VisualSnoop Client module for Python Documentation

Release 0.2

Janoš Guljaš

January 30, 2015

Contents

Release v0.2.

This Python module provides a client for VisualSnoop HTTP API services.

VisualSnoop.com is a Web service for private image searching based on visual similarity. Each user can create personal collections of images and use them as a base for finding duplicate or similar images.

CHAPTER 1

Installation

1.1 Distribute & Pip

Installing VisualSnoop Client is simple with pip, just run this in your terminal:

```
$ pip install visualsnoop
```

1.2 Get the Code

VisualSnoop Client is developed on GitHub, where the code is always available.

You can either clone the public repository:

```
$ git clone git://github.com/visualsnoop/visualsnoop-client-python.git
```

Download the tarball:

```
$ curl -OL https://github.com/visualsnoop/visualsnoop-client-python/tarball/master
```

Or, download the zipball:

```
$ curl -OL https://github.com/visualsnoop/visualsnoop-client-python/zipball/master
```

Once you have a copy of the source, you can embed it in your Python package, or install it into your site-packages easily:

```
$ python setup.py install
```

CHAPTER 2

Usage

Register an account on VisualSnoop.com, create a collection and prepare `public` and `secret` keys.

2.1 Connecting

Import `visualsnoop.models.Collection`:

```
>>> from visualsnoop.models import Collection
```

Initialize a session to interact with that collection using API access keys:

```
>>> collection = Collection(public_key='24d337fe0ed9427cb512', secret_key='sUGSL96ZvNxqHixSPUopX5L1Mt
```

Note: Values in this example are not valid. Use keys for your own collection.

To verify collection attributes, check properties `id`, `name` and/or `user`:

```
>>> collection.id
'6b7c9142'
>>> collection.name
'Example collection'
>>> collection.user
'user@example.com'
```

2.2 Adding images to collection

Adding a new image to the collection:

```
>>> collection.add_image(image='example1.jpg')
{
    'image_time': '2015-01-25T20:43:05.690218+00:00',
    'collection_id': '6b7c9142',
    'image_id': 'c14d4bae93a95c159360e29de998181b',
    'message': 'Added',
    'status': 201
}
```

Make sure to save the `image_id` value as it is the reference to the actual image. If you want to specify your own unique ID, for example an ID from your local database provide it within the call:

```
>>> collection.add_image(image='example2.jpg', image_id='myid12')
{
    'image_time': '2015-01-25T20:47:10.021689+00:00',
    'collection_id': '6b7c9142',
    'image_id': 'myid12',
    'message': 'Added',
    'status': 201
}
```

Valid image IDs are any strings up to 100 characters long, but integers are most often used for referencing.

In case that you want to update your image with a fresh data, set update_existing to True:

```
>>> collection.add_image(image='example3.jpg', image_id='c14d4bae93a95c159360e29de998181b', update_ex
{
    'image_time': '2015-01-25T20:50:36.384015+00:00',
    'collection_id': '6b7c9142',
    'image_id': 'c14d4bae93a95c159360e29de998181b',
    'message': 'Updated',
    'status': 200
}
```

Argument `update_existing` is by default set to False, to prevent accidental image overwriting.

2.3 Searching

Searching can be performed in two ways:

- By ID of the image that is already in collection
- By sending image data of a local image

```
>>> collection.search_images_by_id(image_id='c14d4bae93a95c159360e29de998181b')
{
    'message': 'OK',
    'status': 200,
    'image_time': '2015-01-25T20:50:36.384015+00:00',
    'collection_id': '6b7c9142',
    'results': [],
    'image_id': 'c14d4bae93a95c159360e29de998181b'
}

>>> collection.search_images(image='example3.jpg')
{
    'collection_id': '6b7c9142',
    'results': [{'id': 'c14d4bae93a95c159360e29de998181b', 'rank': 0}],
    'message': 'OK',
    'status': 200
}
```

2.4 Verifying images by ID

```
>>> collection.get_image(image_id='myid12')
{
    'image_time': '2015-01-25T20:47:10.021689+00:00',
```

```
    'collection_id': '6b7c9142',
    'image_id': 'myid12',
    'message': 'OK',
    'status': 200
}
>>> collection.get_images()
{
    'collection_id': '6b7c9142',
    'status': 200,
    'message': 'OK',
    'images': [
        {
            'image_time': '2015-01-25T20:50:36.384015+00:00',
            'id': 'c14d4bae93a95c159360e29de998181b'
        },
        {
            'image_time': '2015-01-25T20:47:10.021689+00:00',
            'id': 'myid12'
        }
    ],
    'next_image_id': None
}
```

2.5 Deleting images by ID

```
>>> collection.delete_image(image_id='myid12')
{
    'image_time': '2015-01-25T20:47:10.021689+00:00',
    'collection_id': '6b7c9142',
    'image_id': 'myid12',
    'message': 'Deleted',
    'status': 200
}
```

CHAPTER 3

Module API Documentation

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

3.1 Main model

class `visualsnoop.models.`**`Collection`**(*public_key*, *secret_key*, *endpoint='http://visualsnoop.com/api/v1'*, *proxies=None*, *max_retries=2*)

Represents a VisualSnoop Collection identified by `public_key`.

Parameters

- **public_key** – Public key generated for a collection on VisualSnoop Keys page.
- **secret_key** – Secret key associated with the public key.
- **endpoint** – Base URL for the API calls. It must be an valid URL including protocol, domain and path.
- **proxies** – Python Requests compatible proxies dictionary.
- **max_retries** (*int*) – The maximum number of retries each connection should attempt. Note, this applies only to failed DNS lookups, socket connections and connection timeouts, never to requests where data has made it to the server. By default, Requests does not retry failed connections.

`add_image`(*image*, *image_id=None*, *update_existing=False*, *timeout=None*)

Adds `image` to collection, in overwrite-safe manner.

This method sends image data. Use argument `image_id` to specify your own ID for the image. If `image_id` is `None` a unique ID will be generated and returned in the response as a value for key `image_id`. Make sure to save the `image_id` value from the API response.

The image with `image_id` will not be overwritten if parameter `update_existing` is False (default). To update an existing image with `image_id`, set `update_existing` to True.

Parameters

- **image** (*file or str*) – A file object with image data or a path to a file on filesystem.
- **image_id** – (optional) A unique ID of an image. It will be self generated if omitted.
- **update_existing** (*bool*) – A flag whether to allow overwriting existing images.
- **timeout** (*float or tuple*) – (optional) How long to wait for the server to send data before giving up, as a float, or a Python Requests timeout tuple.

delete_image(*image_id*, *timeout=None*)
Remove image with `image_id` from collection.

Parameters

- **image_id** – ID of an image previously added to the collection.
- **timeout** (*float or tuple*) – (optional) How long to wait for the server to send data before giving up, as a float, or a Python Requests timeout tuple.

get_image(*image_id*, *timeout=None*)
Retrieve information about indexed image with `image_id`.

Parameters

- **image_id** – ID of an image previously added to the collection.
- **timeout** (*float or tuple*) – (optional) How long to wait for the server to send data before giving up, as a float, or a Python Requests timeout tuple.

get_images(*start=None*, *count=None*, *timeout=None*)
List images in collection sorted by time.

Parameters

- **start** – Image ID from which to start the listing.
- **count** – Number of images in listing.

search_images(*image*, *timeout=None*)
Perform search over collection based on `image` sent in the request body.

Parameters

- **image** (*file or str*) – A file object with image data or a path to a file on filesystem.
- **timeout** (*float or tuple*) – (optional) How long to wait for the server to send data before giving up, as a float, or a Python Requests timeout tuple.

search_images_by_id(*image_id*, *timeout=None*)
Perform search over collection based on image within the same collection with ID `image_id`.

Parameters

- **image_id** – ID of an image previously added to the collection.
- **timeout** (*float or tuple*) – (optional) How long to wait for the server to send data before giving up, as a float, or a Python Requests timeout tuple.

3.2 Python Requests Authentication class

class `visualsnoop.auth.`**VisualSnoopHMACAuth**(*public_key*, *secret_key*)
Attaches VisualSnoop specific HTTP HMAC Authentication to the given Request object.

Parameters

- **public_key** – Public key generated for a collection on VisualSnoop Keys page.
- **secret_key** – Secret key associated with the public key.

3.3 Exception class

exception visualsnoop.exceptions.**VisualSnoopError**
A VisualSnoop error occurred.

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*

V

A

C

D

G

S

V