ウェブコンピューティング プログラミング課題(松) レポート
学籍番号:B8TB2072
梶原颯希

1. 概要

この課題では、自分のいる場所の近くの駅や食事場所の検索を実現するために、施設名を入力として、最寄りの駅やレストランを出力とするプログラムを作成し、それを Web ブラウザ上で動作するようにした、

なお、サービスの URL は https://desolate-crag-93252.herokuapp.com/ である。

2. 利用した Web サービスの情報

この課題で使用した Web サービスは、HeartRails Express と、Google Maps Platform の Geocoding API、Places API である。当初はリクルート Web サービスのグルメサーチをレストラン検索に用いることを検討していたが、ホットペッパーグルメに登録されている店舗が少ない地域では正確な検索結果が得られないことを考慮して Places API による検索に変更した。

3. ソースコード

シェル上の処理と Web サービスの両方に用いた googlemap_test.py というプログラムのソースコードを以下に示す。

```
import googlemaps
from dotenv import load_dotenv
import sys
import os
import pprint
import heartrails_json
load_dotenv()

def create_shopinfo(index, result):
    print("""------ No. {} ------
【店名】{}
【営業情報】{}
【住所】{}""".format(index+1, # No.
                    result["results"][index]["name"], # 店名
                    isopen(result["results"][index]),
                    result["results"][index]["vicinity"]
                    ))

def info_output(index, result):
    return """------ No. {} ------
```

ウェブコンピューティング プログラミング課題(松) レポート
学籍番号:B8TB2072
梶原颯希

```
【店名】{}
【営業情報】{}
【住所】{}""".format(index+1, # No.
                    result["results"][index]["name"], # 店名
                    isopen(result["results"][index]),
                    result["results"][index]["vicinity"]
                    )

def isopen(search_result):
    if (search_result.get('opening_hours', None) == None):
        return "営業情報が存在しません"
    elif(search_result["opening_hours"]["open_now"] == True):
        return "営業中"
    else:
        return "営業時間外"

def search(place, type, distance=200):
    key = os.environ.get('GOOGLE_API_KEY') # 上記で作成した API キーを入れる
    client = googlemaps.Client(key) #インスタンス生成
    geocode_result = client.geocode(place)
    loc = geocode_result[0]['geometry']['location'] # 軽度・緯度の情報のみ取り出す
    s = "" #string
    if type == "food":
        place_result = client.places_nearby(location=loc, radius=distance,
type='restaurant') #半径 200m 以内のレストランの情報を取得
        search_num = len(place_result["results"])

        for index in range(search_num):
            s += info_output(index, place_result)
            s += "¥n"

        return s
    else:
        return heartrails_json.info_search(loc, s)

def main():
```

ウェブコンピューティング プログラミング課題(松) レポート
学籍番号:B8TB2072
梶原颯希

```
    key = os.environ.get("GOOGLE_API_KEY") #上記で作成したAPIキーを入れる
    client = googlemaps.Client(key) #インスタンス生成
    num_of_val = len(sys.argv)
    geocode_result = client.geocode(sys.argv[1])
    loc = geocode_result[0]['geometry']['location'] # 軽度・緯度の情報のみ取り出す
    if (num_of_val >= 3 and sys.argv[2] == "food"):
        place_result = client.places_nearby(location=loc, radius=200,
type='restaurant') #半径200m以内のレストランの情報を取得
        search_num = len(place_result["results"])

        for index in range(search_num):
            create_shopinfo(index, place_result)
    else:
        heartrails_json.heartrailsAPI(loc)

if __name__ == "__main__":
    main()
```

上記のソースコードで import した heartrails_json.py のソースコードを以下に示す。

```
import sys
import urllib.request
import urllib.parse
import json

def create_station_info(index, station):
    return """------ No, {} ------
【駅名】{}
【路線名】{}
【現在地からの距離】{}""".format(index,
                                station["name"], #駅名
                                station["line"], #路線名
                                station["distance"] #現在地からの距離
                               )


def main():
    url = "http://express.heartrails.com/api/json?{}".format(
        urllib.parse.urlencode(
```

ウェブコンピューティング プログラミング課題(松) レポート
学籍番号:B8TB2072
梶原颯希

```
            {"method" : "getStations",
             "y" : "".join(sys.argv[1]),   # 入力からクエリを生成
             "x" : "".join(sys.argv[2]),   # 入力からクエリを生成
            }))
    #print("URL:",url)
    f_url = urllib.request.urlopen(url).read()
    json_result = json.loads(f_url.decode("utf-8"))

    #output
    print("最寄り駅の一覧を表示します。")
    for index, station in enumerate(json_result["response"]["station"]):
        print(create_station_info(index+1, station))

def heartrailsAPI(loc):
    url = "http://express.heartrails.com/api/json?{}".format(
        urllib.parse.urlencode(
            {"method" : "getStations",
             "x" : loc['lng'],   # 入力からクエリを生成(経度)
             "y" : loc['lat'],   # 入力からクエリを生成(緯度)
            }))
    #print("URL:",url)
    f_url = urllib.request.urlopen(url).read()
    json_result = json.loads(f_url.decode("utf-8"))

    #output
    print("最寄り駅の一覧を表示します。")
    for index, station in enumerate(json_result["response"]["station"]):
        print(create_station_info(index+1, station))

def info_search(loc, s):
    url = "http://express.heartrails.com/api/json?{}".format(
        urllib.parse.urlencode(
            {"method" : "getStations",
             "x" : loc['lng'],   # 入力からクエリを生成
             "y" : loc['lat'],   # 入力からクエリを生成
            }))
```

ウェブコンピューティング プログラミング課題(松) レポート
学籍番号:B8TB2072
梶原颯希

```
    #print("URL:",url)
    f_url = urllib.request.urlopen(url).read()
    json_result = json.loads(f_url.decode("utf-8"))

    #output
    for index, station in enumerate(json_result["response"]["station"]):
        s += create_station_info(index+1, station)
        s += "¥n"

    return s

if __name__ == "__main__":
    main()
```

また、シェル上で実行した際の例を 3 つ掲載する。

% python googlemap_test.py 東北大学青葉山キャンパス food
------ No. 1 ------
【店名】 Kotaro
【営業情報】 営業時間外
【住所】 青葉山みどり厚生会館 2F, 荒巻, 青葉区 仙台市
------ No. 2 ------
【店名】 TOHOKU UNIV.COOP LUNCH BOX & HALAL
【営業情報】 営業情報が存在しません
【住所】 青葉山みどり厚生会館 2F, 荒巻, 青葉区 仙台市
------ No. 3 ------
【店名】 Midori Shokudo
【営業情報】 営業時間外
【住所】 Aoba Aramaki, Aoba Ward, Sendai

% python googlemap_test.py 国会議事堂 food
------ No. 1 ------
【店名】 国会中央食堂
【営業情報】 営業時間外
【住所】 1-chōme-7-1 Nagatachō, Chiyoda City

ウェブコンピューティング プログラミング課題(松) レポート
学籍番号:B8TB2072
梶原颯希

------ No. 2 ------
【店名】 The House of Representatives 2nd building Cafeteria
【営業情報】 営業時間外
【住所】 2-chōme-1-2 Nagatachō, Chiyoda City
------ No. 3 ------
【店名】 Issa Soba
【営業情報】 営業時間外
【住所】 1-chōme-7-1 Nagatachō, Chiyoda City
------ No. 4 ------
【店名】 参議院 議員食堂
【営業情報】 営業時間外
【住所】 参議院本館 2 階, 1-chōme-7－1 Nagatachō, Chiyoda City
------ No. 5 ------
【店名】 Mitou-an
【営業情報】 営業時間外
【住所】 1-chōme-7-1 Nagatachō, Chiyoda City
------ No. 6 ------
【店名】 Yoshinoya
【営業情報】 営業時間外
【住所】 1-chōme-7-1 Nagatachō, Chiyoda City

% python googlemap_test.py 東北大学病院
最寄り駅の一覧を表示します。
------ No, 1 ------
【駅名】 北四番丁
【路線名】 仙台市南北線
【現在地からの距離】 680m
------ No, 2 ------
【駅名】 勾当台公園
【路線名】 仙台市南北線
【現在地からの距離】 1090m
------ No, 3 ------
【駅名】 北仙台
【路線名】 JR 仙山線
【現在地からの距離】 1300m
------ No, 4 ------

ウェブコンピューティング プログラミング課題(松) レポート
学籍番号:B8TB2072
梶原颯希

【駅名】 北仙台
【路線名】 仙台市南北線
【現在地からの距離】 1330m

次に、Web サービスを作るために用いた moyori_search.py のコードを掲載する。

```
from flask import Flask,request, render_template, Markup
import os
import googlemap_test

app = Flask(__name__)

@app.template_filter('cr')
def cr(arg):
    return Markup(arg.replace('¥n', '<br>'))

@app.route('/', methods=['GET'])
def get():
    return render_template('map_search.html', ¥
        title = 'Web Computing Assignment', ¥
        subtitle = '最寄りの施設･駅の検索', ¥
        message = "現在地から近い施設名を入力してください。¥n 下のラジオボタンで検索したい店を選ぶことができます。", ¥
        results = "ここに結果が表示されます")

@app.route('/', methods=['POST'])
def post():
    name = request.form['choice']
    if name == "駅":
        search_type = "station"
    else:
        search_type = "food"
    return render_template('map_search.html', ¥
        title = 'Web Computing Assignment', ¥
        subtitle = '最寄りの施設･駅の検索', ¥
        message = '現在地から近い施設名を入力してください。¥n 下のラジオボタンで検索したい店を選ぶことができます。', ¥
```

ウェブコンピューティング プログラミング課題(松) レポート
学籍番号:B8TB2072
梶原颯希

```
      message2 = request.form["place"] + 'の近くの' + request.form["choice"] +
'の検索結果を表示します', ¥
      results = googlemap_test.search(request.form["place"], search_type,
distance=request.form["sel"]))



if __name__ == '__main__':
   # Bind to PORT if defined, otherwise default to 5000.
   port = int(os.environ.get('PORT', 5000))
   app.run(host='0.0.0.0', port=port)
```

4. 考察など

この Web サービスを作る上で工夫した点は、API キーを環境変数として定義することで、API キーの悪用を防いだことである。最初にプログラムを作る際に直接 API キーを記載していたが、Google Maps Platform の API はリクエスト数が増えると有料になってしまうことを考えて環境変数に定義することにした。

苦労した点は、初めて Flask に触れたため、Heroku 上で正しく動作するように port を指定しなければいけないことに気がつかなかったことや、Places API の検索結果のどこに必要な情報が入っているのかを判断することである。この点については、試行錯誤を重ねて把握することができた。

問題点としては、最初に実装した際に距離の選択肢で「選択してください」のまま最寄り駅を検索すると Bad Request になってしまうことがあった。これについては、選択肢の属性が disabled になっていたことが原因で発生したため、距離の初期値である 200 を value に設定することで解決できた。

最後に、感想として、API を使って何らかの便利なプログラムを作ってみることは初めてだったが、さまざまな資料を参考にすることで Web 上で動的にサイトを生成するところまで実現できたので、かなり達成感を感じた。