JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll

DATENAUSTAUSCHFORMATE

SEW-XML Übung
4AHITM 2015/16

Kanyildiz Muhammedhizir

Version 1.0

Note:

Betreuer: Prof. Dolezal Dominik

Begonnen am 26. Februar 2016

Beendet am 2. März 2016

Inhaltsverzeichnis

1 Einführung 3
1.1 Ziele 3
1.2 Voraussetzungen 3
1.3 Aufgabenstellung 3
2 Welche Daten speichert man als Attribut oder Element? 4
2.1 Attribute 4
2.2 Elemente 4
2.3 Attribute vs. Elemente 4
3 XML – Dokument mit Elemente und Attribute (5 Personen/5 Elemente) ... 5
4 Die Wohlgeformtheit Überprüfen 6
5 DOM-Baum 7
5.1 Rechte Hälfte 7
5.2 Linke Hälfte 7
6 Wohlgeformtheit verletzten (5 Beispiele) 8
6.1 Beispiel 1 8
6.2 Beispiel 2 8
6.3 Beispiel 3 9
6.4 Beispiel 4 9
6.5 Beispiel 5 9
7 Java Programm 10
8 Probleme 11
9 Quellen 12

1 Einführung

1.1 Ziele

- Möglichkeiten herauszufinden, wie Daten strukturiert an einem Endgerät(Client) übertragen werden können.
- Sämtlich Eigenschaften von CSV, JSON und vor allem von XML herausfinden/kennenlernen.
- Datenaustauschformate
- Die Unterschiede zwischen XML und HTML

1.2 Voraussetzungen

- Aufmerksames zuhören im Unterricht.
- Syntax von XML
- Syntax von einem Dom-ml

1.3 Aufgabenstellung

- Welche Daten speicherst du als Attribut oder Element?
- Speichere pro Person mindestens sechs Eigenschaften
- Bilde zumindest fünf Personen ab
- Verwende sowohl Elemente als auch Attribute
- Überprüfe, ob dein XML-Dokument wohlgeformt ist (suche ein entsprechendes Tool!)
- Stelle dein Ergebnis als DOM-Baum dar!
- Erweiterung: Erstelle und beschreibe fünf unterschiedliche Beispiele, welche die Wohlgeformtheit verletzen und erkläre die verletzte Regel
- Schreibe ein kleines Java-Programm, welches die

2 Welche Daten speichert man als Attribut oder Element?

2.1 Attribute

Attribute sind Teil der XML- Elemente. Ein Element kann mehrere eindeutige Attribute aufweisen. Attribut geben weitere Informationen über XML-Elemente. Um genauer zu sein, definieren sie Eigenschaften der Elemente.
Ein XML - Attribut ist immer ein Name-Value Paar.
Am häufigsten werden Attribute verwendet, um Informationen zu liefern, die nicht ein Teil des XML-Dokuments ist.

2.2 Elemente

XML Elemente können als Bausteine eines XML-Dokuments definiert werden.
Jedes Element kann als Container betrachtet werde.
Es kann Texte oder auch Attribute beinhalten.
Jedes XML-Dokument enthält ein oder mehrere Elemente, deren Umfang begrenzt werden durch Start- und End-Tags.
Es können natürlich auch in Elemente andere Container reingegeben werden.

2.3 Attribute vs. Elemente

```
<person sex="female">
    <firstname>Anna</firstname>
    <lastname>Smith</lastname>
</person>

<person>
    <sex>female</sex>
    <firstname>Anna</firstname>
    <lastname>Smith</lastname>
</person>
```

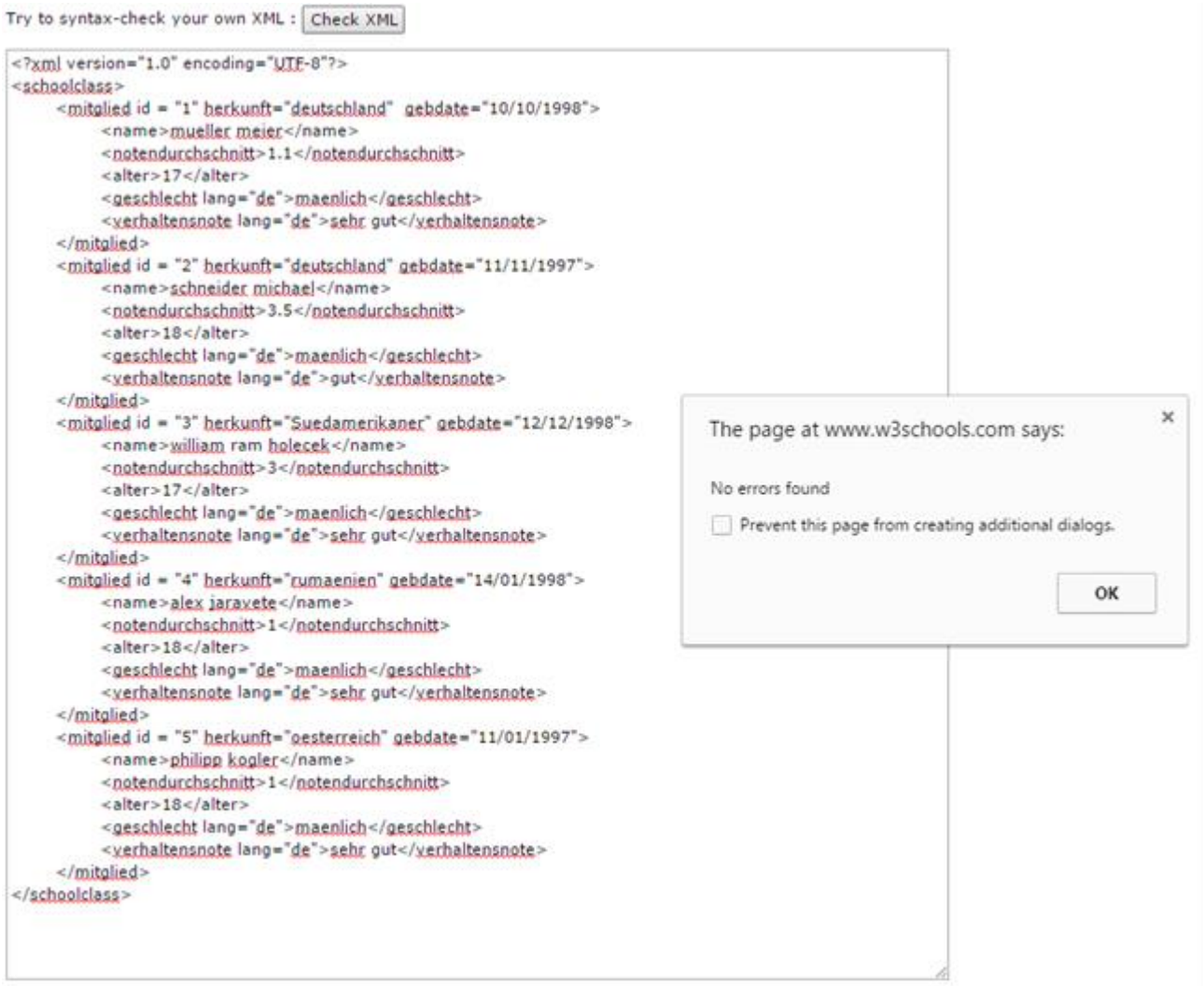
Beim 1. Beispiel wird Sex als Attribut genommen und im 2. Beispiel wird Sex als Element genommen.
Beide Beispiele liefert die gleichen Informationen an den Benutzer.

Es gibt keine festen Regeln, wann man was verwendet.
Man sollte jedoch Attribute vermeiden, solange die gleiche Information durch Elementen ausgedrückt werden kann.

3 XML – Dokument mit Elemente und Attribute (5 Personen/5 Elemente)

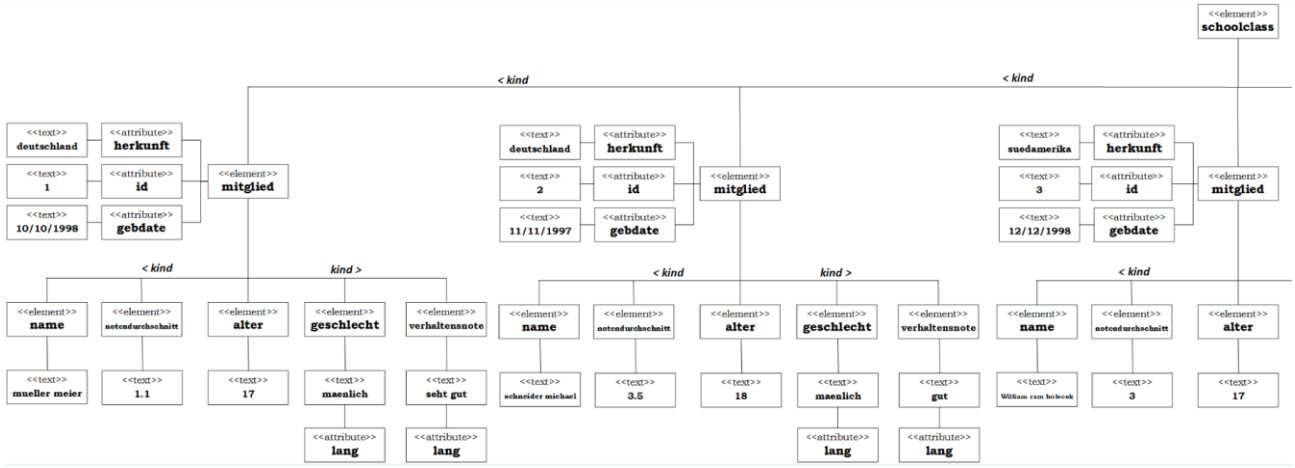
```xml
<?xml version="1.0" encoding="UTF-8"?>
<schoolclass>
    <mitglied id = "1" herkunft="deutschland"  gebdate="10/10/1998">
        <name>mueller meier</name>
        <notendurchschnitt>1.1</notendurchschnitt>
        <alter>17</alter>
        <geschlecht lang="de">maenlich</geschlecht>
        <verhaltensnote lang="de">sehr gut</verhaltensnote>
    </mitglied>
    <mitglied id = "2" herkunft="deutschland" gebdate="11/11/1997">
        <name>schneider michael</name>
        <notendurchschnitt>3.5</notendurchschnitt>
        <alter>18</alter>
        <geschlecht lang="de">maenlich</geschlecht>
        <verhaltensnote lang="de">gut</verhaltensnote>
    </mitglied>
    <mitglied id = "3" herkunft="Suedamerikaner" gebdate="12/12/1998">
        <name>william ram holecek</name>
        <notendurchschnitt>3</notendurchschnitt>
        <alter>17</alter>
        <geschlecht lang="de">maenlich</geschlecht>
        <verhaltensnote lang="de">sehr gut</verhaltensnote>
    </mitglied>
    <mitglied id = "4" herkunft="rumaenien" gebdate="14/01/1998">
        <name>alex jaravete</name>
        <notendurchschnitt>1</notendurchschnitt>
        <alter>18</alter>
        <geschlecht lang="de">maenlich</geschlecht>
        <verhaltensnote lang="de">sehr gut</verhaltensnote>
    </mitglied>
    <mitglied id = "5" herkunft="oesterreich" gebdate="11/01/1997">
        <name>philipp kogler</name>
        <notendurchschnitt>1</notendurchschnitt>
        <alter>18</alter>
        <geschlecht lang="de">maenlich</geschlecht>
        <verhaltensnote lang="de">sehr gut</verhaltensnote>
    </mitglied>
</schoolclass>
```

4 Die Wohlgeformtheit Überprüfen

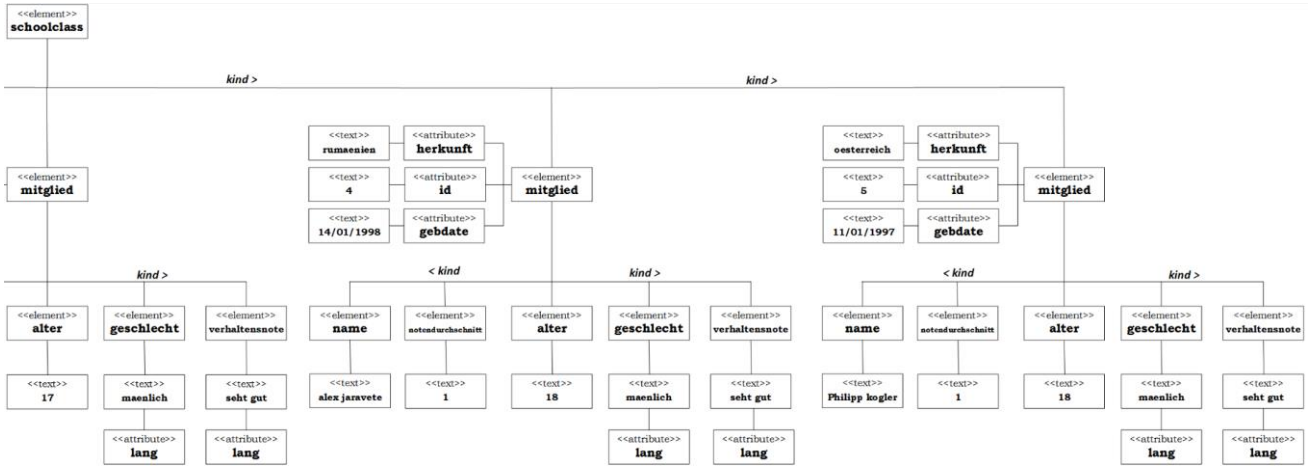


5 DOM-Baum

5.1 Rechte Hälfte



5.2 Linke Hälfte



6 Wohlgeformtheit verletzten (5 Beispiele)

6.1 Beispiel 1

Das Dokument besitzt genau ein Wurzelelement.
Als Wurzelelement wird dabei das jeweils äußerste Element bezeichnet.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<verzeichnis>
    <titel>Beispiel 1</titel>
    <eintrag>
        <stichwort>Bla Bla</stichwort>
        <eintragstext>Eintragstext</eintragstext>
    </eintrag>
    <eintrag>
        <stichwort>Bla Bla Bla</stichwort>
        <eintragstext>Eintragstext</eintragstext>
    </eintrag>
</verzeichnis>
```

Bei diesem Beispiel ist das Wurzelelement verzeichnis.

6.2 Beispiel 2

Alle Elemente mit Inhalt besitzen einen Beginn- und einen End-Auszeichner (-Tag) (z. B. <eintrag>Eintrag 1</eintrag>). Elemente ohne Inhalt können auch in sich geschlossen sein, wenn sie aus nur einem Auszeichner bestehen, der mit /> abschließt (z. B. <eintrag />).

Die Beginn- und End-Auszeichner sind ebenen treu-paarig verschachtelt. Das bedeutet, dass alle Elemente geschlossen werden müssen, bevor die End-Auszeichner des entsprechenden Elternelements oder die Beginn-Auszeichner eines Geschwisterelements erscheinen.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<verzeichnis>
    <titel>Beispiel 1</titel>
    <eintrag>
        <stichwort />
        <eintragstext>Eintragstext</eintragstext>
    </eintrag>
    <eintrag>
        <stichwort />
        <eintragstext>Eintragstext</eintragstext>
    </eintrag>
</verzeichnis>
```

Bei diesem Beispiel kann man erkennen das im Element stickwort nichts drin steht und deswegen gleich geschlossen wird.

Bei den anderen Elementen wird’s geöffnet und wieder geschlossen.

6.3 Beispiel 3

Ein Element darf nicht mehrere Attribute mit demselben Namen besitzen.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<verzeichnis id="1" id="3">
    <titel>Beispiel 1</titel>
    <eintrag>
        <stichwort />
        <eintragstext>Eintragstext</eintragstext>
    </eintrag>
    <eintrag>
        <stichwort />
        <eintragstext>Eintragstext</eintragstext>
    </eintrag>
</verzeichnis>
```

Hier kann man erkennen, dass das Elementen verzeichnis, 2-Attribute die gleichen Namen haben. Dies ist ein Fehler!

6.4 Beispiel 4

Attributeigenschaften müssen in Anführungszeichen stehen.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<verzeichnis id="1" id=3>
    <titel>Beispiel 1</titel>
    <eintrag>
        <stichwort />
        <eintragstext>Eintragstext</eintragstext>
    </eintrag>
    <eintrag>
        <stichwort />
        <eintragstext>Eintragstext</eintragstext>
    </eintrag>
</verzeichnis>
```

6.5 Beispiel 5

Die Beginn- und End-Auszeichner beachten die Groß- und Kleinschreibung (z. B. <eintrag></Eintrag> ist nicht gültig).

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<verzeichnis id="1" id=3>
    <titel>Beispiel 1</titel>
    <eintrag>
        <stichwort />
        <eintragstext>Eintragstext</Eintragstext>
    </eintrag>
    <eintrag>
        <stichwort />
        <eintragstext>Eintragstext</eintragstext>
    </Eintrag>
</Verzeichnis>
```

7 Java Programm

```
import org.w3c.dom.Document;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import java.io.*;

public class validate {
    public static void main (String[] args) {

        String xmlfilename;
        File f1 = null;
        // Ein neues "Schema" erstellen.
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        //Scha Validirung auf false setzten, denn es ist noch nicht Überprüft worden.
        factory.setValidating(false);
        factory.setNamespaceAware(true);

        // Neues DocumentBuilder
        DocumentBuilder builder = null;
        try {
            builder = factory.newDocumentBuilder();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        }

        builder.setErrorHandler(new SimpleErrorHandler());
        try {
            try{
                // Buffer mit dem ich die user Eingabe im Zwischenspeicher Speichere
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                System.out.println("------XML Datei INFO:------");
                System.out.print("PATH:");
                xmlfilename = br.readLine();
                // Eingabe im Typen File einspeichern
                f1 = new File(xmlfilename);
            }catch(IOException e){
                System.out.println("Die Eingabe konnte nicht durchgefuehrt werden. Fehler: " + e.getMessage());
            }
            // Überprüfung der Datei
            Document document = builder.parse(new InputSource(String.valueOf(f1)));
            System.out.println("------OUTPUT:------");
            System.out.println("Die Wohlgeformtheit des Files ist Valide.");
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (FileNotFoundException e){
            System.out.println("Das File wurde nicht gefunden!");
        } catch (IOException e) {
            e.printStackTrace();
        }
```

8 Probleme

Es gab insgesamt keine Probleme und Schwierigkeiten, da die Aufgabe ausführlich und gründlich in der Stunde erklärt wurde.
Bei den Extra Aufgaben sollte man einfach googlen können.

9 Aufwandsabschätzung

Arbeit	Stundenabschätzung
XML Tutorial inklusive alle Aufgaben	0:20-0:30 h
DOM Baum	0:20-0:25 h
Java Programm	0:10 h

10 Literatur Verzeichnisse

Information
GitHub: https://github.com/mkanyildiz01/Tgm4ahitmKanyildizMuhammedhizirXmlValidator.git

11 Quellen

INDEX	Quelltitel + Information
[1]	Wikipedia: http://www.xmlfiles.com/xml/xml_attributes.asp Kapitel: 2 zuletzt abgerufen am [2.3.2016]
[2]	Online: http://www.w3schools.com/xml/ Kapitel: 3,4 zuletzt abgerufen am [2.3.2016]
[3]	PDF: SEW_E14_Datenaustauschformate0 Kapitel: 5 zuletzt abgerufen am [2.3.2016]
[4]	Online: http://www.tutorialspoint.com/java_xml/java_xml_parsers.htm Kapitel: 3,4 zuletzt abgerufen am [2.3.2016]