Verteilte Systeme und Kommunikationsnetze

REST-Services

Praktikum 5

Fachhochschule Bielefeld
Campus Minden
Studiengang Informatik

Beteiligte Personen:

Name	Matrikelnummer
Michael Nickel	1120888
Jan Augstein	1119581

Aufgaben:

Aufgabe	Gelöst
Aufgabe 1	alle Teilpunkte
Aufgabe 2	alle Teilpunkte

---

4. Dezember 2017

Inhaltsverzeichnis

1 Aufgabe 2

[1]

1.1 Frage- bzw. Aufgabenstellung

Unter Verwendung der in der Einleitung erwähnten Daten sollen Sie in Java einen Clienten erstellen, der für einen beliebigen Ort die maximale- (tx) und minimale Temperatur (tn) anzeigen kann. Die restlichen Informationen müssen nicht berücksichtigt werden.

1.2 Lösung

Für die Lösung dieser Aufgabe wurde eine Klasse mit dem Namen *MaxMinTemp* erstellt in der alle Methoden, inklusive der main-Methode, realisiert wurden.

Zunächst wurde eine *readUrl*-Methode geschrieben, der man eine URL-Adresse übergibt und alle Zeichen auf dieser Seite einem StringBuilder hinzufügt und diesen dann auch zurückgibt (siehe Zeile 1 bis 16).

```
1  private static String readUrl(String urlString) throws Exception {
2
3          BufferedReader reader = null;
4          try {
5                  URL url = new URL(urlString);
6                  reader = new BufferedReader(new
                       InputStreamReader(url.openStream()));
7                  StringBuilder sb = new StringBuilder();
8                  int read;
9                  while ((read = reader.read()) != -1)
10                         sb.append((char) read);
11                 return sb.toString();
12         } finally {
13                 if (reader != null)
14                         reader.close();
15         }
16 }
```

Listing 1: readUrl-Methode

Weiterhin wurde eine *getDate*-Methode geschrieben, die später dazu benötigt wird, um später nur die Temperaturen für den aktuellen Tag anzuzeigen. Hierfür wird das richtige Format mit einem Objekt von *DateFormat* gesetzt. Danach wird ein neues Datum erzeugt und mit dem richtigen Format zurückgegeben (siehe Zeile 1 bis 5).

```
1  private static String getDate() {
2          DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
3          Date date = new Date();
4          return dateFormat.format(date);
5  }
```

Listing 2: getDate-Methode

Die letzte Methode namens *ausgabeTemp* übernimmt den wichtigsten Teil der Aufgabe, da sie dafür sorgt, dass wir am Ende die Ausgaben der Temperaturen auf der Konsole haben.

Da wir hier mit dem JSON-Format arbeiten, benutzen wir das *org.json*-package, um mit den JSON-Objekten zu arbeiten.**[2]**

Zu Beginn wird der *Scanner* benutzt, damit der Benutzer den Namen, so wie die Postleitzahl der Stadt eingeben kann, für die der Benutzer die Wetterausgabe haben will. Anschließend wird ein neues JSONObject erstellt, dem man dann mithilfe der *readUrl*-Methode die API der OpenWeatherMap zusammen mit der zuvor eingegebenen Stadt (plus zugehöriger Postleitzahl) übergibt (siehe Zeile 1 bis 9).

Da die Daten im JSON-Format immer in einer untergeordneten Sammlung über Name/Werte-Paare gespeichert sind, muss man zunächst gucken wie man an die Temperaturen der API kommt. Dazu eignet sich die Seite `https://jsonformatter.curiousconcept.com/` (Vorschlag aus dem Praktikumsblatt. Hier werden die JSON-Daten vernünftig formatiert, so dass man einen guten Überblick über alle Daten bekommt. Wie man in Abbildung 1 sieht, sind die Daten für die Temperatur in einem *main*-Objekt, welches wiederum in einem *list*-Array gespeichert ist.


```
{
   "cod":"200",
   "message":0.0036,
   "cnt":40,
   "list":[
      {
         "dt":1485799200,
         "main":{
            "temp":261.45,
            "temp_min":259.086,
            "temp_max":261.45,
            "pressure":1023.48,
            "sea_level":1045.39,
            "grnd_level":1023.48,
            "humidity":79,
            "temp_kf":2.37
         },
```


Abbildung 1: Formatierte JSON-Datei der API

Mit dem Code aus Zeile 10 bis 13 greifen wir also auf das *main*-Objekt zu und der String *dt_txt* speichert das Datum der Wettervorhersage. Anschließend erstellt man noch ein String für das aktuelle Datum mit der *getDate*-Methode und das Datum aus dem *dt_txt*

String, welcher jedoch nicht die Uhrzeit enthält (siehe Zeile 14,15). Diese Strings sind später für die Bedingung in der while-Schleife wichtig. Vor der while-Schleife sucht man sich aus der JSON-Datei noch den Namen der Stadt raus, da wir diese mit ausgeben wollen (siehe Zeile 16 bis 18).**[3]**

Nun gehen wir in die while-Schleife. Diese wird nur ausgeführt wenn *dateOfForecast* mit *date* übereinstimmt. Dies ist wichtig, da wir die API für die 5-Tage Vorhersage benutzen, aber lediglich die Vorhersage für den aktuellen Tag wollen (siehe Zeile 19). Zeile 20 bis 23 stimmen mit den Zeilen 12 bis 15 überein, daher werden diese nicht nochmal erklärt. Um nun die minimale/maximale Temperatur zu bekommen, müssen wir wie man in Abbildung 1 sieht auf das *main*-Objekt zugreifen und von dort dann auf die Integer-Werte *temp_min* und *temp_max* (siehe Zeile 24 bis 26). Anschließend werden noch alle mit einer üblichen *println* ausgegeben und *i* wird erhöht um auf das nächste *main*-Objekt zuzugreifen (siehe Zeile 27,28).

```
1  private static void ausgabeTemp() throws Exception {
2          Scanner scan = new Scanner(System.in);
3          System.out.println("Ã¼Fr welche Stadt wollen Sie das Wetter
               anzeigen?");
4          String city = scan.nextLine();
5          System.out.println("Geben Sie bitte die Postleitzahl ein: ");
6          String zipCode = scan.nextLine();
7          scan.close();
8          try {
9                  JSONObject json = new
                       JSONObject(readUrl("http://api.openweathermap.org/data/2.5/forecastq="
                       + city+ "&zip=" + zipCode +
                       "&units=metric&APPID=722920868a0a0266c859a174da690bc1"));
10                 JSONArray list = json.getJSONArray("list");
11                 int i = 0;
12                 JSONObject main = list.getJSONObject(i);
13                 String dt_txt = main.getString("dt_txt");
14                 String date = getDate();
15                 String dateOfForecast = dt_txt.substring(0, 10);
16                 JSONObject cityNameObject = json.getJSONObject("city");
17                 String cityName = cityNameObject.getString("name");
18                 System.out.println(cityName + " :\n");
19                 while (dateOfForecast.equals(date)) {
20                         main = list.getJSONObject(i);
21                         dt_txt = main.getString("dt_txt");
22                         date = getDate();
23                         dateOfForecast = dt_txt.substring(0, 10);
24                         JSONObject temp = main.getJSONObject("main");
25                         double tn = temp.getDouble("temp_min");
26                         double tx = temp.getDouble("temp_max");
27                         System.out.println("min Temp: " + tn + " C , " +
                               "max Temp: " + tx + " C , " + dt_txt);
28                         i++;
29                 }
30
31                 } catch (JSONException e) {
32                         e.printStackTrace();
33                 }
```

```
34 ||                }
```

Listing 3: ausgabeTemp

Die Ausgabe könnten dann so aussehen:

```
Für welche Stadt wollen Sie das Wetter anzeigen?
Minden
Geben Sie bitte die Postleitzahl ein:
32427
Minden :

min Temp: 5.22 °C , max Temp: 5.44 °C , 2017-12-04 21:00:00
min Temp: 4.94 °C , max Temp: 5.11 °C , 2017-12-05 00:00:00
```

Abbildung 2: Minden, min/max Temperatur

1.3 Ergebnis

Wir können mithilfe der OpenWeatherMap die Daten für die maximale/minimale Temperatur eines beliebigen Ortes anzeigen.

2 Quellen

Literatur

[1] OpenWeatherMap, 5 day weather forecast
`https://openweathermap.org/forecast5`, 4.12.2017

[2] org.json, JSON In Java
`https://mvnrepository.com/artifact/org.json/json`, 4.12.2017

[3] vikingmaster, Java - How to get object value in JSON array?
`https://stackoverflow.com/questions/22386344/java-how-to-get-object-value-in-json-array`, 4.12.2017