Austauschbare Datentypen & Parsing
JSON, XML
DOM/Sax-Parser

Motivation

Programmier-Praktikum 2.Meilenstein Aufgabe 4
Das Level soll aus einem (programmiersprachenunabhängigem) Dateiformat eingelesen werden können, Empfehlenswert hier wäre beispielsweise XML.

Die Aufgabe verlangt, dass wir Austauschbare Datentypen (wie z.B JSON oder XML) kreieren und diese mit Parsern (wie z.B DOM oder Sax-Parser) einlesen.

Austauschformat

bezeichnet ein Dateiformat, welches mit fast allen Anwendungen auf fast jedem Betriebssystem kompatibel ist.

- Betriebssystem-übergreifend
- Datensicherung über Lange Zeit (kompatibel mit Programmen aus unterschiedlichen Generationen von EDV-Systemen)
- relativ simple Struktur

1982: Plain Text (.txt)

1992: Hypertext Markup Language (.htm/.html)

1991: JPEG File Interchange Format (.jpg/.jpeg)

1993: Portable Document Format (.pdf)

JSON JavaScript Object Notation

- kompaktes Datenformat
- einfach lesbare Textform
- unabhängig von der Programmiersprache.
- Jedes gültige JSON-Dokument soll ein gültiges JavaScript sein und per `eval()` interpretiert werden können.
- Parser existieren in praktisch allen verbreiteten Sprachen.

JSON kennt folgende Datentypen:

Nullwert
wird durch das Schlüsselwort **`null`** dargestellt.

boolescher Wert
wird durch die Schlüsselwörter **`true`** und **`false`** dargestellt. Dies sind *keine* Zeichenketten. Sie werden daher, wie **`null`**, *nicht* in Anführungszeichen gesetzt.

Zahl
ist eine Folge der Ziffern **`0–9`**. Diese Folge kann durch ein negatives Vorzeichen **`-`** eingeleitet und einen Dezimalpunkt **`.`** unterbrochen sein. Die Zahl kann durch die Angabe eines Exponenten **`e`** oder **`E`** ergänzt werden, dem ein Vorzeichen **`+`** oder **`-`** und eine Folge der Ziffern **`0–9`** folgt.

Zeichenkette
beginnt und endet mit doppelten geraden Anführungszeichen (**`"`**). Sie kann Unicode-Zeichen und Escape-Sequenzen enthalten.

Array
beginnt mit **`[`** und endet mit **`]`**. Es enthält eine durch Kommata geteilte, geordnete Liste von *Werten*, gleichen oder verschiedenen Typs. Leere Arrays sind zulässig.

Objekt
beginnt mit **`{`** und endet mit **`}`**. Es enthält eine durch Kommata geteilte, ungeordnete Liste von *Eigenschaften*. Objekte ohne Eigenschaften ("leere Objekte") sind zulässig.

Eigenschaft
besteht aus einem Schlüssel und einem Wert, getrennt durch einen Doppelpunkt (`Schlüssel:Wert`). Die Schlüssel aller Eigenschaften in einem Objekt müssen eindeutig, also paarweise verschieden sein.

- Der **Schlüssel** ist eine Zeichenkette.
- Der **Wert** ist ein *Objekt*, ein *Array*, eine *Zeichenkette*, eine *Zahl* oder einer der Ausdrücke `true`, `false` oder `null`.

Leerraum-Zeichen sind beliebig verwendbar.

XML **Extensible Markup Language** („erweiterbare Auszeichnungssprache“)

- stellt hierarchisch strukturierte Daten in Form von Textdateien dar

- wird u. a. für den plattform- und implementationsunabhängigen Austausch von Daten zwischen Computersystemen eingesetzt, insbesondere über das Internet.[1]

– einfach lesbare Textform

Der logische Aufbau entspricht einer Baumstruktur und ist damit hierarchisch organisiert. Als Baumknoten gibt es:

- Elemente, deren physische Auszeichnung mittels
  - einem passenden Paar aus Start-Tag (`<Tag-Name>`) und End-Tag (`</Tag-Name>`) oder
  - einem Empty-Element-Tag (`<Tag-Name />`) erfolgen kann,
- Attribute als bei einem Start-Tag oder Empty-Element-Tag geschriebene Schlüsselwort-Werte-Paare (`Attribut-Name="Attribut-Wert"`) für Zusatz-Informationen über Elemente (eine Art Meta-Information),
- Verarbeitungsanweisungen (`<?Ziel-Name Parameter ?>`, engl. *Processing Instruction*),
- Kommentare (`<!-- Kommentar-Text -->`), und
- Text, der als normaler Text oder in Form eines CDATA-Abschnittes (`<![CDATA[ beliebiger Text]]>`) auftreten kann.

Vergleich

JSON

```
{
  "Herausgeber": "Xema",
  "Nummer": "1234-5678-9012-3456",
  "Deckung": 2e+6,
  "Währung": "EURO",
  "Inhaber": {
    "Name": "Mustermann",
    "Vorname": "Max",
    "männlich": true,
    "Hobbys": [ "Reiten", "Golfen", "Lesen" ],
    "Alter": 42,
    "Kinder": [],
    "Partner": null
  }
}
```

XML

```
<Kreditkarte
  Herausgeber="Xema"
  Nummer="1234-5678-9012-3456"
  Deckung="2e+6"
  Waehrung="EURO">
  <Inhaber
    Name="Mustermann"
    Vorname="Max"
    maennlich="true"
    Alter="42"
    Partner="null">
    <Hobbys>
      <Hobby>Reiten</Hobby>
      <Hobby>Golfen</Hobby>
      <Hobby>Lesen</Hobby>
    </Hobbys>
    <Kinder />
  </Inhaber>
</Kreditkarte>
```

- JASON Dateien benötigen weniger Speicherplatz, sind einfacher zu lesen und zu schreiben

- XML ist vielseitiger einsetzbar als JSON

Parser

(engl. *to parse*, „analysieren“, bzw. lateinisch *pars*, „Teil“; im Deutschen gelegentlich auch **Zerteiler**)

ist ein Computerprogramm, das für die Zerlegung und Umwandlung einer beliebigen Eingabe in ein für die Weiterverarbeitung brauchbares Format zuständig ist.

Wird benötigt um Daten in eine von unserem Programm verwendbare Form zu bringen.

Um unsere XML Dateien einzulesen können wir z.B DOM oder SAX-Parser benutzen

Document Object Model (**DOM**)

- ist eine Spezifikation einer Schnittstelle für den Zugriff auf HTML – oder XML-Dokumente.

- DOM-Schnittstelle ist sehr einfach aufgebaut

- parst ein ganzes XML-Dokument und erstellt eine vollständige „in memory“-Darstellung des Dokuments.

Mit dem DOM-API kann man 'in beide Richtungen' arbeiten, also vom XML zum "in memory" DOM als auch vom DOM zum XML. Es eignet sich also nicht nur zum "parsen" von XML sondern auch zum Generieren von XML (-Streams oder -Files).

SAX-Parser

- erstellt keine "in-memory"-Darstellung eines XML-Dokumentes
- schneller und weniger anspruchsvoll im Speicherverbrauch
- SAXParser informiert den Client der XML-Dokumenten-Struktur durch Rückruffunktionen (Callbacks), d. h. es werden Methoden der DefaultHandler-Instanz, die dem Parser zur Verfügung stehen, ausgeführt.

Die DefaultHandler-Klasse befindet sich im Paket `org.xml.sax.helpers`. Diese implementiert den ContentHandler, den ErrorHandler, den DTDHandler und die EntityResolver-Schnittstelle. Die meisten Clients interessieren sich für die Methoden aus der ContentHandler-Schnittstelle.

Die ContentHandler-Methoden, implementiert durch den DefaultHandler, werden aufgerufen, sobald der SAX-Parser auf die entsprechenden Elemente des XML-Dokumentes trifft. Die wichtigsten Methoden in dieser Schnittstelle sind:

- die `startDocument()` und `endDocument()`-Methode, die am start- und am end-tag eines XML-Dokumentes aufgerufen werden.
- die `startElement()` und `endElement()`-Methode, die am start- und am end-tag eines Dokumentelementes aufgerufen werden.
- die `characters()`-Methode. Diese wird mit dem Inhalt, der sich zwischen start- und end-tag des jeweiligen XML-Dokumentelements befindet, aufgerufen.

Mit dem SAX-API kann man 'nur in eine Richtung' arbeiten, und zwar vom XML 'hinein' in Java. Es eignet sich also ausschließlich zum "parsen" von XML. Mit SAX kann man also kein XML (-Streams oder -Files) erzeugen.

Vielen Dank für Ihre Aufmerksamkeit

Quelle:
Wikipedia
http://www.webmasterpro.de/coding/article/json-als-xml-alternative.html
http://totheriver.com/learn/xml/xmltutorial.html