line-bot-sdk-python を用いたユーザー情報と位置情報の取得

Webhooks

> Receive notifications in real-time when a user sends a message or friends your account.

ユーザがメッセージやアカウントをBotに送信した場合に，通知を受け取るサーバーのこと(?)

> When an event, such as when a user adds your account or sends a message, is triggered, an HTTPS POST request is sent to the webhook URL that is configured on the Channel Console.

ユーザがBotを追加したり，Botへメッセージを送ったりした場合，"event"が発生する．
event の発生時には，HTTPS POST request として Channel console で設定した webhook URL へ通知が送られる．

webhook 経由で event を受け取った場合，handlerを使うことで，event の種類に応じたレスポンスを返すことができる．
以下の例では，MessageEvent が発生したとき，その event がもつ message が TextMessage だったら，定義された関数を実行する．

python

```python
@handler.add(MessageEvent, message=TextMessage)
def handle_message(event):
    line_bot_api.reply_message(
        event.reply_token,
        TextSendMessage(text=event.message.text)
    )

# 次のように書くことでLocationMessageに向けたhandlerも作成できる
@handler.add(MessageEvent, message=LocationMessage)
```

Webhook event object

> JSON object which contains events generated on the LINE

> Platform

event の正体は，webhook へ送られる JSON object である．

event の中には，timestamp や user_id が格納されている．
例えば，event を生成したユーザの user_id は，event.source の中に格納されている．

以下のソースコードでは，受け取った event から user_id を取り出して，表示している．
また受け取った user_id から，profile 情報を取り出して表示している．

```python
@handler.add(MessageEvent, message=TextMessage)
def handle_message(event):
    # get user_id from event
    uId = event.source.user_id
    print uId

    # get profile from user_id
    profile = line_bot_api.get_profile(uId)
    print(profile.display_name)
```

Reply message

> Reply to messages from users.

ユーザから送信されたメッセージに対して，Botが返信する行為
(User -> Bot, then Bot -> User)

sdk 内では，次の命令で Reply message を実現している．

```
reply_message(self, reply_token, messages, timeout=None)
```

> Respond to events from users, groups, and rooms. You can get a reply_token from a webhook event object.

命令内の reply_token は，ユーザから受け取った event に紐付いている固有のトークンを示す．実際のユースケースでは，まず handler を使ってユーザが生成した event を受け取り，それに対して token を付加して，reply_message 関数を呼び出す．

命令内の message には，ユーザに返す具体的な内容（例：文章，ボタン，写真，位置情報など）を引数として与える．
次のソースコードは，文章(TextSendMessage)を返す場合の例である．

```python
line_bot_api.reply_message(reply_token, TextSendMessage(text='Hello World!'))
```

Push message

> Send messages to users at any time.

Botが自発的にユーザへメッセージを送信する行為
(Bot -> User)

位置情報の取得

chatbot.py に以下のスクリプトを追加.
MessageEvent が起こったとき，その message が LocationMessage ならば，
handlerを起動する.
handler では event.message.address から，位置情報の文字列表記を取得し，
TextSendMessage として User に送信する.

```python
# Header 部分に追加
from linebot.models import (
    LocationMessage
)

# TextMessageに対するhandlerの下に追加
@handler.add(MessageEvent, message=LocationMessage)
def handle_message(event):
    line_bot_api.reply_message(
        event.reply_token,
        TextSendMessage(text=event.message.address)
    )
    print 'reply to location message'
```

参考プログラム

```python
from flask import Flask, request, abort

from linebot import (
    LineBotApi, WebhookHandler
```

```
)
from linebot.exceptions import (
    InvalidSignatureError
)
from linebot.models import (
    MessageEvent, TextMessage, TextSendMessage, LocationMessage
)

app = Flask(__name__)

line_bot_api = LineBotApi('token')
handler = WebhookHandler('secret token')


@app.route("/callback", methods=['POST'])
def callback():
    # get X-Line-Signature header value
    signature = request.headers['X-Line-Signature']

    # get request body as text
    body = request.get_data(as_text=True)
    app.logger.info("Request body: " + body)

    # handle webhook body
    try:
        handler.handle(body, signature)
    except InvalidSignatureError:
        abort(400)

    print 'callback'
    return 'OK'


@handler.add(MessageEvent, message=TextMessage)
def handle_message(event):
    line_bot_api.reply_message(
        event.reply_token,
        TextSendMessage(text=event.message.text)
    )
    # get user_id from event
    uId = event.source.user_id
    print uId
    # get profile from user_id
    profile = line_bot_api.get_profile(uId)
    print(profile.display_name)
    # for test
    print 'reply to text message'

@handler.add(MessageEvent, message=LocationMessage)
def handle_message(event):
    line_bot_api.reply_message(
```

```
            event.reply_token,
            TextSendMessage(text=event.message.address)
        )
        # for test
        print 'reply to location message'


if __name__ == "__main__":
    app.run()
```