

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	Fichier call.py	3
4.1	Code	3
4.2	Explication du code	3
5	Fichier add.py	3
5.1	Code	4
5.2	Explication du code	4
6	Conclusion	4

1 Introduction

Flask est un micro-framework léger pour Python qui permet de créer des applications web rapidement et facilement.

2 Configuration du projet

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

```
1 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

```
1 from flask import Flask, jsonify, request
2
3 app = Flask(__name__)
4
5 incomes = [
6     {'description': 'salary', 'amount': 5000}
7 ]
8
9 @app.route('/incomes')
10 def get_incomes():
11     print("appel de la fonction incomes")
12     return jsonify(incomes)
13
14 @app.route('/incomes', methods=['POST'])
15 def add_income():
16     incomes.append(request.get_json())
17     return '', 204
18
19 if __name__ == '__main__':
20     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 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`.

4.1 Code

```

1 import requests
2
3 api_url = "http://localhost:5000/incomes"
4
5 response = requests.get(api_url)
6
7 print(response.json())

```

4.2 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.

5 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`.

5.1 Code

```
1 import requests
2
3 api_url = "http://localhost:5000/incomes"
4
5 newobj = { 'description': 'Salut', 'amount': 5431 }
6
7 response = requests.post(api_url, json=newobj)
8
9 print(response.text)
```

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.
- `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.

6 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.