

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

5. Dezember 2017

Inhaltsverzeichnis

1	Aufgabe 1	3
1.1	Teilaufgabe a)	3
	Frage- bzw. Aufgabenstellung	3
	Lösung	3
	Ergebnis	3
1.2	Teilaufgabe b)	3
	Frage- bzw. Aufgabenstellung	3
	Lösung	3
	Ergebnis	4
1.3	Teilaufgabe c)	4
	Frage- bzw. Aufgabenstellung	4
	Lösung	4
	Ergebnis	4
2	Aufgabe 2	4
2.1	Frage- bzw. Aufgabenstellung	4
2.2	Lösung	4
2.3	Ergebnis	8
3	Quellen	8

1 Aufgabe 1

1.1 Teilaufgabe a)

Frage- bzw. Aufgabenstellung

Machen Sie sich mit der Openweather-Dokumentation vertraut : <https://openweathermap.org/api>

Lösung

Die *Openweather API* kann dazu benutzt werden, das aktuelle Wetter, das Wetter eines bestimmten Tages oder 5/16 Tage Vorhersagen abzurufen. Zur Abfrage vom Wetter wird ein *API-KEY* gebraucht, welcher einzigartig ist und vom Account abhängt. Die abgefragten Informationen werden in verschiedenen Formaten dargestellt: JSON, XML und HTML. Der sogenannte *CALL* kann mit verschiedenen Parametern ausgeführt werden. Zum einen kann das Wetter per Stadtnamen, Stadt ID, geografischen Koordinaten oder ZIP-Code (Postleitzahl) abgerufen werden. Des Weiteren sind Spezifizierungen für einzelne Informationen, wie zum Beispiel Wind und Temperatur möglich.

Ergebnis

Wir wissen was man mit der *Openweather API* machen kann.

1.2 Teilaufgabe b)

Frage- bzw. Aufgabenstellung

Informieren Sie sich über das JSON-Format. Welche Vorteile bietet es gegen über XML und wie ist ein JSON-Objekt aufgebaut?

Lösung

JSON (JavaScript Object Notation) ist ein Datenaustauschformat, welches leicht zu lesen und zu analysieren ist. JSON baut auf zwei Elementen auf: Name/Werte Paare und geordnete Listen von Werten. Ein Objekt ist eine ungeordnete Menge von Name/Werte-Paaren und ist wie folgt aufgebaut : [1]

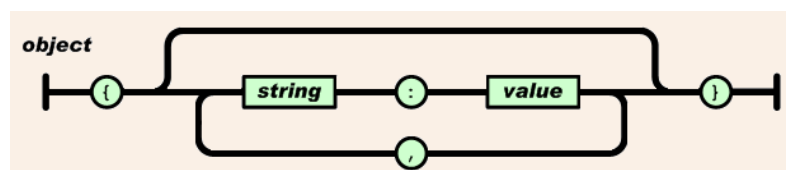


Abbildung 1: Aufbau eines Objekts in JSON

Gegenüber XML ist das JSON-Format leichter zu lesen und zu verstehen, was dazu führt, dass JSON besser für flexible Schnittstellen ist, da es durch die einfache Syntax leichter zu generieren ist. Des Weiteren generiert JSON in der Regel weniger Overhead als XML. [2]

Ergebnis

Wir wissen wie JSON und die Objekte aufgebaut sind und welche Vorteile es bietet.

1.3 Teilaufgabe c)

Frage- bzw. Aufgabenstellung

REST, bzw. auch RESTful sind keine einheitlichen Standards. In der Regel findet der Austausch über das http-Protokoll statt und fußt auf dessen bekannten Methoden. Wie lauten diese Methoden und wie werden diese verwendet?

Lösung

Die Methoden lauten GET, POST, PUT und DELETE. Der GET-Befehl greift lesend auf Information, die auf einem Server hinterlegt sind, zu. Mit POST legt man eine neue Ressource auf dem Server an. Mit dem PUT-Befehl lassen sich vorhandene Ressourcen bearbeiten und DELETE wird dafür benutzt Ressourcen zu löschen. [3]

Ergebnis

Wir wissen wie REST aufgebaut ist und welche Methoden benutzt werden.

2 Aufgabe 2

[4]

2.1 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.

2.2 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  
7 |             InputStreamReader(url.openStream()));  
8 |         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.[5]

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). Wichtig für die URL ist auch der Parameter *units=metric*, damit die Angaben in Celsius erfolgen und nicht in Fahrenheit. Außerdem wäre es sinnvoll den API-Key vom Praktikumsblatt anzugeben. Dazu benutzt man den Parameter *APPID*

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.

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

```

{
  "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 2: Formatierte JSON-Datei der API

sich aus der JSON-Datei noch den Namen der Stadt raus, da wir diese mit ausgeben wollen (siehe Zeile 16 bis 18).[6]

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("¼Fr welche Stadt wollen Sie das Wetter
4         anzeigen?");
5     String city = scan.nextLine();
6     System.out.println("Geben Sie bitte die Postleitzahl ein: ");
7     String zipCode = scan.nextLine();
8     scan.close();
9     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`

In der `main`-Methode wird dann nur noch die `ausgabeTemp`-Methode aufgerufen.

```

1 public static void main(String[] args) throws Exception {
2     ausgabeTemp();
3 }

```

Listing 4: `main`

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: 5.58 °C , max Temp: 6.99 °C , 2017-12-05 15:00:00
min Temp: 5.68 °C , max Temp: 6.74 °C , 2017-12-05 18:00:00
min Temp: 5.57 °C , max Temp: 6.28 °C , 2017-12-05 21:00:00
min Temp: 5.31 °C , max Temp: 5.66 °C , 2017-12-06 00:00:00

```

Abbildung 3: Minden, min/max Temperatur

2.3 Ergebnis

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

3 Quellen

Literatur

- [1] JSON, Einführung in JSON
<https://www.json.org/json-de.html>, 5.12.2017
- [2] w3schools.com, JSON vs XML
https://www.w3schools.com/js/js_json_xml.asp, 5.12.2017
- [3] Thomas Drilling, Konzept, Aufbau und Funktionsweise von REST
<https://www.dev-insider.de/konzept-aufbau-und-funktionsweise-von-rest-a-603152/>, 5.12.2017
- [4] OpenWeatherMap, 5 day weather forecast
<https://openweathermap.org/forecast5>, 4.12.2017
- [5] org.json, JSON In Java
<https://mvnrepository.com/artifact/org.json/json>, 4.12.2017
- [6] 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