BUỔI 4 (2): FASTAPI VÀ HTML

MỤC TIÊU:

- Tương tác FastAPI với HTML

CHUẨN BỊ:

- Requirements: python 3.9+
- Installation: pip install jinja2
- Postman: https://www.postman.com/downloads/

1. Giới thiệu về Jinja2

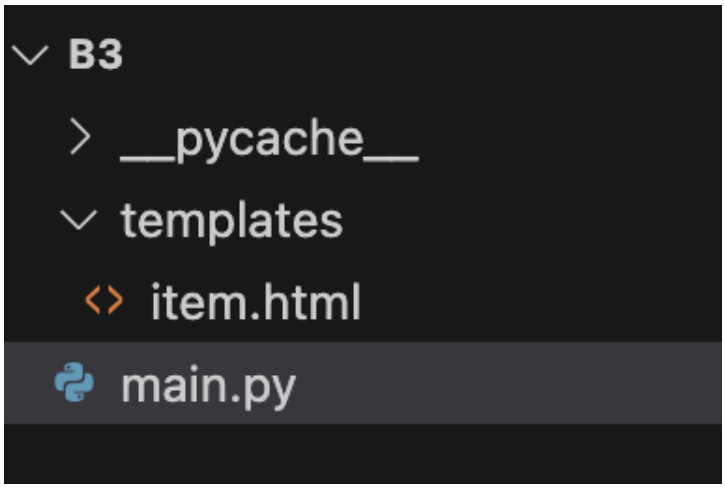
Jinja2 là một hệ thống mẫu Python, cho phép chúng ta tạo các trang HTML động bằng cách nhúng các biểu thức và biến Python vào trong mẫu.

Đặc điểm chính:

- Jinja2 sử dụng cú pháp dễ đọc và rõ ràng, giúp người phát triển dễ dàng theo dõi và sửa đổi các mẫu.
- Có thể được tích hợp dễ dàng với nhiều framework web khác nhau, bao gồm **Flask, Django**, và cả **FastAPI**.

2. Thực hành

Đầu tiên, chúng ta sẽ tạo tệp **main.py** và thư mục **templates** để quản lý các tệp HTML:



Với tệp **main.py**:

- Import **Jinja2Templates**.
- Tạo 1 đối tượng **templates** để có thể tái sử dụng về sau.
- Khai báo một tham số **Request** trong đường dẫn sẽ trả về một template.
- Sử dụng templates vừa tạo để render và return **TemplateResponse**

```
from fastapi import FastAPI, Request
from fastapi.responses import HTMLResponse
from fastapi.templating import Jinja2Templates

app = FastAPI()

templates = Jinja2Templates(directory="templates")

@app.get("/items/{id}", response_class=HTMLResponse)
async def read_item(request: Request, id: str):
    return templates.TemplateResponse("item.html", {"request": request, "id":
id})
```

Giải nghĩa code:

```
from fastapi import FastAPI, Request
from fastapi.responses import HTMLResponse
from fastapi.templating import Jinja2Templates
```

- **FastAPI**: Là framework để phát triển API nhanh chóng với Python.
- **Request**: Được sử dụng để lấy thông tin từ yêu cầu HTTP.
- **HTMLResponse**: Chỉ định rằng endpoint này sẽ trả về một trang HTML.
- **Jinja2Templates**: Là một công cụ giúp chúng ta tương tác với các mẫu - Jinja2 (được sử dụng để tạo trang HTML động).

```
templates = Jinja2Templates(directory="templates")
```

- Khởi tạo một đối tượng **templates** từ **Jinja2Templates**, và chỉ định thư mục templates sẽ chứa các tệp mẫu.

```
@app.get("/items/{id}", response_class=HTMLResponse)
```

- Đây là một decorator dùng để xác định một endpoint. Endpoint này có đường dẫn **/items/{id}** và sẽ trả về một trang HTML.

```
async def read_item(request: Request, id: str):
    return templates.TemplateResponse("item.html", {"request": request, "id":
id})
```

- Hàm **read_item** được gắn liền với đường dẫn endpoint. Nhận vào một đối tượng request và một biến id.
- Hàm này trả về một **TemplateResponse** sử dụng template **item.html**, và cung cấp các dữ liệu (**request** và **id**) để render vào trang HTML.

Lưu ý rằng item.html phải tồn tại trong thư mục templates và có các placeholder như **{{ request }}** và **{{ id }}** để được thay thế bởi dữ liệu thực tế.

Trong **item.html**:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Item Details</title>
</head>
<body>
    <h1>Item ID: {{ id }}</h1>
</body>
</html>
```

Chạy chương trình và đi tới đường dẫn http://127.0.0.1:8000/items/1 trong trình duyệt để xem kết quả: