Verteilte Systeme und Kommunikationsnetze

Aufgabe in der Reflektionswoche

Praktikum 6

Fachhochschule Bielefeld
Campus Minden
Studiengang Informatik

Beteiligte Personen:

Name	Matrikelnummer
Michael Nickel	1120888
Jan Augstein	1119581

Aufgaben:

Aufgabe	Gelöst
Aufgabe 1	alle Teilpunkte

---

17. Dezember 2017

Inhaltsverzeichnis

1 Aufgabe

1.1 Frage- bzw. Aufgabenstellung

Sie erhalten in dieser Woche die Gelegenheit, den bisher im Modul behandelten Stoff zu vertiefen. Die folgende Praktikumsaufgabe soll Ihnen dabei helfen.

Arbeiten Sie die bisher bereitgestellten Vorlesungsmaterialien nochmals nach. Fangen Sie dort an, wo Sie meinen, den größten Nachholbedarf zu haben, bzw. dort, wo Sie unsicher sind ob Sie alles verstanden haben.

1.2 Teilaufgabe 1

Frage- bzw. Aufgabenstellung

Beschreiben Sie mit eigenen Worten und einer Skizze, wie Browser-Cookies funktionieren!

Lösung

[**1**] Cookies sind Informationen die von einer besuchten Website auf dem eigenen Computer abgelegt werden, damit zum Beispiel Daten aus einem Warenkorb in einem Online-Shop nicht verloren gehen wenn man die Seite schließt und wieder öffnet. Dies ist nötig, da HTTP ein zustandsloses Protokoll ist und die Seitenaufrufe so unabhängig voneinander sind. Die Funktionsweise der Cookies ist ziemlich simpel. Der Webserver setzt bei Antworten im Browser einen Cookie. Dies wird über eine zusätzliche Cookie-Zeile geregelt, wodurch der Webserver den User wiedererkennen kann. Der einfachste Aufbau eines Cookies besteht aus einem Namen und einem Wert, was dann in etwa so aussehen könnte:

Set-Cookie: TestCookie = 1236326

Ergebnis

Wir wissen was Browser-Cookies sind und können diese mithilfe einer Skizze veranschaulichen.

1.3 Teilaufgabe 2

Frage- bzw. Aufgabenstellung

Formulieren Sie mindestens fünf Kurzfragen, die in Stichworten zu beantworten sind. Die Aufgaben sollen im Stil von Klausurfragen gestellt werden. Gerne dürfen Sie eine eigene Aufgabe zur UDP Prüfsumme und der Nummerierung von TCP Paketen mit Sequenz- und ACK-Nr. entwerfen und lösen. Zur Ideenfindung beachten Sie auch den Artikel zum Thema Internet Peering, der Stoff aus Vorlesung Nr.2 wiederholt und vertieft.

Lösung

1. Welche Kommunikationsformen gibt es und wie funktionieren sie?

- *Client-Server-Modell*: ein Host (Client) ruft Informationen von einem anderen Host (Server) ab
- *Peer-to-Peer-Modell*: gleichgestellte Hosts (Peers) tauschen gegenseitig Informationen aus

2. Was ist *MPEG-DASH* und wie funktioniert es?

- *Server*: Unterteilt das Video in Chunks, wobei jeder Chunk mit unterschiedlichen Bitraten abgespeichert wird. Eine sogenannte *Manifest-Datei* enthält die URLs für die Chunks.

- *Client*: Misst während des Abspielens die Bandbreite zwischen Client und Server. In der Manifest-Datei wird geguckt welcher Chunk gewählt werden sollte, wobei die maximal mögliche Bitrate gemäß der aktuellen Bandbreite der Verbindung ausgewählt wird.

3. Wie sieht der Protokollstapel des Internets aus?

- (von oben nach unten): Anwendung, Transport, Netzwerk, Sicherung, Bitübertragung

4. Addieren Sie diese zwei 16-Bit-Integer-Werte (1011001110110110,1001101110101100) und erstellen Sie die Prüfsumme.

Tabelle 1: Prüfsummenbeispiel

Zahl 1	1011001110110110
Zahl 2	1001101110101100
Übertrag	10100111101100010
Summe	0100111101100011
Prüfsumme	1011000010011100

5. Warum ist TCP fair?

Wenn zwei Verbindungen im Wettbewerb sind führt *Additive Increase* zu einer Steigerung von 1, wenn der Durchsatz wächst während *Multiplicative Decrease* den Durchsatz proportional reduziert.

1.4 Teilaufgabe 3

Frage- bzw. Aufgabenstellung

Entwickeln Sie eine kurze Programmieraufgabe (Frage und Lösung dazu). Diese Aufgabe soll einen Teilbereich Ihrer Wahl aus dem bisher behandelten Vorlesungsstoff abdecken.

Lösung

Aufgabe:

Erstellen Sie in Java einen Clienten der mithilfe der OpenWeatherMap die aktuelle Temperatur und Wetterlage (wolkig, sonnig etc) einer Stadt ausgeben kann. Übergeben Sie

hierfür die Postleitzahl und den Ländercode (jedoch soll auch er Stadtname ausgegeben werden). Wie die Daten entgegen genommen werden bleibt ihnen überlassen (XML oder JSON).

Lösung:

Für die Lösung dieser Aufgabe wurde eine Klasse mit dem Namen *CurrentTemp* 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

Die zweite Methode namens *ausgabeTemp* übernimmt den wichtigsten Teil der Aufgabe, da sie dafür sorgt, dass wir am Ende die Ausgabe des Wetters 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.

Zu Beginn wird der *Scanner* benutzt, damit der Benutzer die Postleitzahl, so wie den Ländercode eingeben kann. Anschließend wird ein neues JSONObject erstellt, dem man dann mithilfe der *readUrl*-Methode die API der OpenWeatherMap zusammen mit der zuvor eingegebenen Postleitzahl (plus zugehörigem Ländercode) 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 Daten der API kommt. Dazu eignet sich die Seite `https://jsonformatter.curiousconcept.com/` (aus dem letzten Praktikum). 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 zu finden. Die Stadt befindet sich ganz unten in einem *name* String und die Wetterlage ist in dem Array *weather* zu finden.


```
{ ⊟
   "coord":{ ⊞ },
   "weather":[ ⊟
      { ⊟
         "id":803,
         "main":"Clouds",
         "description":"broken clouds",
         "icon":"04n"
      }
   ],
   "base":"stations",
   "main":{ ⊟
      "temp":2.65,
      "pressure":1021,
      "humidity":86,
      "temp_min":2,
      "temp_max":3
   },
   "visibility":10000,
   "wind":{ ⊞ },
   "clouds":{ ⊞ },
   "dt":1513545600,
   "sys":{ ⊞ },
   "id":0,
   "name":"Minden",
   "cod":200
}
```


Abbildung 1: Formatierte JSON-Datei der API

In Zeile 9 bis 15 greifen wir dann auf die verschiedenen Objekte der JSON-Datei zu, die wir für diese Aufgabe brauchen. Dies wären dann die jetzige Temperatur über das Objekt *main* (Zeile 13,14), der Stadtname (Zeile 15) und die Wetterlage über das Array *weather* und den String *main* (Zeile 10 bis 12). Anschließend geben wir alle notwendigen Daten aus.

```
1 private static void ausgabeTemp() throws Exception {
2         Scanner scan = new Scanner(System.in);
3         System.out.println("Geben Sie bitte die Postleitzahl ein: ");
4         String zipCode = scan.nextLine();
5         System.out.println("Geben Sie bitte den Ã¤Lndercode ein: ");
6         String countryCode = scan.nextLine();
7         scan.close();
8         try {
9                 JSONObject json = new
                      JSONObject(readUrl("http://api.openweathermap.org/data/
```

```
                    2.5/weather?zip="+zipCode+",+countryCode+"&units=metric
&APPID=722920868a0a0266c859a174da690bc1"));
                JSONArray weatherArray = json.getJSONArray("weather");
                JSONObject weatherObject = weatherArray.getJSONObject(0);
                String weatherCurrent = weatherObject.getString("main");
                JSONObject main = json.getJSONObject("main");
                int temp = main.getInt("temp");
                String cityName = json.getString("name");
                System.out.println(zipCode + ", "+ cityName + ": " + temp +
                    "Â°C, " + weatherCurrent);
        } catch (JSONException e) {
                System.out.println("Stadt nicht gefunden");
                e.printStackTrace();
        }
}
```

Listing 2: ausgabeTemp

In der main steht schließlich nur noch die statische Methode *ausgabeTemp*.

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

Listing 3: main

Die Ausgabe könnten dann so aussehen:

```
Geben Sie bitte die Postleitzahl ein:
32427
Geben Sie bitte den Ländercode ein:
de
32427, Minden: 2°C, Clouds
```

Abbildung 2: aktuelles Wetter, Minden

Ergebnis

Wir können mithilfe der OpenWeatherMap das aktuelle Wetter einer Stadt anzeigen lassen.

2 Quellen

Literatur

[1] Agnieszka Czernik, Cookies – Funktion und Aufbau einfach erklärt `www.datenschutzbeauftragter-info.de/cookies-funktion-und-aufbau-einfach-erklaert/`, 17.12.2017