Introduction

FastAPI est un framework web moderne, rapide et intuitif pour construire des API avec Python 3.7+ basé sur les annotations de types. Ce guide vous accompagne dans :

— L'installation et la configuration de FastAPI.

— La création d'une première API simple.

— Quelques exemples pratiques avec requêtes et réponses JSON.

Installation de FastAPI et Uvicorn

1. Créer un environnement virtuel

Ouvrez votre terminal (ou PowerShell sous Windows) et tapez :
[bgcolor=gray !5]bash python -m venv venv source venv/bin/activate Sur Linux ou Mac venv Sur Windows

2. Installer FastAPI et le serveur Uvicorn

[bgcolor=gray !5]bash pip install fastapi uvicorn
Pour vérifier l'installation :
[bgcolor=gray !5]bash pip list
Vous devriez voir `fastapi` et `uvicorn` dans la liste.

Création d'une première API

Créez un fichier `main.py` et ajoutez le code suivant :
[bgcolor=gray !5]python from fastapi import FastAPI
app = FastAPI()
@app.get("/") def read$_{r}oot() : return"message" : "Bienvenuesurvotrepremière APIFastAPI!"$
@app.get("/hello/name") def say$_{h}ello(name : str) : return"message" : f"Bonjourname, ravidetevoi$
Démarrage du serveur : uvicorn main :app –reload
Lancez ensuite le serveur :
[bgcolor=gray !5]bash uvicorn main :app –reload
Allez dans votre navigateur à l'adresse :

`http://127.0.0.1:8000`

Documentation automatique

FastAPI fournit automatiquement deux interfaces interactives :

— **Swagger UI** : `http://127.0.0.1:8000/docs`

— **ReDoc** : `http://127.0.0.1:8000/redoc`

Exemple : Requête POST avec modèle Pydantic

Créons un modèle de données pour recevoir des informations depuis le client.
[bgcolor=gray !5]python from pydantic import BaseModel
class Utilisateur(BaseModel) : nom : str age : int email : str
@app.post("/utilisateur/") def creer$_u tilisateur(user : Utilisateur) : return"message" : f"Utilisateur$
Pour tester, vous pouvez utiliser Swagger UI et soumettre le JSON suivant :
[bgcolor=gray !5]json "nom" : "Ammar", "age" : 30, "email" : "ammar@example.com"

Exemple : Paramètres de requête et valeurs par défaut

[bgcolor=gray !5]python @app.get("/produit/") def lire$_p roduit(nom : str = "Inconnu", prix : float = 0.0) : return"produit" : nom, "prix" : prix$
Essayez dans votre navigateur :

`http://127.0.0.1:8000/produit?nom=PC&prix=1500`

Exemple : Simulation d'une base de données simple

[bgcolor=gray !5]python fake$_d b = []$
@app.post("/items/") def create$_i tem(item : dict) : fake_db.append(item)return"db_size" : len(fake_db)$
@app.get("/items/") def list$_i tems() : return"items" : fake_db$

Exemple avancé : API avec dépendance

[bgcolor=gray !5]python from fastapi import Depends
def get$_t oken_h eader(token : str = "12345") : iftoken! = "12345" : raiseHTTPException(status_code 400, detail = "Tokeninvalide")returntoken$
@app.get("/secure-data/", dependencies=[Depends(get$_t oken_h eader)])defsecure_data() : return"message" : "Accèsautorisé"$

Conclusion

Vous avez installé FastAPI, exploré les routes GET et POST, et vu comment utiliser les modèles de données avec Pydantic. Prochaine étape : connecter FastAPI à une base de données (SQLite, PostgreSQL ou autre) et gérer l'authentification.

Bon codage et amusez-vous avec FastAPI !

