dokumenttypdefinition aus
- es existiert kein schließender Tag wie in XML. Ein } reicht aus
- kann direkt in Javascript per eval()geparsed werden
- es existieren keine reservierten Worte
- weniger Schreibaufwand als XML (z.B. Definition von Arrays)
- nutzt Javascriptsyntax

Nachteile

- Nur für äußerst Simple Darstellungen geeignet
- kein Datums- bzw. Zeitformat

Quellen: http://www.w3schools.com/xml/
http://www.w3schools.com/json/

1.7 Quellenverzeichnis

http://de.selfhtml.org/xml/regeln/namensraeume.htm
http://de.selfhtml.org/xml/regeln/begriffe.htm#wohlgeformt
http://de.selfhtml.org/xml/regeln/begriffe.htm#gueltig_vollstaendig
http://www.w3schools.com/xml/
http://www.w3schools.com/json/