topggpy Documentation

Release 1.4.0

Assanali Mukhanov

Nov 22, 2021

CONTENTS

CHAPTER

ONE

API REFERENCE

The following section outlines the API of topggpy.

1.1 Version Related Info

There are two main ways to query version information about the library.

`topgg.`**`version_info`**

A named tuple that is similar to `sys.version_info`.

Just like `sys.version_info` the valid values for `releaselevel` are 'alpha', 'beta', 'candidate' and 'final'.

`topgg.`**`__version__`**

A string representation of the version. e.g. `'0.1.0'`.

1.2 Client

`class topgg.`**`DBLClient`**(*bot: discord.client.Client*, *token: str*, *autopost: bool = False*, *post_shard_count: bool = False*, *autopost_interval: Optional[float] = None*, ***kwargs: Any*)

Represents a client connection that connects to Top.gg. This class is used to interact with the Top.gg API.

Parameters

- **`bot`** (*`discord.Client`*) – An instance of a discord.py Client object.
- **`token`** (*`str`*) – Your bot's Top.gg API Token.
- **`autopost`** (*`bool`*) – Whether to automatically post bot's guild count every 30 minutes. This will dispatch *`on_autopost_success()`* (or *`on_autopost_error()`* in case of an error).
- **`post_shard_count`** (*`bool`*) – Whether to post the shard count on autopost. Defaults to False.
- **`autopost_interval`** (*`Optional[int]`*) – Interval used by autopost to post server count automatically, measured in seconds. Defaults to 1800 (30 minutes) if autopost is True, otherwise None.
- **`**session`** (`aiohttp.ClientSession`) – An aiohttp session to use for requests to the API.

`property` **`guild_count`**: `int`

Gets the guild count from the provided Client object.

`async` **`get_weekend_status`**() → bool

This function is a coroutine.

Gets weekend status from Top.gg.

Returns weekend status – The boolean value of weekend status.

Return type bool

async post_guild_count(*guild_count: Optional[Union[int, List[int]]] = None*, *shard_count: Optional[int] = None*, *shard_id: Optional[int] = None*) → None

This function is a coroutine.

Posts your bot's guild count and shards info to Top.gg.

Parameters

- **guild_count** (`Optional[Union[int, List[int]]]`) – Number of guilds the bot is in. Applies the number to a shard instead if shards are specified. If not specified, length of provided client's property *.guilds* will be posted.
- **shard_count** (`Optional[int]`) – The total number of shards.
- **shard_id** (`Optional[int]`) – The index of the current shard. Top.gg uses 0 based indexing for shards.

async get_guild_count(*bot_id: Optional[int] = None*) → *topgg.types.BotStatsData*

This function is a coroutine.

Gets a bot's guild count and shard info from Top.gg.

Parameters bot_id (`int`) – ID of the bot you want to look up. Defaults to the provided Client object.

Returns stats – The guild count and shards of a bot on Top.gg.

Return type *BotStatsData*

async get_bot_votes() → List[*topgg.types.BriefUserData*]

This function is a coroutine.

Gets information about last 1000 votes for your bot on Top.gg.

Note: This API endpoint is only available to the bot's owner.

Returns users – Users who voted for your bot.

Return type List[*BriefUserData*]

async get_bot_info(*bot_id: Optional[int] = None*) → *topgg.types.BotData*

This function is a coroutine.

Gets information about a bot from Top.gg.

Parameters bot_id (`int`) – ID of the bot to look up. Defaults to the provided Client object.

Returns bot info – Information on the bot you looked up. Returned data can be found here.

Return type *BotData*

async get_bots(*limit: int = 50*, *offset: int = 0*, *sort: Optional[str] = None*, *search: Optional[Dict[str, Any]] = None*, *fields: Optional[List[str]] = None*) → *topgg.types.DataDict*[str, Any]

This function is a coroutine.

Gets information about listed bots on Top.gg.

Parameters

- **`limit`** (*`int`*) – The number of results to look up. Defaults to 50. Max 500 allowed.
- **`offset`** (*`int`*) – The amount of bots to skip. Defaults to 0.
- **`sort`** (*`str`*) – The field to sort by. Prefix with `-` to reverse the order.
- **`search`** (*`Dict[str, Any]`*) – The search data.
- **`fields`** (*`List[str]`*) – Fields to output.

Returns **bots** – Info on bots that match the search query on Top.gg.

Return type *DataDict*

async `get_user_info`(*user_id: int*) → *topgg.types.UserData*

This function is a coroutine.

Gets information about a user on Top.gg.

Parameters **`user_id`** (*`int`*) – ID of the user to look up.

Returns **user data** – Information about a Top.gg user.

Return type *UserData*

async `get_user_vote`(*user_id: int*) → bool

This function is a coroutine.

Gets information about a user's vote for your bot on Top.gg.

Parameters **`user_id`** (*`int`*) – ID of the user.

Returns **vote status** – Info about the user's vote.

Return type bool

async `generate_widget`(*options:* topgg.types.WidgetOptions) → str

This function is a coroutine.

Generates a Top.gg widget from the provided *WidgetOptions* object.

Parameters **`options`** (*WidgetOptions*) – A *WidgetOptions* object containing widget parameters.

Returns **widget** – Generated widget URL.

Return type str

async `close`() → None

This function is a coroutine.

Closes all connections.

1.3 Event reference

`topgg.`**`on_autopost_success`**()

Called when autopost posts server count successfully on Top.gg.

`topgg.`**`on_autopost_error`**(*exception*)

Called when autopost raises an exception during server count posting.

Parameters **`exception`** (*`Exception`*) – The raised exception object.

`topgg.`**`on_dbl_vote`**(*data*)

Called when someone votes for your bot on Top.gg.

Parameters **`data`** (*BotVoteData*) – The data model containing bot vote information.

Example:

```
@bot.event
async def on_dbl_vote(data):
    print(data)
```

`topgg.on_dsl_vote`(*data*)

Called when someone votes for your server on Top.gg.

Parameters `data` (*ServerVoteData*) – The data model containing server vote information.

Example:

```
@bot.event
async def on_dsl_vote(data):
    print(data)
```

CHAPTER

TWO

DATA MODELS

This section explains data models used in topggpy to represent data received from Top.gg.

Note: All listed models subclass `dict` and allow retrieving data via attributes *and* keys (i.e., both `response['id']` and `response.id` are valid).

2.1 DataDict

`class topgg.types.DataDict`(***kwargs: topgg.types.VT*)

Bases: `dict`, `MutableMapping`[`topgg.types.KT`, `topgg.types.VT`]

Base class used to represent received data from the API.

Every data model subclasses this class.

2.2 BotData

`class topgg.types.BotData`(***kwargs: Any*)

Bases: *`topgg.types.DataDict`*[`str`, `Any`]

Model that contains information about a listed bot on top.gg. The data this model contains can be found here.

2.3 UserData

`class topgg.types.UserData`(***kwargs: Any*)

Bases: *`topgg.types.DataDict`*[`str`, `Any`]

Model that contains information about a top.gg user. The data this model contains can be found here.

2.4 SocialData

`class topgg.types.SocialData(**kwargs: topgg.types.VT)`

Bases: `topgg.types.DataDict[str, str]`

Model that contains social information about a top.gg user.

youtube: str
The YouTube channel ID of the user.

reddit: str
The Reddit username of the user.

twitter: str
The Twitter username of the user.

instagram: str
The Instagram username of the user.

github: str
The GitHub username of the user.

2.5 BriefUserData

`class topgg.types.BriefUserData(**kwargs: Any)`

Bases: `topgg.types.DataDict[str, Any]`

Model that contains brief information about a Top.gg user.

id: int
The Discord ID of the user.

username: str
The Discord username of the user.

avatar: str
The Discord avatar URL of the user.

2.6 BotStatsData

`class topgg.types.BotStatsData(**kwargs: Any)`

Bases: `topgg.types.DataDict[str, Any]`

Model that contains information about a listed bot's guild and shard count.

server_count: Optional[int]
The amount of servers the bot is in.

shards: List[int]
The amount of servers the bot is in per shard.

shard_count: Optional[int]
The amount of shards a bot has.

2.7 VoteDataDict

class `topgg.types.`**`VoteDataDict`**`(**kwargs: Any)`

Bases: `topgg.types.DataDict[str, Any]`

Base model that represents received information from Top.gg via webhooks.

type: `str`

Type of the action (`upvote` or `test`).

user: `int`

ID of the voter.

query: `topgg.types.DataDict`

Query parameters in *DataDict*.

2.8 BotVoteData

class `topgg.types.`**`BotVoteData`**`(**kwargs: Any)`

Bases: `topgg.types.VoteDataDict`

Model that contains information about a bot vote.

bot: `int`

ID of the bot the user voted for.

is_weekend: `bool`

Boolean value indicating whether the action was done on a weekend.

2.9 ServerVoteData

class `topgg.types.`**`ServerVoteData`**`(**kwargs: Any)`

Bases: `topgg.types.VoteDataDict`

Model that contains information about a server vote.

guild: `int`

ID of the guild the user voted for.

2.10 WidgetOptions

class `topgg.types.`**`WidgetOptions`**`(id: Optional[int] = None, format: Optional[str] = None, type: Optional[str] = None, noavatar: bool = False, colors: Optional[Dict[str, int]] = None, colours: Optional[Dict[str, int]] = None)`

Bases: `topgg.types.DataDict[str, Any]`

Model that represents widget options that are passed to Top.gg widget URL generated via `DBLClient.generate_widget()`.

id: `Optional[int]`

ID of a bot to generate the widget for. Must resolve to an ID of a listed bot when converted to a string.

`colors: Dict[str, int]`
: A dictionary consisting of a parameter as a key and HEX color (type *int*) as value. `color` will be appended to the key in case it doesn't end with `color`.

`noavatar: bool`
: Indicates whether to exclude the bot's avatar from short widgets. Must be of type `bool`. Defaults to `False`.

`format: str`
: Format to apply to the widget. Must be either `png` and `svg`. Defaults to `png`.

`type: str`
: Type of a short widget (`status`, `servers`, `library`, `upvotes`, and `owner`). For large widget, must be an empty string.

CHAPTER

THREE

WEBHOOKS

Attention: In order for webhooks to work, the port you provide to `WebhookManager.run()` must be accessible, meaning your firewall must allow incoming requests to it.

Note: `WebhookManager` exposes the internal webserver instance via the `WebhookManager.webserver` property.

3.1 WebhookManager

class topgg.**WebhookManager**(*bot: discord.client.Client*)

This class is used as a manager for the Top.gg webhook.

Methods `WebhookManager.dbl_webhook()` and `WebhookManager.dsl_webhook()` return a modified version of the object, allowing for method chaining.

Parameters **bot** (`discord.Client`) – The Client object that will be utilized by this manager's webhook(s) to emit events.

dbl_webhook(*route: str = '/dbl', auth_key: str = ''*) → *topgg.webhook.WebhookManager*

Helper method that configures a route that listens to bot votes.

Parameters

- **route** (`str`) – The route to use for bot votes. Must start with `/`. Defaults to `/dbl`.
- **auth_key** (`str`) – The Authorization key that will be used to verify the incoming requests. All requests are allowed if this is not set.

Returns **Modified version of the object**

Return type *WebhookManager*

dsl_webhook(*route: str = '/dsl', auth_key: str = ''*) → *topgg.webhook.WebhookManager*

Helper method that configures a route that listens to server votes.

Parameters

- **route** (`str`) – The route to use for server votes. Must start with `/`. Defaults to `/dsl`.
- **auth_key** (`str`) – The Authorization key that will be used to verify the incoming requests. All requests are allowed if this is not set.

Returns **Modified version of the object**

Return type *WebhookManager*

run(*port: int*) → asyncio.Task[None]

Runs the webhook.

Parameters **port** (*int*) – The port to run the webhook on.

property webserver: aiohttp.web_app.Application

Returns the internal web application that handles webhook requests.

Returns The internal web application.

Return type `aiohttp.web.Application`

async close() → None

Stops the webhook.

3.2 Examples

3.2.1 Helper methods:

```
import discord
import topgg


bot = discord.Client(...)  # Initialize a discord.py client
# WebhookManager helper methods allow method chaining, therefore the lines below are␣
↪valid
bot.topgg_webhook = topgg.WebhookManager(bot)\
                    .dbl_webhook("/dbl", "dbl_auth")\
                    .dsl_webhook("/dsl", "dsl_auth")
```

3.2.2 Via the `webserver` property:

You can utilize the internal `aiohttp.web.Application` via the *`WebhookManager.webserver`* property to add custom routes manually.

```
import discord
import topgg
from aiohttp import web

dbl_auth = "Your DBL Authorization key"  # Leave empty to allow all requests

bot = discord.Client(...)  # Initialize a discord.py client

async def bot_vote_handler(request):
    auth = request.headers.get("Authorization", "")  # Default value will be empty␣
↪string to allow all requests
    if auth == dbl_auth:
        # Process the vote and return response code 2xx
        return web.Response(status=200, text="OK")
    # Return code 401 if authorization fails
    # 4xx response codes tell Top.gg services not to retry the request
```

(continues on next page)

(continued from previous page)

```
    return web.Response(status=401, text="Authorization failed")

bot.topgg_webhook = topgg.WebhookManager(bot)
bot.topgg_webhook.webserver.router.add_post(path="/dbl", handler=bot_vote_handler)
```

CHAPTER
FOUR
EXCEPTIONS

The following exceptions are thrown by the library.

4.1 TopGGException

exception topgg.**TopGGException**

Bases: `Exception`

Base exception class for topggpy.

Ideally speaking, this could be caught to handle any exceptions thrown from this library.

4.2 UnauthorizedDetected

exception topgg.**UnauthorizedDetected**

Bases: *topgg.errors.TopGGException*

Exception that's thrown when no API Token is provided.

4.3 ClientException

exception topgg.**ClientException**

Bases: *topgg.errors.TopGGException*

Exception that's thrown when an operation in the *DBLClient* fails.

These are usually for exceptions that happened due to user input.

4.4 HTTPException

exception topgg.**HTTPException**(*response: ClientResponse*, *message: Union[dict, str]*)

Bases: *topgg.errors.TopGGException*

Exception that's thrown when an HTTP request operation fails.

response

The response of the failed HTTP request.

Type `aiohttp.ClientResponse`

text
The text of the error. Could be an empty string.

Type str

4.5 Unauthorized

exception topgg.**Unauthorized**(*response: ClientResponse*, *message: Union[dict, str]*)
Bases: *topgg.errors.HTTPException*

Exception that's thrown when status code 401 occurs.

4.6 Forbidden

exception topgg.**Forbidden**(*response: ClientResponse*, *message: Union[dict, str]*)
Bases: *topgg.errors.HTTPException*

Exception that's thrown when status code 403 occurs.

4.7 NotFound

exception topgg.**NotFound**(*response: ClientResponse*, *message: Union[dict, str]*)
Bases: *topgg.errors.HTTPException*

Exception that's thrown when status code 404 occurs.

4.8 ServerError

exception topgg.**ServerError**(*response: ClientResponse*, *message: Union[dict, str]*)
Bases: *topgg.errors.HTTPException*

Exception that's thrown when Top.gg returns "Server Error" responses (status codes such as 500 and 503).

CHAPTER

FIVE

WHAT'S NEW

This page keeps a detailed human friendly rendering of what's new and changed in specific versions.

5.1 v1.4.0

- The type of data passed to `on_dbl_vote` has been changed from `dict` to `BotVoteData`
- The type of data passed to `on_dsl_vote` has been changed from `dict` to `ServerVoteData`

5.2 v1.3.0

- Introduced global ratelimiter to follow Top.gg global ratelimits
  - Fixed an `AttributeError` raised by `HTTPClient.request()`
  - Resource-specific ratelimit is now actually resource-specific

5.3 v1.2.0

- Introduced global ratelimiter along with bot endpoints ratelimiter
- Follow consistency with typing in `HTTPClient` and *`DBLClient`* along with updated docstrings (GH-55)

5.4 v1.1.0

- Introduced data models
  - *`DBLClient.get_bot_votes()`* now returns a list of `BriefUserData` objects
  - *`DBLClient.get_bot_info()`* now returns a `BotData` object
  - *`DBLClient.get_guild_count()`* now returns a `BotStatsData` object
  - *`DBLClient.get_user_info()`* now returns a `UserData` object
- *`WebhookManager.run()`* now returns an `asyncio.Task`, meaning it can now be optionally awaited

5.5 v1.0.1

- *`WebhookManager.webserver`* now instead returns `aiohttp.web.Application` for ease of use

5.6 v1.0.0

- Renamed the module folder from `dbl` to `topgg`
- Added `post_shard_count` argument to *`DBLClient.post_guild_count()`*
- Autopost now supports automatic shard posting (GH-42)
- Large webhook system rework, read the *Webhooks* section for more
  - Added support for server webhooks
- Renamed `DBLException` to *`TopGGException`*
- Renamed `DBLClient.get_bot_upvotes()` to *`DBLClient.get_bot_votes()`*
- Added *`DBLClient.generate_widget()`* along with the `widgets` section in the documentation
- Implemented a properly working ratelimiter
- Added *`on_autopost_error()`*
- All autopost events now follow `on_autopost_x` naming format, e.g. *`on_autopost_error()`*, *`on_autopost_success()`*
- Added handlers for autopost args set when autopost is disabled

5.7 v0.4.0

- *`DBLClient.post_guild_count()`* now supports a custom `guild_count` argument, which accepts either an integer or list of integers
- Reworked how shard info is posted
- Removed `InvalidArgument` and `ConnectionClosed` exceptions
- Added `ServerError` exception

5.8 v0.3.3

- Internal changes regarding support of Top.gg migration
- Fixed errors raised when using *`DBLClient.close()`* without built-in webhook

5.9 v0.3.2

- `Client` class has been renamed to `DBLClient`

5.10 v0.3.1

- Added `on_guild_post`, an event that is called when autoposter successfully posts guild count
- Renamed `get_upvote_info` to `get_bot_upvotes`
- Added `get_user_vote`

5.11 v0.3.0

- *`DBLClient`* now has **`autopost`** kwarg that will post server count automatically every 30 minutes
- Fixed code 403 errors
- Added `on_dbl_vote`, an event that is called when you test your webhook
- Added `on_dbl_test`, an event that is called when someone tests your webhook

5.12 v0.2.1

- Added webhook
- Removed support for discord.py versions lower than 1.0.0
- Made *`DBLClient.get_weekend_status()`* return a boolean value
- Added webhook example in README
- Removed `post_server_count` and `get_server_count`

5.13 v0.2.0

- Added `post_guild_count`
  - Made `post_server_count` an alias for `post_guild_count`
  - Added `get_guild_count`
- Made `get_server_count` an alias for `get_guild_count`
- Added *`DBLClient.get_weekend_status()`*
- Removed all parameters from `DBLClient.get_upvote_info()`
- Added limit to *`DBLClient.get_bots()`*
- Fixed example in README

5.14 v0.1.6

- Bug fixes & improvements

5.15 v0.1.4

- Initial ratelimit handling

5.16 v0.1.3

- Added documentation
- Fixed some minor bugs

5.17 v0.1.2

Initial release

- Working
  - POSTing server count
  - GET bot info, server count, upvote count, upvote info
  - GET all bots
  - GET specific user info
  - GET widgets (large and small) including custom ones. See Top.gg docs for more info.
- Not Working / Implemented
  - Searching for bots via the api

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

INDEX