python-bale-bot

Release 2.5.0

Kian Ahmadian

May 01, 2024

REFERENCE

An API wrapper for Bale written in Python.

CHAPTER

ONE

INTRODUCTION

1.1 What is Bale?

The "Bale" is a messenger-platform for send and receive messages. it's provides services for developers, and they can send or receive messages through `bots` like normal users and These services are provided by web services (`API`).

1.2 What is python-bale-bot?

The "python-bale-bot" is a Python language package optimized for developers to use web services provided by "Bale".

CHAPTER

TWO

INSTALLING

You can install or update ``python-bale-bot`` via:

2.1 PyPi:

```
$ pip install python-bale-bot -U
```

2.2 Git:

```
$ git clone https://github.com/python-bale-bot/python-bale-bot
$ cd python-bale-bot
$ python setup.py install
```

CHAPTER

THREE

QUICK START

To get started, learn how the library works through the library. In addition, there are examples in the "Examples" section of the library.

CHAPTER

FOUR

DOCUMENTATION

The package documentation **is the technical reference for python-bale-bot. It contains descriptions of all available classes, modules, methods and arguments as well as the changelog.**

CHAPTER

FIVE

CONTACT TO DEVELOPERS

5.1 Bale Package

5.1.1 Classes in this package

Bot

Available Types

Update

Message

Chat

Chat Member

Chat Photo

User

Callback Query

Sticker

Wait Context

Webhook info

UI

Inline Keyboard Markup

Menu Keyboard Markup

Reply Markup Item

Inline Keyboard Button

Menu Keyboard Button

Payments

Invoice

Labeled Price

Successful Payment

Attachments

Audio

Voice

Contact

Document

Base File

Input File

Location

Photo Size

Video

Animation

Handlers

Base Handler

class bale.handlers.**BaseHandler**

Bases: `object`

This object shows a Base Handler. This is a base class for all update handlers. You can create custom handlers by inheriting from it.

Important: Follow the steps below to create a custom handler: 1. Create a subclass of `bale.BaseHandler`.

2. Create the method *check_new_update()* inside the class When processing updates, the "Mohammed" method is called. This method must return either None or a tuple.

check_new_update(*update*)

This function determines whether the "update" should be covered by the handler or not.

Parameters
update (`bale.Update`) – The update to be tested.

Returns

- If `False` or `None` is returned, the update should not be wrapped by the handler,
- *otherwise the handler is required to wrapp that update.*

async handle_update(*update*, **args*)

This function works if the handler is required to cover the new Update and calls the `callback` function.

Parameters

- **update** (`bale.Update`) – The update to be tested.
- **args** – Additional objects, if given to this parameter, will be passed directly to the `callback` function.

set_callback(*callback*)

Register new handler callback. It will be called during the new Update process after confirming the *check_new_update()* function.

Parameters
callback (*Callable[[UT, ...], Coroutine[...]]*) – The new callback function.

Message Handler

class bale.handlers.**MessageHandler**(*check=None*)

Bases: *BaseHandler*

This object shows a Message Handler. It's a handler class to handle Messages.

Parameters
check (*Callable, optional*) – The check for this handler.

Hint: Called in *check_new_update()*, when new update confirm. This checker indicates whether the Update should be covered by the handler or not.

check_new_update(*update*)

This function determines whether the "update" should be covered by the handler or not.

Parameters
update (`bale.Update`) – The update to be tested.

Returns

- If `False` or `None` is returned, the update should not be wrapped by the handler,

- *otherwise the handler is required to wrapp that update.*

Command Handler

class bale.handlers.**CommandHandler**(*command*, *check=None*)

Bases: *BaseHandler*

This object shows a Message Handler. It's a handler class to handle Messages.

Parameters

- **command** (*Union[str, List[str]]*) – The list of commands that the handler is required to cover. It can be a string or a list of strings.
- **check** (*Optional[Callable[["Update"], bool]]*) – The check function for this handler.

Hint: Called in *check_new_update()*, when new update confirm. This checker indicates whether the Update should be covered by the handler or not.

---

check_new_update(*update*)

This function determines whether the "update" should be covered by the handler or not.

Parameters
update (bale.Update) – The update to be tested.

Returns

- If False or None is returned, the update should not be wrapped by the handler,
- *otherwise the handler is required to wrapp that update.*

Callback Query Handler

class bale.handlers.**CallbackQueryHandler**(*check=None*)

Bases: *BaseHandler*

This object shows a Callback Query Handler. It's a handler class to handle Callback Queries.

check_new_update(*update*)

This function determines whether the "update" should be covered by the handler or not.

Parameters
update (bale.Update) – The update to be tested.

Returns

- If False or None is returned, the update should not be wrapped by the handler,
- *otherwise the handler is required to wrapp that update.*

Edited Message Handler

class bale.handlers.**EditedMessageHandler**(*check=None*)

Bases: *BaseHandler*

This object shows an Edited Message Handler. It's a handler class to handle Edited Messages.

check_new_update(*update*)

This function determines whether the "update" should be covered by the handler or not.

Parameters
update (bale.Update) – The update to be tested.

Returns

- If False or None is returned, the update should not be wrapped by the handler,
- *otherwise the handler is required to wrapp that update.*

5.2 Helpers

5.2.1 Deep Link

bale.helpers.**create_deep_linked_url**(*bot_username*, *payload*)

Creating a deep link for the bot.

> **Warning:** The username of the robot must be entered in the correct format and invalid characters should not be used in the payload parameter.

Parameters

- **bot_username** (str) – The username of bot.
- **payload** (str) – The Payload of deep link

5.2.2 Find

bale.helpers.**find**(*predicate*, *iterable*)

A helper to return the first element in the sequence that meets the predicate.

Parameters

- **predicate** – A function to return boolean-like result.
- **iterable** (str) – An iterable to search through.

5.3 Event Reference

The list of events that can be received by the bot.

An example of how to listen to events in different situations (for `on_message` event):

```
from bale import Bot, Message

bot = Bot("YOUR_TOKEN")

@bot.event
async def on_message(message: Message):
    if message.content == '/start':
        return await message.reply("Hi, python-bale-bot!")

bot.run()
```

```
from bale import Bot, Message

class MyBot(Bot):
    async def on_message(self, message: Message):
        if message.content == '/start':
            return await message.reply("Hi, python-bale-bot!")

MyBot('YOUR_TOKEN').run()
```

5.3.1 Connection

async on_before_ready()

This event is called before the updater starts.

async on_ready()

When the updater starts working and the Bot information is placed in `bale.Bot.user`.

5.3.2 Updates

async on_update(*update*)

This event is called when an update is received from "Bale" servers.

Parameters

update (`bale.Update`) – update received.

5.3.3 Messages

async on_message(*message*)

This event is called when sending a message in a chat to which the bot is connected.

Parameters
message (bale.Message) – message sent.

async on_message_edit(*message*)

This event is called when the sent message is edited.

Parameters
message (bale.Message) – message edited.

5.3.4 CallbackQuery

async on_callback(*callback*)

This event is called when a callback query is created.

Parameters
callback (bale.CallbackQuery) – callback received.

5.3.5 Groups

async on_member_chat_join(*message*, *chat*, *user*)

This event is called when a user joins the chat.

Parameters

- **message** (bale.Message) – message sent.
- **chat** (bale.Chat) – the chat.
- **user** (bale.User) – the user.

async on_member_chat_leave(*message*, *chat*, *user*)

This event is called when a user leaves the chat.

Parameters

- **message** (bale.Message) – message sent.
- **chat** (bale.Chat) – the chat.
- **user** (bale.User) – the user.

5.3.6 Payments

async on_successful_payment(*payment*)

This event is called when a transaction is completed and its status is successful.

Parameters
successful_payment (bale.SuccessfulPayment) – the payment.

5.4 Examples

In this section, there are some robots that are written with `python-bale-bot`.

5.4.1 examples.basic

This robot will answer you with only some commands.

5.4.2 examples.inlinemarkup

This example sheds some light on inline keyboards, callback queries and message editing.

5.4.3 examples.attachment

A basic example of a bot that can send media

5.4.4 examples.invoice

A basic example of a bot that can accept payments.

5.4.5 examples.conversation

A common task for a bot is to ask information from the user.

INDEX