MongoDb Flask API - Python

OOPs concepts are extensively used to increase reusablity of code

Every Step is logged into a file using custom logging class¶

By Arjun Panwar

Follow me on LinkedIn: https://www.linkedin.com/in/arjun-panwar/

Flask App

In [1]:
```
from flask import Flask, render_template, request, jsonify
app = Flask(__name__)
```

MongoDB Route

In [ ]:
```
@app.route('/mongodb_postman', methods=['POST'])  # for calling the API from Postman/S
def mongodb_via_postman():
    if (request.method == 'POST'):
        operation = request.json['operation']
        url = request.json['url']
        db = request.json['db']
        ob = mongodb(url,db)
        collection_name = request.json['collection_name']


        if (operation == 'create'):
            '''
            for creating collection
            JSON format
            {
                "operation":"create",
                "url":connection url
                "db" : db name
                "collection_name": collection name
            }
            '''

            ob.create_collection(collection_name)
            msg = "Table created"
        elif (operation == 'insert'):
            '''
            for inserting in collection
            JSON format
            {
                "operation":"create",
                "url":connection url
                "db" : db name
                "collection_name": collection name
                "record": for single record a dict,for many record list of dict
            }
            '''
            record = request.json['record']
            ob.insert(collection_name,record)
            msg = "data inserted"
        elif (operation == 'update'):
            '''
            for updating collection
            JSON format
            {
                "operation":"create",
```

```
                "url":connection url
                "db" : db name
                "collection_name": collection name

                "set": "key=value pair of columns & values to be updated"
                "where": "condition"
            }
            '''
            set = request.json['set']
            where = request.json['where']

            ob.update(collection_name,set, where)
            msg = "data updated"


        if (operation == 'delete'):
            '''
           for deleting record
           JSON format
           {
               "operation":"create",
                "url":connection url
                "db" : db name
                "collection_name": collection name
               "where": "condition"
           }
           '''

            where = request.json['where']

            ob.delete(collection_name, where)
            msg = "data deleted"

        if (operation == 'download'):
            '''
                        for downloading table
                        JSON format
                        {
                             "operation":"create",
                             "url":connection url
                             "db" : db name
                             "collection_name": collection name


                        }
                        '''
            link = ob.download(collection_name)
            msg = "you can download data using this link :   http://127.0.0.1:5000/" +

        return jsonify(msg)
```

Treminating Flask App

In [ ]:

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

Logging Class

In [1]:

```
from datetime import datetime  # importing DateTime package


class App_Logger:
    '''
    It is used save logs into a file

    Parameters
```

```
    ----------
    file: log file name Default is logfile.log

    '''

    def __init__(self, file="logfile.log"):
        self.f_name = file

    def log(self, log_type, log_msg):
        '''
        Function log to save logs and log type in file

        Parameters
        ----------
        log_type: Type of log-info,error,warning etc
        log_msg: Log to be saved(message)

        '''
        now = datetime.now()  # current time
        current_time = now.strftime("%d-%m-%Y %H:%M:%S")  # changing time format
        f = open(self.f_name, "a+")  # opening file in append + mode
        f.write(current_time + "," + log_type + "," + log_msg + "\n")  # writing log t
        f.close()  # closing log file
```

MongoDB Main Class

In [ ]:

```
import pymongo
import pandas as pd

class mongodb:
    '''
    mongodb class through which we can perform most of the mongodb tasks using python

    Parameters
    ----------
    connection_url: connection url with password
    db:db name
    '''

    def __init__(self, connection_url,db):
        '''
        init function of sql class
        '''

        # Establish a connection with mongoDB
        self.client = pymongo.MongoClient(connection_url)
        # Create a DB
        self.db = self.client[db]

        self.logger = App_Logger("mongodb_logs.txt")  # creating App_Logger object
        self.logger.log("info", "mongodb object created")  # logging


    def create_collection(self, COLLECTION_NAME):
        '''
        Function create_ table is used to create a new table

        Parameters
        ----------
        COLLECTION_NAME: collection name
        '''
        try:
            self.db[COLLECTION_NAME]
            self.logger.log("info", f"{COLLECTION_NAME}  collection created")  # loggi
        except Exception as e:

            self.logger.log("error", f"collectionqw not created error : {str(e)}")  #
```

```
    def insert(self, collection_name,record):
        '''
        Function insert is used to insert value in table

        Parameters
        ----------
        record: data to be inserted as dict, to insert many data use list of dict
        '''
        try:
            if type(record)==dict:
                collection = self.db[collection_name]
                collection.insert_one(record)
            elif type(record)==list:
                collection = self.db[collection_name]
                collection.insert_many(record)
            self.logger.log("info", f"inserted successfully")  # logging
        except Exception as e:

            self.logger.log("error", f"insert error : {str(e)}")  # logging


    def update(self, collection_name,new_dict,where_dict):
        '''
        Function delete is used to delete record from collection

        Parameters
        ----------
        collection_name: collection name
        where_dict: condition as dict
        new_dict:new values
        '''
        try:

            collection = self.db[collection_name]

            collection.update_many(where_dict,{"$set":new_dict} )
            self.logger.log("info", f"update successfully")  # logging
        except Exception as e:
            self.logger.log("error", f"update error : {str(e)}")  # logging


    def delete(self, collection_name,where_dict):
        '''
        Function delete is used to delete record from collection

        Parameters
        ----------
        collection_name: collection name
        where_dict: condition as dict
        '''
        try:
            query_to_delete = where_dict
            collection = self.db[collection_name]

            collection.delete_one(query_to_delete)
            self.logger.log("info", f"deleted successfully")  # logging
        except Exception as e:
            self.logger.log("error", f"delete error : {str(e)}")  # logging


    def download(self,collection_name):

        # make an API call to the MongoDB server
        collection = self.db[collection_name]
        mongo_docs = collection.find()

        # Convert the mongo docs to a DataFrame
        docs = pd.DataFrame(mongo_docs)
        # Discard the Mongo ID for the documents
        docs.pop("_id")
```

```
        #df = pd.read_sql_query(f"SELECT * FROM {table_name}", self.conn())
        docs.to_csv(f"{collection_name}.csv",index=False)
        return f"{collection_name}.csv"
```

Sample JSON Input

create

{ "operation":"insert", "url":enter your mongodb url here , "db":"testdb", "collection_name":"test_coll" }

insert

{ "operation":"insert", "url":"mongodb+srv://arjun:password@cluster03.mongodb.net/myFirstDatabase?retryWrites=true&w=majority", "db":"testdb", "collection_name":"test_coll", "record":{"companyName": "ERF","name":"Arjun"}, }

insert many

{ "operation":"insert", "url":"mongodb+srv://arjun:password@cluster03.mongodb.net/myFirstDatabase?retryWrites=true&w=majority", "db":"testdb", "collection_name":"test_coll", "record":[{"companyName": "ERF","name":"Abhay"},{"companyName": "ERF","name":"Badal"},{"companyName": "ERF","name":"Arjun"}],

}

update

{ "operation":"update", "url":"mongodb+srv://arjun:password@cluster03.mongodb.net/myFirstDatabase?retryWrites=true&w=majority", "db":"testdb", "collection_name":"test_coll", "set":{"name":"rachit"}, "where":{"name":"Arjun"} }

delete

{ "operation":"delete", "url":"mongodb+srv://arjun:pass@cluster0.mongodb.net/myFirstDatabase?retryWrites=true&w=majority", "db":"testdb", "collection_name":"test_coll", "where":{"name":"Abhay"} }

Download

{ "operation":"insert", "url":enter your mongodb url here , "db":"testdb", "collection_name":"test_coll" }

In [ ]: