Dokumentation WBAII
Phase I

Roy Fochtman
Studiengang: Medieninformatik 4. Semester.
Fach: Webbasierte Anwendungen II.
Abgabedatum: 14.04.2013

Inhaltsverzeichnis

Aufgabe 1

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

Wohlgeformtheit

Wohlgeformtheit eines XML-Dokuments: Die Datei hält die Regeln von XML korrekt. Die Regeln sind die folgenden:

- Gesamtes Dokument muss in ein einzelnes Wurzelelement eingeschlossen sein.
- Alle unbedingt erforderlichen Attribute sind angegeben.
- Die Werte der Attribute befinden sich im richtigen Wertebereich und entsprechen dem angegebenem Typ.
- Korrekte Verschachtelung der Elemente.

Validität

Drei Punkte bestimmen, ob ein XML-Dokument Gültigkeit (Validität) hat, oder nicht:

- Es handelt sich um ein wohlgeformtes Dokument (siehe obere Definition).
- Eine zugehörige interne oder externe DTD (Dokumenttyp-Definition) existiert und ist verfügbar.
- Das Dokument ist in Bezug auf die in der DTD ausgestellten Regeln gültig.

Also: Der Standard definiert ein XML-Dokument als gültig (valid), wenn es wohlgeformt ist, den Verweis auf eine Grammatik enthält und das durch die Grammatik beschriebene Format enthält.

Namespaces

XML-Namespaces (XML-Namensräume) werden benutzt, um das Vokabular eines XML-Dokuments eindeutig zu identifizieren und um in einem einzelnen Dokument mehrere XML-Sprachen zu mischen. Der Standard-Namensraum wird definiert durch xmlns="...".

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.

Erläuterung: Als Wurzelelement habe ich mich für ein Formular entschieden. Innerhalb dieses Formulars können mehrere Anmeldungen übertragen werden. Pro Anmeldung sind verschiedene Felder verfügbar, welche mit Daten gefüllt werden sollen/müssen. Jede Anmeldung enthält folgende Unterelemente:

- Vornamedes Gruppenleiters
- Nachnamedes Gruppenleiters
- Email des Gruppenleiters
- Geburtsdatum des Gruppenleiters
- Erfahrung (Amateur, Fortgeschrittener, Profi)
- Schlagzeug (vorhanden, nicht vorhanden)
- Anmerkung.

```xml
<?xml version="1.0" encoding="UTF-8"?> <!-- XML Decl
<formular> <!-- root element -->
    <anmeldung>
        <vorname>Roy</vorname>
        <nachname>Fochtman</nachname>
        <email>roy@test.de</email>
        <geburtsdatum>16.01.2000</geburtsdatum>
        <erfahrung>amateur</erfahrung>
        <schlagzeug>vorhanden</schlagzeug>
        <anmerkung>keine Anmerkung</anmerkung>
    </anmeldung>

    <anmeldung>
        <vorname>Monica</vorname>
        <nachname>Chou</nachname>
        <email>rmonica@test.de</email>
        <geburtsdatum>16.01.1990</geburtsdatum>
        <erfahrung>amateur</erfahrung>
        <schlagzeug>vorhanden</schlagzeug>
        <anmerkung>Halli Hallo</anmerkung>
    </anmeldung>
</formular>
```

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

Erläuterung: JSON zeichnet sich durch eine gute Lesbarkeit aus. Mit JSON ist es möglich, komplexe Datenstrukturen etwa übersichtlicher als mit XML darzustellen. Wie es gut zu erkennen ist, verzichtet JSON auf Schliessklammer. Die Objekte werden mit der selben Syntax wie bei Javascript beschrieben, das heißt, es ist nicht nötig, eine neue Syntax zu lernen. Sowohl das Wurzelelement, als auch die entsprechenden Elemente der Anmeldungen wurden beibehalten.

```json
{
"formular": [
{ "vorname":"Roy",
"nachname":"Fochtman",
"email":"roy@test.de",
"geburtsdatum":"16.01.2000",
"erfahrung":"amateur",
"schlagzeug":"vorhanden",
"anmerkung":"keine Anmerkung"},

{ "vorname":"Monica",
"nachname":"Chou",
"email":"monica@test.de",
"geburtsdatum":"16.01.1990",
"erfahrung":"amateur",
"schlagzeug":"vorhanden",
"anmerkung":"Halli Hallo"}
]
}
```

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.

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

    <titel>Lenchen's Schokoladenkuchen</titel>
    <zutaten>
        <zutat>
            <menge>
                <anzahl>200</anzahl>
                <einheit>g</einheit>
            </menge>
            <zutatname>Butter</zutatname>
        </zutat>
        <zutat>
            <menge>
                <anzahl>200</anzahl>
                <einheit>g</einheit>
            </menge>
            <zutatname>Zucker</zutatname>
        </zutat>
```

Erläuterung: „Rezept" steht als Wurzelelement. Das erste was im Rezept zu finden ist, ist der Name des Rezeptes. Das Foto bzw. die Menge der Fotos die auf der Website zu sehen ist, habe ich vernachlässigt, und es bei der Abstraktion für die Modellierung der XML-Datei nicht berücksichtigt, denn es wäre auch nichts anderes als ein XML-Tag, wo die Referenz des Bilds zu finden wäre. Außerdem habe ich es nicht miteinbezogen, um die Übersichtlichkeit des Dokuments beizubehalten. Natürlich sollen die Bilder bei einer reellen Umsetzung berücksichtigt werden, aber ich persönlich fand es nicht relevant für diese Übung.

„Zutaten" ist eine Liste von Zutaten. Jede Zutat enthält eine Menge (Anzahl, Einheit) und einen Namen. Dies war meine erste Überlegung, bei der nächsten Aufgabe musste ich jedoch einige Sachen anpassen.

```xml
    <anzahlportionen>16</anzahlportionen>

    <zubereitung>
        <arbeitszeit> ca. 1 Std</arbeitszeit>
        <schwierigkeitsgrad>normal</schwierigkeitsgrad>
        <brennwertpP>295 kcal</brennwertpP>
        <schritte>Butter und Schokolade [...]
            40 - 50 Minuten backen.</schritte>

    </zubereitung>
```

Bei jeder Rezept muss die Anzahl der (gewünschten) Portionen angegeben werden.

Das Element „Zubereitung" enthält die benötigte Vorbereitungszeit, den Schwierigkeitsgrad, den Brennwert (in Kcal.) und die zufolgenden Schritte des Rezepts.

Bei dieser Aufgabe habe ich die Kommentare nicht miteinbezogen, da sie später eine wichtigere Rolle spielen.

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.

Nach der Analyse der Bestandteile verschiedener Rezepten, bin ich auf folgendes Ergebnis gekommen:

Alle Rezepte haben:

- einen Namen
- eine Liste mit Zutaten. Jede Zutat enthält: Menge, Einheit, Zutatsname.
- die Portionenanzahl
- eine Zubereitungsbeschreibung, mit: benötigte Zeit, Schwierigkeitsgrad, Brennwert.
- eine Liste mit Kommentaren

Die Werte dieser Eigenschaften können aber unterschiedlich sein, je nachdem, bei welchem Rezept man sich gerade befindet.

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.

•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?

XML-Schema: Ein XML-Schema beschreibt die Struktur eines XML-Dokuments und ist eine Alternative zu DTD. XML-Schemata benutzen die XML-Syntax.

Kriterien:

Simple-types:

- Zubereitungsschritte
- Kommentar
- Einheit

Complex-types:

- Rezeptliste: Liste, die mehrere Rezepten enthalten kann.
- Rezept: enthält Attribute und Elemente.
- Zutaten: Liste, die mehrere Zutaten enthält.
- Zutat: enthält Attribute „anzahl“, „einheit“, „zutatsname“.
- Zubereitung: enthält Attribute „zeit“, „zeiteinheit“, „schwierigkeitsgrad“, „brennwert“.
- Kommentare: Liste, die mehrere Kommentare enthalten kann.

Attribute:

- Rezeptname
- Anzahlportionen
- Anzahl
- Einheit

- Zutatsname
- Arbeitszeit
- Zeiteinheit
- Schwierigkeitsgrad
- Brennwert

Restriktionen:

- Einheit: g, TL, Pkt., EL, l, ml, Zehe/n, evtl., Bund, Würfel, Becher, …
- Zeiteinheit: Min, Std.
- Schwierigkeitsgrad: Simpel, Normal, Pfiffig.

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.

Es gibt verschiedene Varianten, wie man ein XML-Schema erstellt. Ich habe mich für die einfachste Variante entschieden, wo man der Reihe nach, alle Elemente, mit ihren Attributen und verketteten Elementen modelliert. Es ist zwar nicht die beste Auswahl, was die Lesbarkeit betrifft, jedoch ist es weniger Schreibarbeit.

Die zweite Variante wäre, die Simple-Types, Complex-Types und Attributen voneinander zu trennen. Hier müsste man jedoch immer auf die definierten Types referenzieren, um auf sie zugreifen zu können.

Der Wurzelknoten meines Schemas, ist die Rezeptliste. Die Rezeptliste, enthält null, ein, oder mehrere Rezepte. Sie ist eine Sequenze vom Typ „complexType“, was bewirkt, dass die Rezepte der Reihe nach hinzugefügt werden.

Ein Rezept ist ein Element vom Typ „complexType“, da es eine Zusammensetzung verschiedener Objekte ist. Durch die Eigenschaft „maxOccurs=“unbounded“ bewirkt man, dass man mehrere Rezepte speichern kann. „Titel“ und „AnzahlPortionen“ sind Attribute des Rezeptes. Beide sind Pflichteingabe. Wie bereits oben erwähnt, besteht ein Rezept aus „Zutaten“, „Zubereitung“, „Zubereitungsschritte“, und „Kommentare“. Diese Eigenschaften werden hier als Elemente abgebildet. Durch die Eigenschaft „maxOccurs=“1““, werden diese Elemente jeweils maximal einmal pro Rezept auftauchen können.

```
<xs:element name="rezept" maxOccurs="unbounded">
    <xs:complexType>
        <xs:all maxOccurs="1">
            <xs:element name="zutaten">
            <xs:element name="zubereitung">
            <xs:element name="zubereitungsschritte" type="xs:string"/>
            <xs:element name="kommentare">
        </xs:all>
        <xs:attribute name="titel" type="xs:string" use="required"/>
        <xs:attribute name="anzahlportionen" type="xs:integer" use="required">
    </xs:attribute>
    </xs:complexType>
</xs:element>
```

Zutaten ist eine Liste von Zutaten (maxOccurs= “unbounded“). Jede Zutat ist ein Element von Typ „complexType“, und enthält drei Attribute: „anzahl“ (integer), „einheit“ (string, mit restrictions) und „zutatsname“ (string).

```
<xs:element name="zutaten">
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded"><!-- Hier sollten mehrere Zuta
                <xs:element name="zutat">
                    <xs:complexType>
                        <xs:attribute name="anzahl" type="xs:integer"/>
                        <xs:attribute name="einheit">
                            <xs:simpleType>
                                <xs:restriction base="xs:string">
                            </xs:simpleType>
                        </xs:attribute>
                        <xs:attribute name="zutatsname" type="xs:string"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
```

Sehr ähnlich sieht die „Zubereitung“ aus. Enthält vier Attribute: „arbeitszeit“ (integer, Pflichteingabe), „Zeiteinheit“ (mit Restriktionen, Pflichteingabe), „Schwierigkeitsgrad“ (Auch mit Restriktionen, Pflichteingabe), und einen „Brennwert“ vom Typ Integer, welcher nicht unbedingt eingegeben werden muss.

```
<xs:element name="zubereitung">
    <xs:complexType>
        <xs:attribute name="arbeitszeit" type="xs:integer" use="required"/>
        <xs:attribute name="zeiteinheit" use="required">
        <xs:attribute name="schwierigkeitsgrad" use="required">
        <xs:attribute name="brennwert" type="xs:integer" use="optional"/>
    </xs:complexType>
</xs:element>
```

Die Zubereitungsschritte werden als einfacher String eingegeben.
„Kommentare“ ist eine Liste von Kommentaren. Sie werden innerhalb einer Sequence abgespeichert, damit sie auch der Reihe nach angezeigt werden (Dies kann später entweder im Backend angepasst werden). Jedes Kommentar habe ich als einfacher String abgebildet, und habe somit das Datum, Name des Users und Bild vernachlässigt, um dieses Modell zu vereinfachen.

```
<xs:element name="kommentare">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="kommentar" type="xs:string" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

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.

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, dass in der Konsole angezeigt wird. Über dieses Menü sollen die Auswahl der Funktionen, zum Ausgeben der Daten und Erstellen neuer Kommentare, möglich sein.

Nachdem ich mit Hilfe des JAXB-Plugins alle Java-Klassen erstellt habe, habe ich eine neue Main-Klasse erstellt, wo sich nun mein kleines Programm befindet.

Über die Schnittstellen „marshaller“ und „unmarshaller“ habe ich zwei Objekte erstellt (m, unmarshaller), die benötigt werden, um eine XML-Datei in JAVA einlesen, bzw. eine JAVA in XML übersetzen zu können.

Für weitere Erläuterungen des Programms, siehe bitte die kommentierte RezeptMain.java-Datei.

Das Programm ist nun in der Lage, eine XML-Datei einzulesen, und die Rezepte mit den entsprechenden Elementen auszugeben, neue Kommentare hinzuzufügen und die Datei zu speichern.

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.

Sowohl XML als auch JSON wurden entwickelt, um Daten zu transportieren bzw. speichern, im Gegensatz zu HTML, dessen Zweck lediglich die Darstellung der Daten ist. Somit erhält man eine eindeutige Trennung zwischen Darstellung und Transport. XML und JSON erlauben das Definieren von beliebigen „Tags“. Die Daten werden dann in separaten XML- bzw. JSON-Dateien gespeichert. XML und JSON sind von Computern und Menschen lesbar (bessere Lesbarkeit mit JSON).

	XML	JSON
Vorteile	Standardisiertes Datenaustauschformat	Gute Lesbarkeit
	Eigene Tags	Javascript-basiert
	Erweiterbar	Leichte Javascript-Objekt-Überführung
	Vereinfachte Datenübetragung	Schlanker und schneller als XML
	HTML-Ähnlichkeit	Dynamische Speicherung von Daten
	Modulare Datendefinition	Lizenzfrei
	Lizenzfrei	
	Plattformunabhängig	
	Geeignet für große Datenstrukturen	
Nachteile	Unübersichtlich bei tiefen Datenstrukturen	Ungeeignet für große Datenstrukturen
	Redundant	

Quellen

XML:
http://www.mathematik.uni-ulm.de/sai/ws02/oodb/slides/physxml-07.html
http://msdn.microsoft.com/de-de/library/cc431269.aspx
http://www.w3c.at/Misc/XML-in-10-points.html
http://www.w3schools.com/xml/xml_usedfor.asp
http://www.uzi-web.de/xml/xml_grundlegendes.htm
http://www.uzi-web.de/xml/xml_grundlegendes.htm

JSON:
http://www.webmasterpro.de/coding/article/json-als-xml-alternative.html
http://de.softuses.com/73650

JAXB:
http://javathreads.de/2010/04/mit-jaxb-einfach-von-xml-daten-zu-java-objekten/