

Web-Informationssysteme, WS 2009/10

Seminar-Übung 5: XML, Anfragen auf Text- und HTML Dokumenten

Besprechung am Mi 09.12.2009

— SÜ-5.1 —

Was Ihr wissen müßt über “XML (Basics)”

Wiederholung

- Beschreiben Sie die **Grobstruktur** eines XML Dokuments! Welche Art von XML Information darf auf dem obersten Level in einem XML Dokument vorkommen? In welcher Reihenfolge?
- Charakterisieren Sie die **Unterschiede zwischen Elementen und Attributen** in XML?
- Was bedeutet **“Wohlgeformtheit”** und welche Bedingungen muss ein XML Dokument erfüllen, damit es “wohlgeformt” ist.
- Was ist der Unterschied zwischen einem **“tag”** und einem “element”?
- Geben Sie für das folgende XML Dokument eine **Baumdarstellung** an, die ausreichend detailliert ist, um genau dieses XML Dokument wieder aus der Baumdarstellung zu gewinnen (Tipp: Sie können sich am DOM oder am *XML Information Set* orientieren.)

```
1 <!DOCTYPE html ...>
  <html>
3   ...
  <body>
5     <!-- The -->
     <p id='in'>
7       abc<em>pqr</em>xy
     </p>
9     ...
  </body>
11 </html>
```

— SÜ-5.2 —

Pipe-Separated Values vs. HTML

Programmierung

Immer noch werden tabellarisch oder anders strukturierte Daten oft einfach in delinierten Text-Dateien oder `<pre>` Blöcken in Web-Seiten abgelegt. Solche Dateien werden auch, je nach Separator, *comma / semi-colon / pipe separated value* (kurz CSV, SSV, PSV) Dateien genannt. Sie enthalten pro Zeile einen Datensatz dessen Attribute durch Separatoren (oder *delimiter*), voneinander getrennt abgespeichert werden:

EXAMPLE FLATFILE:

```
# Nachname, Vorname <E-Mail>      | Matrikeln | StR   | Vorl. | Note
#-----+-----+-----+-----+-----
Markmann, Jana <mm@gmail.net>      | 12399201  | Inf   | IN2003 | 1.3
Mut, Hans <mut@uni-tuebingen.de>    | 20923898  | Inf   | IN2003 | 2.3
Ahr, Benjamin <ben86@gmx.net>      | 23894333  | Math  | IN2041 | 1.7
Markmann, Jana <mm@gmail.net>      | 12399201  | Inf   | IN2041 | 1.0
Koller, Matthias <zorro@web.de>    | 12232112  | Inf   | IN2062 | 2.0
Mut, Hans <mut@uni-tuebingen.de>    | 20923898  | Inf   | IN2062 | 2.3
```

Die darin vorhandenen Informationen werden durch | voneinander getrennt. Ein # am Anfang einer Zeile leitet einen Kommentar ein, der für den Rest der Zeile gilt. Die erste Kommentarzeile wird als Header der Tabelle betrachtet.

In einer Reihe von Programmieraufgaben auf diesem und folgenden Themenblättern werden wir untersuchen, wie einfach die Verarbeitung solcher tabellarischer Daten in unterschiedlichen Darstellungen (PSV, HTML, XML, etc.) und mit unterschiedlichen Techniken (DOM, SAX, XPath, etc.) ist.

- Konvertieren Sie dieses Flatfile in ein HTML Dokument, in dem die Informationen mit Hilfe des HTML elements `table` repräsentiert sind.

Sowohl auf dem ursprünglichen Flatfile als auch auf dem erstellten HTML Dokument sollen Sie **die folgenden Recherche-Aufgaben mit einem Werkzeug Ihrer Wahl implementieren.**

1. Geben Sie den Vor- und Nachname, sowie Matrikelnummer der Studenten aus, die in einer Vorlesung eine Note besser als 1.7 erzielt haben.

Auf obigen Daten sollte beispielsweise folgende Ausgabe produziert werden:

```
1 Basic PSV splitter with input emails.psv!
  Jana Markmann, MNr: 12399201
3 Benjamin Ahr, MNr: 23894333
  Jana Markmann, MNr: 12399201
```

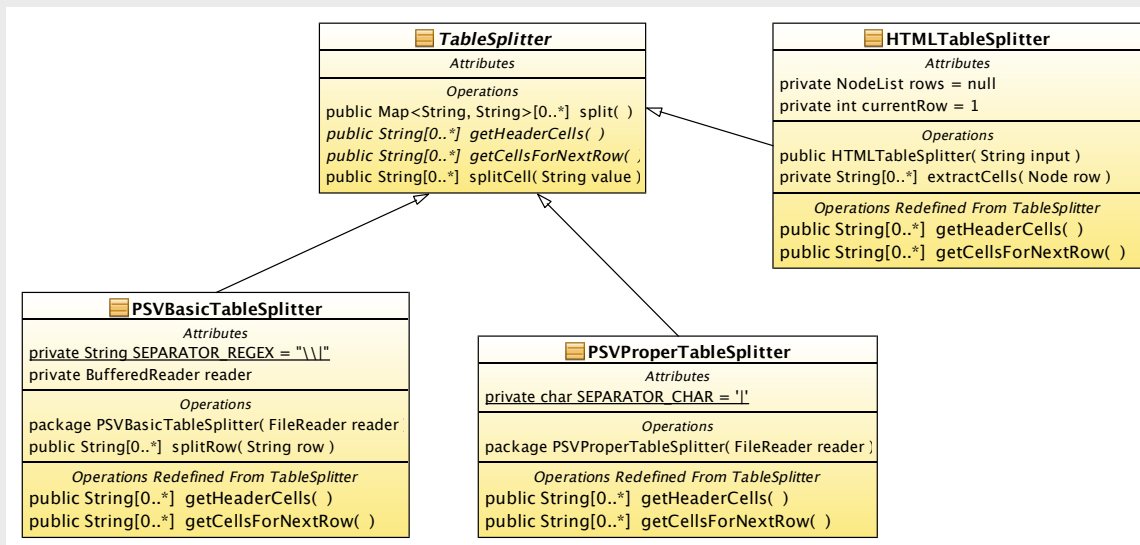
2. Geben Sie die E-Mail Adressen aller Mathematikstudenten aus, deren Matrikelnummer mit 238 beginnt.

3. Überprüfen Sie, ob der Student Hans Mut in allen Vorlesungen, die er besucht hat, eine Note besser als 3.0 erzielt hat.
4. Berechnen Sie den Notendurchschnitt des Studenten Hans Mut.
5. Berechnen Sie den Notendurchschnitt aller Studenten.

Diskutieren Sie die Probleme, welche Sie dabei hatten.

Tipp: Für die Verarbeitung des Flatfile bieten sich *reguläre Ausdrücke* an. Falls Sie Java verwenden wollen, könnte sich ein Blick auf `java.lang.String#split` lohnen. Für den HTML Fall werden Sie sich mit regulären Ausdrücken schon schwerer tun. Am einfachsten ist die Verwendung eines HTML oder XML Parsers (siehe z.B. `javax.xml.parsers` für Java).

Hinweis: Sie finden schon ein Gerüst von .java-Dateien in den Anlagen. Das folgende UML-Modell gibt einen Überblick über den Lösungsvorschlag:



Wie immer finden sich einige mit @TODO markierte Stellen, an denen noch Code ergänzt werden muss.