Introduction to Web Programming with Flask

November 3, 2023

1 Introduction to Web Programming with Flask

What is a Microframework ?

Why are Microframeworks popular ?

Why Flask for Python web development ?

Setting up development environment for Flask

1. Install virtualenv and activate the virtualenv called web_dev.
2. Install Flask using pip install flask
3. Install visual studio code.

Simple Hello World Application in Flask

[3]:
```
from flask import Flask

app = Flask(__name__)  # Create an instance of a Flask Application

@app.route("/")        # Application Routing
def hello_world():
    return "<p>Hello, World!</p>"
```

[ ]:
```
run the app as

(web_dev) $ flask --app hello_app run
```

2 Introduction to REST Framework

REST Framework enables applications to manage state using HTTP as the transport protocol.

Standard:

Method /api/resource defines ways to access a resource.

Example:

Using the URL endpoint /api/employees to manage all the APIs for employees.

POST /api/employees - Create an employee PUT /api/employees - Alter an employee information GET /api/employees - Get information about employees. PATCH /api/employees - Alter an

employee information DELETE /api/employees - Remove an employee information

Status codes are used to indicate application states.

1. 200 - Successful
2. 400 - Data input errors / Auth errors. [User input data errors]
3. 500 - Application state error. [Severe errors]

[ ]:
```
Sample application for multiple HTTP methods.
```

[5]:
```
from flask import Flask

app = Flask(__name__)

@app.route("/", methods=["GET"])
def hello_get():
    return "<p>Hello, GET!</p>"

@app.route("/", methods=["POST"])
def hello_post():
    return "<p>Hello, POST!</p>"

@app.route("/", methods=["PUT"])
def hello_put():
    return "<p>Hello, PUT!</p>"

@app.route("/", methods=["PATCH"])
def hello_patch():
    return "<p>Hello, PATCH!</p>"

@app.route("/", methods=["DELETE"])
def hello_delete():
    return "<p>Hello, Delete!</p>"
```

Accessing Path variables

[7]:
```
from flask import Flask

app = Flask(__name__)

employees = {
    1: "Prashanth",
    2: "Shiva",
    3: "Phani",
    4: "Pranav"
}

@app.route("/<employeeName>", methods=["GET"])
def hello_employee(employeeName):
```

```
    return "<p>Hello, {}!</p>".format(employeeName)

@app.route("/<int:employee_id>", methods=["GET"])
def hello_employee_var(employee_id):
    return "<p>Hello, {}!</p>".format(employees.get(employee_id,
                                                    "Person Not Present"))
```

A simple REST Endpoint for the Employee Resource.

[9]:
```
from flask import Flask
from flask import request, make_response

app = Flask(__name__)

class Employee:
    name = ''
    id = ''
    address = ''

    def __init__(self, name, id, address):
        self.name = name
        self.id = id
        self.address = address

    def __str__(self):
        return str({
            "id": self.id,
            "name": self.name,
            "address": self.address
        })

employees = {
    1: Employee("Prashanth", 1, "Bengaluru"),
    2: Employee("Shiva", 2, "Bengaluru"),
    3: Employee("Phaneendra", 3, "Mysore"),
    4: Employee("Pranav", 4, "Mysore"),
}

count = 4

@app.route("/api/employee/", methods=["POST"])
def create_employee():
    global count
    employee = request.json
    count += 1
    employee['id'] = count
    employees[count] = Employee(employee['name'], count, employee['address'])
```

```
    return employee

@app.route("/api/employee/<int:employee_id>", methods=["PUT"])
def alter_employee_data(employee_id):
    employee = request.json
    employees[employee_id].name = employee.get('name', employees[employee_id].
 ↪name)
    employees[employee_id].address = employee.get('address',␣
 ↪employees[employee_id].address)
    return str(employees[employee_id])

@app.route("/api/employee/<int:employee_id>", methods=["GET"])
def get_employee_information(employee_id):
    return str(employees[employee_id])

@app.route("/api/employee/<int:employee_id>", methods=["DELETE"])
def remove_employee_data(employee_id):
    del employees[employee_id]
    return make_response(""), 200
```

3 Exercises:

Create a Flask microservice to expose mathematical operations. It exposes POST operations for the following operations.

- add -multiply - subtract - divide

Create a Flask microservice to input a list the words and returns the list of words with the word-count.

It exposes a POST API for the following operations.

- POST /wordcount

Create a Flask microservice Library management with ability to manage books.

[ ]: