**Python flask**

Python flask cung cấp các cài đặt cấu hình và tiện ích để bắt đầu 1 ứng dung web đơn giản đến phức tạp

\* Cài đặt python flask và môi trường phát triển.

**1. Khởi động với Python flask**

**1.1. Giới thiệu về cấu trúc 1 file dùng flask**

Một chương trình dùng lớp Flask để dựng web sẽ như sau:

**from** flask **import** Flask

app = Flask**(**\_\_name\_\_**)**

@app.route**(**"/"**)**

**def** hello\_world**():**

**return** "<p>Hello, World!</p>"

Chạy chương trình bằng lệnh:

flask --app hello run

với hello là tên file hello.py

Kết quả trả ra là:

\* Serving Flask app 'hello'

\* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)

Như vậy, đoạn lệnh trên đã dựng thành công 1 web local trên địa chỉ 127.0.01:5000. Chúng ta có thể thấy, chỉ với 1 đoạn lệnh rất ngắn ta đã có thể dựng được 1 web local. Từ đây, ta có thể phát triển thêm các tính năng để hoàn thiện ứng dụng của riêng mình.

**1.2. Giới thiệu về Debug mode**

Chúng ta có thể bật debug mode để các thay đổi trên code sẽ được web server cập nhật tự động và các lỗi từ debugger sẽ được show ra trên trình duyệt.

Để bật mode này ta dùng từ khoá option –debug như sau:

flask --app hello run --debug

Kết quả trên Terminal như sau:

\* Serving Flask app 'hello'

\* Debug mode: on

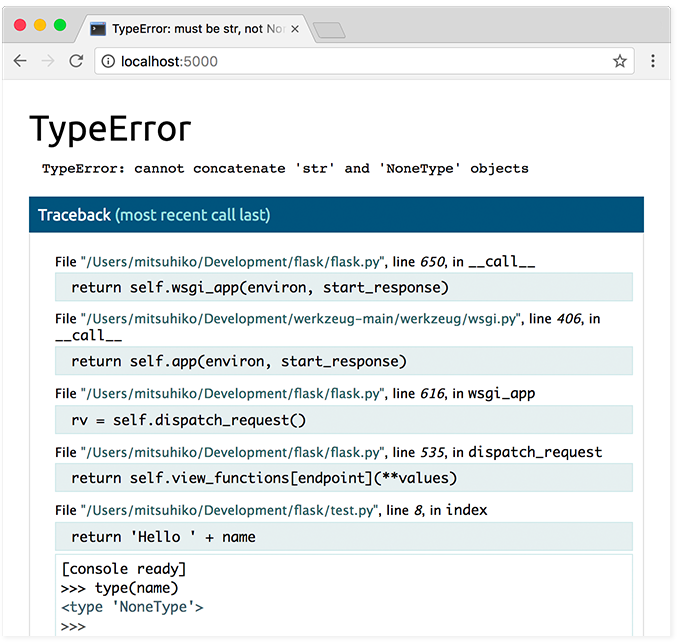
\* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)

\* Restarting with stat

\* Debugger is active!

\* Debugger PIN: nnn-nnn-nnn

và trên web hiển thị như sau:



1.3. HTML escaping

Các giá trị do người dùng truyền vào cần được escaped để tránh các nguy cơ về tấn công bảo mật. HTML render bởi Jinja sẽ làm điều này tự động.

hàm escape được dùng bằng tay như sau:

**from** markupsafe **import** escape

@app.route**(**"/<name>"**)**

**def** hello**(**name**):**

**return** f"Hello, {escape**(**name**)**}!"

Nếu người dung nhập tên từ trên web thì escape sẽ giúp các từ được nhập hiển thi dưới dạng văn bản thay vì hiển thị ra các đoạn code đã được viết.

=> Tóm lại: bước này thực hiện tự động giúp đóng gói và hiển thị dưới định dạng cho trước trên web.

**1.3. Routing trên web**

Các web hiện nay sử dụng các URL có ý nghĩa để giúp người dùng dễ dạng hơn trong việc truy cập trang web hay các tính năng của trang web.

Dùng đến chức năng route() để có thể gắn các chức năng của trang web với các URL tương ứng như sau:

@app.route**(**'/'**)**

**def** index**():**

**return** 'Index Page'

@app.route**(**'/hello'**)**

**def** hello**():**

**return** 'Hello, World'

như trên ta thấy localhost:5000 sẽ trả về “Index Page” và localhost:5000/hello sẽ trả về “Hello, World”

**1.4. Quy tắc biến**

Chúng ta có thể thêm các biến vào trong URL bằng cách thêm đánh dấu như sau <tên biến>. Hàm chức năng sẽ coi tên biến như 1 tham số truyền vào. Chứng ta có thể cấu hình kiểu dữ liệu cho biến truyền vào như sau:<kiểu dữ liệu: tên biến>

Các kiểu dữ liệu có thể chuyển đổi.

|  |  |
| --- | --- |
| string | (kiểu dữ liệu mặc định) chấp nhận mọi loại văn bản không chưa dấu gạch chéo |
| int | Số nguyên dương |
| float | Số thực dương |
| path | Giống dạng chuỗi nhưng chứa dấu gạch chéo |
| uuid | Chuỗi UUID |

**1.5. URL là duy nhất. Chuyển trang khi thực hiện hành động**

các URL là duy nhất và nếu nhập sai url thì sẽ trả về lỗi 404. Có 1 số quy tắc để xác định url đúng cú pháp. Khi nhập URL từ trình duyệt thì flask sẽ tự động thêm dấu gạch chéo vào cuối url. Xem ví dụ sau:

@app.route**(**'/projects/'**)**

**def** projects**():**

**return** 'The project page'

@app.route**(**'/about'**)**

**def** about**():**

**return** 'The about page'

ta có thể hiểu như sau:

với cách viết như trên (/projects/) thì ta có thể truy cập vào trang project bằng cả 2 đường dẫn /projects/ và /projects vì flask sẽ tự động chuyển hướng đến /projects/ khi gõ /projects.

Nhưng với cách viết bên dưới (/about) thì khi gõ /about/ thì sẽ trả về lỗi vì không có /about/ và flask cũng ko chuyển đến /about mà có lẽ là /about//

Khuyến nghị đặt theo kiểu bên dưới vì dễ nhìn và tránh việc dùng chung tài nguyên khi bị truy cập vào cùng 1 trang.

**1.6. Xây dựng đường dẫn**

Đôi khi ta cần xây dựng đường dẫn cho các chức năng nhất định. Để chỉ định 1 đường dẫn url cho các function nhất định ta sử dụng hàm url\_for().

Việc này sẽ đem lại lợi ích như sau:

1. Tạo tính mô tả hơn so với việc hard-code

2. Có thể thay đổi đường dẫn linh hoạt hơn, chỉ cần thay đổi 1 lần thay vì mã hoá theo cách thủ công.

3. Xử lí các kí tự đặc biệt một cách linh hoạt hơn.

4. Các đường dẫn là tuyệt đối, tránh các thao tác từ các đường dẫn tương đối.

5. Nếu đường dẫn nằm ngoài phạm vi đường dẫn gốc thì url\_for () sẽ xử lí được các trường hợp này.

**1.7. HTTP Method**

Ứng dụng web sẽ cho phép sử dụng các phương thức khác nhau của HTTP. Mặc định khi không truyền mothod thì ứng dụng sẽ chỉ phản hồi GET request.

Để thêm các method xử lý bằng cách thêm trường methods vào route() như sau:

**from** flask **import** request

@app.route**(**'/login'**,** methods=**[**'GET'**,** 'POST'**])**

**def** login**():**

**if** request.method == 'POST'**:**

**return** do\_the\_login**()**

**else:**

**return** show\_the\_login\_form**()**

Ở ví dụ trên, tronbg 1 hàm login thì sử dụng cả 2 method POST và GET.

hoặc có thể chia ra làm 2 hàm như sau:

@app.get**(**'/login'**)**

**def** login\_get**():**

**return** show\_the\_login\_form**()**

@app.post**(**'/login'**)**

**def** login\_post**():**

**return** do\_the\_login**()**

**1.8. Các file tĩnh**

Trang web luôn cần các file tĩnh với các mục đích khác nhau. Để tạo 1 đường dẫn cho 1 file tĩnh ta làm như sau:

url\_for**(**'static'**,** filename='style.css'**)**

**1.9. Rendering Templates**

Các template chứa các text file mà từ các file này sẽ sinh ra các trang html. Các template này thường là HTML, CSS. Để có thể render các file này để sử dụng trong python thì ta cần dùng phương thức render\_temple(). Chúng ta chỉ cần thêm tên file cần render để cso thể render các file đó.

**from** flask **import** render\_template

@app.route**(**'/hello/'**)**

@app.route**(**'/hello/<name>'**)**

**def** hello**(**name=**None):**

**return** render\_template**(**'hello.html'**,** name=name**)**

Flask sẽ tìm các template trong thư mục templates. Thư mục này cùng cấp với ứng dụng của bạn (với app.py).

nó sẽ có dạng như sau:



Trong các file template chúng ta cũng có thể truy xuất đến các đối tượng:

config, request, session and g như các dùng url\_for () và get\_flashed\_messages.

**1.10. Truy cập vào dữ liệu yêu cầu**

Để có thể truy cập đến dữ liệu yêu cầu từ client gửi lên server, Flask sẽ cung cấp đối tượng yêu cầu toàn cục. Tuy nhiên trong Python các đối tượng thường sẽ không là toàn cục để đảm bảo an toàn luồng code.

Vì thế cần thêm 1 khái niệm mới là ngữ cảnh cục bộ

**1.10.1. Ngữ cảnh cục bộ:**

Đối tượng request được khởi động trong luồng mà có ngữ cảnh đang hoạt động và kết nối với ứng dụng dùng luồng đó. Nên các ứng dụng khác khi không gọi đến request này thì có thể bỏ qua đối tượng này. Trừ khi gọi đến đối tượng này trong các kịch bản test nếu không đối tượng sẽ không bị ứng dụng truy cập đến, điều này sẽ dảm bảo an toàn luồng (tránh xung đột giữa function và dữ liệu gửi lên).

Vậy các unit test ở đây sẽ chỉ có giá trị cục bộ tại ngữ cảnh để dùng đối tượng toàn cục.

Ví dụ:

**from** flask **import** request

**with** app.test\_request\_context**(**'/hello'**,** method='POST'**):**

*# now you can do something with the request until the*

*# end of the with block, such as basic assertions:*

**assert** request.path == '/hello'

**assert** request.method == 'POST'

ta có thể truyền cài cả môi trường để dùng đối tượng request.

**with** app.request\_context**(**environ**):**

**assert** request.method == 'POST'

**1.10.1. Đối tượng Request:**

Để sử dụng đối tượng request ta cần import đối tượng này vào trong code.

**from** flask **import** request

Phương thức request là khả dụng khi dùng đến từ khoá method. Để có thể truy cập vào dữ liệu kiểu biểu mẫu (dược gửi lên qua POST hoặc PUT) ta có thể dùng từ khoá form.

@app.route**(**'/login'**,** methods=**[**'POST'**,** 'GET'**])**

**def** login**():**

error = **None**

**if** request.method == 'POST'**:**

**if** valid\_login**(**request.form**[**'username'**],**

request.form**[**'password'**]):**

**return** log\_the\_user\_in**(**request.form**[**'username'**])**

**else:**

error = 'Invalid username/password'

*# the code below is executed if the request method*

*# was GET or the credentials were invalid*

**return** render\_template**(**'login.html'**,** error=error**)**

Ở đoạn lệnh trên nếu form ko tồn tại, flask sẽ bật KeyError và ta có thể xử lí trong Key Error tiêu chuẩn. Nếu không xử lí thì mã HTTP 400 Bad Request sẽ được trả về.

Để truy cập và các tham số được gửi lên trong URL ta cso thể dùng từ khoá args như sau:

searchword = request.args.get**(**'key'**,** ''**)**

Để truy cập vào tham số URL thì phương thức get được dùng hoặc bắt lỗi với KeyError.

Để biết rõ hơn về các para này có thể tham khảo tại:

<https://flask.palletsprojects.com/en/2.3.x/api/#flask.Request>

**1.10.2. Tải tệp lên:**

Ta có thể dễ dàng tải tệp lên với Flask dễ dàng. Trong HTML form cần điền từ khoá như sau: enctype="multipart/form-data".

Tệp được tải lên được lưu trong bộ nhớ hoặc tại địa chỉ tạm thời trong hệ thống file. Để có thể truy cập vào các file đó, ta dùng thuộc tính “file” trong đối tượng request. Thuộc tính này giống như đối tượng file tiêu chuẩn trong python. Thuộc tính này cũng có phương thức save() cho phép lưu file trên hệ thống file của server.

**from** flask **import** request

@app.route**(**'/upload'**,** methods=**[**'GET'**,** 'POST'**])**

**def** upload\_file**():**

**if** request.method == 'POST'**:**

f = request.files**[**'the\_file'**]**

f.save**(**'/var/www/uploads/uploaded\_file.txt'**)**

có thể dùng thuộc tính filename để có thể lấy được tên file khi chưa tải lên ứng dụng. Tuy nhiên, trong quá trình gửi tên này có thể bị thay đổi. Để bảo toàn tên file này thì ta nên sử dụng hàm secure\_filename()

**from** werkzeug.utils **import** secure\_filename

@app.route**(**'/upload'**,** methods=**[**'GET'**,** 'POST'**])**

**def** upload\_file**():**

**if** request.method == 'POST'**:**

file = request.files**[**'the\_file'**]**

file.save**(**f"/var/www/uploads/{secure\_filename**(**file.filename**)**}"**)**

**1.10.2. Bộ nhớ đệm:**

Để truy cập vào cookies ta dùng thuộc tính cookies. Để có thể set cookies ta dùng set\_cookie.

Đọc từ cookie:

**from** flask **import** request

@app.route**(**'/'**)**

**def** index**():**

username = request.cookies.get**(**'username'**)**

*# use cookies.get(key) instead of cookies[key] to not get a*

*# KeyError if the cookie is missing.*

lưu vào cookies:

**from** flask **import** make\_response

@app.route**(**'/'**)**

**def** index**():**

resp = make\_response**(**render\_template**(**...**))**

resp.set\_cookie**(**'username'**,** 'the username'**)**

**return** resp

**1.11. Chuyển hướng và xử lí lỗi:**

Dùng hàm redirect () để chuyển hướng luồng code. dùng hàm abort() để báo cáo 1 lỗi.

Như ví dụ ở dưới:

**from** flask **import** abort**,** redirect**,** url\_for

@app.route**(**'/'**)**

**def** index**():**

**return** redirect**(**url\_for**(**'login'**))**

@app.route**(**'/login'**)**

**def** login**():**

abort**(**401**)**

this\_is\_never\_executed**()**

Nếu muốn tuỳ chỉnh trang báo lỗi ta có thể dùng từ khoá errorhandler():

**from** flask **import** render\_template

@app.errorhandler**(**404**)**

**def** page\_not\_found**(**error**):**

**return** render\_template**(**'page\_not\_found.html'**),** 404

**1.12. Response**

Dùng hàm make\_response() để có thể tạo ra các response đồng thời có thể truy xuất tới các thuộc tính trong response như sau:

**from** flask **import** make\_response

@app.errorhandler**(**404**)**

**def** not\_found**(**error**):**

resp = make\_response**(**render\_template**(**'error.html'**),** 404**)**

resp.headers**[**'X-Something'**]** = 'A value'

**return** resp

**1.13. API của JSON**

Định dạng JSON là kiểu phản hồi phổ biến trên web. Nếu hàm trả lại dict và list thì dữ liệu sẽ được chuyển thành phản hồi định dạng JSON.

@app.route**(**"/me"**)**

**def** me\_api**():**

user = get\_current\_user**()**

**return** **{**

"username"**:** user.username**,**

"theme"**:** user.theme**,**

"image"**:** url\_for**(**"user\_image"**,** filename=user.image**),**

**}**

@app.route**(**"/users"**)**

**def** users\_api**():**

users = get\_all\_users**()**

**return** **[**user.to\_json**()** **for** user **in** users**]**

Các dữ liệu trong list hoặc dict thì cần có thể chuyển thành định dạng JSON.

**1.14. Phiên truy cập**

Cùng với đối tượng request, đối tượng session cho phép lưu trữ các thông tin ứng với user từ 1 request cho đến lần request sau. Để dùng đối tượng session ta cần secret key.

ví dụ sử dung đối tượng session:

**from** flask **import** session

*# Set the secret key to some random bytes. Keep this really secret!*

app.secret\_key = b'\_5#y2L"F4Q8z\n\xec]/'

@app.route**(**'/'**)**

**def** index**():**

**if** 'username' **in** session**:**

**return** f'Logged in as {session**[**"username"**]**}'

**return** 'You are not logged in'

@app.route**(**'/login'**,** methods=**[**'GET'**,** 'POST'**])**

**def** login**():**

**if** request.method == 'POST'**:**

session**[**'username'**]** = request.form**[**'username'**]**

**return** redirect**(**url\_for**(**'index'**))**

**return** '''

<form method="post">

<p><input type=text name=username>

<p><input type=submit value=Login>

</form>

'''

@app.route**(**'/logout'**)**

**def** logout**():**

*# remove the username from the session if it's there*

session.pop**(**'username'**,** **None)**

**return** redirect**(**url\_for**(**'index'**))**

**2. Triển khai 1 web với html**