Hochschule für Technik, Wirtschaft und Kultur Leipzig
Fakultät: Informatik, Mathematik und Naturwissenschaften
Studiengang: Informatik
Modul: Softwareprojekt
Gruppe: 6

Kognitive Suche - Technologierecherche

Thema: Ansteuerung der APIs von Faroo und Google

Name: Hendrik Sawade

Matrikel: 13INB

Datum: 24. November 2014

Inhaltsverzeichnis

1 Einleitung

In den folgenden Abschnitten werden die Funktionsweise der APIs von Faroo und Google erläutert. Des weiteren wird der Zugriff auf diesen und die Ausgabe der Suchergebnisse dargestellt. Die weitere Handhabung der Daten wird ebenfalls erklärt.

2 Zugriff auf die Faroo API

Als erstes wird die API der Suchmaschine Faroo näher betrachtet.
Um den Zugriff auf die Faroo API zu erhalten, besorgt man sich zunächst einen API key von der Website http://www.faroo.com/hp/api/api.html#description . Dazu ist eine kostenlose Registrierung bei Faroo notwendig. Des weiteren wird ein Server mit einer festen IP benötigt, da Faroo die Suchanfragen des Servers aufzeichnet. Dies dient zur Begrenzung der Anfragen an Faroo.
Mit Erhalt der API Key kann auf die API zugegriffen werden. Ist kein Server zur Hand, gibt es die Möglichkeit, diese API über einen Fremdanbieter, zum Beispiel Mashape, anzusteuern. Mashape dient dabei als Proxy. Unter http://www.mashape.com ist ebenfalls eine kostenlose Registratur unentbehrlich, um mit Faroo API über Mashape zu kommunizieren. Hierzu ist kein Server mit einer festen IP erforderlich. Nach der Registrierung bei Mashape bekommt man ebenfalls einen API key, den sogenannten „X-Mashape-Key“. Dieser dient wie bei Faroo der Begrenzung der Anfragen.
Bei beiden Verfahren ist die Grenze bei etwa einer Million Anfragen im Monat erreicht.

2.1 Rückgabeformat der Suchanfrage

Die Faroo API gibt die Suchanfrage in den vier folgenden Formaten aus: JSON, JSON-P, XML oder RSS. Aus diesen wählt der Programmierer das gewünschte Format.
Später, im Unterabschnitt 2.3 auf der Seite 6 wird der Datenempfang über das Format XML erklärt. Darauf folgt die Darlegung der Datenspeicherung in einer NoteList und dessen Ausgabe auf der Konsole.

2.2 Integration der einzelnen Komponenten und der Ausgabe

In **??** wird gezeigt, wie in der Klasse „Main“die einzelnen Klassen „LeseFaroo“„Parser“aufgerufen werden. Als Resultat wird das Suchergebnis in einer NoteList gespeichert und auf der Konsole ausgegeben.

Als Erstes wird eine Instanz von der Klasse LeseFaroo und Parser erstellt (Zeilen 8 - 9). Danach wird die URL im Dateityp String aus der Adresse zur API, dem Suchbegriff, der Länge der Ergebnisanzahl und das Rückgabeformat der API zusammengesetzt (Zeile 10). Der Suchbegriff wird dabei mit dem Parameter q zu dem Befehl „q=Suchwort “. Weitere Parameter, die mit übergeben werden sollen, sind durch „&“getrennt. „length=10“gibt die Länge der zurückzugebenden Ergebnisse an. „f=xml “bezeichnet das zurückzugebende Format.
Des weiteren folgt die Erstellung der NodeList zum Speichern der Resultate der Suchanfrage. Die NodeList wird gefüllt, indem die Methode „parse“in der Klasse Parser aufgerufen wird (Zeile 14). Die Suchergebnisse erhalten die Methode parse durch das Aufrufen der Methode getHTML in der Klasse LeseFaroo und diese als String zurück gibt. Außerdem wird der Methode getHTML die URL (Variable a) als String übergeben. Der try und catch block um diesen Anweisungsblock dienen dazu zum Fehler abfang. Genauere Erklärungen zu den Klassen LeseFaroo und Parser steht in den Abschnitten LeseFaroo und Parser.

Nachdem die NoteList gefüllt wurde beginnt die Forschleife (Zeile 20) damit die einzelnen Notes aus der NoteList auf der Konsole zu schreiben.

Der Zugriff auf die einzelnen Notes wird dadurch realisiert, dass ein einzelnes Note erstellt wird. Dies wird in Zeile 21 dargestellt. Danach wird das Note in ein Element gecastet. Durch diesen Schritt ist es möglich über die Klasse Element die Integrierten Methoden zu benutzen. Der zugriff auf die einzelnen Attribute erfolgt nun über „getElementsByTagName(„Argument“).item(0).getTextContent().trim()“. Des weiteren wird über „.trim()“Leerzeichen die eventuell im Tag sind entfernt. Danach wird über printlin die Resultate auf der Konsole ausgeben (Zeilen 25 - 35).

Listing 1: Main.java für Faroo

```
public class Main {

  public static void main(String[] args)  {

    LeseFaroo l = new LeseFaroo();
    Parser p = new Parser();
    //query
    String a = "http://www.faroo.com/api?q=test&src=news&length=10&f=xml";
    //"https://faroo-faroo-web-search.p.mashape.com/api?q=test&src=news&length=10&f=xml";
    NodeList nList = null;
    try {
      nList = p.parse(l.getHTML(a));
    } catch (SAXException | IOException | ParserConfigurationException e1) {
      // TODO Auto-generated catch block
      e1.printStackTrace();
    }
    for(int NodeAtPosition = 0; NodeAtPosition < nList.getLength(); NodeAtPosition++){
      Node result = nList.item(NodeAtPosition);
      Element e = (Element) result;

      System.out.println("\n" +
          "Ergebnis "+ NodeAtPosition +": " + e.getElementsByTagName("title").item(0).getTextContent().trim() +"\n"+
          "Website url: " + e.getElementsByTagName("url").item(0).getTextContent().trim() +"\n"+
          "Domain: " + e.getElementsByTagName("domain").item(0).getTextContent().trim() +"\n"+
          "imageUrl: " + e.getElementsByTagName("imageUrl").item(0).getTextContent().trim() +"\n"+
          "firstIndexed: " + e.getElementsByTagName("firstIndexed").item(0).getTextContent().trim() +"\n"+
          "firstPublished: " + e.getElementsByTagName("firstPublished").item(0).getTextContent().trim() +"\n"+
          "kwic: " + e.getElementsByTagName("kwic").item(0).getTextContent().trim() +"\n"+
          "author: " + e.getElementsByTagName("author").item(0).getTextContent().trim() +"\n"+
          "votes: " + e.getElementsByTagName("votes").item(0).getTextContent().trim() +"\n"+
          "isNews: " + e.getElementsByTagName("isNews").item(0).getTextContent().trim() +"\n"+
          "============================"
       );
    }
  }
}
```

Listing 1: Main.java für Faroo

2.3 LeseFaroo

In der Klasse LeseFaroo wird der Verbindungsaufbau zu der API von Faroo und das senden bzw. das empfangen der Daten realisiert. In dieser Klasse steht die Methode getHTML. Der Methode wird der zusammengesetzte URL String übergeben. Des weiteren wird ein neues URL, eine HttpURLConnection und ein BufferredReader Objekt erzeugt(Zeile 13 - 17). In Zeile 20 wird nun die Http Verbindung zu der API aufgebaut. Dazu kommen noch das setzten der Request Methode und die Request Property. in der Property wird der Faroo API Key eingetragen. Als erstes der Typ und danach der Key. Im Anschluss wird die Verbindung aufgebaut und die Anfrage übermittelt und auf die Antwort gewartet. Die while Schleife nun solange aus dem BufferedReader Daten bis dieser keine Daten mehr empfängt. Dies wird in den Zeilen 24 - 26 dargestellt. Sobald keine Daten mehr empfangen werden wird die Verbindung geschlossen und beendet. Dieser Vorgang liegt in einem try und catch block, damit bei Problemen der Verbindung oder andere Fehler diese Probleme abgefangen werden können und im danach in der Konsole ausgeben. Die empfangenden Daten werden in String Variable result zurückgeben.

Listing 2: LeseFaroo.java für Faroo

```
package faroo;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class LeseFaroo {

  private String key = "";

  public LeseFaroo(){
    //lese txt mit key
    key = "2CJIbhzsHU4nlSqBVZ2OP3fimb4_";
  }

  public String getHTML(String urlToRead) {
    String result = "";
    try {
      URL url;
      HttpURLConnection conn;
      url = new URL(urlToRead + "&key=" + key);
      conn = (HttpURLConnection) url.openConnection();
      conn.setRequestMethod("GET");

      BufferedReader rd;
      String line;
      rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));
      while ((line = rd.readLine()) != null) {
        result += line;
      }
      rd.close();
    } catch (IOException e) {
      e.printStackTrace();
    } catch (Exception e) {
      e.printStackTrace();
    }
    return result;
  }
}

/**
 * Please add the following API key to your query url:
&key=2CJIbhzsHU4nlSqBVZ2OP3fimb4_
 *
 */
```

Listing 2: LeseFaroo.java für Faroo

2.4 Parser

In der Klasse Parser werden die empfangenden Daten, die in der Struktur einer XML Datei sind in eine NoteListe überführt. Die Anweisungen stehen zum Fehlerabfang in einem try und catch block (Zeile 17 - 30). Zu erst wird ein DocumentBuilderFactory in einer neuen Instance erstellt (Zeile 18). Dies wird benötigt um damit arbeiten zu können. Danach wird ein DocumentBuilder erzeugt (Zeile 19), damit die Daten die zu einem Document werden sollen erstellt werden können. Mit dem DocumentBuilder werden auch ein InputSource und ein StringReader erstellt (Zeile 20). Der StringReader liest den XML String ein und übergibt ihn der InputSource. Dieser über gibt ihm dem Builder, der Builder generiert das Document schließlich. Nach der Dokumenten Erzeugung werden die Tags die „result„heißen einer NoteList hinzugefügt (Zeile 22). Danach wird die NoteListe zurückgeben (Zeile 24). Des weiteren wird der DOM Parser aus den Standard Bibliotheken von Java verwendet.

Listing 3: Parser.java für Faroo

```
1  package faroo;
2
3
4  import java.io.IOException;
5  import java.io.StringReader;
6
7  import javax.xml.parsers.DocumentBuilder;
8  import javax.xml.parsers.DocumentBuilderFactory;
9  import javax.xml.parsers.ParserConfigurationException;
10
11 import org.w3c.dom.Document;
12 import org.w3c.dom.NodeList;
13 import org.xml.sax.InputSource;
14 import org.xml.sax.SAXException;
15
16 public class Parser {
17
18   public  NodeList parse(String xmlString) throws SAXException, IOException, ↩
         ParserConfigurationException{
19
20     try
21     {
22       DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
23       DocumentBuilder builder = factory.newDocumentBuilder();
24       Document document = builder.parse( new InputSource( new StringReader( ↩
             xmlString ) ) );
25
26       NodeList nList = document.getElementsByTagName("result");
27
28       return nList;
29
30     }catch (Exception e) {
31       e.printStackTrace();
32       return null;
33     }
34   }
35 }
```

Listing 3: Parser.java für Faroo

2.5 Ausgabe der Ergebnisse

Dies sind die Ergebnisse mit dem Suchbegriff „Test“die die API zurückgibt.

Listing 4: Ergebnisse.txt für Faroo

```
Ergebnis 0: Audi TT Launched With Virtual Test Drive
Website url: http://www.ubergizmo.com/2014/11/audi-tt-launched-with-virtual-test-drive/
Domain: www.ubergizmo.com
imageUrl: http://cdn2.ubergizmo.com/wp-content/uploads/2014/11/audi-samsung.jpg
firstIndexed: 2014-11-24T17:52:20.9363541
firstPublished: 2014-11-24T17:49:57
kwic: ... S Coupe or the TT Roadster, fret not, you can use the Gear VR for a comparative review. [ Press Release ] Audi TT Launched With Virtual Test
author: Edwin Kee
votes: 20
isNews: true
============================

Ergebnis 1: Time in space exposes materials to the test of time
Website url: http://phys.org/news336051933.html
Domain: phys.org
imageUrl: http://cdn.phys.org/newman/gfx/news/tmb/2014/timeinspacee.jpg
firstIndexed: 2014-11-24T17:44:51.8343972
firstPublished: 2014-11-24T16:45:42
kwic: Much like that pickup truck rusting in your backyard thanks to time, rain and the elements, extended stays in the brutal environment of space ...
author:
votes: 20
isNews: true
============================
```

Listing 4: Ergebnisse.txt für Faroo

3 Zugriff der Google custuemer Search API

Als zweite API wird nun die Google custum Search API erläutert. Um auf die API zuzugreifen zu können, benötigt man als erstes die Bibliothek “Jsoup„. Diese muss man sich von der Website http://jsoup.org herunterladen und in eine Programmierumgebung integrieren. Diese Bibliothek ist ein Java HTML 5 Parser der mit dem DOM Parser zusammen arbeitet. Des weiteren unterstützt die Bibliothek CSS und jquery. Damit man nun auf die Google API zugreifen kann, werden mehrere Elemente erzeugt(Zeilen 15 - 18). Dazu gehören Die Strings google, dort wird der zugriff auf den Server gespeichert. Search, hier wird der Suchbegriff abgelegt. Charset, hier wird die Codierung festgelegt, wie die anfrage gesendet wird und zum schluss wird der userAgent festgelegt. Diese dient zur Authentifizierung. Dies Prinzip ist das gleiche wie bei der API von Faroo mit dem API-key.

Danach folgt der Verbindungsaufbau. Hierzu wird ein Jsoup.connect Objekt erzeugt. In dem Connect Objekt werden nun die URL, der Suchbegriff, das charset und der userAgent mit übergeben. An dieser Stelle ist es außerdem möglich viele weitere Parameter mit zu übergeben. Eine Liste dieser Parameter befindet sich oben im Abschnitt Zugriff auf die Faroo API. Die Antwort von der API wird nun in ein Elements Objekt gespeichert (Zeile 20).

Über eine for Schleife werden nun die einzelnen Ergebnisse aus der Suchanfrage die in dem Elements Objekt gespeichert sind ausgelesen ud in einzelne String variablen gespeichert. dazu gehört der Titel der Website und die dazu gehörige URL (Zeile 22 - 24). Da die URL noch in einem falschen Format codiert ist wird diese nun noch über einen URLDecoder entschlüsselt (Zeile 25 - 28). So kann man später damit besser weiter arbeiten. In der Zeilen 26 - 29 wird geprüft ob die Decodierung des Strings erfolgreich war, wenn dies nicht der Fall ist bricht die Suchanfrage ab und es wird ein Exeption ausgeben.

Als letztes wird nun noch der Titel und die URL auf der Konsole ausgeben (Zeile 30 - 31). Die Maximale Anzahl der zurückgebenden Suchergebnisse beträgt in der kostenlosen Variante zehn Ergebnisse.

Listing 5: Main.java für Google Search API

```
package google;

import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URLDecoder;
import java.net.URLEncoder;

import javax.lang.model.util.Elements;

import org.jsoup.Jsoup;
import org.w3c.dom.Element;

public class Main {

  public static void main(String[] args) throws IOException {
    // TODO Auto-generated method stub


    String google = "http://www.google.com/search?q=";
    String search = "Karsten Wicker";
    String charset = "UTF-8";
    String userAgent = "43ndr1k "; // Change this to your company's name and bot homepage!

    org.jsoup.select.Elements links =  Jsoup.connect(google + URLEncoder.encode(search, charset)).userAgent(userAgent).get().select("li.g>h3>a");

    for (org.jsoup.nodes.Element link : links) {
        String title = link.text();
        String url = link.absUrl("href"); // Google returns URLs in format "http://www.google.com/url?q=<url>&sa=U&ei=<someKey>".
        url = URLDecoder.decode(url.substring(url.indexOf('=') + 1, url.indexOf('&')), "UTF-8");
       // String data = link.;
        if (!url.startsWith("http")) {
            continue; // Ads/news/etc.
        }

        System.out.println("Title: " + title);
        System.out.println("URL: " + url);
       // System.out.println("data: " + data);
    }


  }

}
```

Listing 5: Main.java für Google Search API

3.1 Ausgabe der Ergebnisse

Dies sind die Ergebnisse mit dem Suchbegriff „Kasten Weicker“die die API zurückgibt.

Listing 6: Ausgabe.txt für Faroo

```
Title: Karsten Wicker Profile | Facebook
URL: https://de-de.facebook.com/public/Karsten-Wicker

Title: Karsten Wicker | Facebook
URL: https://de-de.facebook.com/people/Karsten-Wicker/100002296276194
```

Listing 6: Ausgabe.txt für Faroo

4 Einstellungen

4.1 Parameter

Bei dem zugriff auf die API können zahlreiche Parameter für Einstellungen, Informationen und Filterung mitsenden. Im Folgenden sind diese Parameter mit einer Erklärung aufgelistet. Die Tabelle stammt von [1].

Parameter	Typ	Erklärung
q	String	Query - Suchwort | Suchwörter -> Schreibweise q= Suchwort
start	number	Bei welchem Suchbegriff soll die suche anfangen (default=1)
length	number	Wie viele Ergebnisse sollen ausgeben werden. Length (default=10; maximum=10)
rlength	number	Related length (default=20) : maximum number of related news per item, only for Trending News
l	string	Language en English (default) de German zh Chinese
src	string	Source **web** Web Search (default) Sorted by relevancy Contains all kinds of results **news** News Search Sorted by publishing date Contains only news articles from newspapers, magazines and blogs **news** Trending News (if empty q) Does a topic aggregation (i.e. it groups news of the same topic together) Sorts the topics by buzz (i.e. the number of different news sources who are reporting on this topic) Sorts the articles inside a topic by publishing time For each topic the latest article is selected as main article, the other related articles are grouped in the related property **topics** Trending Topics Similar to Trending News: Trending News: for each topic a main article with all properties + related articles with title, url, domain only. Trending Topics: for each topic all the related articles are provided with all properties (more data, slower transfer). **trends** Trending Terms Trending terms, sorted by buzz (number of sources reporting on same term). **suggest** Suggestions Suggestions include auto completes for query substrings and corrections for misspelled terms. When using the above searches with parameter i=true, the suggestions are already included in the search result.
kwic	boolean	Keyword in context **false** snippet is selected from the beginning of the article. **true** (default) snippet is selected from the article parts containing the keywords.
i	boolean	Instant search **false** (default) searches for query q **true** searches for best suggestion if query q is substring or misspelled. Slower search!
f	string	Result format **json** JSON (default), JSON-P (JSON-P, if jsoncallback is defined) **xml** XML (only for Web Search, News Search, Trending News, Trending Topics) **rss** RSS (only for News Search, Trending News)
jsoncallback	string	JSON-P callback function name The JSON data is embedded in JavaScript code to support cross-domain requests.
key	string	API key.

4.2 Return Values

Die API liefert die folgenden Rückgabe Werte und Ergebnisse zurück. Die Tabelle stammt von [1].

Property	Type	Description
results	array	Result array
title	string	Article title
kwic	string	Article snippet with keyword in context
url	string	Article url
iurl	string	Main article image url
domain	string	Domain
author	string	Article author
news	boolean	**true** Article is from newspapers, magazines and blogs **false** Article is from other sources
date	number	Publishing date JavaScript equivalent of a DateTime
related	array	Array of related articles For Trending news only (src=news and empty q)
title	string	Title
url	string	URL
domain	string	Domain
query	string	Query suggestion Actually used query, might differ from original query parameter, if instant search i=true.
count	number	Number of results found
start	number	Start position of results requested
length	number	Number of results requested
time	number	Search time Pure search latency in milliseconds, not including the request/response transfer over the Internet.
suggestions	array	Query suggestions String array of query suggestions, if instant search i=true.

4.3 HTTP Status Codes

Falls es bei dem ansprechen der API Fehler entstehen, liefert die Faroo API einen Fehler Code zurück. Die verschiedenen Code Werte werden in der nachfolgenden Tabelle aufgeführt. Die Tabelle stammt von [1].

Code	Description	Explanation
200	OK	Search sucessfully completed.
401	Unauthorized	Please register an API key.
429	Too many requests	The rate limit has been exceeded. Most likely you have exceeded the 1 query/second rate limit. The rate limiter will then block all queries until the average traffic returns below 1 query/second. The blocking period is proportional to the number of exceeding requests. As search is often a random arrival process, it is normal that the distance between queries is sometimes below 1s. Therefore the blocking starts only after the average distance of 10 consecutive queries below 1s.

Quellen

[1] Faroo: *Faroo Homepage*, Abrufdatum: 22.11.2014. `http://www.faroo.com/hp/api/api.htmlkey`.