



Create a RESTful API with Python 3 and FastAPI

In this article, we'll create a simple RESTful API using Python 3 and FastAPI.

But whats is FastAPI?

FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints.

First, lets go to import necessary modules and classes:

pip3 install fastapi uvicorn

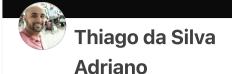
Now the imports:

```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from typing import Optional
from uuid import UUID, uuid4
```

- FastAPI from the fastapi package to create our web application.
- HTTPException to handle error responses.
- BaseModel from pydantic for data validation and
- 👋 Kindness is core times management using Python type annotations.
 - Optional from typing to denote optional fields.
 - UUID and uuid4 for generating and working with universally unique identifiers.

Okay

Defining Data Models



Microsoft MVP

JOINED Apr 9, 2018

More from Thiago da Silva Adriano

Programação Imperativa

Configurando o GraalVM no macOS

Introdução a Event-driven Architecture #eventdriven

... X

We define a Pydantic model named Item to represent the data structure of an item in our API. This model includes fields for name, description, price, and on_offer, with description being optional and on_offer defaulting to False.

```
class Item(BaseModel):
    name: str
    description: Optional[str] = None
    price: float
    on_offer: bool = False
```

FastAPI app instance and in-memory data

```
app = FastAPI()
items = {}
```

We create items = {} to save itens in memory to our test.

RESTful operations

Now Let's go to create our operations to save itens. To do this we'll create a RESTfull API with post, put, get and delete:

starting by post:

```
@app.post("/items/", response_model=Item)
async def create_item(item: Item):
   item_id = uuid4()
   items[item_id] = item
   return item
```

This code will create a new Item with uuid to create a index.

** Kindness is Next, jiets go to create two methods, one to list all itens and other to list by id:

Okay

```
@app.get("/items/", response_model=dict)
```

```
async def read_items():
    return items

@app.get("/items/{item_id}", response_model=Item)
async def read_item(item_id: UUID):
    if item_id not in items:
        raise HTTPException(status_code=404, deta
    return items[item_id]
```

Next step will be the Update and Delete methods:

```
@app.put("/items/{item_id}", response_model=Item)
async def update_item(item_id: UUID, item: Item):
    if item_id not in items:
        raise HTTPException(status_code=404, deta
    items[item_id] = item
    return item

@app.delete("/items/{item_id}", response_model=It
async def delete_item(item_id: UUID):
    if item_id not in items:
        raise HTTPException(status_code=404, deta
    return items.pop(item_id)
```

And now let's go to test. To do this, at your terminal past this command below:

```
uvicorn main:app --reload
```

This command will create a server at port 8000.

Now to create a new Item we can use this curl below:

```
import requests
import json

Kindness is contagious
    url = "localhost:8000/items"

payload = json.dumps({
    "name": "Example",
    "description": "This is an example description
    "price": 10.99,
```

```
Create a RESTful API with Python 3 and FastAPI - DEV Community
      "on_offer": False
   })
   headers = {
    'Content-Type': 'application/json'
   response = requests.request("POST", url, headers=
   print(response.text)
And open localhost:8000/docs and test your endpoints
using swagger:)
FastAPI 01.0 0AS 3.1
 default
  GET /items/ Read Items
  POST /items/ Create Item
      /items/{item_id} Read Item
  PUT /items/{item_id} Update Item
 DELETE /items/{item_id} Delete Item
```

Top comments (0)

Code of Conduct - Report abuse

Nindness is contagious

