Ask Django

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

슬랙 봇 만들기

봇 기능을 위해 메세지를 받을 수 있는 방법

- Real Time Messaging API 활용
  - Slack 서버와 연결을 유지하는 클라이언트 작성 필요
- Incoming Webhooks 활용
  - Slack 서버로부터의 HTTP 요청을 받을 수 있는 웹애플리케이션 필요 (장고/플라스크 등 활용 가능)

Real Time Messaging #api

- 파이썬 프로그램에서 <슬랙서버>로 먼저 접속을 시도하여 연결을 유지
  - 그 연결을 통해 슬랙서버에서 파이썬 프로그램으로 메세지를 보내고 받을 수 있습니다.
- 주의: Bot에 명령을 전달하기 전에 먼저 초대를 해주셔야, Bot에서 메세지를 받을 수 있습니다.

필요한 라이브러리

- 동기식
  - slacker
  - websocket-client #github
- 비동기식
  - 파이썬 3.5 이상 필요
  - slacker
  - websockets #doc

주요코드 (동기식 코드)

```
import json
import websocket
from slacker import Slacker

def bot(token):
    response = Slacker(TOKEN).rtm.start()
    meta = response.body  # 현 Slack Team에 대한 다양한 정보
    ws = websocket.create_connection(meta['url'])

    try:
        while True:
            response = json.loads(ws.recv())
            if 'message' == response.get('type', None):
                if 'channel' in response:
                    ws.send(json.dumps({
                        'channel': response['channel'],
                        'type': 'message',
                        'text': 'Echo : ' + response['text'],
                    }))
    finally:
        ws.close()

if __name__ == '__main__':
    bot(TOKEN)
```

주요코드 (비동기식 코드) #ref

파이썬 3.5 이상만 지원

```
import asyncio
import json
from slacker import Slacker
import websockets

async def bot(token):
    response = Slacker(token).rtm.start()
    meta = response.body  # 현 Slack Team에 대한 다양한 정보
    ws = await websockets.connect(meta['url'])

    try:
        while True:
            response = json.loads(await ws.recv())
            if 'message' == response.get('type', None):
                if 'channel' in response:
                    await ws.send(json.dumps({
                        'channel': response['channel'],
                        'type': 'message',
                        'text': 'Echo : ' + response['text'],
                    }))
    finally:
        await ws.close()

if __name__ == '__main__':
    loop = asyncio.get_event_loop()
    try:
        loop.run_until_complete(bot(TOKEN))  # bot() 코루틴이 종료될 때까지, 블록킹 상태
    except KeyboardInterrupt:
        pass
    finally:
        loop.close()
```

완성코드

bot.py (1/2)

```
import json
from slacker import Slacker
import websocket


class BotBase:
    def __init__(self, token):
        response = Slacker(token).rtm.start()
        meta = response.body
        self.ws = websocket.create_connection(meta['url'])

    def recv(self):
        return json.loads(self.ws.recv())

    def send(self, obj, opcode=websocket.ABNF.OPCODE_TEXT):
        try:
            payload = json.dumps(obj)
        except TypeError:
            payload = obj
        return self.ws.send(payload, opcode)

    def close(self):
        self.ws.close()
```

```
    def on_hello(self):
        pass

    def on_message(self, channel, **kwargs):
        user = kwargs.pop('user', '')
        text = kwargs.pop('text', '')

        if user and text and text.startswith('!'):
            cmd, query = text[1:].split(':', 1)
            fn_name = 'cmd_' + cmd
            fn = getattr(self, fn_name, None)
            if callable(fn):
                fn(channel, user, query)
            else:
                self.post_message(channel, '지원하지않는 명령입니다.')

    def post_message(self, channel, text):
        self.send({
            'channel': channel,
            'type': 'message',
            'text': text,
        })
```

```
def run_forever(self):
    try:
        while True:
            response = self.recv()
            print(response)
            if 'type' in response:
                fn_name = 'on_' + response.pop('type')
                fn = getattr(self, fn_name, None)
                if callable(fn):
                    fn(**response)
    except KeyboardInterrupt:
        pass
    finally:
        self.close()
```

main.py (2/2)

```
import json
import requests
from bot import BotBase


class Bot(BotBase):
    def cmd_멜론검색(self, channel, user, query):
        params = {
            'jscallback': '_',
            'query': query,
        }
        jsonp_string = requests.get('http://www.melon.com/search/keyword/index.json', params=params).text
        json_string = jsonp_string.replace('_(', '').replace(');', '')
        meta = json.loads(json_string)

        messages = []
        for song in meta['SONGCONTENTS']:
            messages.append('''[{ALBUMNAME}] {SONGNAME} by {ARTISTNAME}
- http://www.melon.com/song/detail.htm?songId={SONGID}'''.format(**song))

        if messages:
            message = '\n'.join(messages)
        else:
            message = '검색어 "{}"에 대한 검색결과가 없습니다.'.format(query)

        self.post_message(channel, message)


if __name__ == '__main__':
    token = 'xoxb-----------------------------------------'
    Bot(token).run_forever()
```