RESTful API Project

Driver Application

Course: INFO U762 – 01F

Semester: Spring 2024

Instructor: Dr. Grover Walters

Date: April 22, 2024

David C. Miller

**API Driver App**

**Pulls 3 Public API data sources**

**Pushes the data to the Custom API App**

**Pulls the data from the Custom API App**

----

import pandas as pd

import requests

import json

import geojson

import geopandas

import csv

import objectpath

import numpy as np

----

----

**Weather API Data Retrieval**

----

# API Query to retrieve weather conditions in Los Angeles

----

def fetch\_data\_from\_api(api\_url):

try:

response = requests.get(api\_url)

response.raise\_for\_status() # Raise an exception for 4xx or 5xx status codes

data = response.json()

return data

except requests.exceptions.RequestException as e:

print(f"Error fetching data from API: {e}")

return None

api\_url = "https://api.weather.gov/stations/KCQT/observations/latest?require\_qc=false"

data = fetch\_data\_from\_api(api\_url)

if data:

print(data) # show data collected

else:

print("Failed to fetch data from API.")

----

----

type(data)

----

----

# Converting data from GeoJSON to GeoDataFrame

----

----

gdf = geopandas.GeoDataFrame.from\_features([data])

csv\_filename = 'weather.csv'

gdf.to\_csv(csv\_filename, index=False)

----

----

gdf = gdf.\_\_geo\_interface\_\_

print(gdf)

----

----

keys = gdf.keys()

print(keys)

----

----

items = gdf.items()

print(items)

----

----

values\_list = list(gdf.values())

print(values\_list)

----

----

type(values\_list)

----

----

# Extracting relevant data to pass on

----

----

# Current Conditions STRING Value (CurrentConditionValue)

----

----

tree\_obj = objectpath.Tree(values\_list)

CurrentCondition = tuple(tree\_obj.execute('$..textDescription'))

CurrentConditionValue = CurrentCondition[0]

print(CurrentConditionValue)

type(CurrentConditionValue)

----

----

# Current Temperature FLOAT Value (TempC)

----

----

tree\_obj = objectpath.Tree(values\_list)

temp\_obj = tuple(tree\_obj.execute('$..temperature'))

temp\_value = str(temp\_obj)

tempC = temp\_value[39:]

tempC = tempC[:4]

tempC = float(tempC)

print(tempC)

type(tempC)

----

----

# Converting Temperature FLOAT Value to (TempF)

----

----

tempF = round((tempC \* (9/5)) + 32, 1)

print(tempF)

----

**Currency API Data Retrieval to get EUR exchange Rate**

----

def fetch\_data\_from\_api(api\_url):

try:

response = requests.get(api\_url)

response.raise\_for\_status() # Raise an exception for 4xx or 5xx status codes

data = response.json()

return data

except requests.exceptions.RequestException as e:

print(f"Error fetching data from API: {e}")

return None

api\_url = "https://open.er-api.com/v6/latest/USD"

data = fetch\_data\_from\_api(api\_url)

if data:

print(data) # or do whatever you want with the data

else:

print("Failed to fetch data from API.")

----

----

type(data)

----

----

# Extracting current Eur rates per USD and converting $1000 EUR to USD to pass on

----

----

rates\_data = {}

EUR = 1000

USD = 0

for k in data.keys():

if 'rates' in k:

rates\_data[k] = data[k]

EUREXCH = rates\_data['rates']['EUR']

print(EUREXCH)

print(EUR)

USD = round(EUR \* EUREXCH, 2)

print(USD)

----

**BrewPub API Data Retrieval to identify a random Brew-Pub in LA**

----

# API Query to retrieve a list of Breweries of all types in Los Angeles

----

----

def fetch\_data\_from\_api(api\_url):

try:

response = requests.get(api\_url)

response.raise\_for\_status() # Raise an exception for 4xx or 5xx status codes

data = response.json()

return data

except requests.exceptions.RequestException as e:

print(f"Error fetching data from API: {e}")

return None

api\_url = "https://api.openbrewerydb.org/v1/breweries/search?query=los%20angeles"

data = fetch\_data\_from\_api(api\_url)

if data:

print(data) # or do whatever you want with the data

else:

print("Failed to fetch data from API.")

----

----

# Process to select BrewPubs, which have beer and food, and eliminate BrewPubs that do not have website or phone numbers listed

# Lastly, if there are more than 1 return, I randomly pick one from the returns

----

----

df = pd.DataFrame(data)

#df = pd.DataFrame(data, columns=['id', 'name', 'brewery\_type', 'address\_1', 'address\_2', 'address\_3', 'city', 'state\_province', 'postal\_code', 'country', 'longitude', 'latitude', 'phone', 'website\_url', 'state', 'street'])

df.to\_csv("brewery.csv", index = False)

df\_1 = df[df['website\_url'].notna()]

df\_2 = df\_1[df\_1['phone'].notna()]

df\_3 = df\_2[df\_2['brewery\_type'] == 'brewpub']

df\_4 = df\_3.sample(n=1)

print(df\_4)

----

----

#Extracting relevent data to pass on

----

----

BrewName = df\_4.iat[0,1]

BrewPhone = df\_4.iat[0,12]

BrewURL = df\_4.iat[0,13]

print(BrewName, BrewPhone, BrewURL)

----

----

#Confirming that all values are present

print(CurrentConditionValue, tempF,USD, BrewName, BrewPhone, BrewURL)

----

**Custom API Put / Update**

----

#Update Custom API with API UPDATE

----

----

BASE = " http://127.0.0.1:5000/"

APP\_VERSION = "v1/"

data = [{'wtemp': tempF, 'wdescription': CurrentConditionValue, 'currency': USD, 'brewname': BrewName, 'brewurl': BrewURL, 'brewphone': BrewPhone}]

for i in range(len(data)):

response = requests.patch(BASE + APP\_VERSION + "data/" + str(i), data[i])

print(response.json())

input()

response = requests.patch(BASE + APP\_VERSION + "data/2")

print(response.json())

----

----

#Confirm Update with Customer API GET

----

+\*In[44]:\*+

[source, ipython3]

----

BASE = " http://127.0.0.1:5000/"

APP\_VERSION = "v1/"

response = requests.get(BASE + APP\_VERSION + "data/0")

print(response.json())