Ask Django

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

카카오톡 플러스친구 봇 만들기

카카오톡 플러스친구 자동응답 API v2.0 #ApiDoc

- 플러스친구(구. 옐로아이디) 사용자에 한하여 자동응답 API를 제공
  - 관리자 로그인
  - 내 플러스 친구 > 새 플러스 친구 만들기

앱 등록

관리자 좌측 메뉴 "스마트 채팅" > "API형" 선택

- 앱 이름
- **앱 URL** : 외부에서 접근 가능한 URL
  - ngrok는 처리가 느려서인지 잘 처리되지 않습니다. 직접 연결을 추천
- 앱 설명

봇 요청 클라이언트 정보

- 3대의 Proxy서버. 아이피 대역 #doc
  - 110.76.143.234
  - 110.76.143.235
  - 110.76.143.236
- UserAgent 헤더 : "KakaoTalk/Bot"

유저키 #doc

- user_key는 특정 카카오톡 이용자에 대해 프로필별로 각기 다르게 발급됩니다. 따라서 user_key는 해당 프로필에 대해서만 유효합니다.
- 카카오톡 이용자가 프로필을 차단했다가 다시 추가한 경우에는 user_key가 갱신되지 않으며, **이용자가 카카오톡 탈퇴 후 재가입한 경우 갱신**됩니다.

공개설정

- 관리 / 상세설정 / 공개설정을 통해 공개하셔야, 카카오톡 앱에서 검색이 가능합니다.
  - 홈 공개
  - 검색 허용
  - 노출 허용

API Spec

Spec #doc

- 응답 지연시간 : 최대 5초안에 응답이 오지 않는 경우 응답없음 메시지가 사용자에게 발송됩니다.
- QoS : 만약 응답 실패가 10회 이상 발생하게 되면 자동적으로 해당 모듈은 지정된 장애 메시지를 리턴하게 됩니다. 더불어 앱 등록시 지정된 관리자에게 카카오톡으로 장애 메시지가 발송됩니다.

요청 URL과 실습 프로젝트에서의 뷰 함수

설정한 경로에 덧붙여서 지정 (ex: http(s)://주소/plusfriend)

- http(s)://주소/plusfriend/**keyboard** : views.on_init 함수
  - 자동응답 명령어의 목록을 요구
  - 유저가 최초로 채팅방에 들어오거나, 채팅방을 지우고 재진입시에도 호출
  - 카카오 서버에서 1분동안 캐쉬로 저장

선택유형 1) 아무 응답없음. 유저로부터 TEXT 타이핑 입력을 받을 예정.

```
{"type": "text"}
```

선택유형 2) 버튼 선택지 제공 (봇에서 유저의 TEXT 타이핑 입력을 받을 수 없음.)

```
{"type": "buttons", "buttons": ["선택1", "선택2", "선택3"]}
```

- http(s)://주소/plusfriend/**friend** : viwes.on_added 함수
  - 유저가 친구로 추가할 경우, 호출
  - 요청 예) {"user_key": "bFS8biwaiz-d"}
- http(s)://주소/plusfriend/**chat_room/유저키** : views.on_leave 함수
  - 유저키의 유저가 채팅방에서 나갈 경우, 호출
- http(s)://주소/plusfriend/**friend/유저키** : views.on_block 함수
  - 유저키의 유저가 차단할 경우, 호출

- http(s)://주소/plusfriend/**message** : views.on_message 함수

```
# 요청 예 1) 텍스트 수신 or 버튼 입력
{"type": "text", "content": "받은 버튼/메세지", "user_key": "bFS8biwaiz-d"}

# 요청 예 2) 사진 수신
{"type": "photo", "content": "https://사진URL/blahblah.jpg", "user_key": "bFS8biwaiz-d"}
```

message 응답하기

- "텍스트" + "사진" + "메세지 버튼" 의 임의 조합
- 선택 버튼 목록

Message 부분 : 텍스트 + 이미지[1] #doc + 메세지버튼의 조합

3가지 타입 중 최소 1가지 적용되어야하며, 최대 3가지 타입까지 적용 가능

```
{
    "message": {
        "text": "단순 텍스트 응답",
        "photo": {
            "url": "https://photo.src",
            "width": 640,
            "height": 480
        },
        "message_button": {
            "label": "주유 쿠폰받기",
            "url": "https://coupon/url"
        }
    }
}
```

---

[1] 장고에 대한 자세한 내용은 "장고 기본편 VOD"를 참고하세요.

선택 버튼 목록

```
{
    "keyboard": {
        "type": "buttons",
        "buttons": [
            "처음으로",
            "다시 등록하기",
            "취소하기"
        ]
    }
}
```

장고로 봇 서버 개발

주요 스펙

- 모든 POST요청은 applicatoin/json 요청
  - 즉 request.body를 파싱하여 json처리하여 인자 처리
- 모든 응답은 JSON 응답
- 외부서버로부터의 직접적인 POST요청이므로 csrf_exempt처리 필요

```
import json
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt

@csrf_exempt
def view_fn(request):
    json_obj = json.loads(request.body.decode('utf8'))
    # ...
    return JsonResponse({'type': 'text'})
```

각 구현

장고 프로젝트/장고앱 생성 [1]

- 장고 설치 : `pip3 install django`
- 장고 프로젝트 생성 : `django-admin startproject askbot`
- 디렉토리 이동 : `cd askbot`
- plusfriend 장고앱 생성 : `python3 manage.py startapp plusfriend`

[1] 장고에 대한 자세한 내용은 "장고 기본편 VOD"를 참고하세요.

장고앱 기본 설정

- urls 등록

```
# askbot/urls.py 에 다음 추가
from django.conf.urls import include, url

urlpatterns = [
    url(r'^plusfriend/', include('plusfriend.urls')),
]

# plusfriend/urls.py 파일 생성
from django.conf.urls import url

urlpatterns = [
]
```

plusfriend/functions.py

멜론검색 예시

```
import json
import requests

def melon_search(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 = []
    if 'SONGCONTENTS' in meta:
        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 노래 검색결과가 없습니다.'.format(query)

    return message
```

plusfriend/urls.py

```
from django.conf.urls import url
from . import views

urlpatterns = [
    url(r'^keyboard$', views.on_init),
    url(r'^friend$', views.on_added),
    url(r'^friend/(?P<user_key>[\w-]+)$', views.on_block),
    url(r'^chat_room/(?P<user_key>[\w-]+)$', views.on_leave),
    url(r'^message$', views.on_message),
]
```

plusfriend/decorators.py

각 뷰별로 반복되는 작업을 일괄적으로 처리해주는 bot 장식자

```
from functools import wraps
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt

def bot(view_fn):
    @wraps(view_fn)
    @csrf_exempt
    def wrap(request, *args, **kwargs):
        if request.method == 'POST':
            request.JSON = json.loads(request.body.decode('utf8'))
        else:
            request.JSON = {}
        return JsonResponse(view_fn(request, *args, **kwargs) or {})
    return wrap
```

plusfriend/views.py

각 요청을 처리하는 뷰 함수 구현

```
from .decorators import bot
from . import functions

@bot
def on_init(request):
    return {'type': 'text'}

@bot
def on_message(request):
    user_key = request.JSON['user_key']
    type = request.JSON['type']
    content = request.JSON['content']  # photo 타입일 경우에는 이미지 URL

    if content.startswith('멜론검색:'):
        query = content[6:]
        response = '멜론 "{}" 검색결과\n\n'.format(query) + functions.melon_search(query)
    else:
        response = '지원하는 명령어가 아닙니다.'

    return {
        'message': {
            'text': response,
        }
    }
```

```
@bot
def on_added(request):
    user_key = request.JSON['user_key']

@bot
def on_block(request, user_key):
    pass

@bot
def on_leave(request, user_key):
    pass
```

user_key로 유저를 식별하여,
다양한 기능을 구현해보세요.

Coming soon ...

- 각 대화의 상태(State)를 저장하기
- 카카오톡으로 주문받기
- 클라우드 환경에 봇 배포하기