Social Bots – selber programmieren

Information

- Schaue zur Information das Video zu Social Bots von heuteplus an: https://t1p.de/bmiux

Socialbotnet

- Gehe auf die Seite https://socialbotnet.de und registriere dich unter einem Pseudonym. Erkunde die Möglichkeiten, poste eigene Beiträge, auch auf der Pinnwand von anderen, gib Likes.

API-Schnittstelle

- Die Seite hat eine sog. API-Schnittstelle. Damit können Inhalte der Seite in strukturierter Form abgerufen werden.

- Vergleiche Aufruf https://socialbotnet.de/api/posts?sortby=trending mit der Benutzersicht der Startseite.

- Unter https://socialbotnet.de/docs/index.html findet man sämtliche Möglichkeiten der API-Schnittstelle. Man kann sich z. B. die ersten zehn registrierten User anzeigen lassen:
  - URL: api/users
  - sortby: id
  - order: asc
  - limit: 10

  Als URL im Browser lautet diese Anfrage dann
  https://socialbotnet.de/api/users?sortby=id&order=asc&limit=10
  Probiere dies auf der Dokumentationsseite und mit der Browser-URL aus.

- Stelle folgende GET-Abfragen auf der Dokumentationsseite und mit der entsprechenden URL im Browser. Notiere die URL.
  - Zeige die letzten 30 registrierten User an.

    URL: _____
  - Zeige die letzten 10 Beiträge (Posts) an.

    URL: _____
  - Zeige den Beitrag mit den meisten Likes an.

    URL: _____
  - Zeige die Beiträge auf deiner Pinnwand sortiert nach den meisten Likes an.

    URL: _____

- Außerdem können über die API-Schnittstelle auch sog. POST-Abfragen gestellt werden, z. B. können so Beiträge geschrieben werden. Dies funktioniert aber nicht über eine Browser-URL. Erledige folgende Aufgaben über die Dokumentationsseite und kontrolliere in der Benutzersicht, ob es funktioniert hat.
  - Schreibe einen Beitrag.
  - Schreibe einen Beitrag auf der Pinnwand eines anderen Users.
  - Lasse dir den letzten Beitrag anzeigen, lese die ID ab und gib diesem Beitrag einen Like.

Programmieren der API-Schnittstelle mit der Online-IDE

- Die Klasse SocialBot ermöglicht, dass man sich mit seinen Nutzerdaten beim Socialbotnet anmeldet und stellt die Methoden abfragen (GET) und senden (POST) zur Verfügung. Beispiele zur Verwendung der Klasse SocialBot findet man in der START-Datei.

```
SocialBot socialbot = new SocialBot("https://socialbotnet.de", "meinBot", "meinPasswort");
socialbot.abfragen("/api/users", "sortby=id&order=asc&limit=10");
socialbot.senden("/api/post", "message=Hallo Welt!");
```

- Ändere den Aufruf des Konstruktors mit deinen Nutzerdaten ab und setze die GET- und POST-Abfragen, die du auf der Dokumentationsseite getestet hast, in der START-Datei mit Hilfe der Methoden abfragen und senden um.

- Nun soll die Klasse MeinBot, die von SocialBot erbt, abgewandelt und erweitert werden. Im Konstruktor von MeinBot soll der Konstruktor mit den eigenen Nutzerdaten augerufen werden.

```
public class MeinBot extends SocialBot
{
   public MeinBot()
   {
      super("https://socialbotnet.de", "meinBot", "meinPasswort");
   }
  …
}
```

- Als Beispiel ist die Methode beitragSchreiben(beitrag) zu sehen.

```
   public void beitragSchreiben(String beitrag)
   {
      senden("/api/post", "message=" + beitrag);
   }
```

Schreibe entsprechend die Methode beitragAufUserpinnwandSchreiben(username, beitrag).

- In der Methode userlisteAusgeben wird gezeigt, wie die Ausgabe der API, die im sog. JSON-Format vorliegt, verarbeitet werden kann.

```
   public void userlisteAusgeben()
   {
      JsonElement answer = abfragen("/api/users", "sortby=id&order=asc&limit=10");
      for (JsonElement object : answer.getArrayValues())
      {
         println(object.getAsString("username"));
      }
   }
```

Nebenstehendes Beispiel zeigt die Ausgabe der Userliste im JSON-Format an. Die eckigen Klammern zeigen, dass es sich um ein Feld handelt, in dem in den geschweiften Klammern die Objekte stehen. In der Methode userlisteAusgeben wird dieses JsonElement in der Variable answer abgespeichert. Da es sich um ein Feld handelt kann dieses mit einer for-Schleife durchlaufen werden. Auf der Konsole werden die Werte der Variablen username aller Objekte des Felds ausgegeben.
Schreibe entsprechend eine Methode letzteUserAusgeben(anzahl), die die letzten registrierten User ausgibt, wobei die Anzahl der User angegeben werden kann.

```
[
  {
    "id": 1,
    "username": "root",
    "hobbies": "Netzwerken",
    "about": "Ich bin root!"
  },
  {
    "id": 2,
    "username": "Welcome",
    "hobbies": "",
    "about": ""
  },
  …
]
```

- In der Methode beitraegeAusgeben wird gezeigt, wie die JSON-Ausgabe der Beiträge bearbeitet wird, dabei gibt es für jeden Beitrag ein JsonElement, in dem wiederum u. a. das JsonElement user enthalten ist.

```
public void beitraegeAusgeben()
{
   JsonElement answer = abfragen("/api/posts", "sortby=id&order=asc&limit=10");
   for (JsonElement object : answer.getArrayValues())
   {
      JsonElement userdata = object.getAttributeValue("user");
      println(object.getAsString("message") + " von " + userdata.getAsString("username"));
   }
}
```

  Schreibe eine entsprechende Methode letzteBeitraegeAusgeben(anzahl).
  Schreibe eine weitere Methode beitraegeUserAusgeben(username), die nur die Beiträge eines bestimmten Users ausgibt.

- Weitere mögliche Aufgaben
  - Schreiben zufälliger Beiträge im Stil von Buzz-o-Mat. Dazu kann die Klasse Satzgenerator verwendet werden. Mit Hilfe einer KI z. B. https://chat.lmsys.org (Direct Chat) kann man sich andere Textbausteine generieren lassen.
  - Schreiben von mehreren zufälligen Beiträgen im zeitlichen Abstand von einigen Sekunden.
  - Liken aller Beiträge eines Users.
  - Beiträge nach Schlüsselwörtern durchsuchen und dann liken.
  - Nutzerprofile nach Schlüsselwörter durchsuchen und dann einen vorgefertigten Text auf die Pinnwand des Users schreiben.
  - Anbinden einer externen API-Schnittstelle, um z. B. einen Beitrag über das Wetter mit Hilfe von https://open-meteo.com/en/docs zu schreiben.
  - Eigene Ideen!