UNIVERSITÀ degli STUDI di CATANIA

DIPARTIMENTO DI MATEMATICA E INFORMATICA

Analysis of Social Media Contents

Alessandro Ortis, PhD
ortis@dmi.unict.it

IMAGE PROCESSING LABORATORY
http//iplab.dmi.unict.it

Info&contacts

Alessandro Ortis, *PhD*

Email: *ortis@dmi.unict.it*
Personal webpage: http://www.dmi.unict.it/ortis/

Università di Catania

Aims & Approach

- The aim is to introduce methods, but also best practices and practical tools in the field of *social media content analysis*

- Alternating techniques and code snippets (slides) plus complete code examples (Jupyter notebooks)

- All material will be provided

Outline

- Intro to photo-sharing platforms
- REST Web Services & Social Platform APIs
- OAUTH
- How to use APIs to extract info from social image databases
  - Flickr API
  - Facebook API
  - Instagram API
  - Twitter API

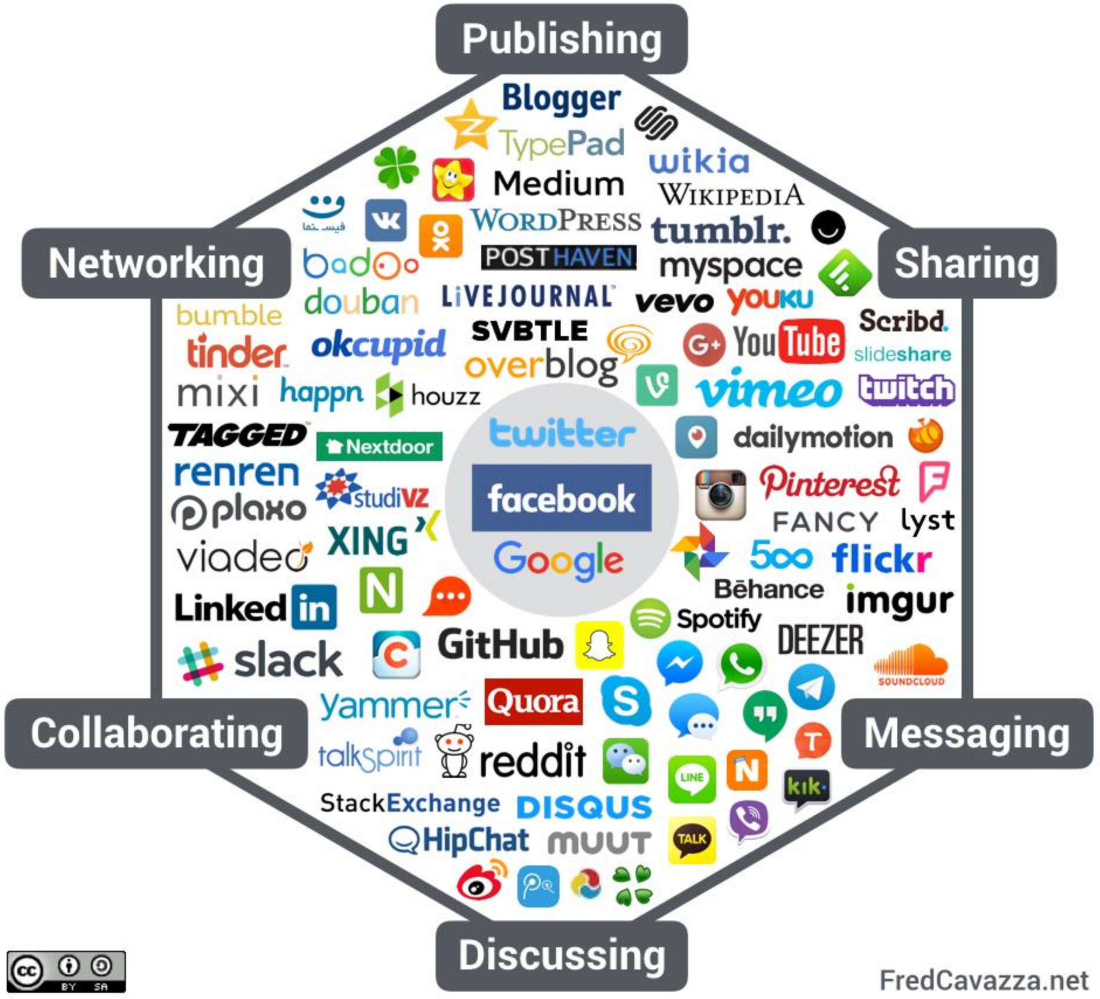
What is a photo-sharing service?

- Photo sharing is the publishing or transfer of user's digital photos online, thus enabling the user to share them with others.

- This function is provided through both websites and applications that facilitate to upload and display the images.

Social Media Platforms

- A lot of data!
- A lot of interaction!
- A lot of images!
- A lot of work for Data Analysts!



Digital image metadata on social media

- Metadata is one of the most powerful tools you have:
  - Camera's info
  - Geolocalized information
  - Rating (e.g., likes)
  - Comments
  - Caption
  - Keywords/tags
  - …

Web Services

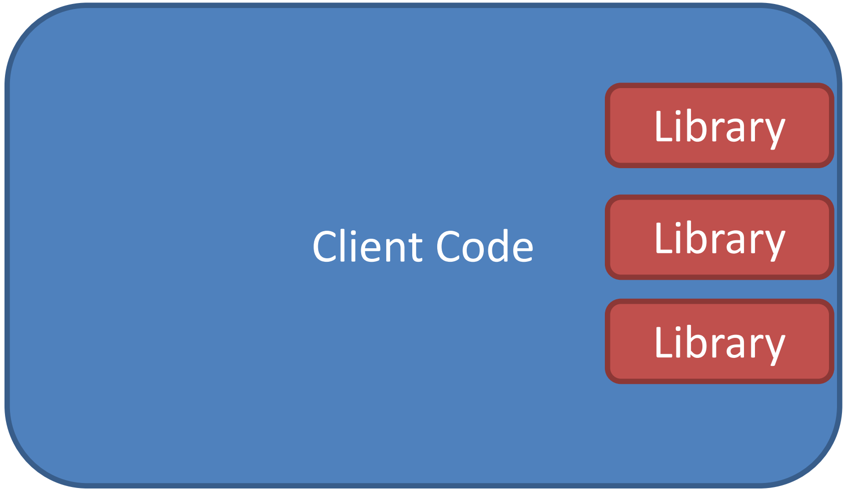
- Services exposed on the Internet for a programmatic access through APIs.

Web Services

- Services exposed on the Internet for a programmatic access through APIs.
- APIs: Application Programming Interface

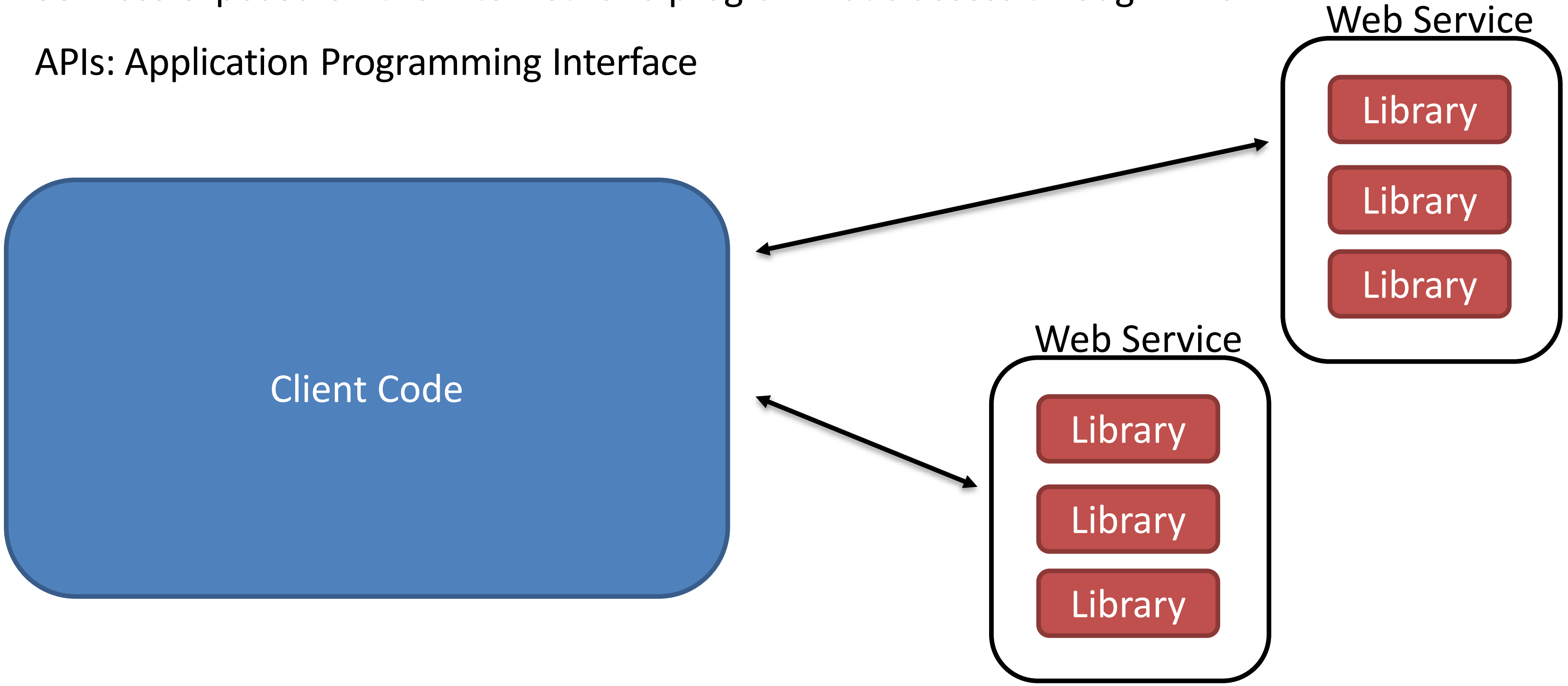
Web Services

- Services exposed on the Internet for a programmatic access through APIs.
- APIs: Application Programming Interface



Web Services

- Services exposed on the Internet for a programmatic access through APIs.
- APIs: Application Programming Interface

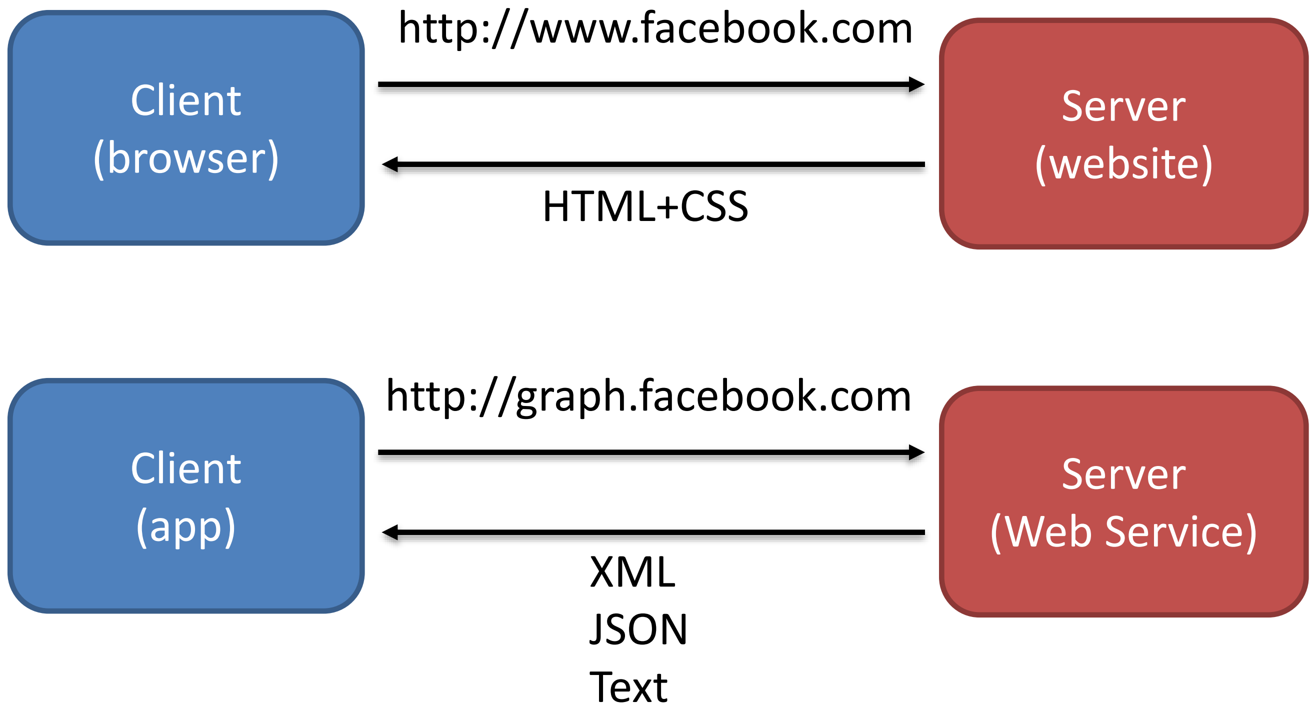


Web Services

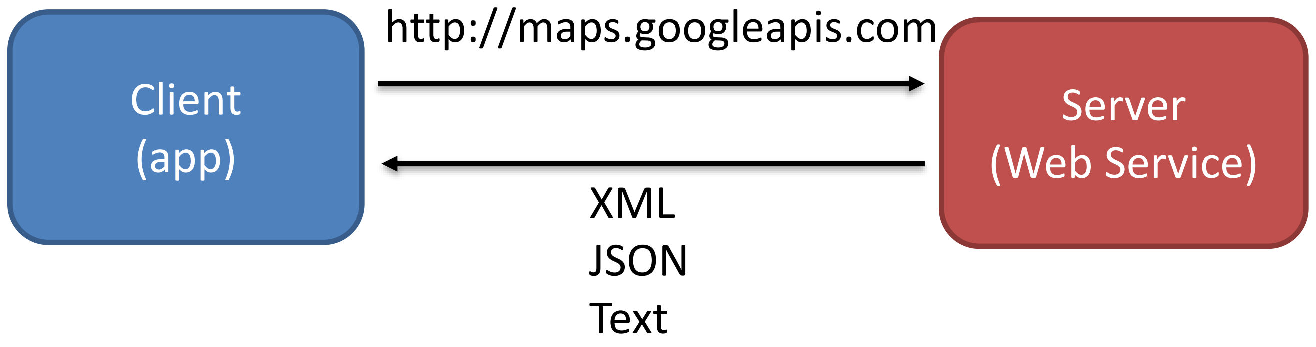
- Services exposed on the Internet for a programmatic access through APIs.
- APIs: Application Programming Interface
- Protocol (Messages format) → REST (Representational State Transfer)

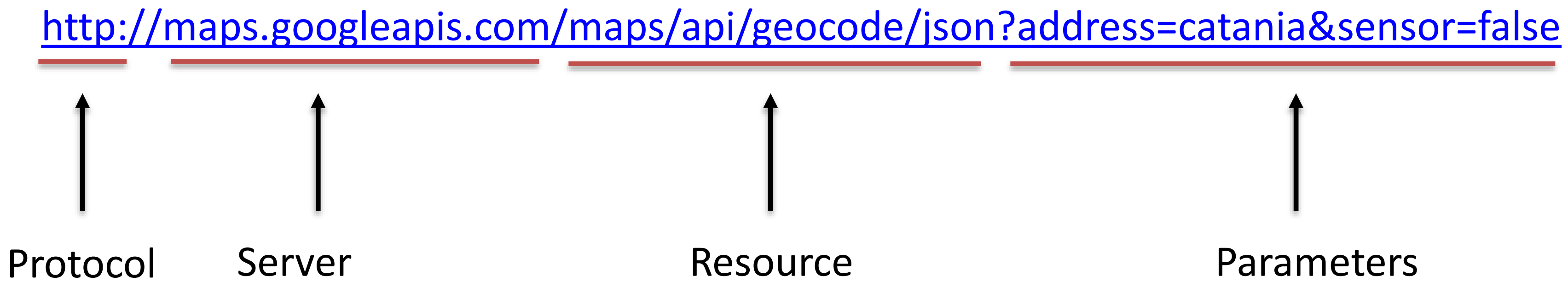
REST Web Services: *"REST-compliant Web services allow requesting systems to access and manipulate textual representations of web resources using a uniform and predefined set of stateless operations."*

REST Web Services



REST Web Services




http://maps.googleapis.com/maps/api/geocode/json?address=catania&sensor=false
Protocol
Server
Resource
Parameters

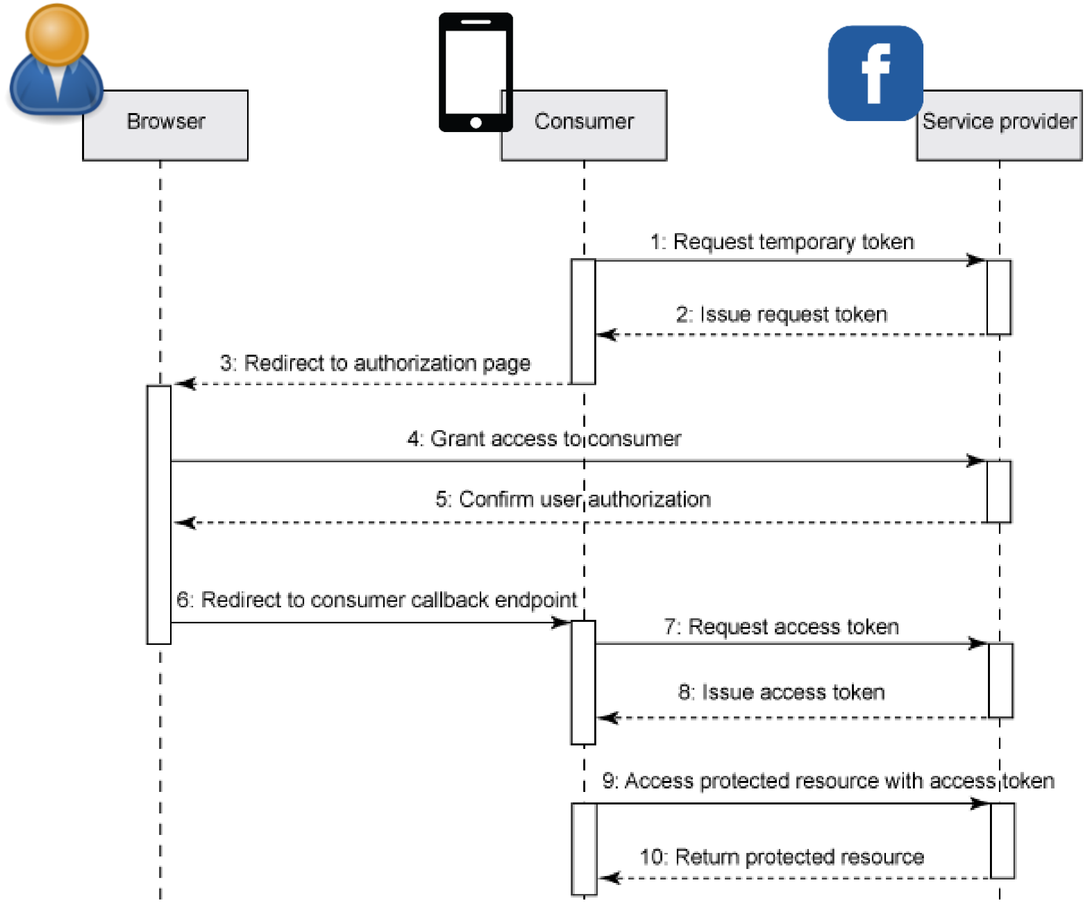
REST Web Services

```
{
   "error_message" : "Keyless access to Google Maps Platform is deprecated. Please
use an API key with all your API calls to avoid service interruption. For further
details please refer to http://g.co/dev/maps-no-account",
   "results" : [],
   "status" : "OVER_QUERY_LIMIT"
}
```

OAUTH

- OAuth (Open Authentication) is an open standard for authorization (access delegation)
  - Allows sharing user’s resources (photos, videos, contact lists) between different websites
  - The user credentials (username and password) are not shared
  - Websites share tokens instead of credentials
  - Each token grants access
    - to a specific website
    - for specific resources
    - for a defined duration

OAUTH


Browser
Consumer
Service provider
1: Request temporary token
2: Issue request token
3: Redirect to authorization page
4: Grant access to consumer
5: Confirm user authorization
6: Redirect to consumer callback endpoint
7: Request access token
8: Issue access token
9: Access protected resource with access token
10: Return protected resource

Request Token

Authorization Token

Access Token

flickr

- Flickr is one of the most widely known photo sharing social networks.

- It offers free image hosting and a great photo management system where it is possible to organize your photos in albums and generate comments on your photos from the community.

Flickr API

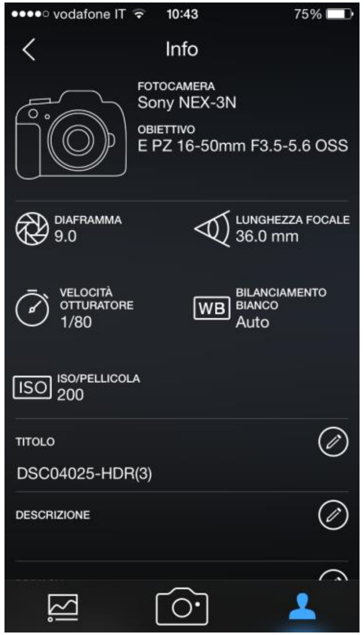
- Much of the success of Flickr is the open system that is available to developers with API’s being free for non-commercial use.

- The Flickr API provides the ability to view, manipulate and search photo tags, display photos from a specific user or group, retrieve tags to construct URLs to particular photos or photo group.

- Almost all the functionality that runs flickr.com is available through the API.

Exchangeable image file format - EXIF

- When you take a picture with a digital device, it automatically saves EXIF data with the photo:
  - Exposure time (Shutter speed)
  - f-number
  - ISO setting
  - …
  - Many devices also include GPS information



FOTOCAMERA
Sony NEX-3N

OBIETTIVO
E PZ 16-50mm F3.5-5.6 OSS

DIAFRAMMA
9.0

LUNGHEZZA FOCALE
36.0 mm

VELOCITÀ OTTURATORE
1/80

BILANCIAMENTO BIANCO
Auto

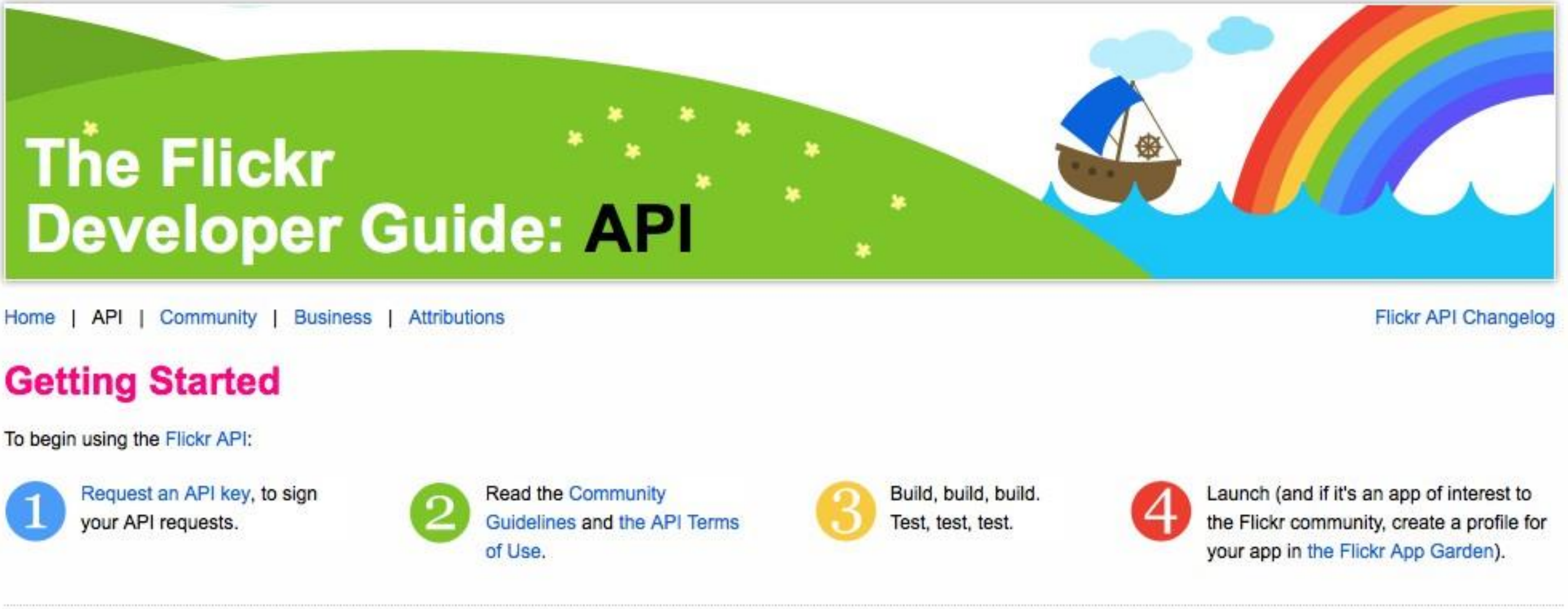
ISO/PELLICOLA
200

TITOLO
DSC04025-HDR(3)

DESCRIZIONE

- The PNG and GIF image formats do not support EXIF data

Flickr API



The Flickr Developer Guide: API

Home | API | Community | Business | Attributions

Flickr API Changelog

Getting Started

To begin using the Flickr API:

1. Request an API key, to sign your API requests.
2. Read the Community Guidelines and the API Terms of Use.
3. Build, build, build. Test, test, test.
4. Launch (and if it's an app of interest to the Flickr community, create a profile for your app in the Flickr App Garden).

https://www.flickr.com/services/developer/api/

Flickr API

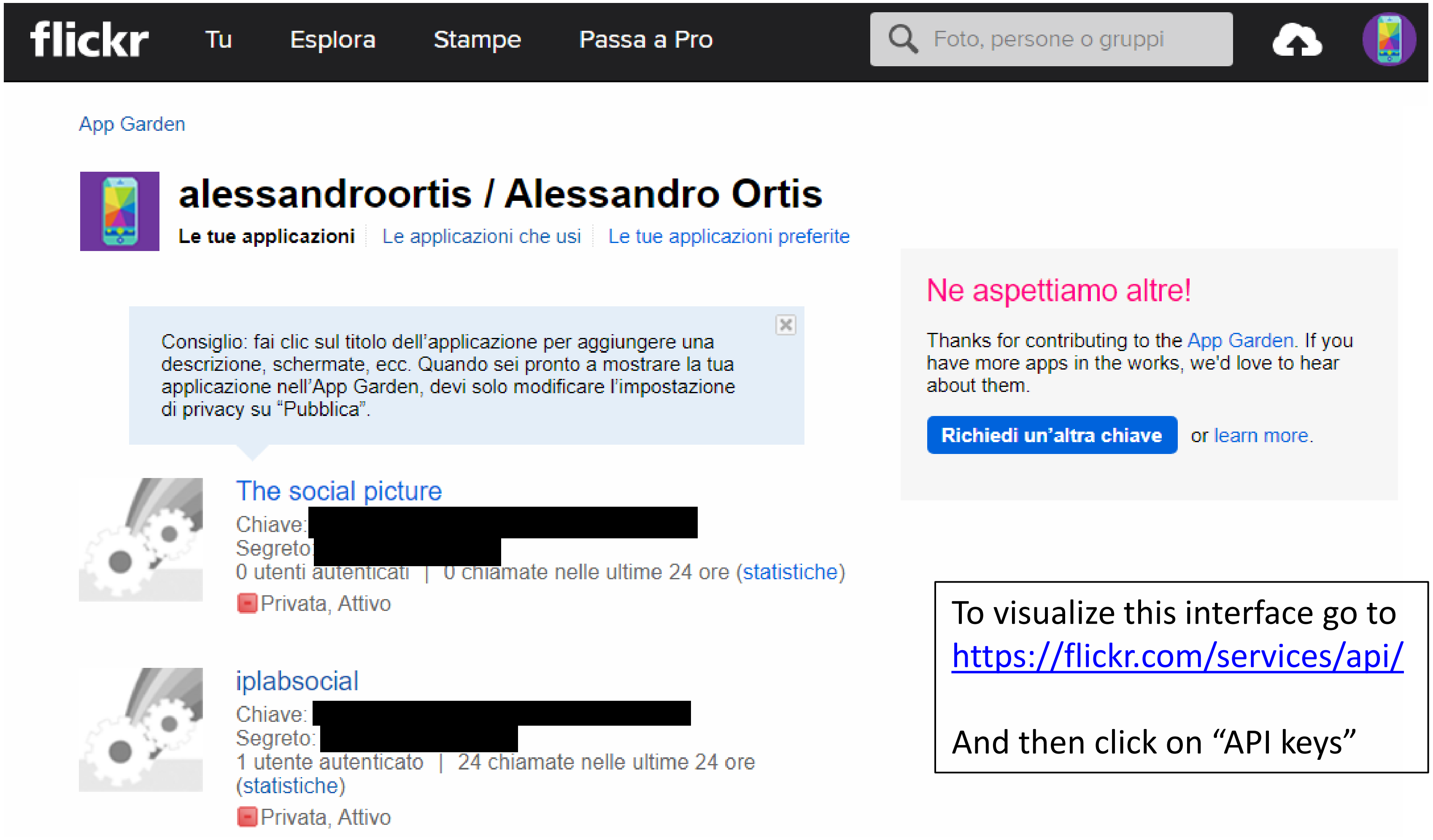
https://www.flickr.com/services/api/

- There are already client libraries available for most languages: PHP, Java, C, Perl, Python, …

Flickr API

How to get an authorization token:

https://www.flickr.com/services/api/auth.oauth.html



alessandroortis / Alessandro Ortis

Le tue applicazioni | Le applicazioni che usi | Le tue applicazioni preferite

Consiglio: fai clic sul titolo dell'applicazione per aggiungere una descrizione, schermate, ecc. Quando sei pronto a mostrare la tua applicazione nell'App Garden, devi solo modificare l'impostazione di privacy su "Pubblica".

The social picture
Chiave: [redacted]
Segreto: [redacted]
0 utenti autenticati | 0 chiamate nelle ultime 24 ore (statistiche)
Privata, Attivo

iplabsocial
Chiave: [redacted]
Segreto: [redacted]
1 utente autenticato | 24 chiamate nelle ultime 24 ore (statistiche)
Privata, Attivo

Ne aspettiamo altre!

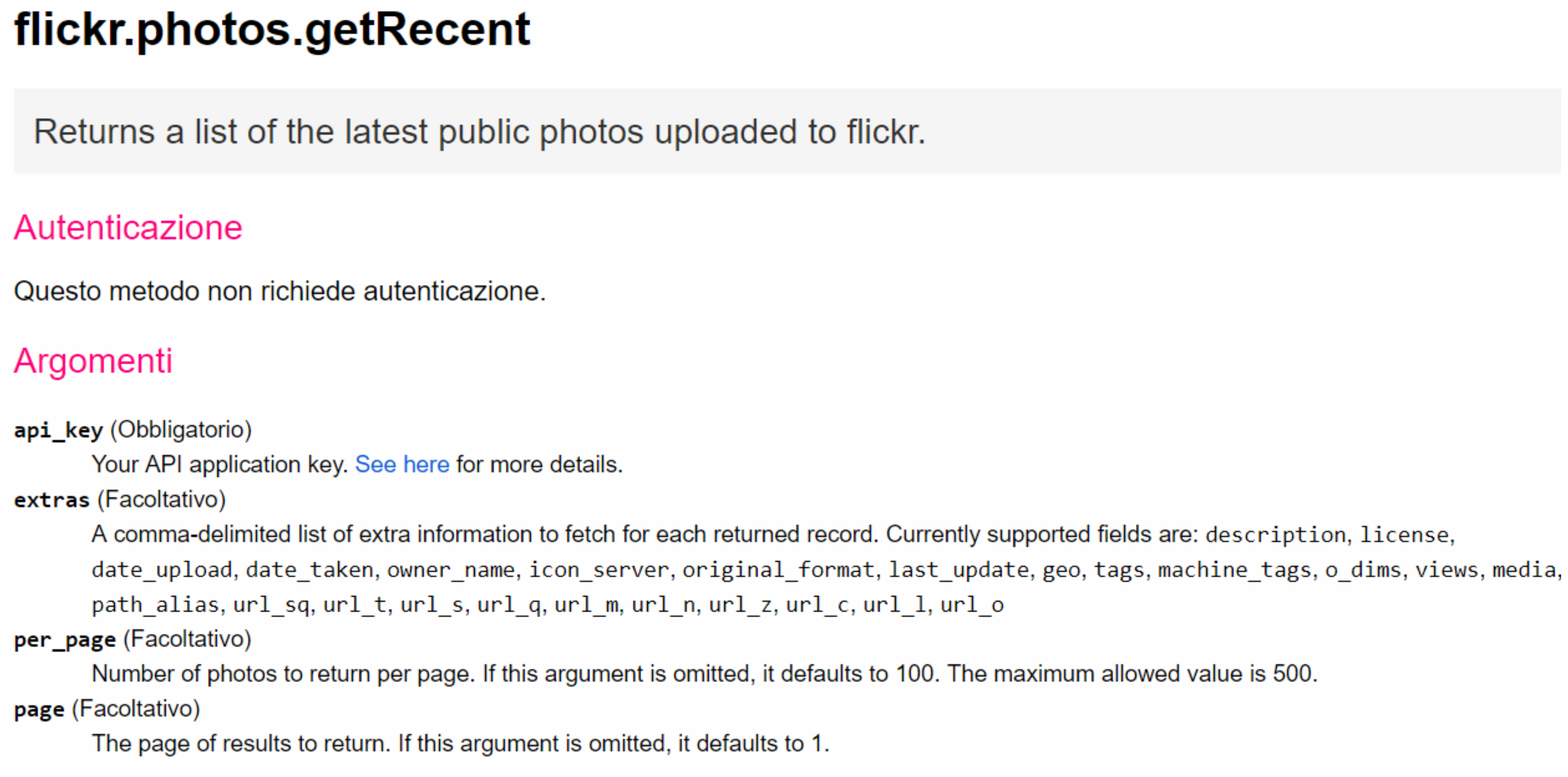
Thanks for contributing to the App Garden. If you have more apps in the works, we'd love to hear about them.

Richiedi un'altra chiave or learn more.

To visualize this interface go to https://flickr.com/services/api/

And then click on "API keys"

Example with Flickr



flickr.photos.getRecent

Returns a list of the latest public photos uploaded to flickr.

Autenticazione

Questo metodo non richiede autenticazione.

Argomenti

`api_key` (Obbligatorio)
Your API application key. See here for more details.

`extras` (Facoltativo)
A comma-delimited list of extra information to fetch for each returned record. Currently supported fields are: `description`, `license`, `date_upload`, `date_taken`, `owner_name`, `icon_server`, `original_format`, `last_update`, `geo`, `tags`, `machine_tags`, `o_dims`, `views`, `media`, `path_alias`, `url_sq`, `url_t`, `url_s`, `url_q`, `url_m`, `url_n`, `url_z`, `url_c`, `url_l`, `url_o`

`per_page` (Facoltativo)
Number of photos to return per page. If this argument is omitted, it defaults to 100. The maximum allowed value is 500.

`page` (Facoltativo)
The page of results to return. If this argument is omitted, it defaults to 1.

flickr.photos.search

- Return a list of photos matching some criteria.
- Unauthenticated calls will only return public photos.
- To return private photos, the caller must be authenticated with “read” privileges.

http://www.flickr.com/services/api/flickr.photos.search.html

Flickr.photos.search - output

- page: the page requested by the user
- pages: total number of pages where the images are distributed
- perpage: each page contains “perpage” photos
- total: total number of images for the selected query
- photo: array of photos, each photo has the following features:
  - Farm: id used to identify the flickr static sub domain name
  - id: id used to identify the photo
  - isfamily, isfriend, ispublic: Who can see the picture?
  - owner: id used to identify the owner of the photo
  - secret: id used as the photo secret code
  - server: id used to identify the server where the photo is hosted
  - title: title of the picture

Photo Source URLs

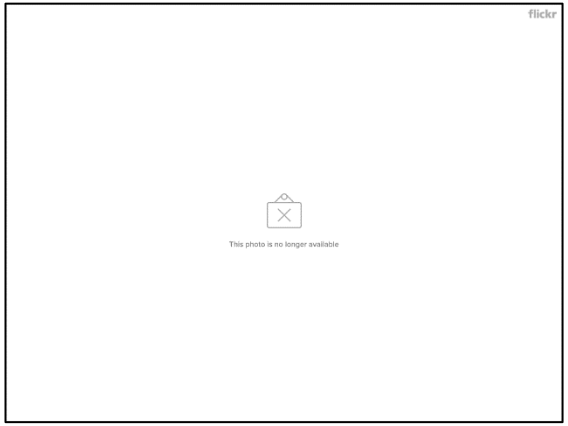
- You can construct the source URL to a photo once you know its ID, server ID, farm ID and secret, as returned by many API methods.
- Example:

https://farm{**farm-id**}.staticflickr.com/{**server-id**}/{**id**}_{**secret**}.jpg

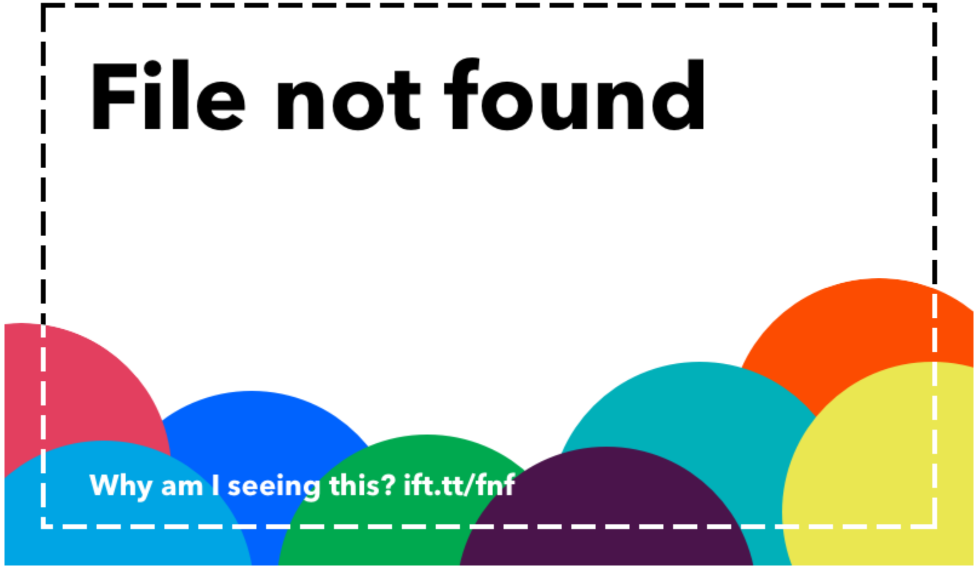
More info on photo URLs:

https://www.flickr.com/services/api/misc.urls.html

Flickr API


flickr

This photo is no longer available

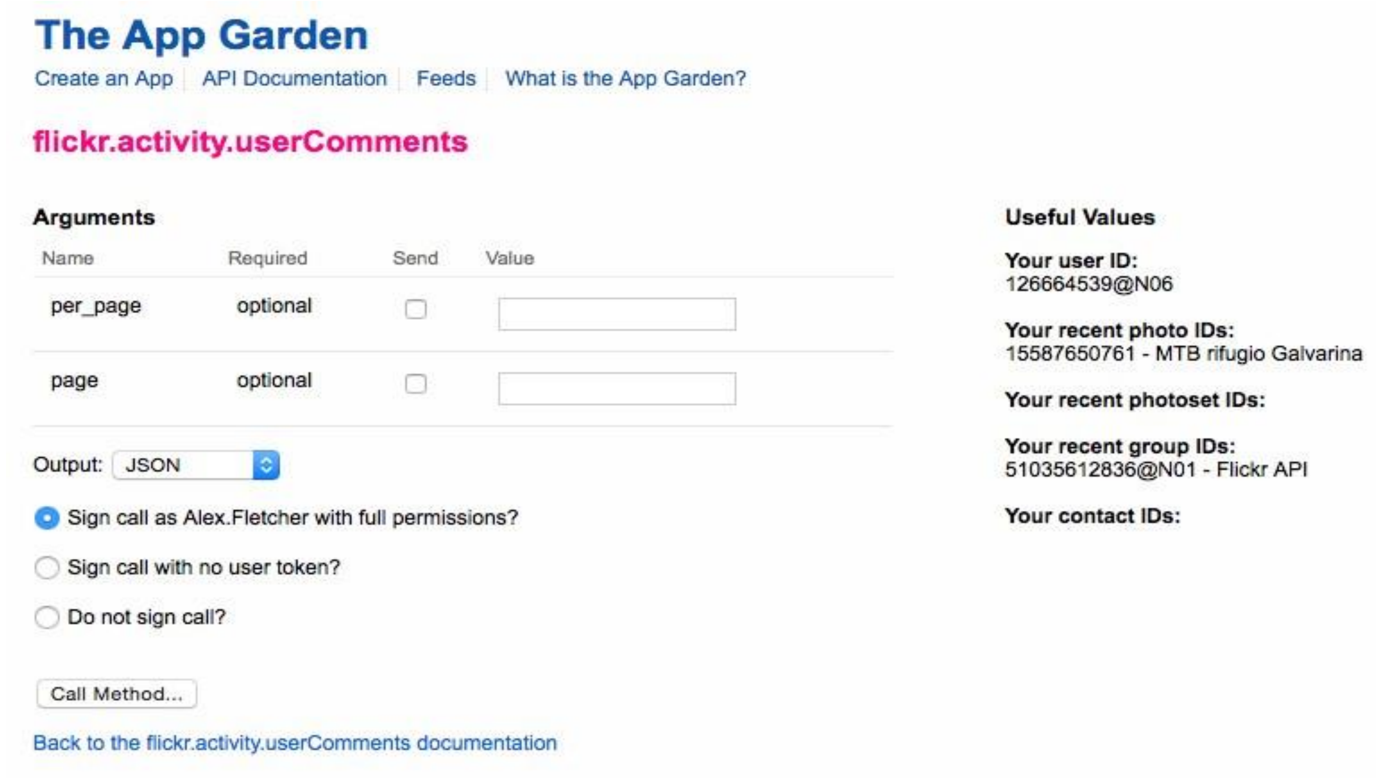

File not found

Why am I seeing this? ift.tt/fnf

IFTTT

Image not found

Flickr API Explorer



The App Garden

Create an App | API Documentation | Feeds | What is the App Garden?

flickr.activity.userComments

Arguments

Name	Required	Send	Value
per_page	optional	☐	
page	optional	☐	

Output: JSON

[✓ Sign call as Alex.Fletcher with full permissions?]

Sign call with no user token?

Do not sign call?

Back to the flickr.activity.userComments documentation

Useful Values

Your user ID:
126664539@N06

Your recent photo IDs:
15587650761 - MTB rifugio Galvarina

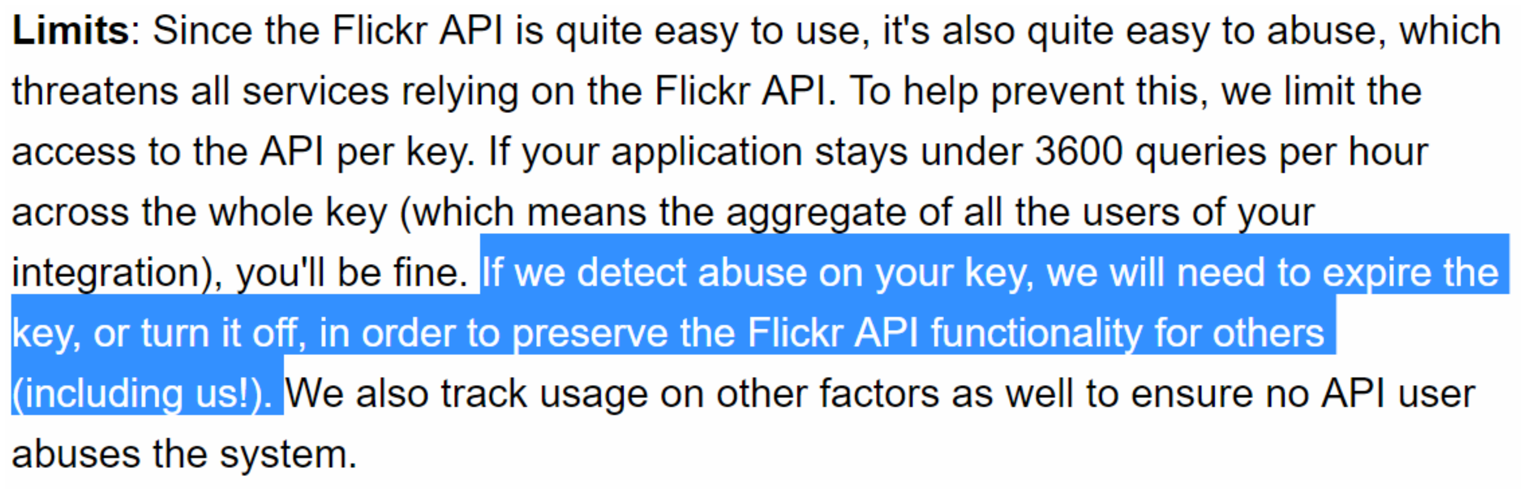
Your recent photoset IDs:

Your recent group IDs:
51035612836@N01 - Flickr API

Your contact IDs:

https://www.flickr.com/services/api/

Flickr API



Limits: Since the Flickr API is quite easy to use, it's also quite easy to abuse, which threatens all services relying on the Flickr API. To help prevent this, we limit the access to the API per key. If your application stays under 3600 queries per hour across the whole key (which means the aggregate of all the users of your integration), you'll be fine. If we detect abuse on your key, we will need to expire the key, or turn it off, in order to preserve the Flickr API functionality for others (including us!). We also track usage on other factors as well to ensure no API user abuses the system.

Source: https://www.flickr.com/services/developer/api/

Facebook Graph API

- Facebook has implemented an API for basic database manipulations named Graph API.
- The Graph API is the primary way to get data in and out of Facebook's social graph.
- It's a low-level HTTP-based API that you can use to query data, post new stories, upload photos and a variety of other tasks that an app might need to do.

https://developers.facebook.com

Facebook Graph API

The Graph API is named after the idea of a “social graph”: a representation of the information on Facebook composed of:

- **nodes** (basically "things" such as a User, a Photo, a Page, a Comment)
- **edges** (the connections between those "things", such as a Page's Photos, or a Photo's Comments)
- **fields** (info about those "things", such as the birthday of a User, or the name of a Page).

How the Graph API is structured

- Each node has a unique **ID** which is used to access it via the Graph API.

- Here's how you'duse the ID to make a request: GET graph.facebook.com
  /{node-id}/{edge-name}

- Full list of root nodes of the Graph API: https://developers.facebook.com/docs/graph-api/reference/v2.2?locale=it_IT

How the Graph API is structured

```
def getPagePosts(ID,since,until):

          ID = str(ID)
          host = "https://graph.facebook.com/v2.8/"

          path = ID +
          "/posts?fields=id,full_picture,shares,
          permalink_url,
          message,created_time,
          caption,description,
          comments{from,comment_count,
                   id,message,created_time,
                   like_count,user_likes},
          reactions{type,name,id}
          &since=" + str(since) +       "&until=" + str(until)

          params = urllib.urlencode({"access_token":
          ACCESS_TOKEN})
          url = "{host}{path}&{params}".format(host=host,
                 path=path, params=params)
```

```
posts
     - id
     - full_picture
     - …
     - comments
          - from
          - message
          - created_time
          - like_count
          - user_likes
     - reactions
          - type
          - name
          - id
```

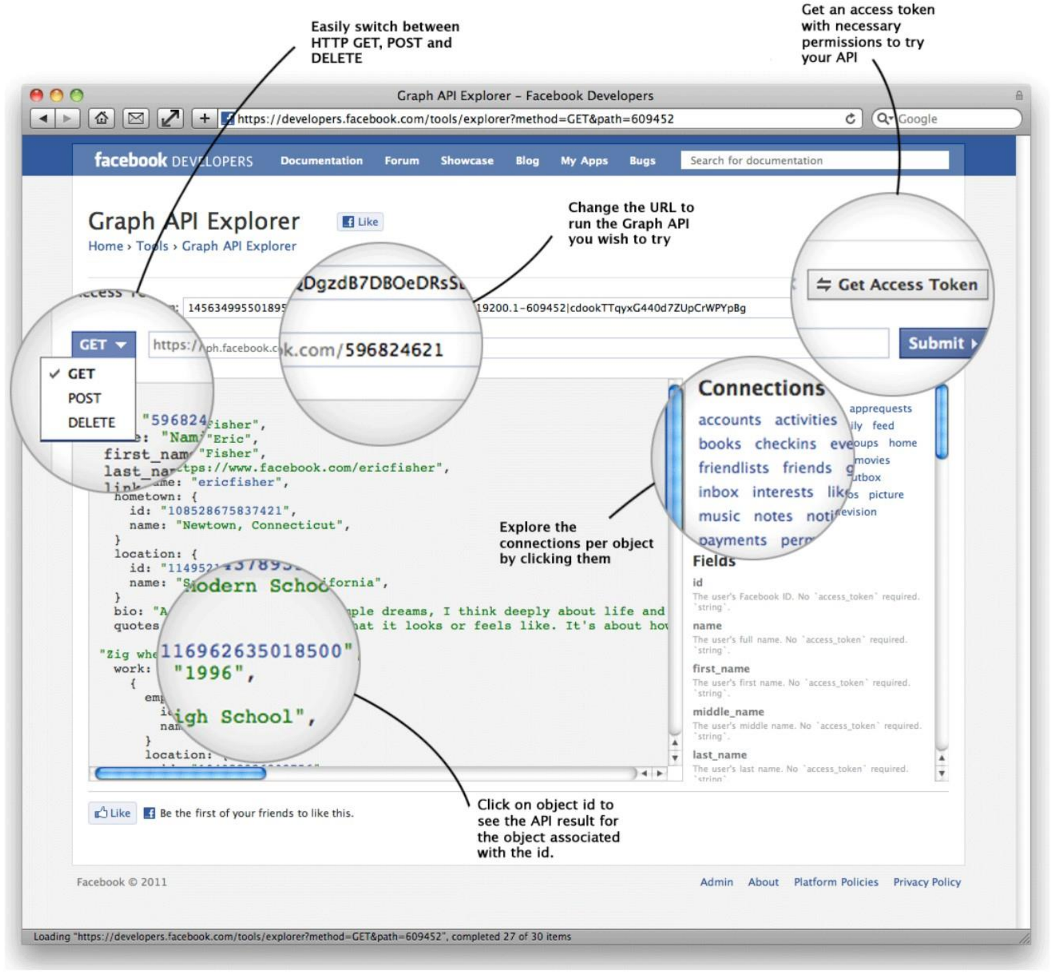
How the Graph API is structured

```
def getPagePosts(ID,since,until):

          ID = str(ID)
          host = "https://graph.facebook.com/v2.8/"

          path = ID +
          "/posts?fields=id,full_picture,shares,
          permalink_url,
          message,created_time,
          caption,description,
          comments{from,comment_count,
                   id,message,created_time,
                   like_count,user_likes},
          reactions{type,name,id}
          &since=" + str(since) +       "&until=" + str(until)

          params = urllib.urlencode({"access_token":
          ACCESS_TOKEN})
          url = "{host}{path}&{params}".format(host=host,
                 path=path, params=params)
```

```
posts
     - id
     - full_picture
     - …
     - comments
          - from
          - message
          - created_time
          - like_count
          - user_likes
     - reactions
          - type
          - name
          - id
```

Graph API Explorer

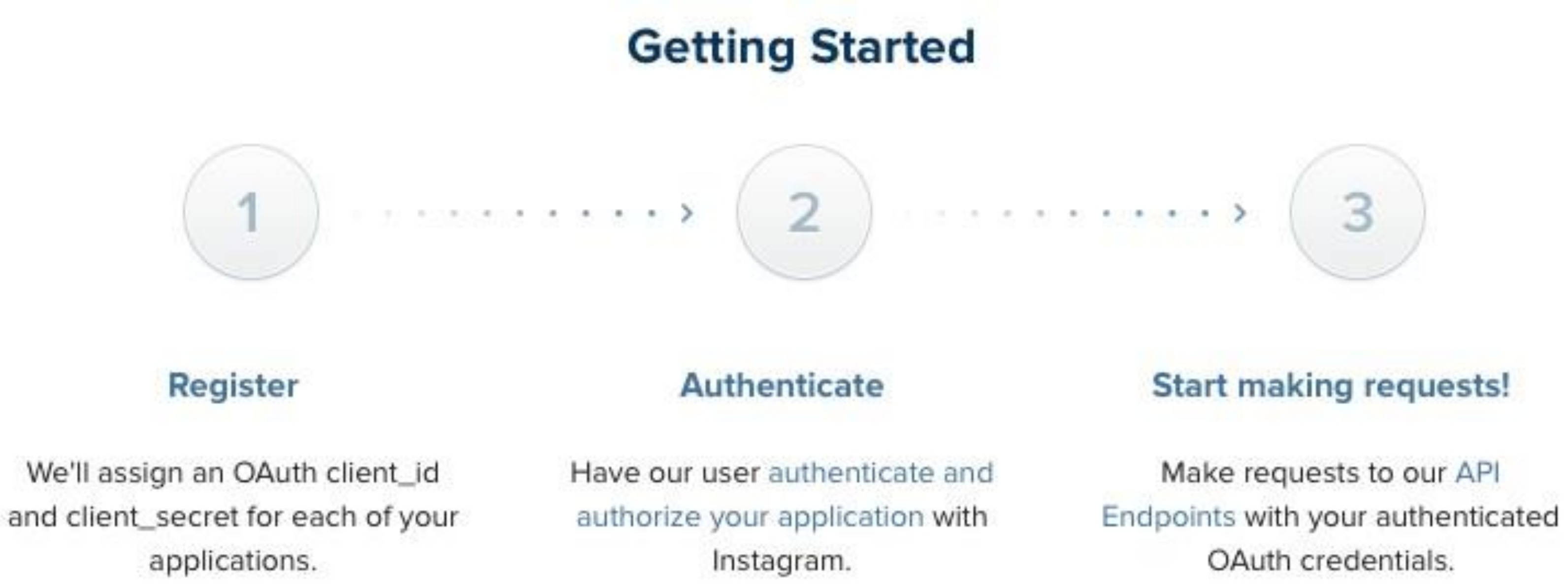
https://developers.facebook.com/tools/explorer/



Easily switch between HTTP GET, POST and DELETE
Get an access token with necessary permissions to try your API
Graph API Explorer – Facebook Developers
https://developers.facebook.com/tools/explorer?method=GET&path=609452
facebook DEVELOPERS
Documentation
Forum
Showcase
Blog
My Apps
Bugs
Search for documentation
Graph API Explorer
Like
Home › Tools › Graph API Explorer
Change the URL to run the Graph API you wish to try
Get Access Token
Submit
GET
POST
DELETE
Connections
accounts activities books checkins friendlists friends inbox interests music notes payments
Fields
id
name
first_name
middle_name
last_name
Explore the connections per object by clicking them
Click on object id to see the API result for the object associated with the id.
Be the first of your friends to like this.
Facebook © 2011
Admin
About
Platform Policies
Privacy Policy

Cambridge Analytica
+
facebook

Instagram API



Getting Started

1 → 2 → 3

Register

We'll assign an OAuth client_id and client_secret for each of your applications.

Authenticate

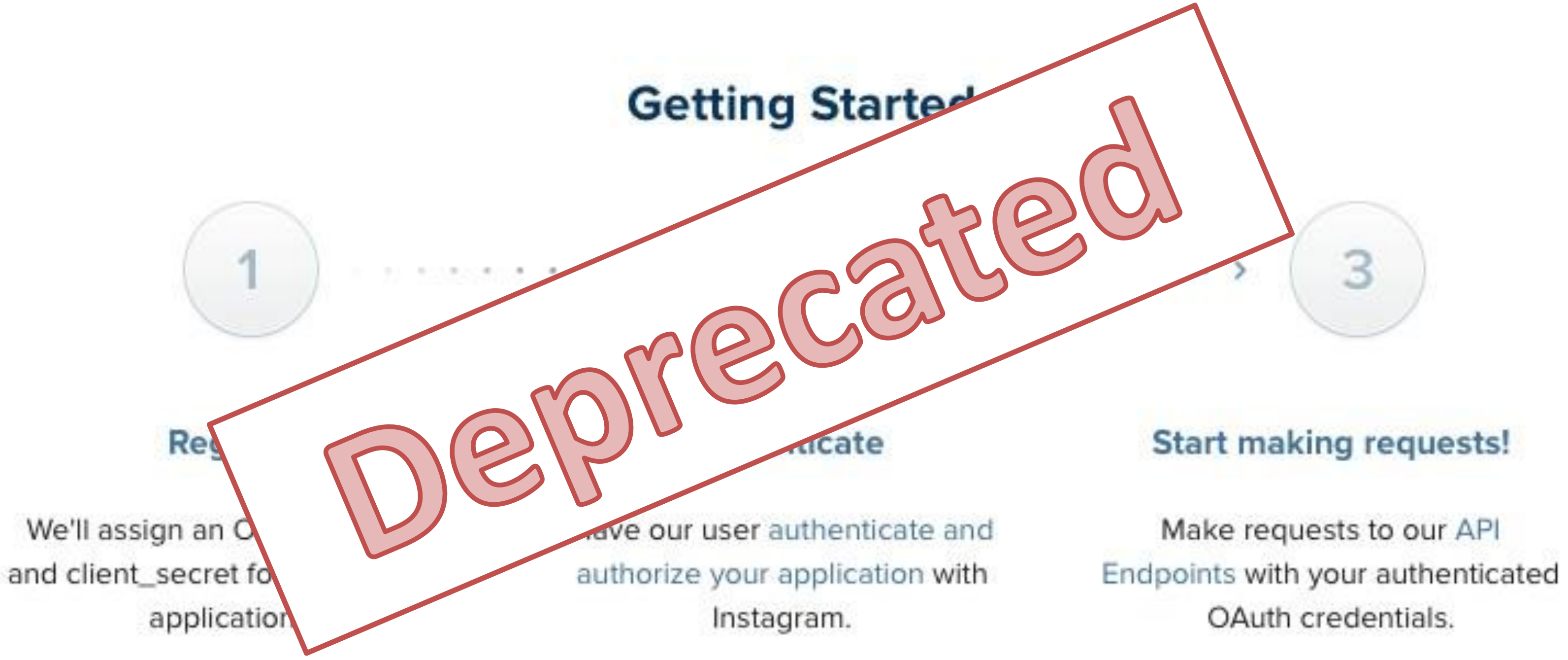
Have our user authenticate and authorize your application with Instagram.

Start making requests!

Make requests to our API Endpoints with your authenticated OAuth credentials.

http://instagram.com/developer

Instagram API



Getting Started

1

3

Reg

We'll assign an O
and client_secret fo
application

icate

ave our user authenticate and
authorize your application with
Instagram.

Start making requests!

Make requests to our API
Endpoints with your authenticated
OAuth credentials.

http://instagram.com/developer

Twitter API - Guidelines

1) Register your app to https://apps.twitter.com to get the credentials needed to perform oauth authentication.

2) Access full documentation of the Twitter API https://dev.twitter.com/docs

Twitter API & Python

1) Tweepy API

2) TwitterSearch API

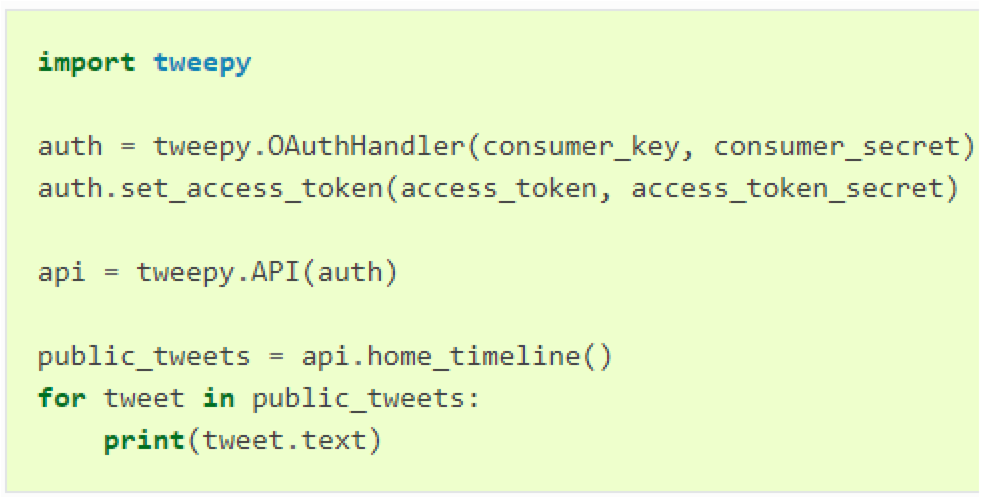
Twitter API & Python

Tweepy API

Tweepy API is an easy-to-use Python library for accessing the Twitter API.

Twitter API & Python

Tweepy API



```
import tweepy

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)

public_tweets = api.home_timeline()
for tweet in public_tweets:
    print(tweet.text)
```

To get your own consumer key, consumer secret, access token and access secret, create a Twitter application: https://apps.twitter.com/app/new.

Twitter API & Python

Tweepy API

Tweepy makes it easier to use the twitter streaming api by handling authentication, connection, creating and destroying the session, reading incoming messages, and partially routing messages.

The Twitter ***streaming API*** is used to download twitter messages in real time. It is useful for obtaining a high volume of tweets, or for creating a live feed using a site stream or user stream.

Twitter API & Python

Tweepy API

In Tweepy, an instance of **tweepy.Stream** establishes a streaming session and routes messages to **StreamListener** instance.

The **on_data** method of a stream listener receives all messages and calls functions according to the message type.

Twitter API & Python

Tweepy API

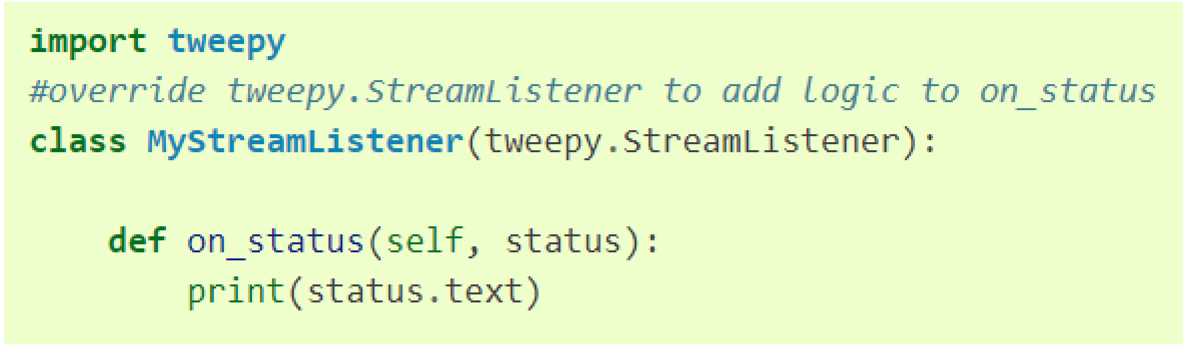
Therefore using the streaming api has three steps.

1. Create a class inheriting from **StreamListener**
2. Using that class create a **Stream** object
3. Connect to the Twitter API using the **Stream**.

Twitter API & Python

Tweepy API

Step 1: creating a StreamListener subclass



```python
import tweepy
#override tweepy.StreamListener to add logic to on_status
class MyStreamListener(tweepy.StreamListener):

    def on_status(self, status):
        print(status.text)
```

The **on_data** method of Tweepy's **StreamListener** conveniently passes data from statuses to the **on_status** method. We just create class **MyStreamListener** inheriting from **StreamListener** and override **on_status** method.

Twitter API & Python

Tweepy API

Step 2: creating a Stream (instance)

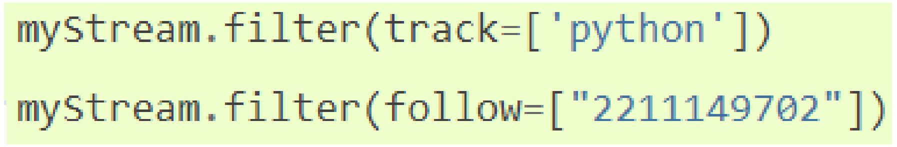
```
myStreamListener = MyStreamListener()
myStream = tweepy.Stream(auth = api.auth, listener=myStreamListener)
```

To create a Stream we need an api object (see authentication steps) and an instance of Listener (MyStreamListener in the example).

Twitter API & Python

Tweepy API

Step 3: starting a Stream



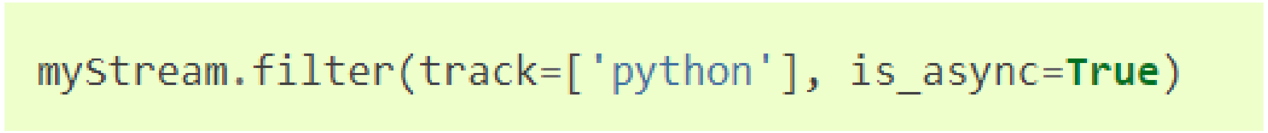
```
myStream.filter(track=['python'])

myStream.filter(follow=["2211149702"])
```

We can ***filter*** the tweets containing some word (e.g., ‘python’) or related to a specific user (e.g., with id = 2211149702).

Twitter API & Python

Tweepy API

More on Twitter Streaming API



```
myStream.filter(track=['python'], is_async=True)
```

Streams do not terminate unless the connection is closed. Tweepy offers a ***is_async*** parameter on filter function, so the stream will run on a new thread (no blocking code).

Twitter API & Python

Tweepy API

More on Twitter Streaming API

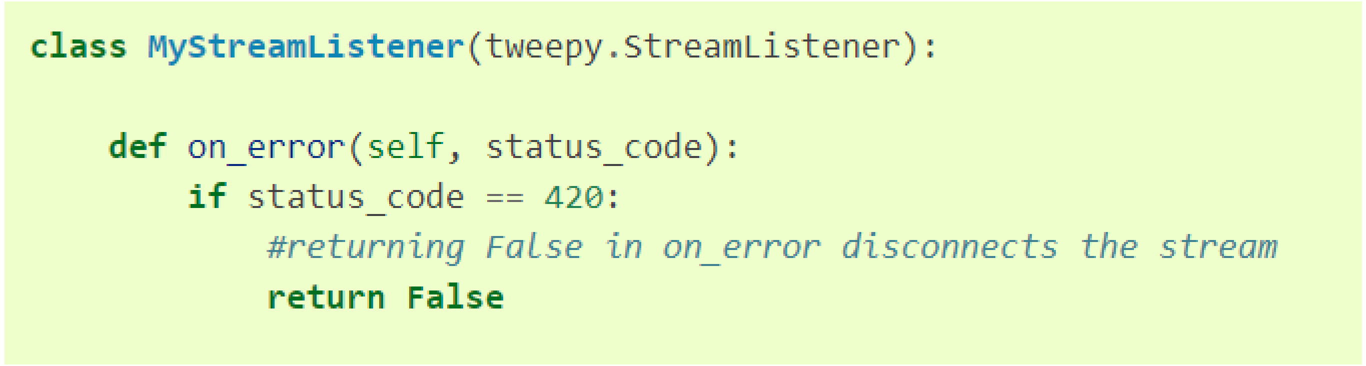
When using Twitter's streaming API one must be careful of the dangers of ***rate limiting***.

If clients exceed a limited number of attempts to connect to the streaming API in a window of time, they will receive error 420. The amount of time a client has to wait after receiving error 420 will ***increase exponentially*** each time they make a failed attempt.

Twitter API & Python

Tweepy API

More on Twitter Streaming API



```python
class MyStreamListener(tweepy.StreamListener):

    def on_error(self, status_code):
        if status_code == 420:
            #returning False in on_error disconnects the stream
            return False
```

Tweepy's **Stream Listener** passes error codes to an **on_error** stub. The default implementation returns **False** for all codes, but we can override it to allow Tweepy to reconnect for some or all codes.

Twitter API & Python

Tweepy API

API Reference:
https://tweepy.readthedocs.io/en/latest/api.html#api-reference

Tweepy Streaming API Reference:
http://docs.tweepy.org/en/latest/streaming_how_to.html

Twitter API & Python

Twitter Search API

This library allows you easily ***<u>create a search</u>*** through the Twitter API without having to know too much about the API details.

Twitter API & Python

Twitter Search API – basic interfaces (Search, User)

```
tso = TwitterSearchOrder() # Basic class for API’s calls
tso.set_keywords(['game of thrones','finale’])
tso.set_language('en')
…
```

```
tuo = TwitterUserOrder(“Cristiano”) # Create a
                                    # TwitterUserOrder
…
```

Twitter API & Python

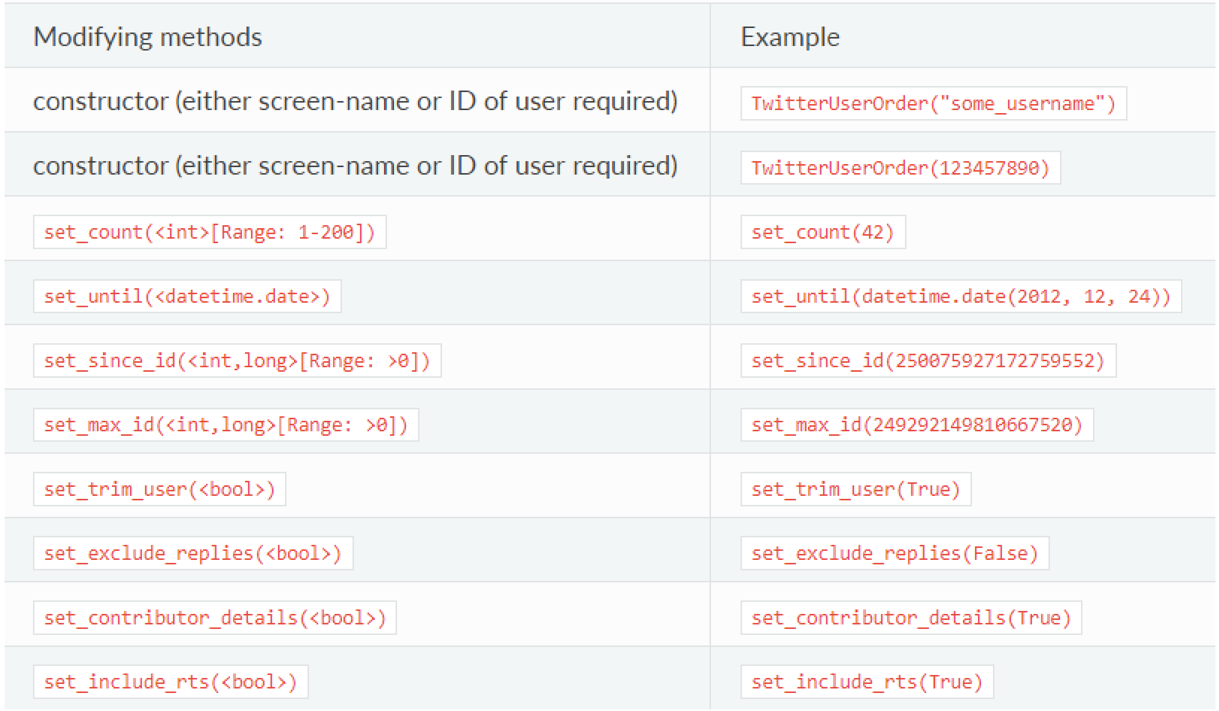
Twitter Search API - Advanced usage:

- *TwitterUserOrder*
  https://twittersearch.readthedocs.io/en/latest/advanced_usage_tuo.html

- *TwitterSearchOrder*
  https://twittersearch.readthedocs.io/en/latest/advanced_usage_tso.html#

Twitter API & Python

Twitter Search API – TwitterUserOrder

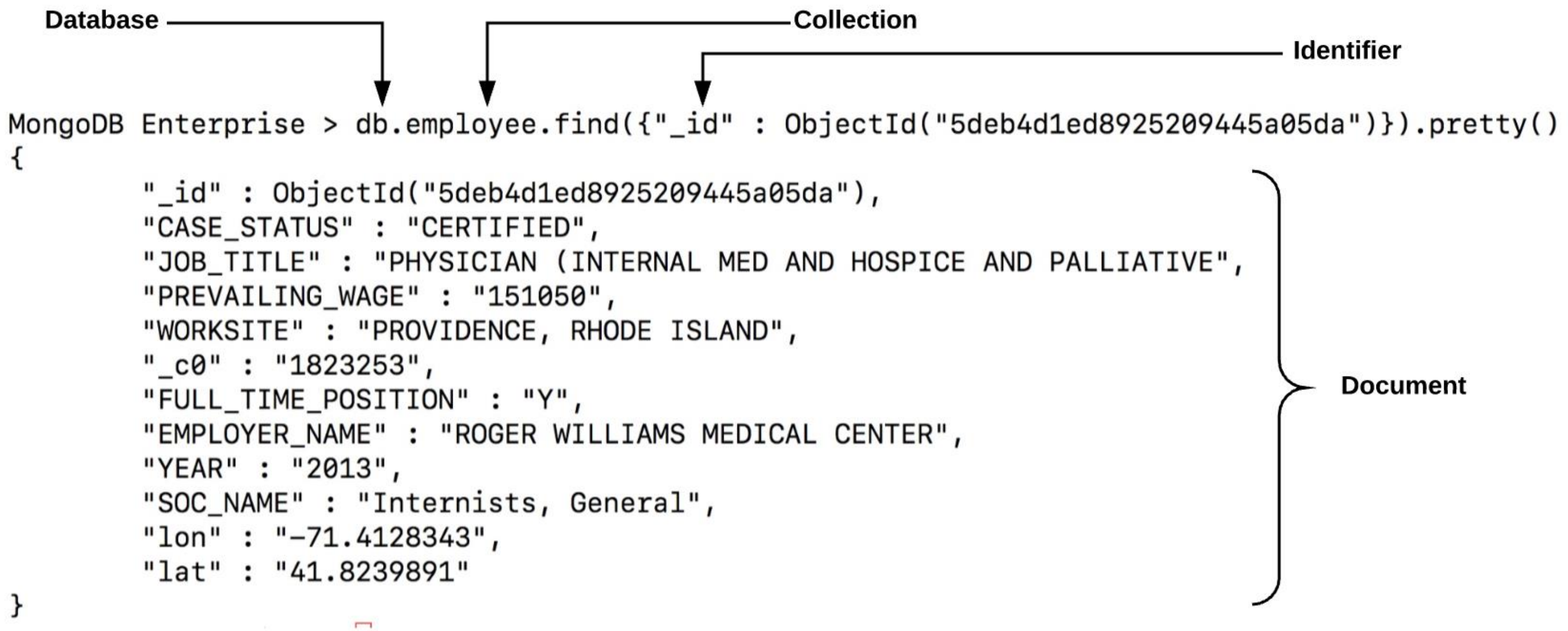


Modifying methods	Example
constructor (either screen-name or ID of user required)	`TwitterUserOrder("some_username")`
constructor (either screen-name or ID of user required)	`TwitterUserOrder(123457890)`
`set_count(<int>[Range: 1-200])`	`set_count(42)`
`set_until(<datetime.date>)`	`set_until(datetime.date(2012, 12, 24))`
`set_since_id(<int,long>[Range: >0])`	`set_since_id(250075927172759552)`
`set_max_id(<int,long>[Range: >0])`	`set_max_id(249292149810667520)`
`set_trim_user(<bool>)`	`set_trim_user(True)`
`set_exclude_replies(<bool>)`	`set_exclude_replies(False)`
`set_contributor_details(<bool>)`	`set_contributor_details(True)`
`set_include_rts(<bool>)`	`set_include_rts(True)`

mongoDB®

MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

https://www.mongodb.com/it



```
MongoDB Enterprise > db.employee.find({"_id" : ObjectId("5deb4d1ed8925209445a05da")}).pretty()
{
        "_id" : ObjectId("5deb4d1ed8925209445a05da"),
        "CASE_STATUS" : "CERTIFIED",
        "JOB_TITLE" : "PHYSICIAN (INTERNAL MED AND HOSPICE AND PALLIATIVE",
        "PREVAILING_WAGE" : "151050",
        "WORKSITE" : "PROVIDENCE, RHODE ISLAND",
        "_c0" : "1823253",
        "FULL_TIME_POSITION" : "Y",
        "EMPLOYER_NAME" : "ROGER WILLIAMS MEDICAL CENTER",
        "YEAR" : "2013",
        "SOC_NAME" : "Internists, General",
        "lon" : "-71.4128343",
        "lat" : "41.8239891"
}
```

https://www.mongodb.com/it

References

- Flickr API: https://www.flickr.com/services/api/
- Instagram API: http://instagram.com/developer/api-console/
- Facebook API: https://developers.facebook.com/docs/graph-api/
- Twitter API: https://dev.twitter.com/docs
- MongoDB: https://www.mongodb.com/it

Practical Session

1) Crawling from Flickr

2) Crawling from Twitter