Listen Documentation

Release 0.2.0

Byron Vanstien

Jul 26, 2017

Contents

1 About **1**

2 Installation **3**

3 Contents **5**

3.1 Full Listen API reference 5
3.2 Find out your version 5
3.3 Client 5
3.4 DataClasses 6
3.5 Exceptions 7
3.6 Examples of Listen API usage 7
3.7 FAQ 7

CHAPTER 1

About

Listen is a fully asynchronous wrapper around the Listen.moe API that's built upon the websockets lib and aiohttp. The official supported versions of Python for this library are anything onwards from 3.5

CHAPTER 2

Installation

```
$ pip install git+https://github.com/GetRektByMe/Listen.git
```

Master branch should always work, you can install dev branches by choice. **These may be broken**.

CHAPTER 3

Contents

Full Listen API reference

Note: Logging should be implemented on your end

Note: A feature that will get previous song names may be added at some point

Find out your version

`listen.__version__`
A string that represents the version of Listen you're using.

Client

class `listen.client.Client`(*loop: asyncio.base_events.BaseEventLoop = None*)
Client class to interface with listen.moe's API

`create_websocket_connection`(*authenticate: bool = False*)
This function is a coroutine

Creates a websocket connection to Listen.moe's socket API

Parameters **authenticate** (*bool*) – Boolean that decides if the authentication to the API is done

Return type None

favorite_toggle(*song_id: int*)
This function is a coroutine

Either favourites or unfavourites a song based on if it's favourited already

Parameters **song_id** (*int*) – The id of the song you want to favourite

Return type bool

get_favorites()
This function is a coroutine

Get all favourites from the current logged in user

Return type `listen.objects.Song`

get_info()
This function is a coroutine

Get a user object that is associated with the logged in user

Return type `listen.objects.User`

get_token(*username: str*, *password: str*)
This function is a coroutine

Get user token for the account you're using to sign in

Parameters

- **username** (*str*) – The username of the account you're getting the token for
- **password** (*str*) – The password of the account you're getting the token for

Return type str

make_request(*song_id: int*)
This function is a coroutine

Requests a song for queueing

Parameters **song_id** (*int*) – The id of the song you want to request

Return type bool

register_handler(*handler*)
Registers a function handler to allow you to do something with the socket API data

Parameters **handler** (*function*) – A function that takes a dictionary that contains Listen.moe's API data

run()
Start the connection to the socket API

DataClasses

Note: These are not for creating yourself, they are handed to you from the library backend

class `listen.`**Song**(*id: int*, *artist: str*, *title: str*, *anime: str*, *enabled: bool*)

class `listen.`**User**(*id: int*, *username: str*)

Exceptions

Note: `listen.Client.start()` throws RuntimeError if the absence of a handler is spotted.

exception `listen.`**`KanaError`**

exception `listen.`**`ListenError`**

Examples of Listen API usage

Coming when people actually use this library

FAQ

Coming when people actually use this library

Index

Symbols

__version__ (in module listen), 5

C

Client (class in listen.client), 5
create_websocket_connection() (listen.client.Client method), 5

F

favorite_toggle() (listen.client.Client method), 5

G

get_favorites() (listen.client.Client method), 6
get_info() (listen.client.Client method), 6
get_token() (listen.client.Client method), 6

K

KanaError, 7

L

ListenError, 7

M

make_request() (listen.client.Client method), 6

R

register_handler() (listen.client.Client method), 6
run() (listen.client.Client method), 6

S

Song (class in listen), 6

U

User (class in listen), 6