Module 02

API with Flask

Data Science Developer

Purwadhika
Startup and Coding School

Membuat database

```sql
create database flask_mysql;

use flask_mysql;

create table employee (
    id int auto_increment primary key,
    username varchar(20) unique,
    name varchar(20),
    gender enum('M', 'F'),
    married tinyint
);
```

Purwadhika
Startup and Coding School

FLASK, FLASK-MYSQLDB, REQUESTS

```
pip install flask flask-mysqldb
pip install requests
```

Purwadhika
Startup and Coding School

Mulai membuat API

Buatlah sebuah folder, kemudian buat satu file bernama app.py

```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'] = 'try_flask'

# 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)

# Selanjutnya kode akan di tambahkan disini
# di antara baris 18 dan 24
# # # # # # # # # # # # # # # # # # # # # #

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

Purwadhika
Startup and Coding School

Membuat route untuk get data

```python
# 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():

    # 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 menjasi json sebelum di kirim
        return jsonify(results)
```

Purwadhika
Startup and Coding School

Melakukan request untuk get data (jupyter notebook)

```
# import library yang ingin digunakan
import requests
```

```
# GET ALL EMPLOYEE
# request ini di tujukan untuk mengambil seluruh data employee
result = requests.get('http://localhost:5000/employee')
# karena response dari API dalam bentuk json
# kita harus menggunakan function json untuk memproses datanya
result = result.json()
print(result)
```

Purwadhika
Startup and Coding School

Membuat route untuk insert 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 delete sama patch)
    mysql.connection.commit()
    cur.close()

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

Melakukan request untuk insert data

```
# POST ONE EMPLOYEE

# data yang akan disimpan
employee = {
    "username" : "deno",
    "name" : "Ryan Node",
    "gender" : "M",
    "married" : True
}

# melakukan proses request ke API dan mendapatkan response
# response dari API akan disimpan ke variable result
result = requests.post(
    'http://localhost:5000/employee',
    data = employee
)

# karena response dikirim dalam bentuk json
# kita gunakan function json untuk meng-extract
result = result.json()

print(result)
```

Purwadhika
Startup and Coding School

Membuat route untuk patch data

```python
# 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'

    if request.method == 'PATCH':

        form = dict(request.form)

        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}"

        cur = mysql.connection.cursor()
        cur.execute(sql)
        mysql.connection.commit()
        cur.close()

        return jsonify({"message" : "Update Success", "user_id" : id})
```

Purwadhika
Startup and Coding School

Membuat request untuk patch data

```
# UPDATE ONE EMPLOYEE WITH ID

# data yang akan disimpan
employee = {
    "username" : "notebook",
    "name" : "Ryan Note Edited",
    "gender" : "M",
    "married" : False
}

# melakukan proses request ke API dan mendapatkan response
# response dari API akan disimpan ke variable result
# melakukan update untuk employee dengan id = 9
result = requests.patch(
    'http://localhost:5000/employee/9',
    data = employee
)

# karena response dikirim dalam bentuk json
# kita gunakan function json untuk meng-extract
result = result.json()

print(result)
```

Purwadhika
Startup and Coding School

Membuat route untuk delete 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})
```

Purwadhika
Startup and Coding School

Membuat request untuk delete data

```
# DELETE ONE EMPLOYEE WITH ID

# melakukan proses request ke API dan mendapatkan response
# response dari API akan disimpan ke variable result
# melakukan delete untuk employee dengan id = 9
result = requests.delete(
    'http://localhost:5000/employee/9'
)

# karena response dikirim dalam bentuk json
# kita gunakan function json untuk meng-extract
result = result.json()

print(result)
```

Purwadhika
Startup and Coding School