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:** 30. November 2014

# Inhaltsverzeichnis

**Tabellenverzeichnis** **2**

**Listings** **2**

**1 Einleitung** **3**

**2 Zugriff auf die Faroo API** **3**
2.1 Rückgabeformat der Suchanfrage . . . . . 3
2.2 Integration der einzelnen Komponenten und der Ausgabe . . . . . 4
2.3 LeseFaroo . . . . . 6
2.4 Parser . . . . . 8
2.5 Ausgabe der Ergebnisse . . . . . 8

**3 Zugriff auf die Google Custom Search API** **9**
3.1 Ausgabe der Ergebnisse . . . . . 11

**4 Einstellungen** **11**
4.1 Parameter . . . . . 11
4.2 Return Values . . . . . 13
4.3 HTTP Status Codes . . . . . 14

**Quellen** **14**

# Tabellenverzeichnis

1 Parameter . . . . . 12
2 Return Values . . . . . 13
3 HTTP Status Codes . . . . . 14

# Listings

1 Main.java für Faroo . . . . . 5
2 LeseFaroo.java für Faroo . . . . . 7
3 Parser.java für Faroo . . . . . 8
4 Ergebnisse.txt für Faroo . . . . . 9
5 Main.java für Google Search API . . . . . 10
6 Ausgabe.txt für Faroo . . . . . 11

# 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. Informationen zur Umsetzung des Programmierens stammen aus [1].

# 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 der API von Faroo über Mashape zu kommunizieren. Hierzu ist kein Server mit einer festen IP erforderlich. Nach der Registrierung bei Mashape erhält 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 NodeList und dessen Ausgabe auf der Konsole.

## 2.2 Integration der einzelnen Komponenten und der Ausgabe

In Listing 1 wird gezeigt, wie in der Klasse „Main“ die einzelnen Klassen „LeseFaroo“ und „Parser“ aufgerufen werden. Als Resultat wird das Suchergebnis in einer NodeList gespeichert und auf der Konsole ausgegeben.

Als Erstes wird eine Instanz von den Klassen LeseFaroo und Parser erstellt (Zeilen 5 - 6). Danach wird die URL im Dateityp String aus der Adresse zur API, dem Suchbegriff, der Länge der Ergebnisanzahl und zum Rückgabeformat der API zusammengesetzt (Zeile 8). Der Suchbegriff wird dabei mit dem Parameter q zu dem Befehl „q=Suchwort“ . Weitere Parameter, die mit übergeben werden sollen, werden durch „&“ getrennt. „length=10“ gibt die Länge der zurückzugebenden Ergebnisse an. „f=xml“ bezeichnet das zurückzugebende Format. Des weiteren folgt die Erstellung der NodeList, um die Resultate der Suchanfrage zu speichern. Die NodeList wird gefüllt, indem der Methode „getHTML“ die URL (Variable a) als String übergeben wird. Die Methode „getHTML“ in der Klasse LeseFaroo gibt die Suchergebnisse als String an die Methode „parse“. Diese wird im Anschluss in der Klasse Parser aufgerufen (Zeile 12) und füllt die NodeList. Der „try and catch block“ um diesen Anweisungsblock (Zeile 12) dienen zum Abfangen von Fehlern. Genauere Erklärungen zu den Klassen LeseFaroo und Parser folgen in den Abschnitten LeseFaroo und Parser.

Nachdem die NodeList gefüllt wird, beginnt die For-Schleife (Zeile 17) die einzelnen Nodes aus der NodeList aufzurufen und sie in ein einzelnes Node zu speichern. Dieses Node wird in ein Element gecastet (Zeile 18). Die Nutzung der integrierten Methoden über das Objekt „Element“ wird so ermöglicht. Deren Zugriff auf die einzelnen Attribute erfolgt nun über „getElementsByTagName(„Argument“).item(0).getTextContent().trim()“. Der Befehl „.trim()“ entfernt die überflüssigen Leerzeichen im Tag. Danach werden über „printIn“ die Resultate auf der Konsole ausgeben (Zeilen 21 - 35).

```
1  public class Main {
2
3    public static void main(String[] args)  {
4
5      LeseFaroo l = new LeseFaroo();
6      Parser p = new Parser();
7      //query
8      String a = "http://www.faroo.com/api?q=test&src=news&length=10&f=xml";
9      //"https://faroo-faroo-web-search.p.mashape.com/api?q=test&src=news&length=10&f=xml";
10     NodeList nList = null;
11     try {
12       nList = p.parse(l.getHTML(a));
13     } catch (SAXException | IOException | ParserConfigurationException e1) {
14       // TODO Auto-generated catch block
15       e1.printStackTrace();
16     }
17     for(int NodeAtPosition = 0; NodeAtPosition < nList.getLength(); NodeAtPosition++){
18       Node result = nList.item(NodeAtPosition);
19       Element e = (Element) result;
20
21       System.out.println("\n" +
22           "Ergebnis "+ NodeAtPosition +": " + e.getElementsByTagName("title").item(0).getTextContent().trim() +"\n"+
23           "Website url: " + e.getElementsByTagName("url").item(0).getTextContent().trim() +"\n"+
24           "Domain: " + e.getElementsByTagName("domain").item(0).getTextContent().trim() +"\n"+
25           "imageUrl: " + e.getElementsByTagName("imageUrl").item(0).getTextContent().trim() +"\n"+
26           "firstIndexed: " + e.getElementsByTagName("firstIndexed").item(0).getTextContent().trim() +"\n"+
27           "firstPublished: " + e.getElementsByTagName("firstPublished").item(0).getTextContent().trim() +"\n"+
28           "kwic: " + e.getElementsByTagName("kwic").item(0).getTextContent().trim() +"\n"+
29           "author: " + e.getElementsByTagName("author").item(0).getTextContent().trim() +"\n"+
30           "votes: " + e.getElementsByTagName("votes").item(0).getTextContent().trim() +"\n"+
31           "isNews: " + e.getElementsByTagName("isNews").item(0).getTextContent().trim() +"\n"+
32           "============================"
33       );
34     }
35   }
36 }
```

Listing 1: Main.java für Faroo

## 2.3 LeseFaroo

In der folgenden Darstellung Listing 2 wird der Verbindungsaufbau zu der API von Faroo und das Senden beziehungsweise das Empfangen der Daten realisiert.

In der Klasse LeseFaroo steht die Methode getHTML. Der Methode wird der zusammengesetzte URL-String und der Identifikations-Key übergeben. Danach folgt das Erzeugen der Objekte URL und HttpURLConnection (Zeile 20 - 27). Hierzu kommt das Setzten der Methode „Request“(GET-Methode). Anschließend wird die Verbindung zur Faroo API aufgebaut. Das Objekt „BufferedReader“ wird im Anschluss erzeugt. Dieses puffert die Daten solange Faroo Daten sendet. Gleichzeitig werden in der While-Schleife die Daten aus dem „BufferedReader“ gelesen, bis der Datenempfang abgeschlossen ist. Das Ergebnis „String variable result“ speichert die gesendeten Daten (Zeile 29 - 31). Ist dies erfolgt, wird die Verbindung geschlossen und beendet. Der beschriebene Vorgang liegt in einem try und catch block. Somit können bei Schwierigkeiten der Verbindung oder anderen Fehlern diese Probleme abgefangen werden.

```java
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

Das beschriebene Programm verwendet den DOM-Parser aus den Standard Bibliotheken von Java. Listing 3 stellt dar, wie die empfangenden Daten, in eine NodeList überführt werden. Sie sind Teil einer XML-Struktur. Die Anweisungen stehen zum Abfangen von Fehler in einem try und catch block (Zeile 20 - 34). Zuerst wird das Objekt „Document-BuilderFactory“ in einer neuen Instanz erstellt (Zeile 22). Als Produkt entsteht das Objekt „DocumentBuilder“ (Zeile 23). Dieses Objekt erstellt die Daten, welche zu einem Document zusammengefügt werden. Des weiteren werden auch ein InputSource und ein StringReader erzeugt(Zeile 24). Der StringReader liest den XML-String ein und übergibt ihn dem Objekt InputSource. InputSource händigt dem Builder die XML-Struktur aus. Schließlich generiert der Builder das Document. Nach der Erzeugung des Dokumentes werden die Tags, „result“ genannt, einer NodeList hinzugefügt (Zeile 26). Danach wird die NodeList an Main zurückgeben (Zeile 28).

```
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

Listing 4 präsentieren die Ergebnisse des Suchbegriffes „Test“, welche die API zurückgibt. In diesem Fall liegt das Resultat als Liste vor, in welcher zu jedem Ergebnis mehrere Tags erscheinen. In dieser Darstellung wird nur ein Ausschnitt aus der Gesamtmenge gezeigt. Die Tabelle im Abschnitt 4 auf der Seite 11 gibt alle Rückgabewerte wieder, die Faroo unterstützt.

```
1  Ergebnis 0: Audi TT Launched With Virtual Test Drive
2  Website url: http://www.ubergizmo.com/2014/11/audi-tt-launched-with-virtual-↩
       test-drive/
3  Domain: www.ubergizmo.com
4  imageUrl: http://cdn2.ubergizmo.com/wp-content/uploads/2014/11/audi-samsung.jpg
5  firstIndexed: 2014-11-24T17:52:20.9363541
6  firstPublished: 2014-11-24T17:49:57
7  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
8  author: Edwin Kee
9  votes: 20
10 isNews: true
11 ============================
12
13 Ergebnis 1: Time in space exposes materials to the test of time
14 Website url: http://phys.org/news336051933.html
15 Domain: phys.org
16 imageUrl: http://cdn.phys.org/newman/gfx/news/tmb/2014/timeinspacee.jpg
17 firstIndexed: 2014-11-24T17:44:51.8343972
18 firstPublished: 2014-11-24T16:45:42
19 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 ...
20 author:
21 votes: 20
22 isNews: true
23 ============================
```

Listing 4: Ergebnisse.txt für Faroo

# 3 Zugriff auf die Google Custom Search API

Als zweite API wird nun „Custom Search“, die API von Google“, erläutert. Die Bibliothek „Jsoup“ bildet die Grundlage für den Zugriff auf die API. Jsoup ist ein Java HTML 5 Parser, der mit dem DOM-Parser zusammenarbeitet. Diese Biblothek wird von der Website `http://jsoup.org` heruntergeladen und in eine Programmierumgebung integriert. Des weiteren unterstützt die Bibliothek die textbasierten Auszeichnungssprachen „CSS“ und „jquery“. Für den Zugriff auf die Google API werden mehrere Elemente erzeugt (Zeilen 19 - 22). Dies kann durch folgende vier Variablen geschehen: Die Variable „String google“ speichert den Zugriff auf den Server. Eine andere Variable, „String Search“, legt den Suchbegriff ab. „String Charset“ legt die Codierung fest, welche die Art der Anfrage bestimmt. Die vierte Variable „String userAgent“ dient zur Authentifizierung und somit der Genehmigung des Zugriffes. Dies Prinzip ist identisch mit der API-Key von Faroo. Danach folgt der Verbindungsaufbau. Hierzu wird das Objekt „Jsoup.connect“ erzeugt. Dem Objekt werden die URL, der Suchbegriff, das charset und der userAgent mit übergeben. An dieser Stelle können viele weitere Parameter zugestellt werden. In der Tabelle im Abschnitt 4 auf der Seite 11 stehen viele Rückgabewerte, welche die API von Google unterstützt. Die Antwort von der API wird nun in Objekte des Typs „Elements“ gespeichert, welches „links“ heißt (Zeile 24). Über eine For-Schleife werden die einzelnen Ergebnisse aus der Suchanfrage, die in dem Objekt „Elements“ gespeichert sind, ausgelesen und in einzelne String -Variablen gespeichert. Dazu gehört der Titel der Website und deren URL (Zeile 27 - 28). Da die URL noch in einem falschen Format codiert ist, muss diese über „URLDecoder“ entschlüsselt werden(Zeile 29). Das erleichtert die weitere Arbeit mit den empfangenden Daten. Nun wird geprüft, ob die Decodierung des Strings erfolgreich war (Zeilen 31 - 33). Ist dies nicht der Fall, bricht die Codierung ab und eine Exeption wird ausgegeben. Zum Schluss wird der Titel und die URL auf der Konsole dargestellt (Zeile 35 - 36). Die maximale Anzahl der zurückgebenden Suchergebnisse beträgt in der kostenlosen Variante zehn Ergebnisse.

```java
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

In der Veranschaulichung Listing 6 sind die Ergebnisse mit dem Suchbegriff „Kasten Weicker“ notiert, welche die API Custom Search zurückgibt. Sie haben den Titel der Website und den dazu gehörigen Link zum Inhalt.

```
1 Title: Karsten Wicker Profile | Facebook
2 URL: https://de-de.facebook.com/public/Karsten-Wicker
3
4 Title: Karsten Wicker | Facebook
5 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 mitgesendet werden. Im Folgenden sind diese Parameter mit einer Erklärung aufgelistet. Die Tabelle stammt von [2].

| 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<br>**web** Web Search (default) Sorted by relevancy Contains all kinds of results<br><br>**news** News Search<br>Sorted by publishing date Contains only news articles from newspapers, magazines and blogs<br><br>**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).<br><br>**trends** Trending Terms Trending terms, sorted by buzz (number of sources reporting on same term).<br><br>**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.<br>**true** (default) snippet is selected from the article parts containing the keywords. |
| i | boolean | Instant search **false** (default) searches for query q<br>**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)<br>**xml** XML (only for Web Search, News Search, Trending News, Trending Topics)<br>**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. |

Tabelle 1: Parameter

## 4.2 Return Values

Die API liefert folgende Rückgabewerte und Ergebnisse zurück. Die Tabelle stammt von [2].

| 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. |

Tabelle 2: Return Values

## 4.3 HTTP Status Codes

Falls es bei dem Ansprechen der API Fehler entstehen, sendet die Faroo API einen Fehlercode. Die verschiedenen Code-Werte werden in der nachfolgenden Tabelle aufgeführt. Die Tabelle stammt von [2].

| 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.<br>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. |

Tabelle 3: HTTP Status Codes

# Quellen


[1] PRESS GALILEO: *galileo-press*, Abrufdatum: 22.11.2014. `http://openbook.galileo-press.de/javainsel/`.

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