EasyDiscord Documentation

Release 0.1a0

Taku

May 11, 2018

Contents

EasyDiscord is a simple to use wrapper for Discord.py library.

Targeted audiences:

- Beginner developers with little to no experience in programming
- Anyone who wish to make a simple discord bot

CHAPTER 1

References

1.1 API Reference

1.1.1 Importing

The recommended way to import `easydiscord` is as followed:

```
import easydiscord
from easydiscord import *
```

This will import all the necessary items into your global scope. Version Info ————

`easydiscord.__version__`
A string representation of the current version.

Returns:

`str`: The version info.

Example:

```
print(easydiscord.__version__)
"0.1a0.dev0"
```

1.1.2 Initiative Functions

`easydiscord.get_bot` (*token: str*, **args*, ***kwargs*)
Generates a new *Bot* instance. You should avoid calling *Bot()* directly, instead, use this *get_bot()* constructor.

Parameters **token** – (`str`): Your bot's API token.

Keyword Arguments **verbose** – (`bool`): Whether or not certain messages should be printed using `print()`. Defaults to `True`.

Returns An instance of *Bot*.

Examples:

```
bot = easydiscord.get_bot("MY_API_TOKEN")
```

1.1.3 The Bot Class

class easydiscord.**Bot**(*token*, *, *verbose: bool = True*, *severity='high'*)

The Bot object that represents a discord bot.

Call *config()* method after that bot is initiated.

Parameters

- **token** – (str): Your bot's API token.
- **verbose** – (bool): Whether or not to receive some print() messages. Defaults to True.
- **severity** – (str): How should none-breaking error be handled. If set to 'low', a warning will be raised; if set to 'high' an exception would be raised. Defaults to 'high'.

Raises AttributeError – When severity is incorrectly set.

add_command(*func*, *, *name=None*)

Adds a handler to a command. The name keyword argument can be used to override the function name.

Parameters

- **func** – (function): The command handler itself, it can be a coroutine or not.
- **name** – (str): The optional replacement name for your command. If None is passed, the function name will be used.

Returns The function provided by argument func.

Examples:

```
def hi(ctx):
    print('hi')

bot.add_command(hi)
```

add_event(*func*, *, *name=None*)

Adds an event handler. The name keyword argument can be used to override the function's name.

Parameters

- **func** – (function): The event handler itself, it can be a coroutine or not.
- **name** – (str): The optional replacement name for your event handler. If None is passed, the function name will be used.

Returns The function provided by argument func.

Examples:

```
def on_message(message):
    print('hi')

bot.add_event(on_message)
```

add_group(*group: easydiscord.core.Group*, *, *name=None*)
Adds a group of commands. The `name` keyword argument can be used to override the class name.

Parameters

- **group** – (*Group*): The class of the commands.
- **name** – (`str`): The optional replacement name for your group. If `None` is passed, the class name will be used.

Returns The class provided by argument `group`.

Raises *EasyDiscordError* – When the `group` argument isn't a subclass of *Group*.

Examples:

```
class Greetings(easydiscord.Group):
    @property
    def register(self):
        return [self.hi]

    def hi(self, ctx):
        print('hi')

bot.add_group(Greetings)
```

config(*prefix='$'*, *default_on_ready=True*, *desc=''*, *help_format=None*)
Configures this *Bot*.

Parameters

- **prefix** – (`str`): The chat commands prefix from your *Bot*. Defaults to `'$'`.
- **default_on_ready** – (`bool`): Whether nor not use the default *on_ready()* message. Defaults to `True`.
- **desc** – (`str`): The description for the *Bot*.
- **help_format** – *This function has not been implemented yet.*

Returns The *Bot* itself.

Return type *Bot*

on_ready()
This method is a coroutine.

The default `on_ready` event. Nothing will be printed if initial `verbose` is set to `False`.

The following will be printed:

```
Logged in as:
Bot : <bot-name>
ID  : <bot-id>
------
```

Returns `None`

process_message(*message*)
This method is a coroutine.

This function has not been implemented yet.

reload()
Resets the bot. Shuts the bot down and restarts it.

Returns None

reply(*current*, *reply_message*)

setup(*prefix='$'*, *default_on_ready=True*, *desc=''*, *help_format=None*)
An alias of *config()*.

Configures this *Bot*.

Parameters

- **prefix** – (str): The chat commands prefix from your *Bot*. Defaults to '$'.
- **default_on_ready** – (bool): Whether nor not use the default *on_ready()* message. Defaults to True.
- **desc** – (str): The description for the *Bot*.
- **help_format** – *This function has not been implemented yet.*

Returns The *Bot* itself.

Return type *Bot*

start_bot()
Starts the main loop of the discord bot. Do not add anything after this command.

Returns None

bot
Retrieve the background discord.py Bot instance. Do not use this unless you have a clear idea on integrating this into your code.

Returns A discord.py Bot.

Return type Bot

Raises *EasyDiscordError* – When *Bot.config()* is not called first.

prefix
The prefix from your *Bot*'s chat commands.

Returns The string representation of your *Bot*'s prefix.

Return type str

Raises *EasyDiscordError* – When *Bot.config()* is not called first.

1.1.4 Other Objects

class easydiscord.**Command**(*name*, *callback*, ***kwargs*)
A subclass of python.py's Command. This should be used as the Command object.

class easydiscord.**Group**
This is the superclass for all grouping of commands. See *add_group()* for exmaples.

Raises TypeError – When *register()* is not overwritten by subclasses.

set_name(*meth*, *name*)
Sets/changes the name from the method. This function is not required, the command name will remain to be the method's name if *set_name()* is not called. After the method's name has been changed, the command will use the new name. Only use *set_name()* in *register()*.

Parameters

- **meth** – (method): The method whom name will be changed.
- **name** – (str): The name to change it to.

Returns The method provided by argument meth.

Examples:

```
class Greetings(easydiscord.Group):
    @property
    def register(self):
        self.set_name(self.hi, 'hello') # The registered command is not
↪called 'hello'
        return [self.hi]

    def hi(self, ctx):
        print('hi')

bot.add_group(Greetings)
```

register

This class must be overwritten.

This method should return a list of commands to register. If there's no command to register, this should return an empty list.

Returns: list: A list of commands to register, all commands needs to be instances.

1.1.5 Exceptions

exception easydiscord.exceptions.**EasyDiscordError**

This is an overall exception that all easydiscord functions raises when encountered a problem. More Exception will be added in the future.

classmethod no_coro()

exception easydiscord.exceptions.**EasyDiscordWarning**

This is an overall warning that all easydiscord functions raises when encountered a minor problem. More UserWarning will be added in the future.

classmethod no_coro()

CHAPTER 2

Search

- genindex
- modindex
- search

Python Module Index

e

Index