Nếu 2 app cùng chạy thì ưu tiên cái chạy trước.

Nếu 2 app chạy cùng lúc thì sẽ chia sẻ tài nguyên (port)

Chạy 2 app cùng lúc khi có nhiều terminal (alt ctrl O)

ở HTML thì giữa {{}} code như python

Cách truy cập 1:

<h3>{{ post['title'] }}</h3>

<span>{{ post['author'] }}</span>

<p>{{ post['content'] }}</p>

Cách 2: dùng object

<h3>{{ post.title }}</h3>

<div>{{ post.author }}</div>

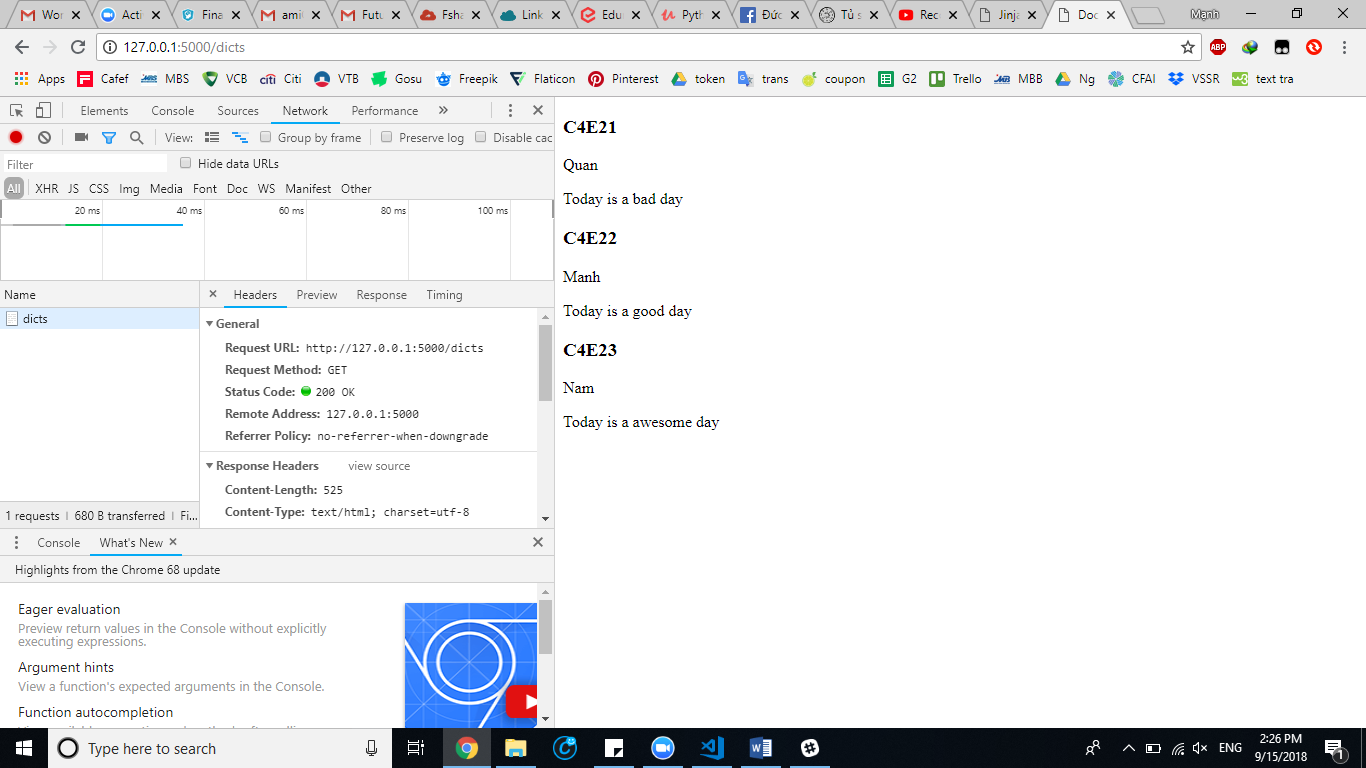
<p>{{ post.content }}</p>

Render:

Single variable

List

Dict

Last = dict + List

Request method: GET => chỉ đọc, xem (view)

Flow

Client bảo server: T muốn xem

Server response JS, CSS, HTML

R C U D

Mỗi cái sẽ có method khác:

Tương ứng là: GET, POST, PUT, DELETE

Phần lớn là động đến GET, POST

POST

Giao diện cho người dùng nhập vào: đơn giản nhất là form

Thẻ cho post:

Form

<http://127.0.0.1:5000/new-post?title=C4E21&author=M%E1%BA%A1nh>

Thông tin không cần bảo mật, không cần riêng tư, không quá dài (<= 255 ký tự)

Link sau dấu ‘?’ là arguments url

Cấu hình:

@app.route('/new-post')

def new\_post():

return render\_template('new\_post.html')

Chỉ nhận GET

Phải cấu hình thêm mới nhận POST

Response: GET

Request: POST

Response: POST

Server

Client

Request: GET

|  |
| --- |
| Form |
| Form |
| Form |

Có thể server và data base nằm cạnh nhau hoặc cùng máy

Cơ chế ORM (Object Relational Mapping)

Phần delete + Put thì cần con khác