SAPIENZA
UNIVERSITÀ DI ROMA

Giovanni Stilo, Ph.D.
stilo@di.uniroma1.it

140 Chars to Fly

Twitter API 1.1 and Twitter4J introduction

Requirements

- Twitter (Mandatory)
  - Account
  - General operation
- REST principles
  - Give every “thing” an ID
  - Link things together **<product ref='http://example.com/products/4554' />**
  - Use standard methods **GET, POST, PUT, DELETE, HEAD**
  - Resources with multiple representations **XML, JSON**
  - Communicate statelessly
- Maven 2 (Optional)

Twitter Api 1.1

- **Timelines**: are collections of Tweets, ordered with the most recent first.
- **Tweets:** are the atomic building blocks of Twitter, 140-character status updates with additional associated metadata. People tweet for a variety of reasons about a multitude of topics.
- **Friends & Followers:** Users follow their interests on Twitter through both one-way and mutual following relationships.
- **Users**: are at the center of everything Twitter: they follow, they favorite, and tweet & retweet.
- **Oauth:** Twitter uses OAuth for authentication. See first *Authentication & Authorization*

Complete information available at:

https://developer.twitter.com/en/docs

REST API Rate Limiting in v1.1

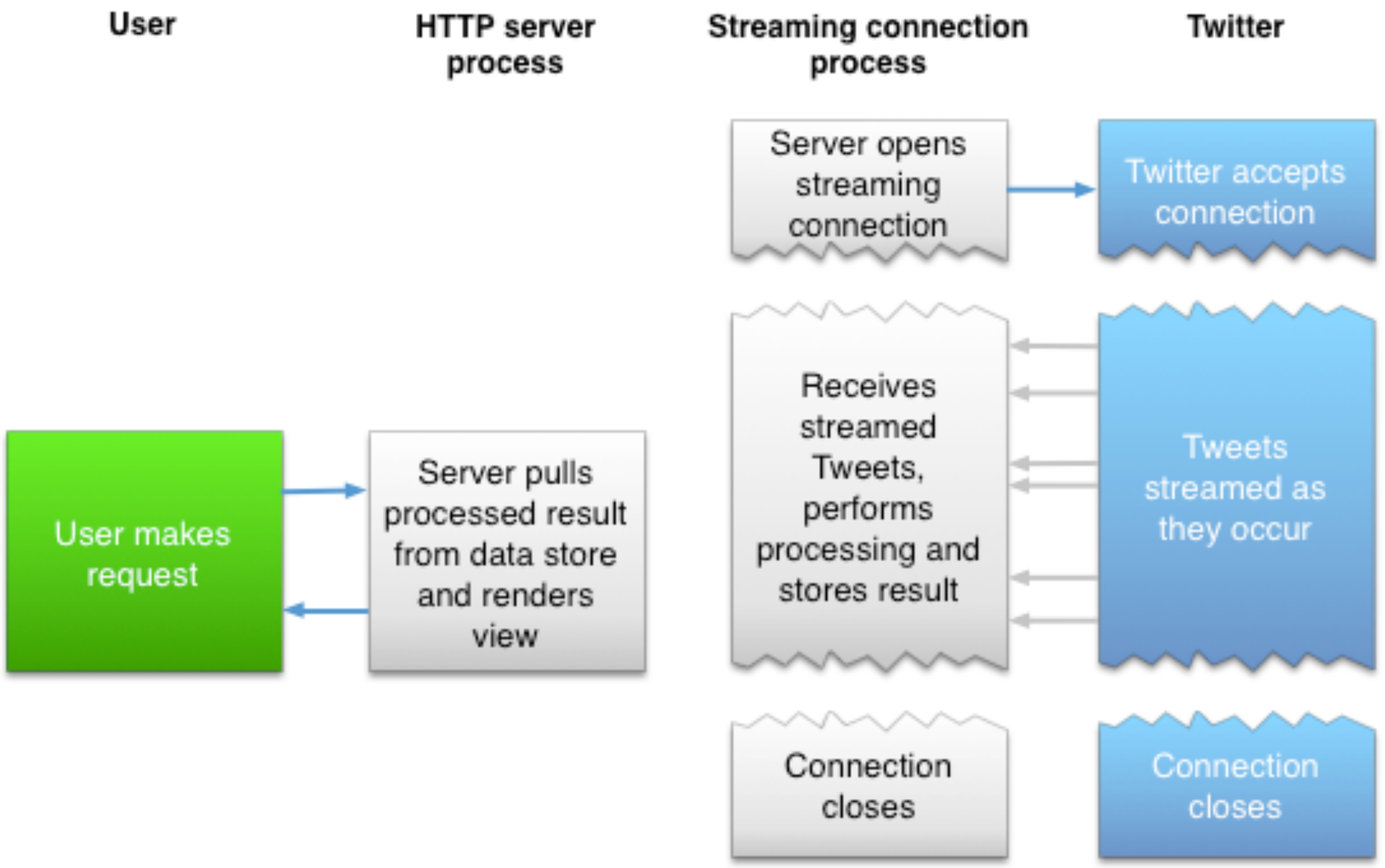
- **Per User**
  - Rate limiting in version 1.1 of the API is primarily considered on a per-user basis — or more accurately described, per access token in your control.
  - If a method allows for 15 requests per rate limit window, then it allows you to make 15 requests per window per leveraged access token.
- **Per Application**
  - When using application-only authentication, rate limits are determined globally for the entire application.
  - If a method allows for 15 requests per rate limit window, then it allows you to make 15 requests per window
- **15 Minute Windows**

https://developer.twitter.com/en/docs/basics/rate-limits

The Streaming APIs

- The set of streaming APIs give low latency **access** to Twitter's **global stream** of Tweet data.
  - Public streams Streams of the public data flowing through Twitter. Suitable for **following** specific **topics**, and data mining.
  - User streams Single-user streams, containing roughly all of the data corresponding with a **single user's** view of Twitter.
- Connecting to the streaming API **requires** keeping a **persistent HTTP connection** open. In many cases this involves thinking about your application **differently** than if you were interacting with **the REST API**.

Streaming Flow



User
HTTP server process
Streaming connection process
Twitter
Server opens streaming connection
Twitter accepts connection
Receives streamed Tweets, performs processing and stores result
Tweets streamed as they occur
User makes request
Server pulls processed result from data store and renders view
Connection closes
Connection closes

Public streams

- The following streams offer **samples** of the **public data** flowing through Twitter. Once applications establish a connection to a streaming endpoint, they are delivered a feed of Tweets, without needing to worry about polling or REST API rate limits.
- **Endpoints**
  - POST statuses/filter
  - GET statuses/sample
- **Connections**
  - **Each** account may create **only one standing** connection to the public endpoints
  - Clients which make **excessive connection** attempts (both successful and unsuccessful) run the risk of having their IP automatically **banned**.

POST statuses/filter

- **Resource URL**
  - https://stream.twitter.com/1.1/statuses/filter.json
- **Parameters**
  - **follow** see note A comma separated list of user IDs, indicating the users to return statuses for in the stream.
  - **track** see note Keywords to track. Phrases of keywords are specified by a comma-separated list.
  - **locations** see note Specifies a set of bounding boxes to track.
- At least one predicate parameter (follow, locations, or track) must be specified.

- **Example Request**
  - https://stream.twitter.com/1.1/statuses/filter.json&track=twitter
    (does not work anymore as is)

Twitter4J

- Twitter4J is an **unofficial** Java **library** for the **Twitter API**.
  - 100% Pure Java - works on any Java Platform version 5 or later
  - Android platform and Google App Engine ready
  - Zero dependency : No additional jars required
  - Built-in OAuth support
  - Out-of-the-box gzip support
  - 100% Twitter API 1.1 compatible

```
<dependency>
        <groupId>org.twitter4j</groupId>
        <artifactId>twitter4j-core</artifactId>
        <version>[2.2,)</version>
 </dependency>

 <dependency>
        <groupId>org.twitter4j</groupId>
        <artifactId>twitter4j-stream</artifactId>
        <version>[2.2,)</version>
</dependency>
```

Authentication

```
ConfigurationBuilder cfg= new ConfigurationBuilder();
cfg.setOAuthAccessToken(access_token)
cfg.setOAuthAccessTokenSecret(access_token_secret);
cfg.setOAuthConsumerKey(consumer_key);
cfg.setOAuthConsumerSecret(consumer_secret);
```

Example Stream

```
public static void main(String[] args) throws TwitterException, IOException{
    StatusListener listener = new StatusListener(){
            public void onStatus(Status status) {
                System.out.println(status.getUser().getName() + " : "
                + status.getText());
            }
            public void onDeletionNotice(StatusDeletionNotice statusDeletionNotice) {}
            public void onTrackLimitationNotice(int numberOfLimitedStatuses) {}
            public void onException(Exception ex) { ex.printStackTrace(); }
            public void onScrubGeo(long arg0, long arg1) {}
            public void onStallWarning(StallWarning warning) {}
    };
    ConfigurationBuilder cfg= new ConfigurationBuilder(); //SET AUTH PARAMETER
    TwitterStream twitterStream = new TwitterStreamFactory(cfg.build()).getInstance();
    twitterStream.addListener(listener );
    // sample() method internally creates a thread which manipulates TwitterStream
    // and calls these adequate listener methods continuously.

    twitterStream.sample();
}
```

Example Filter

```
public static void main(String[] args) throws TwitterException, IOException{
    StatusListener listener = new StatusListener(){
            public void onStatus(Status status) {
               if (status.getLang().equals("it"))
                      System.out.println(status.getUser().getName() + "  :  "
                      + status.getText());
            }
            public void onDeletionNotice(StatusDeletionNotice statusDeletionNotice) {}
            public void onTrackLimitationNotice(int numberOfLimitedStatuses) {}
            public void onException(Exception ex) { ex.printStackTrace(); }
            public void onScrubGeo(long arg0, long arg1) {}
            public void onStallWarning(StallWarning warning) {}
    };
    ConfigurationBuilder cfg= new ConfigurationBuilder(); //SET AUTH PARAMETER
    TwitterStream twitterStream = new TwitterStreamFactory(cfg.build()).getInstance();
    twitterStream.addListener(listener );

     FilterQuery fq = new FilterQuery();
     fq.track(new String[]{"renzi"});
     twitterStream.filter(fq);
}
```

Exercise Stream

- Using Twitter4J, Maven, create classes that allow to use Twitter Streaming API.

- These classes allow to use different type of *streaming endpoint* and allow to chose different configuration parameters.

- These classes are abstract and well organized.

GET users/show

Returns a variety of **information** about the user specified by the required **user_id** or **screen_name** parameter. The author's most recent Tweet will be returned inline when possible.

GET users / lookup is used to retrieve a bulk collection of user objects.

You must be following a protected user to be able to see their most recent Tweet. If you don't follow a protected user, the user's Tweet will be removed. A Tweet will not always be returned in the current_status field.

GET friends/ids

Returns a **cursored collection** of user **IDs** for every user the specified user is following (otherwise known as their "friends").

Results are given in groups of 5,000 user IDs and multiple "pages" of results can be navigated through using the **next_cursor** value in subsequent requests. See Using cursors to navigate collections for more information.

*This method is especially powerful when used in conjunction with **GET users / lookup**, a method that allows you to convert user IDs into full user objects in bulk.*

GET users/lookup

Returns **fully-hydrated** user objects for up to **100** users per **request**, as specified by comma-separated values passed to the user_id and/or screen_name parameters.

This method is especially useful when used in conjunction with collections of user IDs returned from GET friends / ids and GET followers / ids.

Example REST

```
public static void main(String[] args) throws TwitterException, IOException{
    ConfigurationBuilder cfg= new ConfigurationBuilder();
    cfg.setOAuthAccessToken(access_token)
    cfg.setOAuthAccessTokenSecret(access_token_secret);
    cfg.setOAuthConsumerKey(consumer_key);
    cfg.setOAuthConsumerSecret(consumer_secret);

    Twitter twitter = new TwitterFactory(cf.build()).getInstance();
    long [] ids=twitter.getFriendsIDs(USERID,-1).getIDs();

    PRINT IDS…

}
```

Example REST

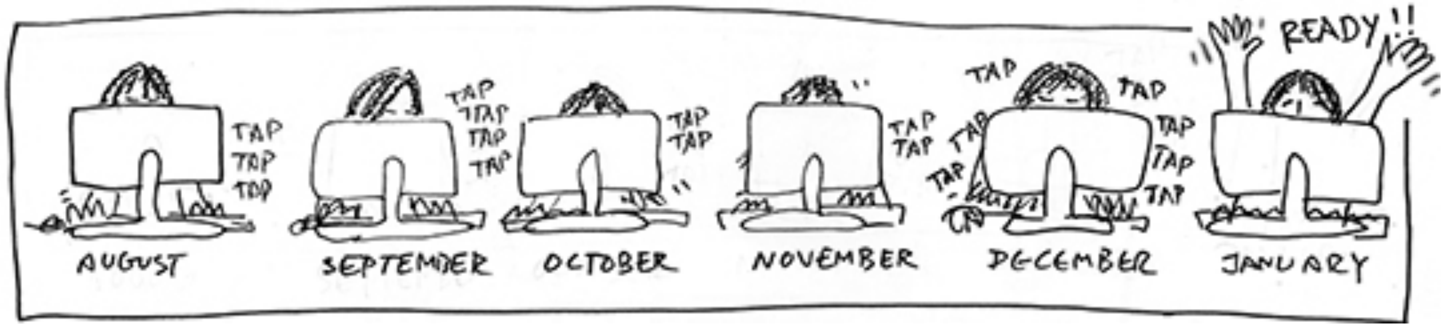
```
public static void main(String[] args) throws TwitterException, IOException{
    ConfigurationBuilder cfg= new ConfigurationBuilder();
    cfg.setOAuthAccessToken(access_token)
    cfg.setOAuthAccessTokenSecret(access_token_secret);
    cfg.setOAuthConsumerKey(consumer_key);
    cfg.setOAuthConsumerSecret(consumer_secret);

    Twitter twitter = new TwitterFactory(cf.build()).getInstance();
    long [] ids=twitter.getFriendsIDs(USERID,-1).getIDs();
    for(long id: ids){
        User u = twitter. showUser(id);
        … PRINT USER SCREEN NAME
    }
}
```

Exercise REST

- Using Twitter4J, Maven, create classes that collect user-name of your Friends.
- These classes allow to use different type of *streaming endpoint* and allow to chose different configuration parameters.
- These classes are abstract and well organized.

Let's Try?!?!


TAP TAP TAP
AUGUST
TAP TAP TAP TAP
SEPTEMBER
TAP TAP
OCTOBER
TAP TAP
NOVEMBER
TAP TAP TAP TAP TAP TAP TAP TAP
DECEMBER
READY!!
JANUARY

Twitter4J add-on

- Enable JSON support:
  - Invoke setJSONStoreEnabled(true) on the ConfigurationBuilder object.

- To extract original JSON String from the Status object:
  - TwitterObjectFactory.getRawJSON(status)

- To build Status object from a JSON:
  - TwitterObjectFactory.createStatus(JSON_String);

Maven Tip!

```
<build>
    <pluginManagement>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.0</version>
          <configuration>
            <source>1.6</source>
            <target>1.6</target>
          </configuration>
        </plugin>
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>exec-maven-plugin</artifactId>
          <configuration>
            <executable>java</executable>
            <arguments>
              <argument>-Xmx128m</argument>
              <argument>-classpath</argument>
              <classpath/>
              <mainClass>YOURCLASS</mainClass>
              <argument>arg0</argument>
            </arguments>
          </configuration>
        </plugin>
      </plugins>
    </pluginManagement>
</build>
```