Tutoriel Flask 2

Net and Light

2023 - 2024

Contents

1 Introduction **2**

2 Configuration du Projet **2**

3 Fichier `serveur.py` **2**
3.1 Code 2
3.2 Explication du Code 2

4 Tester l'API **3**
4.1 Obtenir la liste des revenus 3
4.2 Ajouter un nouveau revenu 3

5 Fichier `call.py` **3**
5.1 Code 3
5.2 Explication du Code 4

6 Fichier `add.py` **4**
6.1 Code 4
6.2 Explication du Code 4

7 Conclusion **5**

1 Introduction

Flask est un micro-framework léger pour Python qui vous permet de créer des applications web rapidement et facilement. Dans ce tutoriel, nous allons créer une simple API RESTful avec Flask.

2 Configuration du Projet

Pour commencer, assurez-vous d'avoir Flask installé. Vous pouvez l'installer via pip:

```
pip install Flask
```

3 Fichier `serveur.py`

Le fichier `serveur.py` contient le code principal pour notre application Flask. Ce fichier initialise l'application et définit deux routes: une pour obtenir la liste des revenus et une autre pour ajouter un revenu.

3.1 Code

```python
from flask import Flask, jsonify, request

app = Flask(__name__)

incomes = [
    {'description': 'salary', 'amount': 5000}
]

@app.route('/incomes')
def get_incomes():
    print("appel de la fonction icomes")
    return jsonify(incomes)

@app.route('/incomes', methods=['POST'])
def add_income():
    incomes.append(request.get_json())
    return '', 204

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

3.2 Explication du Code

- `from flask import Flask, jsonify, request`: Nous importons les modules nécessaires de Flask.
- `app = Flask(__name__)`: Nous créons une instance de l'application Flask.

- `incomes`: Une liste initiale contenant un revenu.
- `@app.route('/incomes')`: Cette route répond aux requêtes GET à l'URL `/incomes` et retourne la liste des revenus.
- `@app.route('/incomes', methods=['POST'])`: Cette route répond aux requêtes POST à l'URL `/incomes` et ajoute un nouveau revenu à la liste.
- `app.run(debug=True)`: Démarre le serveur Flask en mode debug.

4 Tester l'API

Pour tester notre API, nous pouvons utiliser des outils comme curl ou Postman. Voici comment vous pouvez tester avec curl:

4.1 Obtenir la liste des revenus

```
curl http://127.0.0.1:5000/incomes
```

4.2 Ajouter un nouveau revenu

```
curl -X POST -H "Content-Type: application/json" -d '{"description
    ": "bonus", "amount": 1500}' http://127.0.0.1:5000/incomes
```

5 Fichier `call.py`

Le fichier `call.py` contient un exemple de code Python pour effectuer des appels à notre API Flask en utilisant le module `requests`.

5.1 Code

```
import requests

api_url = "http://localhost:5000/incomes"

response = requests.get(api_url)

print(response.json())
```

5.2 Explication du Code

- `import requests`: Nous importons le module `requests` qui permet de faire des requêtes HTTP en Python.
- `api_url = "http://localhost:5000/incomes"`: Nous définissons l'URL de l'API que nous allons appeler.
- `response = requests.get(api_url)`: Nous effectuons une requête GET à l'URL de l'API.
- `print(response.json())`: Nous affichons la réponse JSON obtenue de l'API.

6 Fichier `add.py`

Le fichier `add.py` contient un exemple de code Python pour effectuer une requête POST à notre API Flask en utilisant le module `requests`.

6.1 Code

```python
import requests

api_url = "http://localhost:5000/incomes"

newobj = { 'description': 'Salut', 'amount': 5431 }

response = requests.post(api_url, json=newobj)

print(response.text)
```

6.2 Explication du Code

- `import requests`: Nous importons le module `requests` qui permet de faire des requêtes HTTP en Python.
- `api_url = "http://localhost:5000/incomes"`: Nous définissons l'URL de l'API que nous allons appeler.
- `newobj = {'description': 'Salut', 'amount': 5431}`: Nous créons un nouvel objet JSON à envoyer à l'API.
- `response = requests.post(api_url, json=newobj)`: Nous effectuons une requête POST à l'URL de l'API avec l'objet JSON.
- `print(response.text)`: Nous affichons la réponse obtenue de l'API.

7 Conclusion

Nous avons créé une simple API RESTful avec Flask qui permet de gérer une liste de revenus, et nous avons également montré comment effectuer des appels à cette API en utilisant le module `requests` en Python. Vous pouvez étendre cette application en ajoutant plus de fonctionnalités et en structurant le projet en plusieurs fichiers pour une meilleure organisation.