import sys

import irc.bot

import requests

class TwitchBot(irc.bot.SingleServerIRCBot):

def \_\_init\_\_(self, username, client\_id, token, channel):

self.client\_id = client\_id

self.token = token

self.channel = '#' + channel

self.hogumastack = 0

#채널 ID를 얻기 위해 v5 API 호출

url = 'https://api.twitch.tv/kraken/users?login=' + channel

headers = {'Client-ID': client\_id, 'Accept': 'application/vnd.twitchtv.v5+json'}

r = requests.get(url, headers=headers).json()

self.channel\_id = r['users'][0]['\_id']

# IRC bot 연결 생성

server = 'irc.chat.twitch.tv'

port = 6667

print('서버 ' + server + ', 포트 ' + str(port) + '에 연결 중...')

irc.bot.SingleServerIRCBot.\_\_init\_\_(self, [(server, port, 'oauth:' + token)], username, username)

def on\_welcome(self, c, e):

print(self.channel + '에 연결되었습니다.')

#봇을 사용하기 전에 채널 권한 부여가 필요

c.cap('REQ', ':twitch.tv/membership')

c.cap('REQ', ':twitch.tv/tags')

c.cap('REQ', ':twitch.tv/commands')

c.join(self.channel)

def on\_pubmsg(self, c, e):

# If a chat message starts with an exclamation point, try to run it as a command

if e.arguments[0][:1] == '!':

cmd = e.arguments[0\].split(' ')[0][1:]

print('Received command: ' + cmd)

self.do\_command(e, cmd)

return

def do\_command(self, e, cmd):

c = self.connection

# Poll the API to get current game.

if cmd == "game":

url = 'https://api.twitch.tv/kraken/channels/' + self.channel\_id

headers = {'Client-ID': self.client\_id, 'Accept': 'application/vnd.twitchtv.v5+json'}

r = requests.get(url, headers=headers).json()

c.privmsg(self.channel, r['display\_name'] + ' is currently playing ' + r['game'])

# Poll the API the get the current status of the stream

elif cmd == "title":

url = 'https://api.twitch.tv/kraken/channels/' + self.channel\_id

headers = {'Client-ID': self.client\_id, 'Accept': 'application/vnd.twitchtv.v5+json'}

r = requests.get(url, headers=headers).json()

c.privmsg(self.channel, r['display\_name'] + ' channel title is currently ' + r['status'])

# Provide basic information to viewers for specific commands

elif cmd == "스팀잇":

message = "안녕! 스팀잇!"

c.privmsg(self.channel, message)

# The command was not recognized

else:

c.privmsg(self.channel, "Did not understand command: " \+ cmd)

* **\_\_init\_\_** 함수는 봇의 정보를 입력받고 인증 및 서버에 연결 해주는 함수입니다.
* **on\_welcome** 함수는 채널에 제대로 접속했는지를 pycharm에 출력해줍니다.
* **on\_pubmsg**는 봇이 채팅방에 상주하면서 메세지가 올라올 때 마다 파싱하는 함수입니다. 기본적으로 느낌표(!) 로 시작하는 글자를 명령어로 인식합니다. 모든 채팅에 대해 대응하려면 조건문을 고쳐주면 됩니다. 이 함수 코드의 이미는 ! 로 시작하는 글을 파싱해서 **do\_command** 함수로 보내라는 뜻입니다.
* **do\_command** 함수는 파싱되어 들어온 명령어에 대한 기능 처리 부분입니다. 채팅창에서 누군가 !title 라고 적으면 그걸 인식해서 방송 제목을 보여준다던지 합니다. 예제 코드에 **!스팀잇** 명령어를 추가해서 누군가 !스팀잇 이라고 적으면 **안녕! 스팀잇!** 이라고 반응하는 코드를 추가했습니다.

def main():

username = "Steemit" # 별 의미 없습니다. 봇 계정의 이름은 twitch 사이트에서 조정해주면 됩니다.

client\_id = "xxxxxxxxxxxx" # Client ID

token = "xxxxxxxxxxxxx" # oauth: 는 빼고 뒷부분만 적어주시면 됩니다.

channel = "twitch\_account" # 봇이 접속할 채널입니다. 테스트할 땐 본인의 트위치 계정을 적어주세요.

bot = TwitchBot(username, client\_id, token, channel)

bot.start()

if \_\_name\_\_ == "\_\_main\_\_":

main()

|  |
| --- |
| TwitchIRC::TwitchIRC(UFC32 nick, UFC32 usr, UFC32 pass, UFC32 addr, U32 port, UFC32 channel) : |
|  | \_connectedChannel(channel), serverAddr(), serverPort(0), \_socketObj(NULL), autoping\_thread(NULL) { |
|  | cout << "IRCClient: Establishing" << endl; |
|  | Lib::writeToLog("PhantomBotLog.txt", "{C++} Establishing TwitchIRC Instance"); |
|  | //Create the buffer |
|  | buffer = new ACHAR[\_MAXRECV]; |
|  | //Create the socket |
|  | \_socketObj = Lib::createSocketAndConnect(addr, port); |
|  | if(!\_socketObj) { |
|  | cout << "Failed to establish connection to " << addr << endl; |
|  | Lib::writeToLog("PhantomBotLog.txt", "{C++} Failed to connect to " + string(addr)+ "."); |
|  | return; |
|  | } |
|  | serverAddr = addr; |
|  | serverPort = port; |
|  | cout << "Initializing Command Instances" << endl; |
|  | //Init the command limit instance... |
|  | TwitchCommandLimit::fetchInstance().Init(\_socketObj, channel); |
|  | ChatCommandDecs::fetchInstance().init(); |
|  | cout << "Creating Command Interfaces..." << endl; |
|  | //Construct the login token |
|  | cout << "IRCClient: Establishing login token" << endl; |
|  | Lib::writeToLog("PhantomBotLog.txt", "{C++} Establishing TwitchIRC Login Token"); |
|  | const string pS = string("PASS " + string(pass) + "\r\n"); |
|  | const string nS = string("NICK " + string(nick) + "\r\n"); |
|  | const string uS = string("USER " + string(usr) + "\r\n"); |
|  | //Password must be sent first, then our information |
|  | if(pS.size()) { |
|  | TwitchCommandLimit::fetchInstance().SendCommand(pS); |
|  | } |
|  | TwitchCommandLimit::fetchInstance().SendCommand(nS); |
|  | TwitchCommandLimit::fetchInstance().SendCommand(uS); |
|  | //Wait for the welcome reply... |
|  | fetchServerMessage(); |
|  | if(response.find("Welcome") == string::npos) { |
|  | //We failed... |
|  | cout << "BOT: Failed to connect to TwitchIRC" << endl << endl << response.c\_str() << endl << endl; |
|  | Lib::writeToLog("PhantomBotLog.txt", "{Twitch} We did not recieve a welcome message"); |
|  | CloseSocket(); |
|  | } |
|  | else { |
|  | //Set the socket to non-blocking mode to allow the program to continue execution |
|  | \_socketObj->setNonBlocking(); |
|  | //Enable advanced commnads |
|  | const string aCS1 = string("CAP REQ :twitch.tv/commands\r\n"); |
|  | const string aCS2 = string("CAP REQ :twitch.tv/membership\r\n"); |
|  | const string aCS3 = string("CAP REQ :twitch.tv/tags\r\n"); |
|  | TwitchCommandLimit::fetchInstance().SendCommand(aCS1); |
|  | TwitchCommandLimit::fetchInstance().SendCommand(aCS2); |
|  | TwitchCommandLimit::fetchInstance().SendCommand(aCS3); |
|  | Lib::writeToLog("PhantomBotLog.txt", "{Twitch} Connected to TwitchIRC, connecting to channel '#" + string(channel) + "'."); |
|  | //And finally... connect to the channel |
|  | const string cS = string("JOIN " + string(channel) + "\r\n"); |
|  | TwitchCommandLimit::fetchInstance().SendCommand(cS); |
|  | //Send a intro message to init stuff... |
|  | SendChatMessage("PhantomBot Now Connected to channel..."); |
|  | //Set up AutoPing command |
|  | autoping\_thread = new thread(&TwitchIRC::AutoPing, this); |
|  | } |
|  | } |