Twitter 데이터 수집

우리는 2. Twitter API 사용하기에서에서 트위터 App 을 설정하였다. 이제 해당 앱 ID 와 비밀키를 가지고 파이썬을 이용하여 데이터를 가지고 오는 방법에 대하여 설명하고자 한다. 페이스북의 경우에는 앱 아이디와 토큰을 요청 URL 에 파라미터로 전송하여 간단하게 데이터를 가지고 올 수 있는 기능을 제공한다. 그러나 트위터의 경우에는 OAuth1.0a 의 기본에 충실하게 억세스 토큰을 가지고 와야 실제 원하는 데이터를 조회 할 수 있다.

1 OAuth 란?

사용자는 특정 서비스를 사용하기 위하여 아이디와 비밀번호라는 고전적인 방법을 이용하였다. 이를 위하여 서비스 제공자는 사용자의 아이와 비밀번호를 관리하여야 하는데 서버의 공격등으로 인해 아이디와 비밀번호가 노출되는 사고가 빈번히 발생하고 이를 위하여 데이터베이스에 비밀번호를 암호화 하는 등 다양한 형태의 보안 기술이 발생하였다.

OAuth 의 시작은 2006 년에 트위터와 소셜 북마크 서비스인 Gnolia 사의 개발자들은 접속한 사용자들이 서비스를 사용하기 위하여 사용자를 인증(Authentication)하고 특정한 서비스만 사용하기 위한 권한(Authorization)을 주기 위한 적당한 인증 알고리즘이 없다고 판단하여 개발에 착수, 2007 년 10 월에 OAuth1.0 을 발표하였다. 그 이후 세션 고정 공격(Session Fixation Attack: 세션 하이재킹(Hijacking) 기법중의 하나로 유요한 유저 세션을 탈취하여 인증을 회피하는 공격 방법)에 취약함이 발견되어 이를 개선하고 “The OAuth 1.0 Protocol”이라는 이름으로 2010 년에 표준안이 RFC5849 로 IETF 에 의해 체택 되었다. 기존의 1.0 에서 보안된 개념으로 실제 1.0 이라고 제정되었지만 많은 사람들은 OAuth 1.0a 라고 부른다. 현재 OAuth 는 처음 설계를 한 트위터를 필두로, 구글, 페이스북, 마이크로 소프트 등의 외국 거대 서비스 회사 및 국내의 네이버, 다음 카카오 등에서 사용하고 있다.

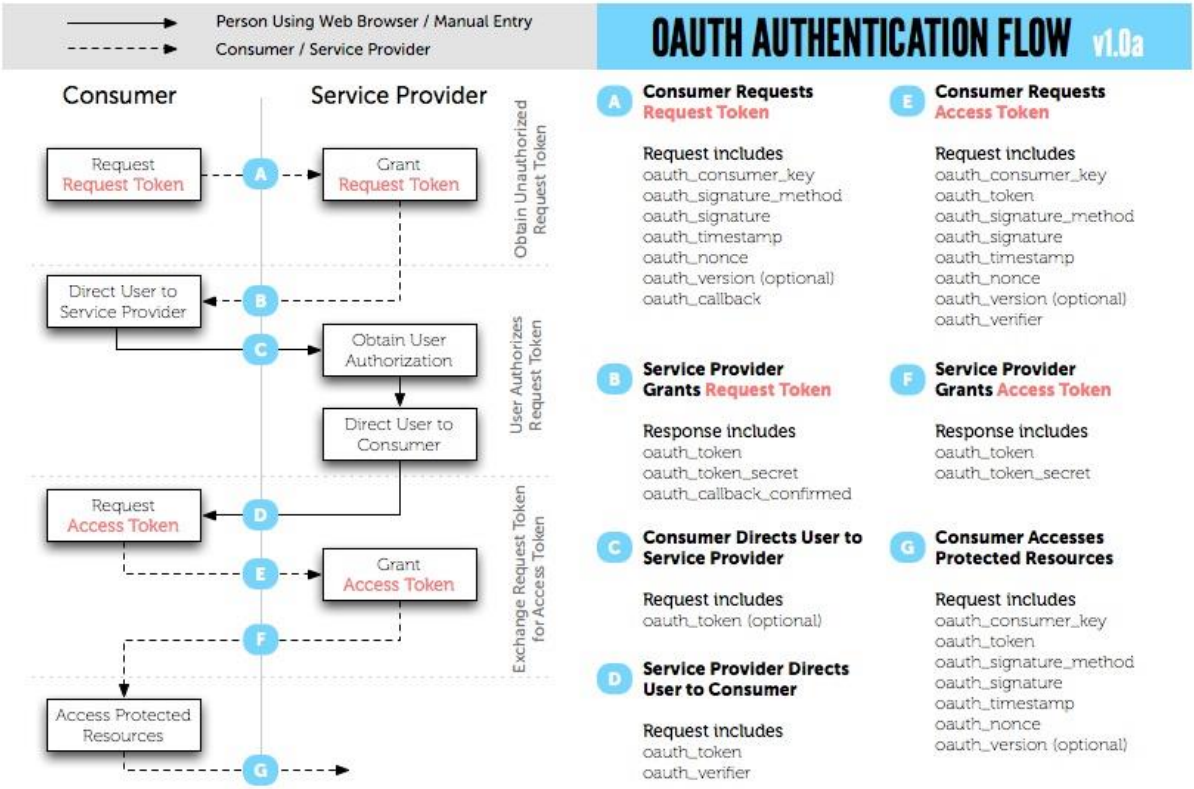
OAuth 의 장점은 사용자의 아이디와 비밀번호를 이용하여 인증을 하는 것이 아니고, 특정 서비스를 사용하고자 할 때 임시 사용 티켓을 발급하여, 서비스가 만료되거나 특정 기간이 종료하면 티켓을 사용할 수 없는 구조로 되어있다.

1.1 OAuth 1.0a 인증 과정

소비자 Consumer	OAuth 를 사용해 서비스 제공자의 기능을 사용하려는 프로그램 : 우리가 제작하는 프로그램의 입장
서비스 제공자 Service Provider	OAuth 를 사용하는 Open API 제공자 : 트위터, 페이스 북 등
요청 토큰 Request Token	OA 소비자(Consumer)가 서비스제공자(Service Provider)에게 접근 권한을 받기 위해 사용하는 값, 인증이 완료되면 접근 토큰(Access Token)으로 변경된다
접근 토큰 Access	인증 후 소비자(Consumer)가 서비스제공자(Service Provider)의 자원에 접근하기 위한 키를 포함한 값

Token	
사용자 User	서비스제공자(Service Provider)에 계정을 가지고 있으면서, 소비자(Consumer)를 이용하려는 사용자

OAuth 를 이용하여 사용자를 인증하는 과정을 OAuth Dance – '춤'이라고 표현할 수 있는 개발자들의 자유스러움이 부럽다 - 라고 한다. 먼저 OAuth 인증을 위하여 사용하는 주체를 표현하는 용어는 다음과 같다.



[그림 1] OAuth 1.0a 인증 과정 (http://oauth.net/core/diagram.png)

1) Request Token 요청 및 발급

- Consumer Key 와 요청 시간(timestamp), 요청자료를 암호화 한 방식(HMAC-SHA1 등), 버전(1.0) 정보, 악의적인 정보 요청을 방지하기 위한 임의 문자열과 서명값을 만든후 해당 서버로 요청한다.
- 서명값(oauth_signature) 앞에서 설명한 각 변수값과 HTTP 요청 방식(POST 또는 GET)을 지정한 암호화 방식으로 암호화하여 만들어 낸다.
- 서비스 제공자에게 Request Token 을 요청하면, 서비스 제공자는 'oauth_token'과 'oauth_token_secret'값을 회신한다.

2) 사용자 인증 페이지 호출 및 수락

- 수신한 'oauth_token'을 가지고 서비스 제공자가 지정한 인증페이지로 인증 수락을 요청한다. 인증 페이지는 각 서비스제공자가 제공한다.

3) Access Token 요청 및 발급

- ‘consumer_key’와 ‘oauth_token’ 및 기타 변수들을 이용하여 ‘oauth_signature’ 를 생성한 후 요청하면 서비스 제공자는 ‘oauth_token’과 ‘oauth_token_secret’ 및 부가 정보를 회신한다.

4) 해당 API 접근

- 수신한 ‘oauth_token’을 이용하여 서비스제공자의 API 를 사용한다.

1.2 OAuth 2.0 의 탄생

아쉽게도 OAuth 2.0 은 OAuth 1.0a 와 호환성을 가지고 개발되지 않았다. OAuth 2.0 은 시그네처(signature)의 생성이 복잡하고 CPU 부하를 많이 가지고 있어서 모바일 환경에 적합하지 않아 시작된 표준이다. 그러나 하나의 표준이 존재하지 않고 프레임워크(Framework)라고 부르며, 여러 개의 표준 작업이 기업의 목적에 따라 이루어지고 있다. 간단하고 더 많은 인증 방법을 제공하고 있으나 각 기업이나 단체가 자신들의 서비스를 목적으로 표준 작업을 진행하다 보니 복잡하고 방대해져서 OAuth 1.0a 를 대체하는데 많은 시간이 걸릴 것이라 관측하는 사람들도 적지 않다.

2 트윗(Tweet) 가지고 오기

파이썬에서 트위터를 사용하기 위해서는 다양한 모듈이 공개되고, 업그레이도 되고 있다. 그중에서 가장 보편적으로 사용하는 것은 “Tweepy” 모듈이다. 그러나 본 책에서는 최대한 모듈의 종속성을 배제하고 기본적인 접근 방법을 인지하기 위하여 최소한의 모듈만을 사용하는 것을 원칙으로 한다.

앞 절에서 언급하였듯이 트위터는 OAuth 1.0a 표준을 사용하여 응용 프로그램의 인증을 하고, 서비스 API 를 사용할 수 있는 키값을 제공하여 준다. 이를 위하여 OAuth 를 이해하는 것이 이전 절의 목적이었다면, 이번절에서는 OAuth 를 이용하여 인증하는 방법에 대하여 알아 볼 것이다. OAuth 는 암호화 알고리즘 및 시그네처(signature)의 구성을 위하여 많은 코딩이 필요하므로 현재 OAuth 인증을 위하여 개발되고 업그레이드 되고 있는 파이썬 “OAuth2” 모듈을 import 하여 사용하고, 각 인증 과정은 3 절에서 살펴 보도록 하겠다.

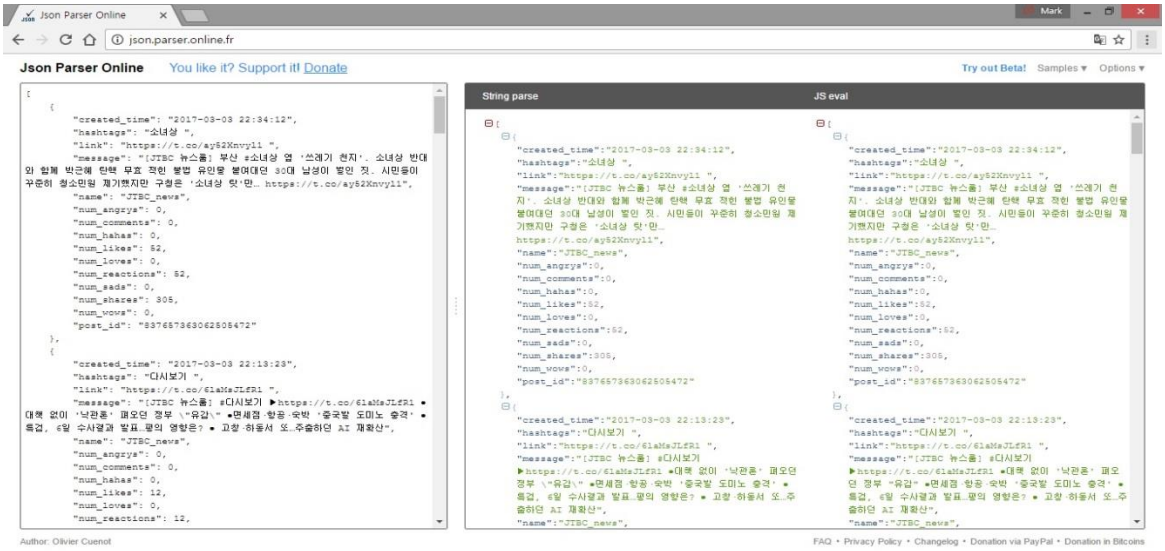
특정 트위터의 트윗을 가져오기 위하여 다음의 코드를 작성한다.

```
import oauth2
import json
import datetime
import time
from config import *

#[CODE 1]

def oauth2_request(consumer_key, consumer_secret, access_token, access_secret):
    try:
        consumer = oauth2.Consumer(key=consumer_key, secret=consumer_secret)
        token = oauth2.Token(key=access_token, secret=access_secret)
        client = oauth2.Client(consumer, token)
        return client
    except Exception as e:
        print(e)
        return None

#[CODE 2]

def get_user_timeline(client, screen_name, count=50, include_rts='False'):
    base = "https://api.twitter.com/1.1"
    node = "/statuses/user_timeline.json"
    fields = "?screen_name=%s&count=%s&include_rts=%s" % (screen_name, count, include_rts)
```

```
        #fields = "?screen_name=%s" % (screen_name)

        url = base + node + fields

        response, data = client.request(url)

        try:
            if response['status'] == '200':
                return json.loads(data.decode('utf-8'))
        except Exception as e:
            print(e)
            return None

#[CODE 3]

def getTwitterTwit(tweet, jsonResult):

        tweet_id = tweet['id_str']
        tweet_message = '' if 'text' not in tweet.keys() else tweet['text']

        screen_name = '' if 'user' not in tweet.keys() else tweet['user']['screen_name']

        tweet_link = ''
        if tweet['entities']['urls']: #list

            for i, val in enumerate(tweet['entities']['urls']):
                tweet_link = tweet_link + tweet['entities']['urls'][i]['url'] + ' '
        else:
            tweet_link = ''

        hashtags = ''
        if tweet['entities']['hashtags']: #list

            for i, val in enumerate(tweet['entities']['hashtags']):
                hashtags = hashtags + tweet['entities']['hashtags'][i]['text'] + ' '
        else:
            hashtags = ''

        if 'created_at' in tweet.keys():
            # Twitter used UTC Format. EST = UTC + 9(Korean Time) Format ex: Fri Feb 10 03:57:27 +0000 2017

            tweet_published = datetime.datetime.strptime(tweet['created_at'],'%a %b %d %H:%M:%S +0000 %Y')
            tweet_published = tweet_published + datetime.timedelta(hours=+9)
            tweet_published = tweet_published.strftime('%Y-%m-%d %H:%M:%S')
        else:
            tweet_published = ''

        num_favorite_count = 0 if 'favorite_count' not in tweet.keys() else tweet['favorite_count']
        num_comments = 0
        num_shares = 0 if 'retweet_count' not in tweet.keys() else tweet['retweet_count']
        num_likes = num_favorite_count
        num_loves = num_wows = num_hahas = num_sads = num_angrys = 0

        jsonResult.append({'post_id':tweet_id, 'message':tweet_message,
                           'name':screen_name, 'link':tweet_link,
                           'created_time':tweet_published, 'num_reactions':num_favorite_count,
                           'num_comments':num_comments, 'num_shares':num_shares,
                           'num_likes':num_likes, 'num_loves':num_loves,
                           'num_wows':num_wows, 'num_hahas':num_hahas,
                           'num_sads':num_sads, 'num_angrys':num_angrys, 'hashtags': hashtags})

def main():
```

```
    screen_name = "jtbc_news"

    num_posts = 50

    jsonResult = []

    client = oauth2_request(CONSUMER_KEY, CONSUMER_SECRET, ACCESS_TOKEN,
ACCESS_SECRET)
    tweets = get_user_timeline(client, screen_name)

    for tweet in tweets:
        getTwitterTwit(tweet, jsonResult)

    with open('%s_twitter.json' % (screen_name), 'w', encoding='utf8') as outfile:
        str_ = json.dumps(jsonResult,
                          indent=4, sort_keys=True,
                          ensure_ascii=False)
        outfile.write(str_)

    print ('%s_twitter.json SAVED' % (screen_name))

if __name__ == '__main__':
    main()
```

코드를 수행하면 jtbc_news_twitter.json 파일이 .py 가 있는 폴더에 저장되어 있을 것이다. 해당 파일을 JSON 뷰어를 이용하여 읽어 보면 [그림 2]와 같이 트윗(tweet)이 JSON 형태로 저장된 것을 확인할 수 있다.



[그림 2] JTBC_NEWS 트위터를 크롤링한 JSON 모습

[그림 2]의 처음 JSON 데이터는 [그림 3]의 내용을 나타내고 있다.

[그림 3] 저장한 JTBC 뉴스 트윗

실제 이전 절에서 페이스북 API 를 이용하여 JSON 형태의 데이터를 수신하고, 취급하는 방법에 대하여 충분히 이해하였으리라 생각하여 중복되는 내용은 5. Facebook 데이터 수집을 확인하기 바란다.

먼저 빈번하게 사용하는 환경변수의 값인 Cusomer_key 와 Customer_Secret, Access_token 과 Access_token 은 congi.py 라는 파일로 따로 작성한 후 import 하여 사용하였다.

```
CONSUMER_KEY = "[COUSUMER_KEY]"
CONSUMER_SECRET = "[COSUMER_SECRET]"
ACCESS_TOKEN = "[ACCESS_TOKEN]"
ACCESS_SECRET = "[ACCESS_SECRET]"
```

해당 .py 파일의 키값은 트위터에서 제공받은 값으로 변경하기 바란다.(Part I.의 트위터 개발자 계정 부분을 참조하기 바란다)

```
#[CODE 1] oauth2 모듈을 이용한 키 인증

def oauth2_request(consumer_key, consumer_secret, access_token, access_secret):
    try:
        consumer = oauth2.Consumer(key=consumer_key, secret=consumer_secret)
        token = oauth2.Token(key=access_token, secret=access_secret)
        client = oauth2.Client(consumer, token)
        return client
    except Exception as e:
        print(e)
        return None
```

oauth2 모듈은 OAuth 의 복잡한 과정을 간단하게 해결해주는 모듈을 제공한다. 우리는 부여받은 consumer_key 와 해당 secret, 그리고 acces_token 과 secret 을 oath2 에 전달해 줌으로써 기본적인 인증을 위한 준비를 마친다.

기본값을 지정한 후 consumer 와 token 오브젝트(object)를 Client 클래스로 전달하면 복잡한 댄싱과정을 마치고 access_token 을 포함하는 OAuthClient 를 반환한다.

```
#[CODE 2] 사용자의 timeline 트윗을 수신

def get_user_timeline(client, screen_name, count=50, include_rts='False'):
    base = "https://api.twitter.com/1.1"
    node = "/statuses/user_timeline.json"
    fields = "?screen_name=%s&count=%s&include_rts=%s" % (screen_name, count, include_rts)
    url = base + node + fields
```

```
    response, data = client.request(url)

    try:
        if response['status'] == '200':
            return json.loads(data.decode('utf-8'))
    except Exception as e:
        print(e)
        return None
```

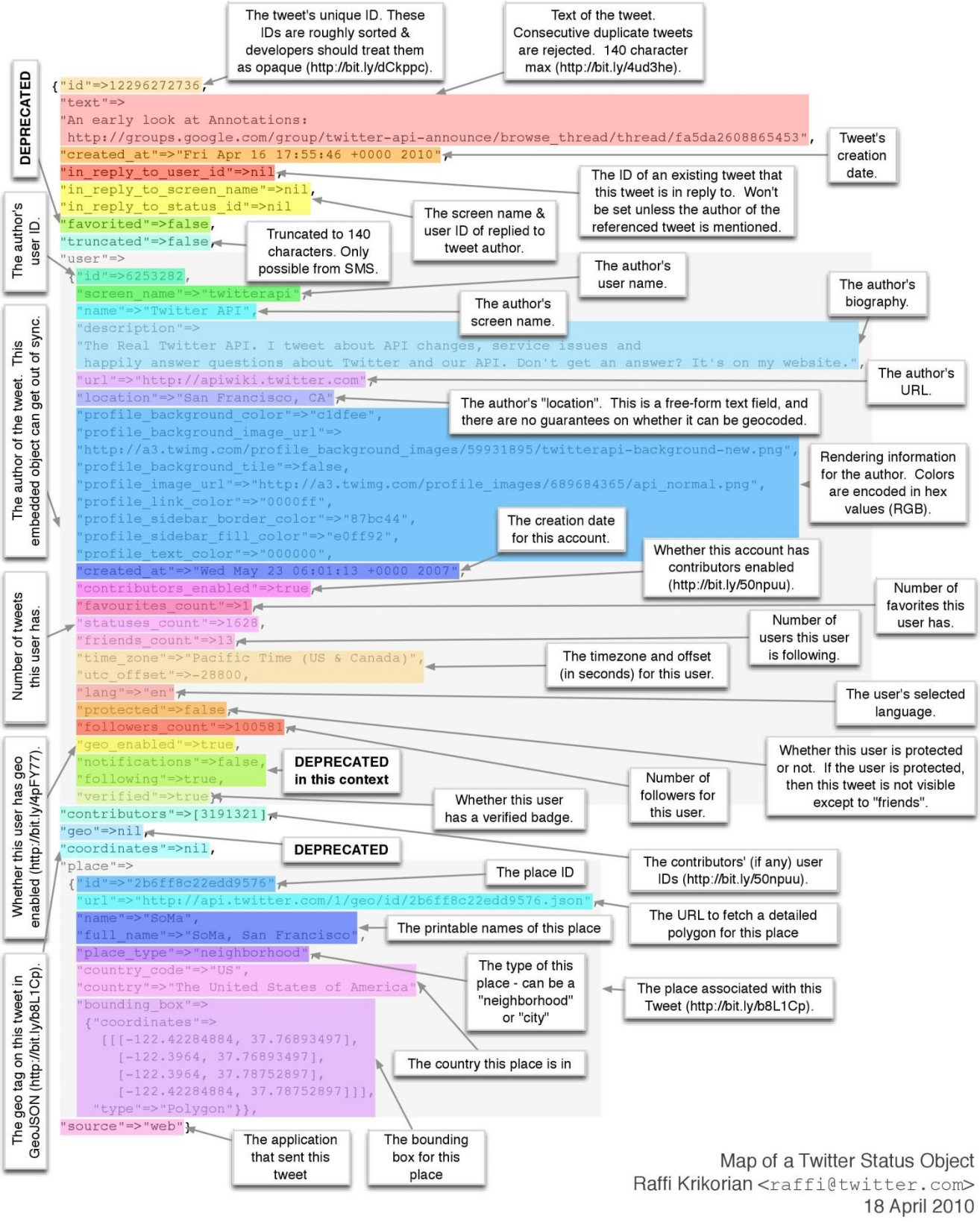
[CODE 2]의 경우에는 페이스북의 타임라인을 얻어 오는것과 동일한 형식을 가진다. 여기서 screen_name 은 트위터에서 사용하는 공식 이름(영문)이다. 페이스북은 Numeric ID 형식의 page id 를 가지고 타임라인을 요청하여야 하기 때문에 변환 과정을 거쳐야 하나, 트위터는 편하게 screen_name 을 가지고 요청이 가능하다.

수신받은 데이터는 페이스북과 마찬가지로 'utf-8'로 인코딩 되어있는 바이너리 JSON 데이터 형식이여서 'utf-8'로 디코딩 한 후 반환한다.

```
#[CODE 3] 사용자의 timeline 트윗을 수신

def getTwitterTwit(tweet, jsonResult):

    tweet_id = tweet['id_str']
    tweet_message = '' if 'text' not in tweet.keys() else tweet['text']

    screen_name = '' if 'user' not in tweet.keys() else tweet['user']['screen_name']

    tweet_link = ''
    if tweet['entities']['urls']: #list

        for i, val in enumerate(tweet['entities']['urls']):
            tweet_link = tweet_link + tweet['entities']['urls'][i]['url'] + ' '
    else:
        tweet_link = ''

    hashtags = ''
    if tweet['entities']['hashtags']: #list

        for i, val in enumerate(tweet['entities']['hashtags']):
            hashtags = hashtags + tweet['entities']['hashtags'][i]['text'] + ' '
    else:
        hashtags = ''

    if 'created_at' in tweet.keys():
        # Twitter used UTC Format. EST = UTC + 9(Korean Time) Format ex: Fri Feb 10 03:57:27 +0000 2017

        tweet_published = datetime.datetime.strptime(tweet['created_at'],'%a %b %d %H:%M:%S +0000 %Y')
        tweet_published = tweet_published + datetime.timedelta(hours=+9)
        tweet_published = tweet_published.strftime('%Y-%m-%d %H:%M:%S')
    else:
        tweet_published = ''

    num_favorite_count = 0 if 'favorite_count' not in tweet.keys() else tweet['favorite_count']
    num_comments = 0
    num_shares = 0 if 'retweet_count' not in tweet.keys() else tweet['retweet_count']
    num_likes = num_favorite_count
    num_loves = num_wows = num_hahas = num_sads = num_angrys = 0

    jsonResult.append({'post_id':tweet_id, 'message':tweet_message,
                       'name':screen_name, 'link':tweet_link,
```

```
'created_time':tweet_published, 'num_reactions':num_favorite_count,
'num_comments':num_comments, 'num_shares':num_shares,
'num_likes':num_likes, 'num_loves':num_loves,
'num_wows':num_wows, 'num_hahas':num_hahas,
'num_sads':num_sads, 'num_angrys':num_angrys, 'hashtags': hashtags})
```

[CODE 3]은 수신한 JSON 형식에서 우리가 원하는 데이터의 값을 키값과 그에 해당하는 데이터값을 조회한다. 페이스북은 reaction 이라는 레코드셋을 가지고 “좋아요, 사랑해요” 등의 리액션 데이터를 가지고 있지만 트위터는 단순하게 ‘좋아요’의 데이터만 가지고 있다. [그림 4]는 트위터의 JSON 구성 형태를 알아보기 편하도록 작성된 맵이다.



[그림 4] 트위터 Status 객체 구성 요소

트위터에서는 API 를 활용하기 위한 다양한 페이지 URL(endpoint)를 제공한다(코드에서 사용한 node 부분).

용도	Endpoint
내 타임라인	/statuses/home_timeline.json allowed_param:'since_id', 'max_id', 'count'
특정 사용자 타임라인	/statuses/user_timeline.json allowed_param:'id', 'user_id', 'screen_name', 'since_id'
맨션 타임라인	/statuses/mentions_timeline.json allowed_param:'since_id', 'max_id', 'count'
나를 리트윗한 정보	/statuses/retweets_of_me.json allowed_param:'since_id', 'max_id', 'count'
사용자 정보	/users/show.json allowed_param:'id', 'user_id', 'screen_name'
사용자 검색	/users/search.json allowed_param:'q', 'count', 'page'
팔로워	/followers/list.json allowed_param: 'id', 'user_id', 'screen_name', 'cursor', 'count', 'skip_status', 'include_user_entities'
검색	/search/tweets.json allowed_param: 'q', 'lang', 'locale', 'since_id', 'geocode', 'max_id', 'since', 'until', 'result_type', 'count', 'include_entities', 'from', 'to', 'source'

트위터는 큰 결과 세트를 검색하기 위하여 페이징 기법으로 커서링(cursoring)이라는 기법을 사용한다. 먼저 처음에는 cursor 의 값을 '-1'로 지정하여 결과를 검색하면 'previous_cursor', 'next_cursor', 'previous_cursor_str' 및 'next_cursor_str'을 반환한다. [그림 4]에서 JSON 구조를 잘 살펴 보았다면 트위터는 숫자문자열의 경우에는 항상 해당 '키'값과 '키_str'을 제공하는 것을 확인할 수 있다.

다음 페이지를 얻기 위하여 우리는 'cursor'에 'next_cursor'의 값을 전달하여 페이지를 요청하면 된다. 만약 'next_cursor'의 값이 '0'으로 반환되면 더 이상의 페이지는 없는 것을 의미한다(페이스북에서는 다음 페이지를 요청하는 URL 을 보내주므로 단순하게 해당 URL 을 호출하는 것으로 데이터를 획득할 수 있었다). 또한 페이스북의 since 와 until 의 개념은 트위터에서는 Search API 에서 사용되며, 'statuses'에서는 지원하지 않는다. 자세한 부분은 Search API 를 참조하기 바란다.