SlackBot プログラム 仕様書

2017 年 4 月 20 日
乃村研究室 西 良太

1 概要

本資料は，平成 29 年度 GN グループ B4 新人研修課題にて作成した SlackBot プログラムの仕様についてまとめたものである．本プログラムは以下の 2 つの機能をもつ．

(1) “「○○」と言って” という発言に対して，“○○” と発言する機能

(2) “○○付近の△△” という発言に対して，○○で指定された場所の近くの△△に関連する施設 3 件の情報を発言する機能

2 対象とする利用者

本プログラムは以下のアカウントを所有する利用者を対象としている．

(1) Slack アカウント

(2) Google アカウント

Google アカウントは本プログラムで使用する API のキー取得に必要である．

3 機能

本プログラムは Slack での “@NBot” から始まるユーザの発言を受信し，それに対して返信する．返信される内容は “@NBot” に続く文字列により決定される．以下に本プログラムがもつ 2 つの機能について述べる．

(機能 1) “「○○」と言って” という発言に対して，“○○” と発言する機能

この機能はユーザの “「○○」と言って” という発言に対して，一番外側の鉤括弧内の文字列を発言したユーザに返信する．

表 1: sinatra1.4.8 が必要とする Gem

Gem	バージョン
rack	1.5 以上 2.0 未満
rack-protection	1.4 以上 2.0 未満
tilt	1.3 以上 3.0 未満

(機能 2) “○○付近の△△” という発言に対して，○○で指定された場所の近くの△△に関連する施設 3 件の情報を発言する機能

この機能はユーザの “○○付近の△△” という発言に対して，○○で指定された場所周辺の△△という施設について以下の 3 つの情報をそのユーザに返信する．

(1) ○○からの距離が近い△△に関連する施設 3 件の施設名と住所．

(2) ○○からそれぞれの施設までの経路を見ることができる GoogleMap へのリンク．

(3) ○○と検索された 3 件の施設にピンを立てた地図の画像．

上記の情報は Googole Maps Geocoding API[1]，Google Places API[2]，Googole Static Maps API[3] を利用して取得または作成している．また，地図画像の URL については Google URL Shortener API[4] を用いて短縮したものを使用する．

上記の (機能 1) と (機能 2) のどちらにも当てはまらない文字列を受信したときは，以下のメッセージを発言する．

```
Hi! @ユーザー名
Usage:"○○付近の○○", "「○○」と言って"
```

4 動作環境

本プログラムは Ruby 2.1.5 で動作する．また，Web アプリケーションフレームワークとして sinatra 1.4.8 を利用しているため sinatra と依存関係にある表 1 に示す Gem を必要とする．

表 2: 動作確認済み環境

デプロイ先 (Heroku) の環境	
OS	Ubuntu 14.04.5 LTS
CPU	Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz
メモリ	64GB
Ruby	2.1.5p273
Ruby Gem	bundler 1.13.7
	sinatra 1.4.8
	rack 1.6.5
	rack-protection 1.5.3
	tilt 2.0.7

5 動作確認済み環境

動作確認済み環境を表 2 に示す．bundler 以外の Gem は Gemfile と Gemfile.lock に記述されている依存関係を用いてインストールされる．

6 環境構築

6.1 概要

本プログラムの動作のために必要な環境構築の項目を以下に示す．

(1) Heroku の設定

(2) Slack の WebHook の設定

(3) 各種 Google API の API キー取得

次節でそれぞれの具体的な環境構築手順について述べる．

6.2 具体的な手順

6.2.1 Heroku の設定

(1) 以下の URL より Heroku にアクセスし,「Sign up」から新しいアカウントを登録する．

https://www.heroku.com/

(2) 登録したアカウントでログインし,「Getting Started with Heroku」の使用する言語として「Ruby」を選択する.

(3) 「I 'm ready to start」をクリックし,「Download Heroku CLI for...」から CLI をダウンロードする.

(4) 以下のコマンドを実行し Heroku にログインする.

```
$ heroku login
```

(5) 以下のコマンドを実行し Heroku 上にアプリケーションを生成する.

```
$ heroku create <app_name>
```

6.2.2 Slack の WebHook の設定

Slack が提供する Incoming Webhooks と Outgoing Webhooks の設定手順は以下の通りである.

Incoming WebHooks の設定

(1) 以下の URL にアクセスする.
https://XXXXX.slack.com/apps/manage/custom-integrations
ただし，XXXXX はチーム名.

(2) 「Incoming WebHooks」をクリックする.

(3) 「Add Configuration」をクリックし発言先のチャンネルを選択した後「Add Incoming WebHooks integration」をクリックすることで WebHook URL を取得する.

(4) 取得した URL は以下のコマンドにより Heroku の環境変数として設定する.

```
$ heroku config:set INCOMING_WEBHOOK_URL="https://XXXXXXXX"
```

Outgoing WebHooks の設定

(1) 以下の URL にアクセスする.
https://XXXXX.slack.com/apps/manage/custom-integrations
ただし，XXXXX はチーム名.

(2) 「Outgoing WebHooks」をクリックする.

(3) 「Add Configuration」をクリックし「Add Incoming WebHooks integration」をクリックする．ここで,「Integration Settings」の以下の項目を設定する．

(A) Channel にて発言を監視するチャンネルを選択する．

(B) Trigger Word(s) に WebHook が動作する契機となる単語を設定する．

(C) URL(s) に WebHook が動作した際に POST を行う URL を設定する．
今回は Heroku 上で動作させるため以下の URL を設定する．
https://XXXXX.herokuapp.com/slack
ただし，XXXXX は Hroku に登録したアプリケーション名．

6.2.3 各種 Google API の API キー取得

本プログラムが使用する Google API のキー取得方法について述べる．また，API キーを取得するためには Google アカウントが必要である．

(1)Google Maps Geocoding API

以下の URL にアクセスし「標準 API 向けの認証」の「キーを取得する」より API キーを取得する．

https://developers.google.com/maps/documentation/geocoding/get-api-key?hl=ja

(2)Google Places API Web Service

以下の URL にアクセスし「標準 Google Places API Web Service を使用する場合」の「キーを取得する」より API キーを取得する．

https://developers.google.com/places/web-service/get-api-key?hl=ja

(3)Googole Static Maps API

以下の URL にアクセスし「標準 API 向けの認証」の「キーを取得する」より API キーを取得する．

https://developers.google.com/maps/documentation/static-maps/get-api-key?hl=ja

(4)Google URL Shotener API

以下の URL にアクセスし「Acquiring and using an API key」の「GET A KEY」より API キーを取得する．

https://developers.google.com/url-shortener/v1/getting_started

また，取得したAPIキーは“APIconfig.yml”という名称のファイルを用意し，以下のように記述する．

```
GooglePlaces : API key
GoogleGeocoding : API key
GoogleStaticMaps : API key
GoogleURLShortener : API key
```

7　使用方法

本プログラムの使用方法について述べる．本プログラムはHeroku上で動作するため，Herokuへデプロイすることで実行できる．

Hrokuには以下のコマンドを用いてデプロイできる．

```
$ git push heroku master
```

8　エラー処理と保証しない動作

本プログラムにおけるエラー処理と保証しない動作について述べる．

8.1　エラー処理

本プログラムで行ったエラー処理を以下に示す．

(1) (機能2)について，“○○付近の△△”というメッセージの○○の座標をGoogle Maps Geocoding APIが見つけられなかった場合，以下のようにユーザに返信する．

> @ユーザ名
> 地点が特定できませんでした．

(2) (機能2)について，“○○付近の△△”について○○周辺の△△に関連する施設がGoogle Places APIによって発見できなかった場合，以下のようにユーザに返信する．

> @ユーザ名
> 結果が見つかりませんでした．

8.2 保証しない動作

本プログラムが保証しない動作を以下に示す.

(1) SlackのOutgoing WebHooks以外からのPOSTリクエストをブロックする動作.

参考文献

[1] Google Inc.: Google Maps Geocoding API, Google (online), available from ⟨https://developers.google.com/maps/documentation/geocoding/start⟩ (accessed 2017-4-17).

[2] Google Inc.: Google Places API, Google (online), available from ⟨https://developers.google.com/places/⟩ (accessed 2017-4-17).

[3] Google Inc.: Google Static Maps API, Google (online), available from ⟨https://developers.google.com/maps/documentation/static-maps/⟩ (accessed 2017-4-17).

[4] Google Inc.: Google URL Shotener API, Google (online), available from ⟨https://developers.google.com/url-shortener/⟩ (accessed 2017-4-17).