

---

# EzTg

*Release 0.2.0*

Hexye

Mar 01, 2023



**CONTENTS:**

<b>1</b>	<b>EzTg</b>	<b>1</b>
1.1	EzTg package . . . . .	1
<b>2</b>	<b>InlineKeyboard Quick Start</b>	<b>9</b>
<b>3</b>	<b>Quick Start</b>	<b>11</b>
3.1	How to use the send method . . . . .	11
<b>4</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>
	<b>Index</b>	<b>17</b>



## 1.1 EzTg package

### 1.1.1 Submodules

### 1.1.2 EzTg.inlinekeyboard module

**class** EzTg.inlinekeyboard.**InlineKeyboard**

Bases: object

**callback**(*text, callback\_data*)

Add a new callback button to the keyboard.

#### Parameters

**text**

[str] The text of the button

**callback\_data**

[str] The callback data of the button

**callback\_new\_row**(*text, callback\_data*)

Add a new callback button to the keyboard in a new row.

#### Parameters

**text**

[str] The text of the button

**callback\_data**

[str] The callback data of the button

**send**()

Return the keyboard.

**url**(*text, url*)

Add a new url button to the keyboard.

### Parameters

**text**  
[str] The text of the button

**url**  
[str] The url of the button

**url\_new\_row**(*text*, *url*)  
Add a new url button to the keyboard in a new row.

### Parameters

**text**  
[str] The text of the button

**url**  
[str] The url of the button

## 1.1.3 EzTg.main module

**class** EzTg.main.Parse

Bases: dict

**class** EzTg.main.TelegramClient(*token*)

Bases: object

**async copyMessage**(*chat\_id*, *message\_id*, *caption=None*, *disable\_notification=False*,  
*reply\_to\_message\_id=None*, *allow\_sending\_without\_reply=False*,  
*reply\_markup=None*)

Copy a message.

### Parameters

**chat\_id: int**  
The chat id you want to forward the message to.

**message\_id: int**  
The message id you want to forward.

**caption: str**  
The caption you want to send.

**disable\_notification: bool**  
Sends the message silently. Users will receive a notification with no sound.

**reply\_to\_message\_id: int**  
If the message is a reply, ID of the original message.

**allow\_sending\_without\_reply: bool**  
Pass True, if the message should be sent even if the specified replied-to message is not found.

**reply\_markup: InlineKeyboard.send**  
Additional interface options. Use the InlineKeyboard class to create a keyboard and use the send method to send it.

**async createChatInviteLink**(*chat\_id*, *name=None*, *expire\_date=None*, *member\_limit=None*, *creates\_join\_request=False*)

Create a chat invite link.

### Parameters

**chat\_id: int**

The chat id you want to make the invite link.

**name: str**

The name of the invite link.

**expire\_date: int**

The expire date of the invite link.

**member\_limit: int**

The member limit of the invite link.

**creates\_join\_request: bool**

Pass True, if the link should be created without a chat member limit.

**async deleteMessage**(*chat\_id*, *message\_id*)

Delete a message.

### Parameters

**chat\_id: int**

The chat id you want to delete the message from.

**message\_id: int**

The message id you want to delete.

**async editMessageText**(*chat\_id*, *message\_id*, *text*, *inline\_message\_id=None*, *parse\_mode='Markdown'*, *entities=None*, *disable\_web\_page\_preview=False*, *reply\_markup=None*)

Edit a message.

### Parameters

**chat\_id: int**

The chat id you want to edit the message from.

**message\_id: int**

The message id you want to edit.

**text: str**

The text you want to send.

**inline\_message\_id: int**

The inline message id you want to edit.

**parse\_mode: str**

The parse mode you want to use.

**entities: list**

List of special entities that appear in message text, which can be specified instead of parse\_mode.

**disable\_web\_page\_preview: *bool***

Disable link previews for links in this message.

**reply\_markup: *InlineKeyboard.send***

Additional interface options. Use the `InlineKeyboard` class to create a keyboard and use the `send` method to send it.

**async exportChatInviteLink(*chat\_id*)**

Export a chat invite link.

### Parameters

**chat\_id: *int***

The chat id you want to make the invite link.

**async forwardMessage(*chat\_id, from\_chat\_id, message\_id, disable\_notification=False*)**

Forward a message.

### Parameters

**chat\_id: *int***

The chat id you want to forward the message to.

**from\_chat\_id: *int***

The chat id you want to forward the message from.

**message\_id: *int***

The message id you want to forward.

**disable\_notification: *bool***

Sends the message silently. Users will receive a notification with no sound.

**async getMe()**

Get information about the bot.

**async get\_author\_id(*message*)**

Get author id from message.

### Parameters

**message: *dict***

The message object.

**async get\_chat\_id(*message*)**

Get chat id from message.



### Parameters

**message: dict**  
The message object.s

**async leaveChat(chat\_id)**  
Leave a chat.

### Parameters

**chat\_id: int**  
The chat id you want to leave.

**async pinChatMessage(chat\_id, message\_id, disable\_notification=False)**  
Pin a message.

### Parameters

**chat\_id: int**  
The chat id you want to pin the message.

**message\_id: int**  
The message id you want to pin.

**disable\_notification: bool**  
Sends the message silently. Users will receive a notification with no sound.

**async send(method, \*\*kwargs)**  
Send a request to the telegram api.

### Parameters

**method: str**  
The method you want to use.

**\*\*kwargs: dict**  
The parameters you want to send to the method.

**async sendMessage(chat\_id, text, parse\_mode='Markdown', disable\_web\_page\_preview=False, disable\_notification=False, reply\_to\_message\_id=None, reply\_markup=None)**  
Send a message to a chat.

### Parameters

**chat\_id: int**  
The chat id you want to send the message to.

**text: str**  
The text you want to send.

**parse\_mode: str**  
The parse mode you want to use.

**disable\_web\_page\_preview:** *bool*

Disable link previews for links in this message.

**disable\_notification:** *bool*

Sends the message silently. Users will receive a notification with no sound.

**reply\_to\_message\_id:** *int*

If the message is a reply, ID of the original message.

**reply\_markup:** *dict*

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

**async setChatPhoto**(*chat\_id, photo*)

Set the chat photo.

### Parameters

**chat\_id:** *int*

The chat id you want to set the photo.

**photo:** *str*

The photo you want to use.

**async start\_polling**(*callback, callback\_query=None*)

Start polling for updates.

### Parameters

**callback:** *function*

The function you want to call when a message is received.

**callback\_query:** *function*

The function you want to call when a callback query is received.

**async unpinChatMessage**(*chat\_id, message\_id*)

Unpin a message.

### Parameters

**chat\_id:** *int*

The chat id you want to unpin the message.

**message\_id:** *int*

The message id you want to unpin.

**exception EzTg.main.TokenError**

Bases: Exception

#### 1.1.4 Module contents



## INLINEKEYBOARD QUICK START

Here is a quick example of how to use EzTg.InlineKeyboard:

```
from EzTg import TelegramClient, InlineKeyboard
import asyncio

bot = TelegramClient("TOKEN")

async def on_message(update):
    message = update.message
    if message.text == "/start":
        keyboard = InlineKeyboard()
        keyboard.url("Google", "https://google.com")
        keyboard.callback("Click me", "click1")
        keyboard.url_new_row("GitHub", "https://github.com")
        keyboard.callback_new_row("Click me", "click2")
        await bot.send_message(message.chat.id, "Hello World!", reply_markup=keyboard)

async def on_callback(update):
    callback = update.callback_query
    if callback.data == "click1":
        await bot.send("answerCallbackQuery", callback_query_id=callback.id, text="You_
↳clicked the first button!")
    elif callback.data == "click2":
        await bot.send("answerCallbackQuery", callback_query_id=callback.id, text="You_
↳clicked the second button!", show_alert=True)

async def main():
    await bot.start_polling(on_message, on_callback)

asyncio.run(main())
```



## QUICK START

This is a quick start guide to get you started with EzTg.:

```
from EzTg import TelegramClient
import asyncio

bot = TelegramClient("TOKEN")

async def on_message(update):
    message = update.message
    if message.text == "/start":
        await bot.sendMessage(chat_id=message.chat.id, text="Hello World!")

async def main():
    await bot.start_polling(on_message)

if __name__ == "__main__":
    asyncio.run(main())
```

### 3.1 How to use the send method

Here is a little example of using the send method.:

```
from EzTg import TelegramClient
import asyncio

bot = TelegramClient("TOKEN")

async def on_message(update):
    message = update.message
    if message.text == "/start":
        await bot.send("sendMessage", chat_id=message.chat.id, text="Hello World!")

async def main():
    await bot.start_polling(on_message)

if __name__ == "__main__":
    asyncio.run(main())
```

Here the send method has been used to send a message. You can use the send method for any method of the telegram api and can be useful for example if such method is not yet implemented in EzTg.



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### e

EzTg, [7](#)

EzTg.inlinekeyboard, [1](#)

EzTg.main, [2](#)



## C

callback() (*EzTg.inlinekeyboard.InlineKeyboard method*), 1  
 callback\_new\_row() (*EzTg.inlinekeyboard.InlineKeyboard method*), 1  
 copyMessage() (*EzTg.main.TelegramClient method*), 2  
 createChatInviteLink() (*EzTg.main.TelegramClient method*), 2

## D

deleteMessage() (*EzTg.main.TelegramClient method*), 3

## E

editMessageText() (*EzTg.main.TelegramClient method*), 3  
 exportChatInviteLink() (*EzTg.main.TelegramClient method*), 4  
 EzTg  
   module, 7  
 EzTg.inlinekeyboard  
   module, 1  
 EzTg.main  
   module, 2

## F

forwardMessage() (*EzTg.main.TelegramClient method*), 4

## G

get\_author\_id() (*EzTg.main.TelegramClient method*), 4  
 get\_chat\_id() (*EzTg.main.TelegramClient method*), 4  
 getMe() (*EzTg.main.TelegramClient method*), 4

## I

InlineKeyboard (*class in EzTg.inlinekeyboard*), 1

## L

leaveChat() (*EzTg.main.TelegramClient method*), 5

## M

module  
   EzTg, 7  
   EzTg.inlinekeyboard, 1  
   EzTg.main, 2

## P

Parse (*class in EzTg.main*), 2  
 pinChatMessage() (*EzTg.main.TelegramClient method*), 5

## S

send() (*EzTg.inlinekeyboard.InlineKeyboard method*), 1  
 send() (*EzTg.main.TelegramClient method*), 5  
 sendMessage() (*EzTg.main.TelegramClient method*), 5  
 setChatPhoto() (*EzTg.main.TelegramClient method*), 6  
 start\_polling() (*EzTg.main.TelegramClient method*), 6

## T

TelegramClient (*class in EzTg.main*), 2  
 TokenError, 6

## U

unpinChatMessage() (*EzTg.main.TelegramClient method*), 6  
 url() (*EzTg.inlinekeyboard.InlineKeyboard method*), 1  
 url\_new\_row() (*EzTg.inlinekeyboard.InlineKeyboard method*), 2