

Flask-API Task

Flask-API

CRUD

Create
Read
Update
Delete

Create APIs for:

- 1, creating Table
- 2, Inserting data into table
- 3, update data in table
- 4, bulk inserting data into table
- 5, Delete from table

→ MySQL
→ MongoDB
→ Cassandra

***NG rock
↓
to make API as Global

6, → for download data from the table.
(select * from)

$6 \times 3 = 18$ APIs.

@app.route

Json format

table: tablename

col1: text1

col2: col2

2D: text 2D

pass: password.

We can use 'mixed JSON'
to give col data types

We can also use:

@app.route('/table/create')

@app.route('/table/update')

Databases - APIs

Flask - API Task

↳ Giving the credentials in .py file & using them in our flask-API code.

```

Databases APIs credspy
Project
  Databases APIs [Databases API]
    credspy
    credtest.py
    flask API Task.pdf
    mongodb_APIs.py
    mongotestdata.csv
    mydata.csv
    mysql_APIs.py
    test_databaseclass.py
    testdata.csv
  External Libraries
  Scratches and Consoles

credtest.py
1 """ This contains the credentials for the MySQL database. """
2 sqlhost = 'localhost'
3 sqluser = 'root'
4 sqlpassword = 'MySQL@123'

credspy
1 import mysql
2 from flask import Flask, render_template, request, jsonify
3
4 import creds
5
6 # install mysql-connector-python
7 import mysql.connector as connection
8 from mysql.connector import DatabaseError
9
10 app = Flask(__name__)
11
12 """Here we are connecting to the database but getting credentials from other python file (creds.py)
13 instead of getting from the postman (to make sure that no one can see our creds in the postman)"""
14
15 @app.route('/mysql/testconnection', methods=['POST'])
16 def db_connection():
17     if request.method == 'POST':
18         mydb = connection.connect(host=creds.sqlhost, user=creds.sqluser, passwd=creds.sqlpassword, use_pure=True)
19         res = mydb.is_connected()
20         result = 'The connection to MySQL database is established: ' + str(res)
21         return jsonify(result)
22
23 if __name__ == '__main__':
24     app.run(debug=True)
  
```

② Creating a class & then using it to connect to db by getting the host, user, ID, dbname from postman.

```

Databases APIs databaseclass.py
1 class sql():
2     def __init__(self, host, user, password):
3         self.host = host
4         self.user = user
5         self.password = password
6         self.mydb = connection.connect(host=self.host, user=self.user, passwd=self.password, use_pure=True)
7         self.cur = self.mydb.cursor()
8
9     def db_connection(self, dbname):
10         self.dbname = dbname
11         query = f"Create database {self.dbname}"
12         self.cur.execute(query)
13         print(f"Database '{self.dbname}' is created")
14
15 app = Flask(__name__)
16 @app.route('/mysql/oops_createdb/', methods=['POST'])
17 def db_connect():
18     if request.method == 'POST':
19         hostn = request.json['hostname']
20         userr = request.json['userID']
21         password = request.json['pwd']
22         dbn = request.json['dbn']
23         try:
24             sqlcreate = sql(hostn, userr, password)
25             sqlcreate.db_connection(dbn)
26
27             return jsonify(f"Database {dbn} is created")
28         except Exception as err:
29             return jsonify('Error: ' + str(err))
30
31 if __name__ == '__main__':
32     app.run(debug=True)
  
```

Database: MySQL-APIs / mysql_oops_createdb

POST http://127.0.0.1:5000/mysql/oops_createdb/

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ▼

```

1 {
2   "hostname": "localhost",
3   "userID": "root",
4   "pwd": "MySQL@123",
5   "dbn": "sql7894"
6 }
  
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON ▼

1 "Error: 1007 (HY000): Can't create database 'sql7894'; database exists"

Creating a file to write the classes, other file to write the Ap2.

③

```

classfile.py
1 import mysql.connector as connection
2 class sql():
3     def __init__(self, host, user, password):
4         self.host = host
5         self.user = user
6         self.password = password
7         self.mydb = connection.connect(host=self.host, user=self.user, passwd=self.password, use_pure=True)
8         self.cur = self.mydb.cursor()
9
10    def db_connection(self, dbname):
11        self.dbname = dbname
12        query = f"Create database {self.dbname}"
13        self.cur.execute(query)
14        print(f"Database '{self.dbname}' is created")
  
```

Database: MySQL-APIs / mysql_oops_createdb2

POST http://127.0.0.1:5000/mysql/oops_createdb2/

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ▼

```

1 {
2   "hostname": "localhost",
3   "userID": "root",
4   "pwd": "MySQL@123",
5   "dbn": "sql7894"
6 }
  
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON ▼

1 "Error: 1007 (HY000): Can't create database 'sql7894'; database exists"

classfile_api.py

```

1 import mysql
2 import csv
3 import pandas as pd
4 from flask import Flask, render_template, request, jsonify
5
6 # install mysql-connector-python
7 import mysql.connector as connection
8 from mysql.connector import DatabaseError
9 import classfile
10
11 app = Flask(__name__)
12
13 @app.route('/mysql/oops_createdb2/', methods=['POST'])
14 def db_connect():
15     if request.method == 'POST':
16         hostn = request.json['hostname']
17         userr = request.json['userID']
18         password = request.json['pwd']
19         dbn = request.json['dbn']
20         try:
21             sqlcreate = classfile.sql(hostn, userr, password)
22             sqlcreate.db_connection(dbn)
23             return jsonify(f"Database {dbn} is created")
24         except Exception as err:
25             return jsonify('Error: ' + str(err))
26
27 if __name__ == '__main__':
28     app.run(debug=True)
  
```

Enter data → Send to server → The server Executes & returns the Result.

Result

Here we give all the details in the postman.
↓
(host, user, pwd, dbname). → Server gets those details from the postman, executes & returns the result.

(I had used this approach to design all the APIs for this task)

```
364
365 @app.route('/mysql/dropdb', methods=['POST'])
366 def drop_db():
367     if (request.method == 'POST'):
368         hostname = request.json['hostname']
369         usern = request.json['userID']
370         password = request.json['pwd']
371         db = request.json['dbname']
372
373         mydb = connection.connect(host=hostname, user=usern, passwd=password, use_pure=True)
374         cur = mydb.cursor()
375
376         try:
377             cur.execute(f"drop database {db}")
378             # mydb.commit()
379             return jsonify(f"Database: {db} is dropped")
380
381         except Exception as err:
382             return jsonify('Error: ' + str(err))
383
384
385 if __name__ == '__main__':
386     app.run(debug=True)
387
```

POST mysqloops.created POST mysql_dropdb

Database: MySQL-APIs / mysql_dropdb

POST http://127.0.0.1:5000/mysql/dropdb

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1
2 {
3   "hostname": "localhost",
4   "userID": "root",
5   "pwd": "MySQL@123",
6   "dbname": "sqldb99"
7 }
```