API Blender

A Uniform Interface to Social Platform APIs

{georges.gouriten, pierre.senellart}@telecom-paristech.fr
Outline

Interacting with social platforms ...

is usually painful, ...

but API Blender makes it easy!
Generic interactions with a Web API

Authenticate → Make request → Process response

Make request → Parse response

Parse response → Make request
Generic interactions with a Web API
flickr.photos.search

Return a list of photos matching some criteria. Only photos visible to the calling user will be returned. To return private or semi-private photos, the caller must be authenticated with 'read' permissions, and have permission to view the photos. Unauthenticated calls will only return public photos.

Authentication

This method does not require authentication.

Arguments

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

user_id (Optional)
The NSID of the user who's photo to search. If this parameter isn't passed then everybody's public photos will be searched. A value of "me" will search against the calling user's photos for authenticated calls.

tags (Optional)
A comma-delimited list of tags. Photos with one or more of the tags listed will be returned. You can exclude results that match a term by prepending it with a - character.

tag_mode (Optional)
Either 'any' for an OR combination of tags, or 'all' for an AND combination. Defaults to 'any' if not specified.

text (Optional)
A free text search. Photos who's title, description or tags contain the text will be returned. You can exclude results that match a term by prepending it with a - character.

min_upload_date (Optional)
Minimum upload date. Photos with an upload date greater than or equal to this value will be returned. The date can be in the form of a unix timestamp or mysql datetime.

max_upload_date (Optional)
Maximum upload date. Photos with an upload date less than or equal to this value will be returned. The date can be in the form of a unix timestamp or mysql datetime.
Graph API
Core Concepts > Graph API

At Facebook’s core is the social graph; people and the connections they have to everything they care about. The Graph API presents a simple, consistent view of the Facebook social graph, uniformly representing objects in the graph (e.g., people, photos, events, and pages) and the connections between them (e.g., friend relationships, shared content, and photo tags).

Every object in the social graph has a unique ID. You can access the properties of an object by requesting https://graph.facebook.com/ID. For example, the official page for the Facebook Platform has id 19292868552, so you can fetch the object at https://graph.facebook.com/19292868552:

```json
{
    "name": "Facebook Platform",
    "website": "http://developers.facebook.com",
    "username": "platform",
    "founded": "May 2007",
    "company_overview": "Facebook Platform enables anyone to build...",
    "mission": "To make the web more open and social.",
    "products": "Facebook Application Programming Interface (API)...",
    "likes": 449921,
    "id": 19292868552,
    "category": "Technology"
}
```
Heterogeneous
Sometimes obscure
Error-prone
Generic interaction with a Web service
Home-made development

What most people do

Redundant efforts
What about collective efforts?

**Top-down**

- WSDL
- WADL

  Error 404

**Bottom-up**

- Spring Social
- SPORE-DL

  Promising direction
API Blender

Social Web APIs description files

Python implementation
API Blender

Social Web APIs description files

Python implementation
Lightweight social Web API description format

{
  "name": "twitter-search",
  "host": "search.twitter.com",
  "port": 80,
  "authentication": authentication_object,
  "policy": policy_object,
  "interactions": [interaction_object]
}
Simple Authentication

```json
{
  "type": "simple",
  "path": "/oauth/access_token",
  "parameters": {
    "client_id": "x2836",
    "client_secret": "z45725",
    "grant_type": "client_credentials"
  }
}
```
Oauth Authentication

{
    "type": "oauth",
    "consumer_key": "x4565482s",
    "consumer_secret": "tsouintsouin",
    "request_token_url": "https://api.twitt...",
    "access_token_url": "https://api.twitt...",
    "authorize_url": "https://api.twitt..."
}
Policy

{
    "active": true,
    "requests_per_hour": 400,
    "too_many_calls_response_code": 420,
    "too_many_calls_waiting_seconds": 600
}
Interaction

```json
{
  "name": "search",
  "description": "search for tweets",
  "request": {
    "url_root_path": "/search.json",
    "method": "GET",
    "url_params": [
      [ "q", "string", false, null ],
      [ "rpp", "int", true, 100 ]
    ]
  },
  "response": {
    "expected_status_code": 200,
    "serialization_format": "JSON",
    "extractor": extractor_object
  }
}
```
Extractor

{
  "data.field1": "mydata.there.fieldx",
  "data.field2": "mydata.there.fieldy"
}

{  "data.field1": "mydata.there.fieldx",
  "data.field2": "mydata.there.fieldy" }

{  "data": {  "field1": <field1_value>,
    "field2": <field2_value>  }
  }

{  "mydata": {  "there": {  "fieldx": <field1_value>
    "fieldy": <field2_value>  }
  }
  }

"mydata": {  "there": {  "fieldx": <field1_value>
    "fieldy": <field2_value>  }
  }
}
results": { 
  "from_user": ...
 }

"posts": { 
  "user_name": ...
}

"data": { 
  "from": { 
    "name": ...
  }
}
Lightweight social Web API description format

{
  "name": "twitter-search",
  "host": "search.twitter.com",
  "port": 80,
  "authentication": authentication_object,
  "policy": policy_object,
  "interactions": [interaction_object]
}
API Blender

Social Web APIs description files

Python implementation
How the apiblender works (1)

```python
import apiblender

blender = apiblender.Blender()

blender.list_servers().load_server('flickr')
  .list_interactions().load_interaction('photos_search')
```
How the apiblender works (2)

```python
blender.list_url_params()
blender.set_url_params({'tags': 'good spirit'})

result = blender.blend()

result['raw_content']
result['prepared_content']
result['headers']
```
Authenticate

Ensure authentication

Respect policy

Make request

Parse response

Process response

Generic interaction with a Web service
Example
interactions made easy (1)

```python
blender.load_server("twitter-search")
blender.load_interaction("search")

users = set()
for p in range(1, 3):
    blender.set_url_params({"q": 'good spirit',
                           "page": p})
    response = blender.blend()
    results = response["raw_content"]['results']
    for twitt in results:
        users.add(twitt['from_user'])
```
Interactions made easy (2)

```python
blender.load_server("twitter-generic")

for user in users:
    blender.load_interaction('followers')
    blender.set_url_params({"screen_name": user})
    followers = blender.blend()
```
demo
Summary

Policy management
Persistent authentication
Data integration
User-friendly interactions
chaining
Coming soon

> 5 APIs (and more methods)

Request chaining libraries
It's open source!

github.com/netiru/apiblender/