

Discord ヘビーユーザによる

discord.py 解説

おまえ誰よ

- 塚田 貴史
- Discord 歴 4 年(2016 年から)
 - (歴が長いだけ...とはいわせない!!)
- お仕事・趣味で Python に触れている
- **重度(重度)**のゲーマー
- `github/takapdayon`

LTで持って帰ってもらいたいもの

✌️ bot開発は簡単だよ! ✌️

突然ですが!

Discord 使ってますか？



DISCORD

Discord

元々はゲーマー向けコミュニケーションツールとしてデビュー
今では、大学・企業・ゲーム以外のコミュニティでも幅広く活躍!!!

- [Python.jp](https://python.jp/)
- discord.py
- etc...

そして、なんとサーバ費は驚きの ZERO!(ブーストはあります)
そんな Discord サーバ...

豪華に...したくないですか？

便利にしたいく...ないですか？

その願い、 discord.pyで叶うかもしれません!

discord.py

Discord API をラップして Python から使えるようにしたライブラリ
中で Discord との認証等ごによってくれているためとてもありがたい

<https://discordpy.readthedocs.io/ja/latest/index.html>

始め方

- discord bot を作成する
- pip で [discord.py](https://pypi.org/project/discord.py/) を入れる

これだけです!

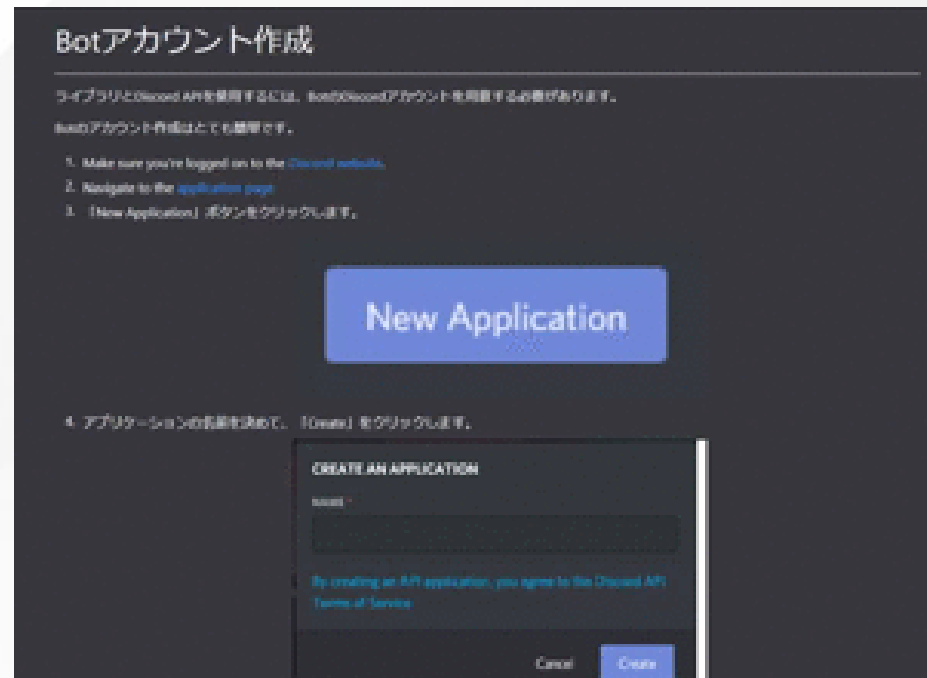
1: discord bot を作成する

[discord.py](https://discord.py.readthedocs.io/ja/latest/discord.html) で紹介されています

<https://discord.py.readthedocs.io/ja/latest/discord.html>

見てわかる通り、やることは凄く少ないです

1. botの名前を決めて作成
2. botの**トークン**を入手
3. botをDiscordサーバに招待する



2: pip で discord.py を入れる

おなじみパッケージ管理ツール pip を使います

```
$ pip install discord.py
```

```
# 音声系を使う場合
```

```
$ pip install discord.py[voice]
```

Hello World!

公式からサンプルで出されている最小限コードです

<https://discordpy.readthedocs.io/en/latest/quickstart.html>

```
import discord

client = discord.Client()

@client.event
async def on_ready():
    print('We have logged in as {0.user}'.format(client))

@client.event
async def on_message(message):
    if message.author == client.user:
        return

    if message.content.startswith('$hello'):
        await message.channel.send('Hello!')

client.run('Botトークン')
```


Hello World!

公式からサンプルで出されている最小限コードです

<https://discordpy.readthedocs.io/en/latest/quickstart.html>

```
import discord

client = discord.Client()

@client.event
async def on_ready():
    print('We have logged in as {0.user}'.format(client))

@client.event
async def on_message(message):
    if message.author == client.user:
        return

    # ここ!
    if message.content.startswith('$hello'):
        await message.channel.send('Hello!')

# ここ!
client.run('Botトークン')
```

動かしてみよう

先ほどのコードを実行してみたいと思います

```
$ python Main.py  
We have logged in as test-slack
```

これだけで、[discord.py](#) 側で、よしなにしてくれます

試す

bot を導入した Discord サーバで\$hello と投稿してみましょう
Bot が Hello!と返してくれれば成功です



t_takafumi 2020/12/11

\$hello



test-slack ボット 2020/12/11

Hello!

何をトリガーにできるのか

一覧は [discord.py](https://discordpy.readthedocs.io/ja/latest/api.html) の API で紹介されています

<https://discordpy.readthedocs.io/ja/latest/api.html>

The screenshot shows the discord.py API documentation for the `Message` class. On the left is a sidebar with a tree view of the API, where `Message` is selected. The main content area is titled "Message" and shows the class definition `class discord.Message`. It is divided into two columns: "Attributes" and "Methods".

Attributes	Methods
<code>activity</code>	<code>async ack</code>
<code>application</code>	<code>async add_reaction</code>
<code>attachments</code>	<code>async clear_reaction</code>
<code>author</code>	<code>async clear_reactions</code>
<code>call</code>	<code>async delete</code>
<code>channel</code>	<code>async edit</code>
<code>channel_mentions</code>	<code>def is_system</code>
<code>clean_content</code>	<code>async pin</code>
<code>content</code>	<code>async publish</code>
<code>created_at</code>	<code>async remove_reaction</code>
<code>edited_at</code>	<code>async reply</code>
<code>embeds</code>	<code>def to_reference</code>
<code>flags</code>	<code>async unpin</code>
<code>guild</code>	
<code>id</code>	
<code>jump_url</code>	
<code>mention_everyone</code>	
<code>mentions</code>	
<code>nonce</code>	
<code>pinned</code>	
<code>raw_channel_mentions</code>	
<code>raw_mentions</code>	
<code>raw_role_mentions</code>	
<code>reactions</code>	
<code>reference</code>	
<code>role_mentions</code>	
<code>stickers</code>	
<code>system_content</code>	
<code>tts</code>	
<code>type</code>	
<code>webhook_id</code>	

Below the class definition, there is a description: "Represents a message from Discord." and a section for "Supported Operations" with the example `x == y`. In the bottom right corner, there are links for "to top" and "v. latest".

ちらっと中身解説

```
client = discord.Client()

@client.event
async def on_message(message):
    if message.author == client.user:
        return
```

- @client.event
 - 対象の関数が**コルーチン関数**か判定し、コルーチンの場合は client にインスタンス変数として保持させます

ちらっと中身解説

```
client = discord.Client()

@client.event
async def on_message(message):
    if message.author == client.user:
        return
```

- `async def on_message(message):`
 - `discord.py`側で実行する関数です。`message`の中にチャットしたサーバ等の情報が入ってます
- `if message.author == client.user:`
 - メッセージを発信した対象が bot 自身か判定(無限ループ防止)

ちらっと中身解説

```
client.run('Botトークン')
```

- client.run()
 - 中でstart関数(login関数とconnect関数)がTaskとして登録され、run_forever()で永続化されて動いています。
asyncio の低レベルAPI()がゴリゴリ動いているので、興味ある方は見てみると面白いかもしれません!

まとめ

✌️ bot開発は簡単だよ! ✌️

You are the server customizer Champion!

(ご清聴ありがとうございました!)