Aufgabe RegEx 3

Zielsetzung

Diese Aufgabe schafft eine Übungsgrundlage für den Umgang mit regulären Ausdrücken. Nach einer grundlegenden Einführung in die Thematik wird die Konstruktion von regulären Ausdrücken für verschiedene Zwecke sowie die Anwendung auf Zeichenketten aus unterschiedlichen Quellen geübt. In dieser Aufgabe wird besonders die Verarbeitung mehrzeiliger Daten sowie das Parsen eines standardisierten Datenaustauschformats geübt.

Vorbemerkung

Anbei ist eine Kopie einer Log-Datei eines Linux-Webservers zu finden.

Diese Protokolldaten stammen von einem echten Server aus dem „echten Internet“ und zeigen tatsächlich so abgelaufenen Datenverkehr. In dieser Aufgabe sollen unterschiedliche Analysen durchgeführt werden, um Erkenntnisse über den Datenverkehr zu gewinnen.

Alle Analyzer-Klassen

Alle Analyzer implementieren das `Analyzer`-Interface. Dieses schreibt eine öffentliche Methode `String analyze( String logContent )`vor. Die Rückgabe enthält den Analysebericht in „menschenlesbarer” Form. Unterschiedliche Analyzer untersuchen unterschiedliche Dinge. Mittels `String.format()` lässt sich die Ausgabe gut gestalten. Diese Aufgaben lassen sich auch sehr elegant mit Java-8-Streams lösen.

Achten Sie darauf, die verwendeten Patterns mit dem Flag `Pattern.MULTILINE` (= im Modus für Strings mit Zeilenumbrüchen) zu erzeugen!

HTTP-Protokollversion-Analyzer

Dieser Analyzer findet die unterschiedlichen verwendeten Versionen des HTTP-Protokolls, und gibt einen String zurück, in dem die vorkommenden Versionen aufgeführt sind. Denken Sie daran, dass es seit Mai 2015 mit RFC 7540 auch einen HTTP/2.0-Standard und seit November 2020 zusätzlich HTTP/3 als Standard-Draft gibt. Analysieren Sie insbesondere die prozentualen Anteile der einzelnen Protokollversionen.

HTTP-Command-Analyzer

Dieser Analyzer findet und zählt die unterschiedlichen protokollierten HTTP-Befehle (mindestens `GET`, `POST`, `HEAD` und `PUT`).
Finden Sie heraus, was mit dem `POST`-Command versucht wurde (dies ist keine Java-Aufgabe!).

UserAgent-String-Analyzer

Dieser Analyzer findet die einzigartigen UserAgent-Strings. Die Rückgabe besteht aus der Anzahl der gefundenen unterschiedlichen UserAgent-Strings, der Anzahl der UserAgent-Strings, die behaupten, von einem Bot (etwa Googlebot, bingbot) zu stammen, und aus allen einzigartigen UserAgent-Strings.

Bestimmen Sie den prozentualen Anteil mobiler UserAgents und den prozentualen Anteil an Bots.

IP-Analyzer

Dieser Analyzer extrahiert die IPv4-Adressen am Zeilenbeginn (also die Adressen der Clients). Er findet die einzigartigen Adressen, und gibt deren Anzahl sowie sämtliche einzigartigen Adressen mit der Anzahl der Zugriffe von dieser IP zurück.

In der nächsten Aufgabe wird dieser Analyzer um eine Geolokation über einen Webdienst erweitert. Dazu benötigt der Analyzer eine zusätzliche öffentliche Methode, die aus einem übergebenen Log-String sämtliche einzigartigen IP-Adressen extrahiert und eine geeignete Datenstruktur mit diesen Daten zurückgibt.

Starter-Klasse

Die Starter-Klasse enthält die `main()`-Methode. Hier werden die Logs *einmal* eingelesen, zu einem einzigen String mit Zeilenumbrüchen zusammengefügt und dann durch die verschiedenen Analyzer analysiert. Die Ausgabe könnte so aussehen:

```
HTTP-Protokoll-Versionen
HTTP/1.0 : 7431 (64,12%)
HTTP/1.1 : 3219 (27,77%)
HTTP/2.0 : 940 (8,11%)
HTTP/3 : 0 (0%)

HTTP-Commands
HEAD : 26 (0,22%)
POST : 1 (0,01%)
GET : 11563 (99,77%)

User-Agent-Analyse
Einzigartige User-Agent-Strings : 125
User-Agent-Strings, die Bots identifizieren : 12 (0,10
User-Agent-Strings, die mobile Geräte : 24 (0,19

Mozilla/5.0 (Linux; Android 6.0.1; Nexus 5X Build/MMB29P) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/41.0.2272.96 Mobile Safari/537.36 (compatible;
Googlebot/2.1; +http://www.google.com/bot.html)
... gekürzt ...

Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/57.0.2987.133 Safari/537.36

IP-Analyzer
Einzigartige IPs : 325

80.155.138.194 (6170)
81.173.228.188 (638)
.. gekürzt ...

91.90.192.195 (1)
```

Erweiterung: Geolokation der IP-Adressen

In dieser Erweiterung soll ein geographische Zuordnung der IP-Adressen aus dem Webserver-Log erfolgen. Hierzu wird die beiliegende Klasse `IPLocator` verwendet, die die Kommunikation mit dem Webdienst durchführt.

Bitte beachten Sie, dass dieser Webdienst nur 1_000 Abfragen pro Tag und Client zulässt. Seien Sie also sparsam!

Weitere Informationen:

- db-ip.com Free IP geolocation API
- db-ip.com Free IP geolocation API Documentation

Eine dokumentierte Klasse zum Abfragen der REST-API des Webdienstes db-ip.com liegt der Aufgabenstellung bei. Diese Klasse enthält zusätzlich Konstanten, die für die Analyse der Antwort nützlich sein könnten. REST-APIs sind für maschinelle Abfrage von Daten über das HTTP-Protokoll gedacht.

Die HTTP-Anfragen haben die Form `Basis-URL/IP-Adresse`:

```
http://api.db-ip.com/v2/free/80.155.138.194
```

Als Antwort erhält der Client einen Text im JSON-Format:

```
 {“ipAddress”: “80.155.138.194”,“continentCode”: “EU”,“continentName”:
“Europe”,“countryCode”: “DE”,“countryName”: “Germany”,“stateProv”: “Hesse”,“city”:
“Frankfurt am Main”}
```

Die Klasse `IPLocator` kommuniziert mit dem Webdienst (auch durch den Proxy, hierzu müssen möglicherweise die auskommentierten Angaben im Konstruktor einkommentiert ung ggf. befüllt werden (Benutzername/Passwort)).

Der Webdienst unterstützt Batch-Abfragen für mehrere IPs in einer Abfrage:

```
http://api.db-ip.com/v2/free/80.155.138.194,188.165.209.216
```

`IPLocator` hat dafür eine entsprechende Methode.
Die Antwort einer Batch-Abfrage sieht etwas anders aus, da hier die Einzelantworten zusammengefasst werden:

```
{“80.155.138.194”: {“continentCode”: “EU”,“continentName”: “Europe”,“countryCode”:
“DE”,“countryName”: “Germany”,“stateProv”: “Hesse”,“city”: “Frankfurt
am Main”},“188.165.209.216”: {“continentCode”: “EU”,“continentName”:
“Europe”,“countryCode”: “FR”,“countryName”: “France”,“stateProv”:
“Hauts-de-France”,“city”: “Roubaix”}}
```

Wählen Sie nach einem durch Sie festzulegenden Kriterium 10 Adressen aus dem Log aus (etwa die 10 häufigesten oder die 10 verdächtigsten). Stellen Sie eine entsprechende Liste zusammen, und lassen Sie eine Abfrage laufen.

Parsen Sie das Ergebnis so, dass eine `Map<String, Map<String, String»` entsteht. In der „äußeren” Map dient die IP als Schlüssel, die „innere” Map enthält die Informationen, die der Webdienst geliefert hat.

Zusatzaufgabe

Diese Zusatzaufgabe deckt keinen momentan prüfungsrelevanten Stoff ab, sondern dient der Erweiterung des Horizonts. Lösen Sie sie daher erst, wenn alle originären Aufgaben gelöst und verstanden sind.

Im richtigen Leben gibt es für die Umwandlung von und nach JSON (und XML, und diversen anderen Formaten) natürlich Programmbibliotheken. Lassen Sie das Einlesen der JSON-Daten von GSON erledigen.