**Lập trình web cơ bản với Flask**

Cài đặt flask

* Sử dụng pip để cài đặt flask. Từ cửa sổ cmd gõ lệnh:

pip install flask

* Kiểm tra flask đã được cài đặt:

>> import flask

>> print(flask.\_\_version\_\_)

1. **First\_app**

****

Thay đổi địa chỉ và cổng của server:

Khi một ứng dụng web chạy, server sẽ lắng nghe tại một địa chỉ (IP) và cổng (Port). Với flask, ứng dụng sẽ sử dụng địa chỉ IP mặc định là 127.0.0.1 (localhost) và port mặc định là 5000. Để thay đổi các giá trị IP và port này, có thể sử dụng các tham số truyền vào từ hàm khởi tạo ứng dụng:

app.run(host='<IP>', port=<port>)

Các tham số:

* IP : địa chỉ IP mà server sẽ dùng để hoạt động. Các giá trị cho phép của IP chỉ được lấy từ danh sách sau:
* Địa chỉ nội mạng 127.0.0.1 (localhost). Khi sử dụng địa chỉ này, chỉ có thể truy nhập được ứng dụng web từ bản thân máy đang chạy ứng dụng
* Các địa chỉ của máy hiện tại trong các mạng LAN mà máy kết nối đến. Ví dụ : 192.168.1.xx . Khi sử dụng địa chỉ dạng này, chỉ có thể truy nhập ứng dụng web từ các máy trong mạng LAN tương ứng
* Địa chỉ public IP của máy. Trường hợp này thường dùng cho các máy server đặt trên cloud có địa chỉ IP public ra internet. Khi dùng public IP, có thể truy nhập ứng dụng web từ bất kể nơi nào có kết nối internet.
* Địa chỉ dạng tổng quát 0.0.0.0. Địa chỉ này tương đương với sử dụng tất cả các địa chỉ dạng trên.
* Port : giá trị cổng sử dụng, có thể nhận giá trị từ 1 đến 65535 nhưng không được trùng với các port đã có ứng dụng khác sử dụng.

Ví dụ:

app.run(host='0.0.0.0', port=8080)

Tạo mới một endpoint

Khi truy nhập ứng dụng web theo url (nhập vào trình duyệt), mỗi endpoint tương ứng với phần sau của url truy nhập (sau khi đã bỏ đi IP và port). Mỗi endpoint sẽ thực hiện một chức năng khác nhau và trả về kết quả tương ứng cho người dùng.

Để khai báo endpoint trong flask, sử dụng kí hiệu @app.route('<endpoint>'), ví dụ:

from flask import Flask  
app = Flask(\_\_name\_\_)  
  
@app.route('/hello')  
def hello\_world():  
   return 'Hello World'  
  
if \_\_name\_\_ == '\_\_main\_\_':  
   app.run()

Sau khi chạy chương trình trên, nếu từ trình duyệt truy nhập vào địa chỉ [http://localhost:5000](https://www.google.com/url?q=http://localhost:5000&sa=D&ust=1557762779830000) sẽ gặp lỗi **Not found**, do endpoint mặc định '/' đã không được khai báo trong chương trình. Để sử dụng ứng dụng, phải truy nhập vào địa chỉ của endpoint mới là : [http://localhost:5000/hello](https://www.google.com/url?q=http://localhost:5000/hello&sa=D&ust=1557762779830000)

Một ứng dụng flask có thể khai báo nhiều endpoint để phục vụ các chức năng khác nhau cho người dùng truy nhập đến:

from flask import Flask  
app = Flask(\_\_name\_\_)  
  
@app.route('/')  
def homePage():  
   return 'Home page'  
     
@app.route('/hello')  
def hello\_world():  
   return 'Hello World'  
  
@app.route('/goodbye')  
def goodbye():  
   return 'Goodbye, see you later.'  
     
if \_\_name\_\_ == '\_\_main\_\_':  
   app.run()

Với ứng dụng trên, có thể truy nhập đến các URL tương ứng với các endpoint đã khai báo : [http://localhost:5000](https://www.google.com/url?q=http://localhost:5000&sa=D&ust=1557762779833000), [http://localhost:5000/hello](https://www.google.com/url?q=http://localhost:5000/hello&sa=D&ust=1557762779834000), [http://localhost:5000/goodbye](https://www.google.com/url?q=http://localhost:5000/goodbye&sa=D&ust=1557762779834000)

## Lấy tham số từ request:

Trong ứng dụng web, với cùng một endpoint truy nhập, người dùng vẫn có thể truyền lên server các tham số khác nhau. Có thể hiểu đây là các tham số sẽ được truyền vào hàm xử lý của endpoint.

Để thực hiện truyền tham số từ trình duyệt (client) lên server, có 2 phương thức truyền:

* GET : tham số được mã hóa dưới dạng cặp giá trị 'key=value' và đặt ở phía cuối của URL gốc của endpoint. Ví dụ :

[http://localhost:5000/hello?name=NVA&address=Hanoi](https://www.google.com/url?q=http://localhost:5000/hello?name%3DNVA%26address%3DHanoi&sa=D&ust=1557762779835000)

Các cặp giá trị ngăn cách nhau bởi kí tự **&**, toàn bộ phần tham số được ngăn cách với URL gốc của endpoint bằng dấu **?**

* POST : tham số được mã hóa trong form của file html và gửi lên server khi hành động form submit xảy ra. Ví dụ:

<form action='http://localhost:5000/hello' method='POST'>  
   <p>Name : <input name='name'></p>  
   <p>Address : <input name='address'></p>  
   <input type='submit'>  
</form>

Để lấy tham số do client truyền lên endpoint xử lý của flask, trước hết cần khai báo sử dụng module request của flask :

from flask import Flask, request

### Với GET method:

Các tham số do client gửi lên server được lấy qua đối tượng request.args , đối tượng này có kiểu dữ liệu là **dictionary**. Ví dụ:

from flask import Flask, request  
app = Flask(\_\_name\_\_)  
  
@app.route('/hello')  
def hello\_world():  
   name = request.args.get('name', '')  
   return f'Hello {name}'  
     
if \_\_name\_\_ == '\_\_main\_\_':  
   app.run()

Để sử dụng ứng dụng trên, truyền tham số name vào phía cuối của URL truy nhập endpoint, ví dụ:

[http://localhost:5000/hello?name=Nguyễn+Văn+An](https://www.google.com/url?q=http://localhost:5000/hello?name%3DNguy%25E1%25BB%2585n%2BV%25C4%2583n%2BAn&sa=D&ust=1557762779841000)

**dictionary –** ta có thể giải thích như sau, cái link trên nó sẽ đưa cho ta dict dữ liệu

<http://website.com/index?arg1=hello&arg2=world>

print request.args

>> {"arg1": "hello", "arg2": "world"}

Key và value

**args là** viết tắt của arguments, **là** tham số truyền vào khi chạy chương trình

### ới POST method:

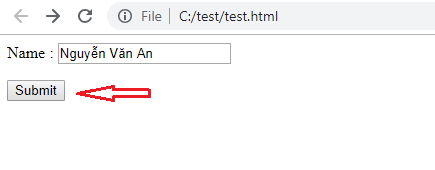
Các tham số do client gửi lên server được lấy qua đối tượng **request.form** , đối tượng này có kiểu dữ liệu là dictionary. Ví dụ:

from flask import Flask, request  
app = Flask(\_\_name\_\_)  
  
@app.route('/hello', methods=['POST'])  
def hello\_world():  
   name = request.form.get('name', '')  
   return f'Hello {name}'  
     
if \_\_name\_\_ == '\_\_main\_\_':  
   app.run()

Để sử dụng ứng dụng trên, tạo mới một file test.html với nội dung sau:

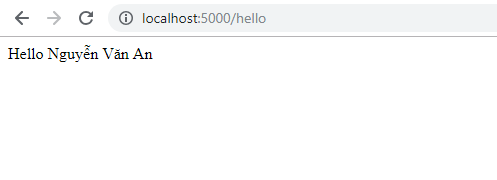
<form action='http://localhost:5000/hello' method='POST'>  
   <p>Name : <input name='name'></p>  
   <input type='submit'>  
</form>

Dùng trình duyệt mở file test.html trên (Từ File Explorer → Open with → Edge/Chrome/FireFox), trong màn hình ứng dụng trên trình duyệt, nhập giá trị cho trường input, sau đó chọn submit :



Hình 17. Mở file html bằng trình duyệt

Sau khi submit xong, trình duyệt sẽ nhận được kết quả do server trả về:



Hình 18. Kết quả do server trả về sau khi submit form

### Với POST method và mã hóa dữ liệu dạng json

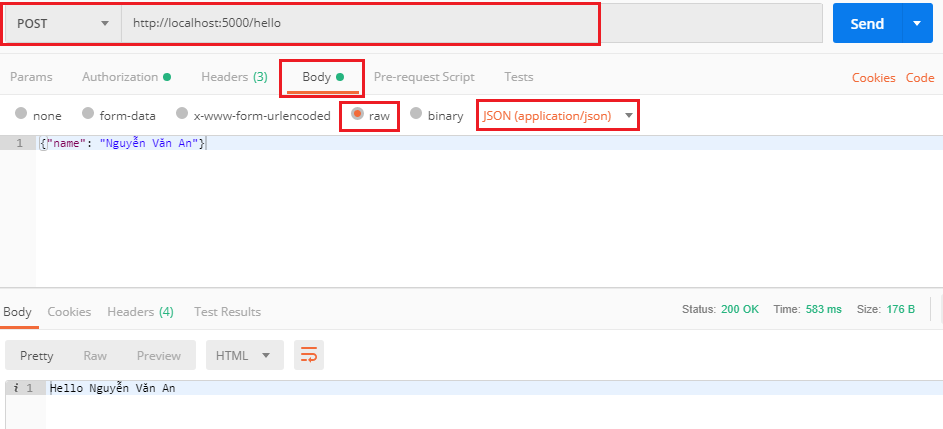
Thông thường mã hóa dữ liệu khi gọi POST method có dạng chuẩn là form-data. Tuy nhiên, trong một số trường hợp, ví dụ giao tiếp giữa các server hoặc gọi service từ client theo dạng AJAX thì dạng mã hóa dữ liệu thường sử dụng là JSON. Trong trường hợp này để lấy tham số trong request, chúng ta thay đối tượng request.form bằng đối tượng request.json :

from flask import Flask, request  
app = Flask(\_\_name\_\_)  
  
@app.route('/hello', methods=['POST'])  
def hello\_world():  
   name = request.json.get('name', '')  
   return f'Hello {name}'  
     
if \_\_name\_\_ == '\_\_main\_\_':  
   app.run()

Để gọi ứng dụng trên, có thể dùng curl hoặc postman :

Sử dụng curl :

curl -X POST -H "Content-Type: application/json" --data "{\"name\":\"Nguyễn Văn An\"}" http://localhost:5000/hello

Sử dụng Postman: 

Hình 19. Sử dụng postman để gọi POST method của flask

Trường hợp tổng quát, một endpoint có thể hỗ trợ nhiều phương thức (GET/POST) và nhiều dạng mã hóa dữ liệu, để lấy tham số từ request, chúng ta cần kiểm tra phương thức & loại mã hóa dữ liệu mà client gửi tới server:

from flask import Flask, request  
app = Flask(\_\_name\_\_)  
  
@app.route('/hello', methods = ['POST', 'GET'])  
def hello\_world():  
   if request.method == 'POST':  
       if 'name' in request.form:  
           name = request.form['name']  
           return f'Hello {name} from POST method (form-data)'  
                          
       elif 'name' in request.json:  
           name = request.json['name']  
           return f'Hello {name} from POST method (JSON)'         
                          
       else:  
           return 'Hello from POST method'  
   else:  
       name = request.args.get('name', '')  
       return f'Hello {name} from GET method'  
  
if \_\_name\_\_ == '\_\_main\_\_':  
   app.run()

**Định nghĩa về FormData:**

Với HTML5 chúng ta có thể sử dụng FormData, nó cho phép chúng ta xây dựng các đối tượng form tự động, có thể bao gồm cả các tệp tin từ hệ thống tệp tin của người dùng và sau đó gửi form này thông qua AJAX.

Túm váy lại, với FormData, chúng ta có thể submit dữ liệu lên server thông qua AJAX như là đang submit form bình thường.

## Mapping tham số trong URL

Ngoài cách lấy tham số từ request, có thể mapping tham số trong khai báo url. Sử dụng cách này, tham số do client gửi lên sẽ được chuyển thành tham số trong hàm xử lý của endpoint. Ví dụ:

from flask import Flask  
app = Flask(\_\_name\_\_)  
  
@app.route('/hello/<name>')  
def hello(name):  
  return f'Hello {name}'  
   
if \_\_name\_\_ == '\_\_main\_\_':  
  app.run(debug = True)

Để sử dụng ứng dụng, truyền tham số name vào cuối URL theo cấu trúc : http://localhost:5000/hello/<name>,  ví dụ:

[http://localhost:5000/hello/Nguyễn Văn An](https://www.google.com/url?q=http://localhost:5000/hello/Nguy%25E1%25BB%2585n%2BV%25C4%2583n%2BAn&sa=D&ust=1557762779857000)

## Sử dụng template

Khi endpoint trả về kết quả dưới dạng string, nếu client là trình duyệt thì nội dung này sẽ được hiện lên trong cửa sổ trình duyệt để người dùng xem. Trên thực tế, nội dung trả về thường khá dài (kèm theo style và format) do đó phía server sẽ không trả về kết quả dưới dạng một biến string, mà trả về nội dung thông qua các file template với tên mở rộng html.

Thư mục chứa template có tên là templates và đặt ngang hàng với file chạy ứng dụng (app.py) , ví dụ:

app.py

templates/

    Index.html

Chương trình sử dụng template để trả về nội dung html cho client:

from flask import Flask, render\_template  
app = Flask(\_\_name\_\_)  
  
@app.route('/')  
def hello():  
  return render\_template('index.html')  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  app.run()

File index.html đặt trong thư mục templates và chứa nội dung muốn trả về cho client, ví dụ:

<html>  
   <body>  
       <b>Hello, world!</b>  
   </body>  
</html>

### Truyền biến từ hàm xử lý endpoint xuống template

Trong hàm xử lý endpoint thường thực hiện một số tính toán (truy nhập database, file, …), sau đó sẽ tạo ra một số biến để chứa dữ liệu kết quả. Các biến này cần được truyền xuống template để tạo ra nội dung tùy biến cho mỗi lần request khác nhau từ người dùng. Việc truyền biến từ hàm xử lý endpoint (còn gọi là controller), xuống các file template (còn gọi là lớp view), được thực hiện bằng cách truyền biến trực tiếp vào hàm render\_template. Ví dụ :

from flask import Flask, render\_template  
app = Flask(\_\_name\_\_)  
  
@app.route('/hello/<user>')  
def hello\_name(user):  
  return render\_template('index.html', user=user)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  app.run()

Trong template, nội dung các biến từ server truyền xuống sẽ được đặt trong cặp dấu {{ }}, ví dụ file index.html ứng với chương trình phía trên cần có nội dung sau:

|  |
| --- |
| <html>    <body>        <b>Hello, {{ user }}!</b>    </body> </html> |

Để sử dụng ứng dụng trên, từ trình duyệt truy nhập đến địa chỉ http://localhost:5000/hello/<name> , chương trình sẽ hiện ra lời chào theo tham số <name> đã truyền vào.

Ngoài cấu trúc lấy giá trị biến được truyền xuống template như trên, trong template còn có các cấu trúc lệnh tương tự với chương trình python thông thường. Hai cấu trúc thường xuyên sử dụng là :

***Cấu trúc điều khiển if:***

{% if <cond> %}

    <html content>

{% else %}

    <html content>

{% endif %}

***Cấu trúc lặp for:***

{% for item in item\_list %}

    {{ item }}

{% endfor %}

Ví dụ : Chương trình dưới đây trả về một danh sách sinh viên trong nội dung hiển thị:

from flask import Flask, render\_template  
app = Flask(\_\_name\_\_)  
  
@app.route('/get\_students')  
def getStudents():  
  students = [  
            {'id' : '100001', 'name' : 'Nguyễn Văn An', 'address' : 'Hà Nội'},  
            {'id' : '100002', 'name' : 'Nguyễn Thị Bình', 'address' : 'TP. HCM'}  
           ]  
             
  return render\_template('students.html', students=students)  
   
if \_\_name\_\_ == '\_\_main\_\_':  
  app.run(debug = True)

File app.py

{% if students %}  
  <table border='1'>  
      <tr>  
           <th>Id</th>  
           <th>Name</th>  
           <th>Address</th>  
      </tr>  
      {% for student in students %}  
          <tr>  
               <td> {{ student.id }} </td>  
               <td> {{ student.name }} </td>  
               <td> {{ student.address }} </td>  
          </tr>  
      {% endfor %}  
  </table>  
{% else %}  
  No students  
{% endif %}

File templates/students.html

## Chuyển hướng trang với redirect và abort

### Chuyển hướng với redirect

Khi đang trong một endpoint, thay vì chạy đến hết quá trình xử lý mà chúng ta muốn chuyển qua một url khác thì có thể sử dụng hàm redirect . Hàm này nhận tham số là url cần chuyển hướng tới. Ví dụ :

from flask import Flask, request, redirect, url\_for  
app = Flask(\_\_name\_\_)  
  
@app.route('/hello/<name>')  
def hello(name):  
   return f'Hello {name}'  
  
@app.route('/')  
def homePage():  
   user = request.args.get('user', 'world')  
   return redirect(url\_for('hello', name=user))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  app.run()

Trong chương trình trên, hàm url\_for của flask có tác dụng tạo ra url dựa trên tên hàm của một endpoint xử lý và các tham số đi kèm với hàm.

Truy nhập ứng dụng tại địa chỉ : http://localhost:5000?user=<user>, chương trình sẽ được chuyển qua địa chỉ http://localhost:5000/hello/<user>

### Chuyển hướng với abort

Trong trường hợp xảy ra lỗi trong hàm xử lý endpoint, để kết thúc nhanh chương trình, có thể dùng hàm abort, kèm theo mã lỗi. Ví dụ :

|  |
| --- |
| from flask import Flask, request, redirect, url\_for app = Flask(\_\_name\_\_)  @app.route('/hello') def hello():    if 'name' not in request.args:        abort(404)            name = request.args['name']    return f'Hello {name}'     if \_\_name\_\_ == '\_\_main\_\_':   app.run() |

Ở ứng dụng trên, nếu truy nhập vào địa chỉ http://localhost:5000/hello?name=<name> , chương trình sẽ đưa ra lời chào theo tham số name đã truyền vào. Nếu truy nhập vào địa chỉ http://localhost:5000/hello, chương trình sẽ không xác định được tham số name và đưa ra thông báo lỗi.

## Upload file

Ở html form, để thực hiện upload file, cần thêm thuộc tính enctype='multipart/form-data'. Ở phía server, trong hàm xử lý endpoint, các file do client upload lên có thể lấy được từ đối tượng request.files. Đối tượng này có kiểu dữ liệu là dictionary.

Ví dụ:

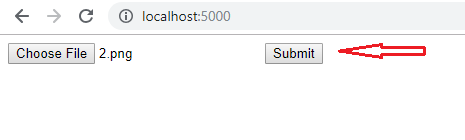
from flask import Flask, request, render\_template  
app = Flask(\_\_name\_\_)  
  
@app.route('/upload', methods=['POST'])  
def uploadFile():  
   file = request.files.get('file')  
   if file and file.filename != '':  
       file.save(file.filename)  
   return 'File uploaded'     
  
@app.route('/')  
def homePage():  
   return render\_template('index.html')  
     
if \_\_name\_\_ == '\_\_main\_\_':  
  app.run()

File app.py

<form method='POST' enctype='multipart/form-data' action='/upload'>  
   <input type='file' name='file'>  
   <input type='submit'>  
</form>

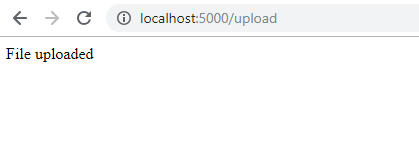
File templates/index.html

Để sử dụng ứng dụng trên,  truy nhập địa chỉ http://localhost:5000, màn hình upload file sẽ hiện lên :



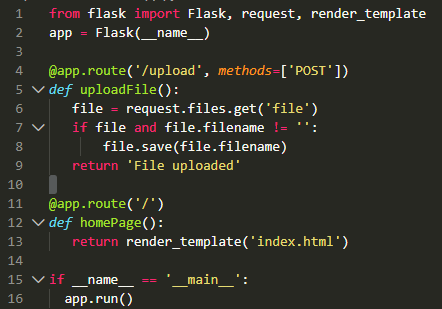
Hình 21. Màn hình upload file

Sau khi chọn file và submit form, chương trình sẽ đưa ra thông báo file đã upload thành công:



Hình 22. Thông báo file đã upload thành công

File do client gửi lên sẽ được lưu trong thư mục chạy ứng dụng (thư mục chứa file app.py)



File index.html