# spotifylib Documentation

***Release 0.1.1***

**Oriol Fabregas**

**Oct 09, 2017**

# Contents

Contents:

CHAPTER 1

# spotifylib

Spotify API Client

This library aims to implement Spotify's Authorization flow without the user needing to create a third party application to authorize the application, redirect it to the callback and then manually authorize it with username and password.

This library goal is to make authorization transparent but using Spotipy's functionality. It is implemented in a non-standard way that Spotify wouldn't recommend so we can't guarantee this would work forever.

Read more on USAGE.rst or Read the docs or check the code for substantial docstrings.

## Features

- Same features as Spotipy's library but with transparent authentication
- Renew's the token transparently

CHAPTER 2

# Installation

At the command line:

```
$ pip install spotifylib
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv spotifylib
$ pip install spotifylib
```

CHAPTER 3

# Usage

To use spotifylib in a project:

## Instructions

Go to your account's developer site and create an application. Give it a name and get the *Client ID*, *Client Secret* and provide a *Redirect URI - http://127.0.0.1/callback* would just work.

You will also need to use a *scope* to get a token that has access to the resources. Read more about scopes here

Scopes can be appended by using a white space. Let's assume we will use *playlist-modify-public playlist-modify-private* as *scope*.

```
$ pip install spotifylib
```

```
from spotifylib import Spotify
import os

spotify = Spotify(client_id=os.environ.get('CLIENT_ID'),
                  client_secret=os.environ.get('CLIENT_SECRET'),
                  username=os.environ.get('USERNAME'),
                  password=os.environ.get('PASSWORD'),
                  callback=os.environ.get('CALLBACK_URL'),
                  scope=os.environ.get('SCOPE'))
print(spotify.user_playlists(os.environ.get('USERNAME')))
```

Your linked app can then be found under user's profile

CHAPTER 4

# Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

## Submit Feedback

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.

### Get Started!

Ready to contribute? Here's how to set up *spotifylib* for local development.

1. Clone your fork locally:

```
$ git clone git@spotifylib
```

2. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your clone for local development:

```
$ mkvirtualenv spotifylib
$ cd spotifylib/
$ python setup.py develop
```

3. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. Commit your changes and push your branch to the server:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

5. Submit a merge request

CHAPTER 5

spotifylib

# spotifylib package

## Submodules

## spotifylib.constants module

defines constants

Static URL's and variables

## spotifylib.spotifylib module

This module makes use of Spotipy's methods but modifying the authentication in a simple and transparent way from the user without any need of 3rd party application to follow the OAuth flow as mentioned in the following documentation page.

https://developer.spotify.com/web-api/authorization-guide/#authorization_code_flow

**class** spotifylib.spotifylib.**Spotify**

Bases: object

Library's interface object

Instantiates the authentication object to figure out the token and passes it alongside the session to Spotipy's in order to use its methods.

**class** spotifylib.spotifylib.**SpotifyAuthenticator**(*client_id*, *client_secret*, *username*, *password*, *callback*, *scope*)

Bases: object

Authenticator object

This object handles authentication for all requests. In order to retrieve all values for this to work, one has to create a new application under his/her account.

https://developer.spotify.com/my-applications/#!/applications

**token**

**class** spotifylib.spotifylib.**Token**(*access_token*, *token_type*, *expires_in*, *refresh_token*, *scope*)
Bases: tuple

**access_token**
Alias for field number 0

**expires_in**
Alias for field number 2

**refresh_token**
Alias for field number 3

**scope**
Alias for field number 4

**token_type**
Alias for field number 1

**class** spotifylib.spotifylib.**User**(*client_id*, *client_secret*, *username*, *password*)
Bases: tuple

**client_id**
Alias for field number 0

**client_secret**
Alias for field number 1

**password**
Alias for field number 3

**username**
Alias for field number 2

## spotifylib.spotifylibexceptions module

Main module Exceptions file

Put your exception classes here

**exception** spotifylib.spotifylibexceptions.**SpotifyError**
Bases: exceptions.Exception

# Wrong client_id (<Response [400]>, 'INVALID_CLIENT: Invalid client')

# Wrong response_type (<Response [400]>, 'response_type must be code or token')

# Invalid scope (<Response [400]>, 'INVALID_SCOPE: Invalid scope')

# Invalid CSRF cookie (<Response [400]>, '{"error":"errorCSRF"}')

# Invalid redirect_uri (<Response [400]>, 'Illegal redirect_uri') "Error while accepting APP to Spotify API"

## Module contents

spotifylib package

Imports all parts from spotifylib here

CHAPTER 6

# Credits

## Development Lead

- Costas Tyfoxylos <https://github.com/costastf>

## Contributors

- Oriol Fabregas <fabregas.oriol@gmail.com>

CHAPTER 7

# History

# CHAPTER 8

## 0.1 (18-09-2017)

- First release

# CHAPTER 9

## 0.1.1 (09-10-2017)

- Docstrings

CHAPTER 10

# Indices and tables

- genindex
- modindex
- search
- search

# Python Module Index

## S

# Index