

Fast API

Doc FastAPI : <https://fastapi.tiangolo.com/>

Mise en place :

Version de Python : Python 3.6 minimum

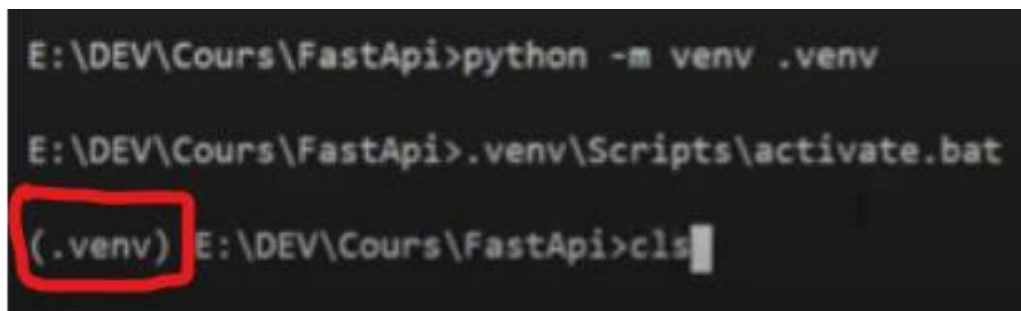
Installation :

Environnement virtuel :

Dans le terminal :

```
python -m venv .venv // « .venv » étant le nom du dossier
```

```
.venv\Scripts\activate.bat
```

A screenshot of a Windows command prompt terminal. The first two lines show the commands 'python -m venv .venv' and '.venv\Scripts\activate.bat' being executed. The third line shows the prompt changing to '(.venv) E:\DEV\Cours\FastApi>' and the user typing 'cls'. A red rectangle is drawn around the '(.venv)' part of the prompt, indicating that the virtual environment is now active.

```
E:\DEV\Cours\FastApi>python -m venv .venv  
E:\DEV\Cours\FastApi>.venv\Scripts\activate.bat  
(.venv) E:\DEV\Cours\FastApi>cls
```

Signifie que venv est activé

FastAPI :

```
pip install "fastapi[all]"
```

toivicornmain :app--reload

Installer FastAPI

Code API :

Import :

```
from fastapi import FastAPI #importation du module FastAPI  
from data import * #importation de l'Exemple BD :  
from fastapi import Query, Path # voir Contraintes :  
from pydantic import BaseModel #voir Création Classe :
```

Import uvicorn

Créer l'API :

```
nomApp = FastAPI()
```

Création Classe :

Ici nous prenons en exemple la classe Concessionnaire

```
class Concessionnaire (BaseModel) :
```

```
    id : int
```

```
    marque : str
```

```
    naissance : str
```

```
    modele : list[str]
```

Première Fonction (Racine) :

```
@app.get(« / »)
```

```
def index() :
```

```
    return {« message » : « Bienvenue »}
```

Exemple de GET :

GET permet d'accéder à certaine valeur des documents (par exemple le nom de chaque concessionnaire).

```
@app.get(« /concessionnaires »)
def get_concessionnaires() :
    return get_all_concessionnaires()
```

Contraintes :

Pour les contraintes utilisant Query ou Path, import nécessaire :

```
from fastapi import Query, Path
```

Plusieurs exemples :

```
@app.get(« /concessionnaires »)
def get_concessionnaires(skip : int = 0, limit : int = 10) :
    concessionnaires = get_all_concessionnaires()
    return concessionnaires[skip :limit]
```

Appel : localhost :8000/concessionnaire?skip=2&limit=9

Renvoie tous les concessionnaires si skip et limit ne sont pas précisés. Sinon, renvoie les concessionnaires compris entre skip et limit.


```
@app.get(« /teams »)
```

```
def get_concessionnaire(skip : int = Query(0, le=9, description = « number you want to skip » ), limit : int = 10) :
```

```
    concessionnaires = get_all_concessionnaires()
```

```
    return concessionnaires [skip :limit]
```

Appel : localhost :8000/concessionnaire?skip=2

Renvoie tous les concessionnaires si skip n'est pas précisé. Skip doit être compris entre 0 et 10. Si skip vaut x, les x premiers concessionnaires ne seront pas affichés.

```
@app.get(« /concessionnaire/{marque} »)
```

```
Def get_marque(marque : str) :
```

```
    Result = get_marque(marque)
```

```
    If result is None :
```

```
        Return f « la marque{marque} est inconnue »
```

```
    Return result
```

Appel : localhost :8000/concessionnaire/Porsche

Renvoie le document ayant pour nom de marque « Porsche ».

```
@app.get(« /concessionnaire/{marque} »)
```

```
Def get_marque(marque : str = Path(..., min_length=5, description = « name of  
the team »)) :
```

```
    Result = get_marque(marque)
```

```
    If result is None :
```

```
        Return f « la marque{marque} est inconnue »
```

```
    Return result
```

Il est possible de mettre des critères sur la variable rentrée dans le Path grâce à la fonction « Path ». Ici, il faut que le paramètre ait une taille minimale de 5 caractères.

Appel : localhost :8000/concessionnaire/Porsche

GET avec PATH :

```
@app.get(« /id/{id} »)
```

```
Def get_concessionnaire (id : int) :
```

```
    result = get_id (id)
```

```
    If result is None :
```

```
        return f « le concessionnaire {id} est inconnue »
```

```
    Return result
```

Appel : localhost :8000/id/1 -> concessionnaire ayant pour id 1

Exemple de POST :

POST permet d'ajouter un document (ici un concessionnaire).

```
@app.post(« /concessionnaire/ »)
```

```
Def create_concessionnaire (concessionnaire: Concessionnaire) :
```

```
    create_concessionnaire (concessionnaire)
```

```
    return concessionnaire
```

Exemple de PUT :

PUT permet de modifier un document (par exemple ajouter un modèle a un concessionnaire).

```
@app.put("/id/{id}")

def edit_concessionnaire(id: int, cons_modifie: Concessionnaire, response:
Response):

    for concessionnaire in concessionnaires:

        if concessionnaire ["id"] == id:

            concessionnaire ['marque'] = cons_modifie.marque

            concessionnaire ['naissance'] = cons_modifie.naissance

            concessionnaire ['modele'] = cons_modifie.modele

            response.status_code = 200

            return concessionnaire

        else:

            response.status_code = 404

            return " concessionnaire Not found"
```

Exemple DELETE :

DELETE permet de supprimer un document.

```
@app.delete("/id/{id}")

def delete_concessionnaire(id: int, response: Response):

    for concessionnaire in concessionnaires:

        if concessionnaire ["id"] == id:
```



```
concessionnaires.remove(concessionnaire)
```

```
response.status_code = 204
```

```
return " concessionnaire supprimé"
```

```
else:
```

```
response.status_code = 404
```

```
return " concessionnaire Not found"
```

Lancement Serveur

Avec Main :

Main qui exécute le script :

```
if __name__ == '__main__':
```

```
    uvicorn.run(app) #par défaut port 8000
```

```
    # ou : uvicorn.run(app, host= »127.0.0.1 », port=8000)
```

Lancer le serveur dans terminal :

```
python main.py #nom du fichier
```

Sans Main :

Dans le terminal

```
uvicorn main:app --reload #Reload permet de relancer automatiquement le serveur
```

Accès à l'API :

Sur Google :

localhost:8000/

Ou :

localhost:8000/docs : génération auto de la doc

Exemple BD :

Pour l'exemple, nous allons remplacer la bd par un fichier python « data.py » ayant ce contenu :

```
from typing import Union
```

```
# concessionnaire
```

```
concessionnaires = [
```

```
    {
```

```
        « marque » : « Porsche »,
```

```
        « naissance » : « 1948 »,
```

```
        « modele » : [« GT3 RS », « 918 Spyder »]
```

```
    },
```

```
    {
```

```
        « marque » : « Peugeot »,
```

```
        « naissance » : « 1896 »,
```

```
        « modele » : [« 308 », « 206 bicolore »]
```

```
    },
```

```
    {
```

```
        « marque » : « Chevrolet »,
```

```
        « naissance » : « 1911 »,
```

```
        « modele » : [« Corvette c6 ZR1 », « Camaro ZL1 »]
```

```
    },
```

```
]
```

```
def get_all_marques() -> list :
```

```
    « « « Renvoie une liste des marques de concessionnaire » » »
```

```
    return concessionnaire
```

```
def get_marque(marque : str) -> Union[dict, None] :
```

```
    for concessionnaire in concessionnaires:
```

```
        if marque.lower == concessionnaire.get('marque').lower() :
```

```
            return concessionnaire
```

```
    return None
```