SESSION 7

Flask REST API

Install Module, Create Database, Create API

Data Science Program

Purwadhika
Startup and Coding School

SESSION 7

Flask REST API

Python Module to Create API

Purwadhika
Startup and Coding School

PYTHON MODULE TO CREATE API

Flask

Requests

http for humans

How To Install :

Open CMD (Windows) or terminal (Linux/MacOS) then type/write code below :

- pip install flask flask-mysqldb
- pip install request

Purwadhika

Startup and Coding School

SESSION 7

Flask REST API

Create Simple API

Purwadhika
Startup and Coding School

STEP 1 : CREATE DATABASE

```
1  create database flask_mysql ;
2
3  use flask_mysql ;
4
5  create table employee (
6      id int auto_increment primary key,
7      username varchar(20) unique,
8      name varchar(20),
9      gender enum ('M','L'),
10     married tinyint
11 );
```

Create Database “flask_mysql”

Use Database “flask_mysql”

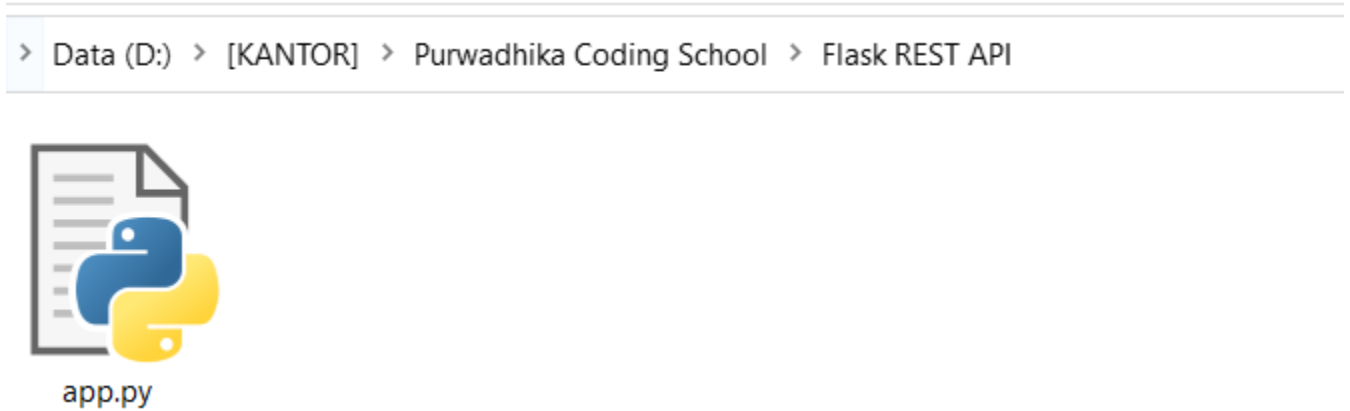
Create table “employee” in “flask_mysql” database, consist of :

1. id (integer type, primary key)
2. username (varchar type, unique key)
3. name (varchar type)
4. Gender (enum type)
5. Married (tinyint type)

Purwadhika
Startup and Coding School

STEP 2 : CREATE API

1. Create a folder.
2. Create a file with python extension (.py). Try by create a file namely “app.py”



3. Import module that has been installed before.

```
# import library yang akan digunakan
from flask import Flask, request, jsonify
from flask_mysqldb import MySQL
```

Purwadhika
Startup and Coding School

STEP 2 : CREATE API

4. Create an object (name it with "app"). This object will then be used as the main object to be able to run the API

```
# membuat object app, sebagai object utama untuk menjalankan API
app = Flask(__name__)
```

5. **Configure to connect to MySQL so that it can connect to the database**

```
# config untuk terhubung dengan mysql
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'Mysql123'
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_DB'] = 'flask_mysql'
```

Purwadhika
Startup and Coding School

STEP 2 : CREATE API

6. Create a config to get data from MySQL as list of dictionaries. (optional)

```
# config optional untuk mendapatkan data dari mysql dalam bentuk list of dictionary
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'
```

7. **Create an object to connect to MySQL.**

```
# membuat object untuk membuat koneksi terhadap mysql
mysql = MySQL(app)
```

8. **Create a script to run API.**

```
# running api
if __name__ == '__main__':
    app.run(debug=True)
```

Purwadhika
Startup and Coding School

FULL CODE CREATE API

```python
# import library yang akan digunakan
from flask import Flask, request, jsonify
from flask_mysqldb import MySQL

# membuat object app, sebagai object utama untuk menjalankan API
app = Flask(__name__)

# config untuk terhubung dengan mysql
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'Mysql123'
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_DB'] = 'flask_mysql'

# config optional untuk mendapatkan data dari mysql dalam bentuk list of dictionary
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'

# membuat object untuk membuat koneksi terhadap mysql
mysql = MySQL(app)

##############################################
#         Route Akan Dibuat Di Sini          #
##############################################

#
# running api
if __name__ == '__main__':
    app.run(debug=True)
```

Purwadhika
Startup and Coding School

SESSION 7

Flask REST API

Route Get, Insert, Patch, and Delete Data

Purwadhika
Startup and Coding School

ROUTE TO GET DATA

1. Create a route that connects to the table in the previously configured database. In this case what will be used is the "employee" table in the database "flask_mysql"

```
# membuat route untuk melakukan operasi penambahan data (Create) dan membaca data (Read) terhadap mysql.
# route hanya dapat di akses dengan method GET dan POST
@app.route('/employee', methods=['GET', 'POST'])
def employee():
```

2. Use request method “GET”

```
# jika request yang datang menggunakan method GET
if request.method == 'GET':
```

Purwadhika
Startup and Coding School

ROUTE TO GET DATA

3. In "GET“ request method , make a connection to MySQL then perform a query. Convert the data obtained into json.

```
# membuka koneksi ke database
cur = mysql.connection.cursor()
# menjalankan query mysql
cur.execute('''SELECT * FROM employee''')
# hasil dari query akan tersimpan di results
results = cur.fetchall()
# mengirim response berisi data hasil query
# merubah data menjadi json sebelum di kirim
return jsonify(results)
```

Purwadhika
Startup and Coding School

ROUTE TO GET DATA

```python
@app.route('/employee', methods=['GET', 'POST'])
def employee():

    # jika request yang datang menggunakan method GET
    if request.method == 'GET':
        # membuka koneksi ke database
        cur = mysql.connection.cursor()
        # menjalankan query mysql
        cur.execute('''SELECT * FROM employee''')
        # hasil dari query akan tersimpan di results
        results = cur.fetchall()
        # mengirim response berisi data hasil query
        # merubah data menjadi json sebelum di kirim
        return jsonify(results)
```

Purwadhika
Startup and Coding School

ROUTE TO POST DATA

1. Still with the same router as the previous "GET" request method, all you need to add is the "POST" request method section.

```
# jika request yang datang menggunakan method POST
elif request.method == 'POST':
```

2. Make a variable namely "form" which consist of "request form" command. It will be declared as dictionary type. This variable will be used to get every fields in the 'employee' table.

```
# data yang dikirim saat request akan terdapat pada request.form
# data pada request.form diubah menjadi bentuk dictionary agar mudah dikelola
form = dict(request.form)

# menyimpan setiap satu data yang dikirim ke dalam variable
username = form['username']
name = form['name']
gender = form['gender']
# data dikirim dalam bentuk string, maka dari itu harus di ubah ke boolean secara manual
married = bool(form['married'])
```

Purwadhika
Startup and Coding School

ROUTE TO POST DATA

3. Just like "GET" method, perform a query for "POST" method. In this case, “insert into” query command will be used.

```
# membuat query untuk input data, dimana semua karakter %s akan digantikan oleh data yang ada di tuple user
sql = "INSERT INTO employee (username, name, gender, married) VALUES (%s,%s,%s,%s)"
# data yang akan disimpan
data = (username, name, gender, married)
```

4. Last is to connect to the database, then run the previous query. Don't forget to close the connection to the database after running the query

```
# membuka koneksi ke database
cur = mysql.connection.cursor()
# menjalankan query mysql
cur.execute(sql,data)

# menutup koneksi ke mysql (untuk patch dan delete)
mysql.connection.commit()
cur.close()

return jsonify({"message" : "Insert Success"})
```

ROUTE TO POST DATA

```
# jika request yang datang menggunakan method POST
elif request.method == 'POST':

    # data yang dikirim saat request akan terdapat pada request.form
    # data pada request.form diubah menjadi bentuk dictionary agar mudah dikelola
    form = dict(request.form)

    # menyimpan setiap satu data yang dikirim ke dalam variable
    username = form['username']
    name = form['name']
    gender = form['gender']
    # data dikirim dalam bentuk string, maka dari itu harus di ubah ke boolean secara manual
    married = bool(form['married'])

    # membuat query untuk input data, dimana semua karakter %s akan digantikan oleh data yang ada di tuple user
    sql = "INSERT INTO employee (username, name, gender, married) VALUES (%s,%s,%s,%s)"
    # data yang akan disimpan
    data = (username, name, gender, married)

    # membuka koneksi ke database
    cur = mysql.connection.cursor()
    # menjalankan query mysql
    cur.execute(sql,data)

    # menutup koneksi ke mysql (untuk patch dan delete)
    mysql.connection.commit()
    cur.close()

    return jsonify({"message" : "Insert Success"})
```

ROUTE TO POST DATA

1. Create a route that leads to the table in database that have configured before. In this case table namely “employee” in the “flask_mysql” database will be used. In addition to patching and deleting data, data delivery will go through the route path variable "id". Then, create the request method "PATCH" in this router.

```
# membuat route untuk melakukan operasi pengubahan data (Update) dan delete data (Delete) terhadap mysql.
# route hanya dapat di akses dengan method PATCH dan DELETE
# khusus route ini, dapat mengirim data (angka) via route path yang disimpan di variable 'id'
@app.route('/employee/<id>', methods=['PATCH', 'DELETE'])
def employeeid(id): # function ini menerima satu parameter yaitu 'id'

    # jika diakses menggunakan method PATCH
    if request.method == 'PATCH':
```

Purwadhika
Startup and Coding School

ROUTE TO POST DATA

2. Change the data received into a dictionary type. After that, do the same as in the previous "POST" request method to define each variable as listed in the table in the database.

```
# data yang diterima harus diubah menjadi dictionary agar mudah di proses
form = dict(request.form)

# simpan masing - masing data dalam satu variable
username = form['username']
name = form['name']
gender = form['gender']
married = form['married']
```

Purwadhika
Startup and Coding School

ROUTE TO POST DATA

3. Just like before, make a query to update data based on the employee's “id”. Connect to MySQL and then run the query. Finally, close the connection to MySQL and respond to the update in json form.

```
# membuat query untuk mengupdate data berdasarkan id employee
sql = f"UPDATE employee SET username='{username}', name='{name}', gender='{gender}', married={married} WHERE id = {id}"

# membuka koneksi ke mysql
cur = mysql.connection.cursor()
# running query mysql
cur.execute(sql)

# menutup koneksi ke mysql (untuk delete, patch, dan delete)
mysql.connection.commit()
cur.close()

# memberikan respon dalam bentuk json
return jsonify({"message" : "Update Success", "user_id" : id})
```

ROUTE TO POST DATA

```python
@app.route('/employee/<id>', methods=['PATCH', 'DELETE'])
def employeeid(id): # function ini menerima satu parameter yaitu 'id'

    # jika diakses menggunakan method PATCH
    if request.method == 'PATCH':

        # data yang diterima harus diubah menjadi dictionary agar mudah di proses
        form = dict(request.form)

        # simpan masing - masing data dalam satu variable
        username = form['username']
        name = form['name']
        gender = form['gender']
        married = form['married']

        # membuat query untuk mengupdate data berdasarkan id employee
        sql = f"UPDATE employee SET username='{username}', name='{name}', gender='{gender}', married={married} WHERE id = {id}"

        # membuka koneksi ke mysql
        cur = mysql.connection.cursor()
        # running query mysql
        cur.execute(sql)

        # menutup koneksi ke mysql (untuk delete, patch, dan delete)
        mysql.connection.commit()
        cur.close()

        # memberikan respon dalam bentuk json
        return jsonify({"message" : "Update Success", "user_id" : id})
```

ROUTE TO POST DATA

1. The router used is still the same as "PATCH“ before. Add a request method "DELETE" in this section.

```
# jika request menggunakan method DELETE
elif request.method == 'DELETE':
```

2. Write query to delete data in MySQL by making "id" as the main parameter to connect to data. Open a connection to MySQL and then run the previous query. Finally, close the connection to MySQL after running the query and provide a response in json type as before

```
# membuat query untuk mengupdate data berdasarkan id employee
sql = f"DELETE FROM employee WHERE id = {id} "

# membuka koneksi ke mysql
    cur = mysql.connection.cursor()
    # running query mysql
    cur.execute(sql)

# menutup koneksi ke mysql (untuk delete, patch, dan delete)
    mysql.connection.commit()
    cur.close()

# memberikan respon dalam bentuk json
return jsonify({"message" : "Delete Success", "user_id" : id})
```

Purwadhika
Startup and Coding School

ROUTE TO POST DATA

```
# jika request menggunakan method DELETE
elif request.method == 'DELETE':

# membuat query untuk mengupdate data berdasarkan id employee
sql = f"DELETE FROM employee WHERE id = {id} "

# membuka koneksi ke mysql
    cur = mysql.connection.cursor()
    # running query mysql
    cur.execute(sql)

# menutup koneksi ke mysql (untuk delete, patch, dan delete)
    mysql.connection.commit()
    cur.close()

# memberikan respon dalam bentuk json
return jsonify({"message" : "Delete Success", "user_id" : id})
```