**HƯỚNG DẪN LẬP TRÌNH AN TOÀN TRONG PHÁT TRIỂN ỨNG DỤNG WEB TRÊN FRAMEWORK FLASK-ANGULARJS**

1. **Quản lý xác thực**

* ***Thông tin định danh***
  1. Tên đăng nhập phải là duy nhất, chỉ nên chứa tập các ký tự là chữ cái, chữ số, dấu gạch dưới.
     + Kiểm tra tên đăng nhập là hợp lệ:

**def check\_valid\_username**(username):  
 pattern = re.compile("^[a-zA-Z\_0-9]{5,30}$")  
 **return** (False, True)[pattern.match(username) **is not** None]

* + - Tại chức năng đăng ký cần kiểm tra tên đăng ký đã tồn tại hay chưa:

res = user.find\_one({'username': data['username']})   
**if** res **is** None:  
 # Thực hiện tạo tài khoản mới

* 1. Thiết lập chính sách mật khẩu mạnh:
     + Mật khẩu có độ dài tối thiểu là 8 kí tự.
     + Chứa chữ cái, chữ số và kí tự đặc biệt.
     + Thiết lập blacklist các mật khẩu yếu (VD: 123456A@, 123456a@,…).

# Sinh mật khẩu mạnh khi tạo tài khoản  
**def gen\_random\_str**(size, chars):  
 **return** ''.join(random.SystemRandom().choice(chars) **for** \_ **in** range(size))  
**def gen\_strong\_pass**():  
 **return** gen\_random\_str(12, (  
 gen\_random\_str(4, string.ascii\_letters) + gen\_random\_str(4, string.digits) + gen\_random\_str(4, "\*&%$#@!")))

# Kiểm tra mật khẩu mạnh tại chức năng đổi mật khẩu  
**def check\_valid\_password**(password):  
 pattern = re.compile("^((?=(.\*[a-zA-Z]){1,})(?=(.\*[\d]){1,})(?=(.\*[\W]){1,})(?!.\*\s)).{8,}$")  
 check = (False, True)[pattern.match(password) **is not** None]  
 **if** check:  
 **with** open("black\_list\_pass.txt") **as** f:  
 **for** line **in** f:  
 **if** password == line.rstrip():  
 **return** False  
 **return** True  
 **else**:  
 **return** False

* 1. Thiết lập thời gian hết hiệu lực cho mật khẩu tối đa 90 ngày, mật khẩu mới không được trùng với mật khẩu hiện tại.

# Kiểm tra khi người dùng đăng nhập

today = datetime.datetime.utcnow().date()  
user = mongo.db.user  
res = user.find\_one({'username': data['username']})  
**if** res **is not** None:  
 last\_change = res['lastChange'].date()  
 delta = today - last\_change  
 **if** delta.days > 90:

# Người dùng cần thực hiện đổi mật khẩu

# Tại chức năng đổi mật khẩu

user = mongo.db.user  
res = user.find\_one({'username': session['username']})

**if** check\_password\_hash(res['hash\_pass'], data['new\_password']):  
 # Yêu cầu người dùng nhập mật khẩu khác

* 1. Đối với chức năng reset/ quên mật khẩu:
     + Đường dẫn reset/quên mật khẩu được gửi qua email phải bị mất hiệu lực sau lần truy cập đầu tiên hoặc sau 8 giờ nếu không được truy cập.
     + Nếu chức năng reset/quên mật khẩu thực hiện gửi mật khẩu qua email thì mật khẩu phải được sinh ngẫu nhiên và phải tuân theo chính sách mật khẩu mạnh tại mục 2.

# Tạo đường dẫn reset tại chức năng reset password, gửi đường dẫn reset tới

# email của người dùng

reset\_path = gen\_random\_str(20, string.ascii\_letters)   
user = mongo.db.user  
user.update({'username': data['username']}, {'$set': {'reset\_path': reset\_path}}, False, True)

# Xóa đường dẫn reset khi người dùng gọi tới chức năng này

user = mongo.db.user  
res = user.find\_one({'username': data['username'], 'reset\_path': reset\_path\_param})  
**if** res **is not** None:  
 user.update({'username': data['username']}, {'$set': {'reset\_path': reset\_path}}, False, True)

* 1. Chỉ lưu dạng mã hash của mật khẩu trong DB. Khuyến nghị sử dụng các phương pháp như pbkdf2, bcrypt với tùy chọn hàm băm: SHA-256, số vòng lặp tối thiểu 1000 lần, kích thước salt tối thiểu 10 bytes:

**from** werkzeug.security **import** generate\_password\_hash

data = {'username': 'anhvtm4',  
 'password': generate\_password\_hash(password, method='pbkdf2:sha256', salt\_length=10)}

* ***Xử lý xác thực***
  1. Trả về thông báo chung cho trường hợp người dùng đăng ký thông tin định danh (username, email,…) đã tồn tại tại chức năng đăng ký, hoặc gửi sai thông tin định danh tại các chức năng đăng nhập, reset/quên mật khẩu, đổi địa chỉ email,… Ví dụ:
     + Tại chức năng đăng ký: *Vui lòng kiểm tra hộp thư của bạn để nhận được hướng dẫn kích hoạt tài khoản.*
     + Tại chức năng đăng nhập: *Sai tên đăng nhập hoặc mật khẩu*.
     + …
  2. Bật cơ chế bảo vệ bằng Captcha hoặc các hình thức tương đương khi đăng nhập sai quá 5 lần liên tiếp. Cần triển khai cơ chế này tại các chức năng quan trọng khác của ứng dụng.
     + Sử dụng Captcha an toàn theo Chỉ thị sử dụng Captcha an toàn Tập đoàn đã ban hành.
     + Thực hiện kiểm tra tính hợp lệ của Captcha trước khi thực hiện chức năng được request.
  3. Chỉ sử dụng phương thức POST để submit thông tin định danh, khuyến nghị sử dụng HTTPS cho đường truyền để tăng tính bảo mật.

1. **Quản lý phiên đăng nhập**
   1. Session phải được quản lý bởi server, sinh ngẫu nhiên và độ dài tối thiểu là 128 bit.

# Trong file config.py

SESSION\_KEY\_BITS = 128

* 1. Session phải được thiết lập thời gian timeout, giá trị timeout cần cân bằng giữa nhu cầu thương mại và yếu tố bảo mật.

Ví dụ thiết lập timeout là 100 phút:

# Trong file config.py

PERMANENT\_SESSION\_LIFETIME = timedelta(minutes=100)

* 1. Tạo mới session sau khi đăng nhập thành công.

# Sau khi xác thực thành công

session.regenerate()

* 1. Xóa giá trị sessionid và các dữ liệu gắn với session đó khi người dùng đăng xuất.

# Sau khi người dùng đăng xuất

session.clear()

* 1. Cấu hình thuộc tính “Secure” đối với các ứng dụng sử dụng HTTPS và “HTTP-Only” cho trường session cookie.

# config.py

SESSION\_COOKIE\_HTTPONLY = True  
SESSION\_COOKIE\_SECURE = True

* 1. Đối với các chức năng quan trọng có tương tác với database, ứng với mỗi phiên cần sinh thêm 1 token ngẫu nhiên, và thực hiện kiểm tra tính hợp lệ của token này trước khi xử lý truy vấn từ người dùng.

### ------- Flask ------------###

# Cấu hình tên, timeout và enable cơ chế chống CRSF

WTF\_CSRF\_HEADERS = ['CSRF-TOKEN']  
WTF\_CSRF\_TIME\_LIMIT = 1800  
WTF\_CSRF\_ENABLED = True

# Khởi tạo CsrfProtect

csrf\_protect = CsrfProtect()

csrf\_protect.init\_app(app)

# Chèn token vào mỗi response

@app.after\_request  
**def set\_csrf\_cookie**(response):  
 response.set\_cookie('X-CSRFToken', generate\_csrf())  
 **return** response

###-------- AngularJS -----------###

app.run(**function**($http){  
 $http.defaults.xsrfCookieName = 'X-CSRFToken';  
 $http.defaults.xsrfHeaderName = 'CSRF-TOKEN';  
});

1. **Phân quyền**
   1. Kiểm tra phân quyền dựa trên các đối tượng được lưu tại server (VD: tham số lưu trên session server, dữ liệu lưu trên DB,…).

**# Tạo hàm kiểm tra quyền người dùng, thông tin kiểm tra lấy từ session**

**def is\_admin**(function):  
 @wraps(function)  
 **def wrap**(\*args, \*\*kwargs):  
 data = {}  
 username = session['username']  
 user = NnhUser.query.join(NnhGroupUser).filter(NnhUser.user\_name == username).first()  
  
 **if** user **is** None **or** user.user\_priv != 1:  
 data['success'] = False  
 data['message'] = PRIV\_ERR  
 **return** jsonify(data)  
 **else**:  
 **return** function(\*args, \*\*kwargs)  
  
 **return** wrap

* 1. Phân quyền tối thiểu, chỉ đáp ứng đủ chức năng và tài nguyên cho người dùng/ứng dụng.

|  |  |
| --- | --- |
| Tất cả các chức năng | Sau khi đăng nhập thành công |
| Quản lý người dùng | admin |
| Thêm, sửa, xóa dữ liệu A | admin, managerA |
| Xem, tìm kiếm | admin, manager, user |

* 1. Phía giao diện người dùng: Chỉ hiển thị các thành phần giao diện, đường dẫn, hàm,... tương ứng với quyền của người dùng.

# Cách 1:

@func\_ctrl.route('/some\_function')  
@login\_required  
**def dashboard**():  
 **if** is\_admin() **or** is\_manager():  
 **return** render\_template('full\_function.html')  
 **else**:  
 **return** render\_template('limited\_function.html')

# Cách 2:

# server

**return** render\_template('sensor.html', has\_admin\_role=has\_admin\_role)

**# html**

{% if has\_admin\_role %}  
 <th width="6%">Sửa</th>  
 <th width="6%">Xóa</th>  
{% endif %}

* 1. Phía server: Kiểm tra quyền tác động của người dùng/ứng dụng trên các hàm và tài nguyên tương ứng trước khi thực hiện bất cứ tác vụ nào tới hệ thống.

@user\_ctrl.route('/user-delete/<int:user\_id>')  
@login\_required  
@is\_admin  
**def delete\_user**(user\_id):  
 data = {}  
 user\_del = delete\_user\_md(user\_id)  
 **if** user\_del **is** None:  
 data['success'] = False  
 data['message'] = ERR\_MSG   
 **else**:  
 data['success'] = True  
 **return** jsonify(data)

* 1. Phải có tính năng xóa phiên làm việc hiện tại của người dùng hoặc các cơ chế tương đương đối với các trường hợp quyền người dùng bị thay đổi hoặc bị disable bởi người dùng có thẩm quyền.

Kiểm tra quyền người dùng thông qua các dữ liệu không đổi, ví dụ: user\_id, username, user\_email:

**# Bổ sung việc kiểm tra user có đang được active hay không**

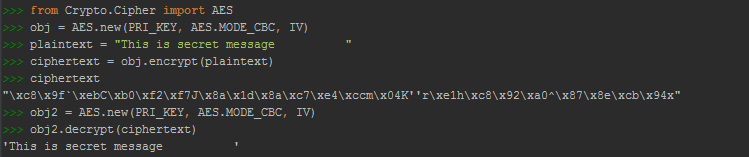
**if** "username" **not in** session **or not** is\_enable\_user(session["username"]):

**# Chuyển hướng về trang đăng nhập**

* 1. Không đặt trang quản trị public internet, trong trường hợp bắt buộc phải đặt public cần giới hạn các IP được phép truy cập hoặc sử dụng cơ chế xác thực đa nhân tố (multiple authentications).

1. **Tương tác với back-end**

* Mã hóa các dữ liệu nhạy cảm trước khi lưu trữ (thông tin tài khoản ngân hàng, private key,..).
  + - Ví dụ sử dụng thuật toán mã hóa AES, secret key lưu trữ trong file cấu hình hoặc DB hoặc sử dụng định dạng file p12 bảo vệ bằng password để lưu trữ privatekey.



* ***SQL***
  1. Sử dụng mô hình truy vấn prepared statement (parameterized query) hoặc các hình thức tương đương.

# sử dụng thư viện sqlalchemy

user = NnhUser.query.join(NnhGroupUser, NnhGroup).filter(NnhUser.user\_name == username, NnhGroup.group\_index == 1).first()

* 1. Trong 1 số trường hợp không sử dụng được các mô hình ở trên, cần thiết lập danh sách whitelist các đầu vào mong muốn.

# order\_by

**if not** check\_none\_data(["news\_col"], search\_data):  
 """  
 {value: 1, desc: 'Thời gian crawl'},  
 {value: 2, desc: 'Đối tượng'},  
 {value: 3, desc: 'Nhóm crawler'},  
 {value: 4, desc: 'Tag'},  
 {value: 5, desc: 'Trạng thái'}  
 """  
 value = search\_data['news\_col']  
 **if** value == 1:  
 **if not** check\_none\_data(["news\_order"], search\_data) **and** search\_data["news\_order"] == 2:  
 news\_search = news\_search.order\_by(NnhNews.news\_crawl\_date.asc())  
 **else**:  
 news\_search = news\_search.order\_by(NnhNews.news\_crawl\_date.desc())  
 **if** value == 2:  
 **if not** check\_none\_data(["news\_order"], search\_data) **and** search\_data["news\_order"] == 2:  
 news\_search = news\_search.order\_by(NnhNews.object\_idobject.asc())  
 **else**:  
 news\_search = news\_search.order\_by(NnhNews.object\_idobject.desc())  
 **if** value == 3:  
 **if not** check\_none\_data(["news\_order"], search\_data) **and** search\_data["news\_order"] == 2:  
 news\_search = news\_search.order\_by(NnhNews.news\_crawl\_group.asc())  
 **else**:  
 news\_search = news\_search.order\_by(NnhNews.news\_crawl\_group.desc())  
 **if** value == 4:  
 **if not** check\_none\_data(["news\_order"], search\_data) **and** search\_data["news\_order"] == 2:  
 news\_search = news\_search.order\_by(NnhNews.news\_tag.asc())  
 **else**:  
 news\_search = news\_search.order\_by(NnhNews.news\_tag.desc())  
 **if** value == 5:  
 **if not** check\_none\_data(["news\_order"], search\_data) **and** search\_data["news\_order"] == 2:  
 news\_search = news\_search.order\_by(NnhNews.news\_status.asc())  
 **else**:  
 news\_search = news\_search.order\_by(NnhNews.news\_status.desc())  
**else**:  
 news\_search = news\_search.order\_by(NnhNews.news\_crawl\_date.desc())

* ***NoSQL***
  1. Không mở service ra ngoài public internet, cài đặt trong môi trường mạng an toàn (trusted enviroments).
     + Cấu hình chỉ mở localhost interface trong trường hợp ứng dụng cùng host với database, tắt các các dịch vụ không cần thiết. Trường hợp, cần kết nối database remote, cấu hình firewall giới hạn ip cho phép kết nối.
     + Ví dụ, với mongodb 3.0.7, cấu hình chỉ lắng nghe trên interface 127.0.0.1, tắt http, json, rest interface.

|  |
| --- |
| net:  port: 27017  bindIp: 127.0.0.1  http:  enabled: false  JSONPEnabled: false  RESTInterfaceEnabled: false |

* 1. Đối với các hệ NoSQL có hỗ trợ xác thực, cần cấu hình xác thực khi truy cập.

Mặc định các hệ NoSQL tắt tính năng xác thực, cụ thể đối với mongodb 3.0.7, sau khi cài đặt, server chưa có người dùng và cũng không bật xác thực, thực hiện các bước sau để tạo người dùng với các quyền tương ứng và bật tính năng xác thực:

* + - Tạo người dùng quản trị server

|  |
| --- |
| use admin  db.createUser(  {  user: "administrator",  pwd: "M@tkhauKh0Do@n",  roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]  }  ) |

* + - Tạo người dùng quản trị cho 1 database (ví dụ DB baseline)

|  |
| --- |
| use baseline  db.createUser(  {  user: "admin\_baseline",  pwd: "M@tkhauKh0Do@n",  roles: [ { role: "userAdmin", db: "baseline" } ]  }  ) |

* + - Tạo người dùng sử dụng DB đó (có quyền thêm, sửa, xóa truy vấn)

|  |
| --- |
| use baseline  db.createUser(  {  user: "user\_baseline",  pwd: "M@tkhauKh0Do@n",  roles: [ { role: "readWrite", db: "baseline" } ]  }  ) |

* + - Cuối cùng, cấu hình mongodb bật tính năng xác thực và phân quyền

|  |
| --- |
| security:  authorization: enabled |

* 1. Phụ thuộc vào hệ NoSQL sử dụng, sử dụng các api hỗ trợ truy vấn an toàn hoặc thực hiện escape các ký tự đặc biệt khi xây dựng câu truy vấn.

Đối với Flask, sử dụng PyMongo khi thực hiện các thao tác thêm, sửa, xóa, tìm kiếm:

* + - Thêm document cho collection

last\_change = datetime.datetime.utcnow()  
user = {'username': data['username'], 'password': data['password'],  
 'last\_change': last\_change}  
mongo.db.user.insert(user)

* + - Sửa document trong collection

user = mongo.db.user  
user.update({'username': data['username']}, {'$set': {'reset\_path': reset\_path}}, False, True)

* + - Xóa document trong collection

user = mongo.db.user  
res = user.remove({'username': 'anhvtm4'})

* + - Tìm kiếm document trong collection

user = mongo.db.user  
res = user.find\_one({'username': data['username']})

* ***XPath***
  1. Thiết lập danh sách whitelist các ký tự đầu vào mong muốn, đầu vào nên là tập hợp của chữ cái, chữ số.

pattern = re.compile("^[a-zA-Z0-9]\*$")  
**if** pattern.match(data) **is not** None:

# data hợp lệ

* 1. Lập danh sách blacklist các ký tự đặc biệt (( ) = ‘[ ] : , \* / và dấu cách), loại bỏ các đầu vào có chứa các ký tự nằm trong blacklist.

black\_list = ["(", ")", "=", "'", "[", "]", ":", "\*", "/", " ", ","]  
**for** s **in** black\_list:  
 **if** s **in** data:

# data không hợp lệ

* ***LDAP***
  1. Thiết lập danh sách whitelist các ký tự đầu vào mong muốn, đầu vào nên là tập hợp của chữ cái, chữ số.

pattern = re.compile("^[a-zA-Z0-9]\*$")  
**if** pattern.match(data) **is not** None:

# data hợp lệ

* 1. Lập danh sách blacklist các ký tự đặc biệt (( ) ; , \* | & = và nullbyte), loại bỏ các đầu vào có chứa các ký tự nằm trong blacklist.

black\_list = ["(", ")", ";", ",", "\*", "|", "&", "="]  
**for** s **in** black\_list:  
 **if** s **in** data:

# data không hợp lệ

* ***Tương tác với OS***
  1. Sử dụng các API hỗ trợ việc thực thi câu lệnh hệ thống.

filename = user\_input["file\_name"] # dữ liệu không tin cậy

# Thiết lập giá trị shell=False

Popen(["ls", "-l", filename], shell=False)

# Trong trường hợp phải enable tính năng shell, cần thực hiện xử lý tham số do

# người dùng điều khiển

**from** pipes **import** quote  
Popen(["ls -l " + quote(filename)], shell=True)

* 1. Không truyền trực tiếp dữ liệu người dùng truyền lên tới OS, trong trường hợp bắt buộc cần thiết lập danh sách whitelist các đầu vào mong muốn.

pattern = re.compile(  
 "^(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$")  
**if** pattern.match(ip):  
 Popen(["ping -c 1 " + ip], shell=True)

* ***Tương tác với file***
  1. Không truyền trực tiếp dữ liệu từ người dùng đến các hàm include file.
  2. Lập danh sách whitelist các định dạng file được phép upload.

# Ví dụ là chức năng upload file ảnh

ALLOWED\_EXTENSIONS = set(['jpg', 'png', 'jpeg'])

* 1. Validate file hợp lệ bằng cách kiểm tra đồng thời file header, phần mở rộng của file và nội dung file.

**def allowed\_file**(filename):  
 **return** '.' **in** filename **and** filename.rsplit('.', 1)[1].lower() **in** ALLOWED\_EXTENSIONS

@dashboard\_ctrl.route('/upload', methods=['GET', 'POST'])  
**def upload\_file**():  
 **if** request.method == 'POST':  
 file\_upload = request.files['file']  
 **if** file\_upload **and** allowed\_file(file\_upload.filename):  
 filename = secure\_filename(file\_upload.filename)  
 file\_upload.save(os.path.join(UPLOAD\_FOLDER, filename))

* 1. Với các trường hợp không bắt buộc thì không lưu file upload trong thư mục web, bỏ quyền thực thi trên thư mục upload.
  2. Khi cần refer tới các file tồn tại trên hệ thống cần thiết lập danh sách whitelist đầu vào mong muốn hoặc gán các giá trị định danh tương ứng cho các file thay vì truyền tên file.

**def is\_white\_list\_file**(filename):  
 filename = secure\_filename(filename)  
 **return** filename **in** WHITE\_LIST\_FILE

* 1. Không trả về đường dẫn tuyệt đối của file.
  2. Tất cả dữ liệu, tài nguyên hệ thống (báo cáo, file upload, file cấu hình...) không được lưu trong thư mục cho phép truy cập trực tiếp không qua xác thực.
* ***Xử lý back-end HTTP request***
  1. Khi tạo http request phía server, các tham số GET, POST cho request đó tránh tạo từ dữ liệu phía người dùng, hoặc phải được kiểm tra cẩn thận để chống ghi đè các tham số khác.

unsafePostData = "action=transfer&amount=" + urllib.quote(amount) + "&recipient=" + urllib.quote(recipient)

Ví dụ: amount = '2&recipient=10', recipient = '1'  
unsafePostData : action=transfer&amount=2%26recipient%3D1&recipient=1

* **Tương tác với XML**
  1. Tắt tính năng external entity resolve và remote doctype retrival của xml parser khi đọc dữ liệu xml.

**# Sử dụng thư viện** defusedxml thay vì thư viện xml mặc định của Python

**from** defusedxml.ElementTree **import** parse  
et = parse(xmlfile)

* 1. Kiểm tra dữ liệu người dùng, encode các kí tự đặc biệt (<>/) khi tạo dữ liệu xml.

1. **Kiểm soát dữ liệu đầu vào**
   1. Việc kiểm tra dữ liệu đầu vào phải được thực hiện phía server.

Việc kiểm tra tính hợp lệ của dữ liệu như định dạng, kích thước, cú pháp,… phải được thực hiện phía server.

* 1. Thực hiện việc kiểm tra dữ liệu từ tất cả các nguồn dữ liệu có tương tác với người dùng (Các tham số lấy từ GET/POST request, HTTP Headers, dữ liệu lấy từ DB, dữ liệu từ file upload,…).
  2. Xác định rõ chuẩn định dạng encode của dữ liệu đầu vào, thực hiện validate dữ liệu sau khi đã decode đầu vào về 1 định dạng chuẩn và nhất quán.
  3. Validate kiểu dữ liệu, phạm vi, độ dài dữ liệu và định dạng dữ liệu.

**def is\_valid\_email**(email):  
 pattern = re.compile("^([\w-]+(?:\.[\w-]+)\*)@((?:[\w-]+\.)\*\w[\w-]{0,66})\.([a-z]{2,6}(?:\.[a-z]{2})?)$")  
 **return** (False, True)[pattern.match(email) **is not** None]

**def is\_valid\_password**(password):  
 pattern = re.compile("^((?=(.\*[a-zA-Z]){1,})(?=(.\*[\d]){1,})(?=(.\*[\W]){1,})(?!.\*\s)).{8,50}$")  
 check = (False, True)[pattern.match(password) **is not** None]  
 **if** check:  
 **with** open("black\_list\_pass.txt") **as** f:  
 **for** line **in** f:  
 **if** password == line.rstrip():  
 **return** False  
 **return** True  
 **else**:  
 **return** False

* 1. Nếu dữ liệu đầu vào bắt buộc là các ký tự đặc biệt, cần thiếp lập danh sách whitelist các ký tự đầu vào mong muốn.

1. **Kiểm soát dữ liệu đầu ra**
   1. Phải chỉ rõ character encoding cho dữ liệu đầu ra.

# Thiết lập encoding tại file index.html, login.html

<meta charset="utf-8"/>

* 1. Response body phải được encode theo ngữ cảnh sử dụng. Một số trường hợp phổ biến:
     + Đầu ra là html, thực hiện html encode các kí tự đặc biệt (<>’”&) từ các nguồn dữ liệu không an toàn (Các tham số lấy từ GET/POST request, HTTP Headers, dữ liệu lấy từ DB, dữ liệu từ file upload,… có thể điều khiển được bởi người dùng).

#\*\* TH1: Dữ liệu được render từ server \*\*#

init = {'username': session['username']}  
**return** render\_template('tpl/blocks/header.html', init=init)

Thêm directive ng-non-bindable vào thẻ html hiển thị dữ liệu

<div class="header-username" ng-non-bindable>  
 <span>[[ init.username ]]</span>  
</div>

#\*\* TH2: Hiển thị dữ liệu thông thường \*\*#

Phải sử dụng các directive builtin của AngurlarJS để đảm bảo dữ liệu trước khi

hiển thị đã được escape html:

- Đặt dữ liệu trong cặp {{ }}: {{ data }}

- Sử dụng các directive đã hỗ trợ escapse html: ng-model, ng-bind

#\*\* TH3: Sử dụng các thư viện js của bên thứ 3 \*\*#

Phải escape dữ liệu trước khi trả về client

html\_escape\_table = {  
 "&": "&amp;",  
 '"': "&quot;",  
 "'": "&apos;",  
 ">": "&gt;",  
 "<": "&lt;",  
}  
**def html\_escape**(text):  
 **return** "".join(html\_escape\_table.get(c, c) **for** c **in** text)

* + - Đầu ra là json, thực hiện encode dữ liệu trả về dạng object, không trả về dạng mảng.

data = {'key': 1, 'value': res}  
**return** jsonify(data)

* 1. Response header: lọc bỏ các kí tự đặc biệt (\n, \r) do dữ liệu người dùng truyền vào.

header\_to\_response.replace('\n', '').replace('\r', '')

* 1. Cookie trả về cần giới hạn tối thiểu nhất các thuộc tính (domain, path, httponly, expire, secure). Tránh lưu trữ các dữ liệu nhạy cảm trên cookie, nếu cần lưu trữ các dữ liệu nhạy cảm thì phải thực hiện mã hóa các dữ liệu này với thuật toán đối xứng mạnh và key chỉ được lưu trên server.

@app.after\_request  
**def set\_csrf\_cookie**(response):   
 response.set\_cookie('X-CSRFToken', generate\_csrf(), expires=timedelta(minutes=30), path='/some/path', domain='some.domain.com', secure=True, httponly=True)  
 **return** response

* 1. Hạn chế việc chuyển hướng, chuyển tiếp đến các URI khác. Nếu ứng dụng có chức năng này cần phải lập danh sách whitelist các URI được phép thực hiện chuyển hướng, chuyển tiếp.

1. **Kiểm soát ngoại lệ và ghi log ứng dụng**
   1. Xử lý các ngoại lệ bằng try-catch và trả về các thông báo lỗi chung đã custom, thông báo lỗi trả về không được chứa các thông tin nhạy cảm của người dùng, hệ thống,...

**def commit\_db**(app):  
 session = db\_session()  
 **try**:  
 session.commit()  
 **except**:  
 app.logger.error('Error’s Content!')  
 session.rollback()  
 **raise  
 finally**:  
 session.close()

**# Cấu hình thông báo lỗi với nội dung tổng quát**

@app.errorhandler(Exception)  
**def default\_exception**(error):  
 app.logger.debug(error)  
 **return** jsonify(error={'code': 500, 'message': 'Some errors were happened! Please contact with administrator to be supported!'}), 500

* 1. Các thông tin lỗi, ngoại lệ này phải được log lại để phục vụ bảo trì, xác định nguyên nhân lỗi ứng dụng.

**# Cấu hình việc xử lý log**

**def configure\_log\_handlers**(app):log\_filename = 'template-frontend.log'  
 current\_file\_path = os.path.abspath(\_\_file\_\_)  
 log\_dir = os.path.dirname(current\_file\_path)  
 log\_dir = os.path.dirname(log\_dir)  
 log\_filename = os.path.join(log\_dir, log\_filename)  
 formatter = logging.Formatter("[%(asctime)s] {%(pathname)s:%(lineno)d} %(levelname)s - %(message)s")  
 handler\_rotating\_file = RotatingFileHandler(log\_filename, maxBytes=10000000, backupCount=5)  
 handler\_rotating\_file.setLevel(logging.INFO)  
 handler\_rotating\_file.setFormatter(formatter)  
  
 handler\_console = logging.StreamHandler(sys.stdout)  
 app.logger.addHandler(handler\_rotating\_file)  
 app.logger.addHandler(handler\_console)  
 app.logger.setLevel(logging.DEBUG)  
 app.logger.info('App log file: {0}'.format(log\_filename))

# Cấu hình cho app sử dụng log handler

app = CustomFlask(\_\_name\_\_)  
configure\_log\_handlers(app)

# Log các thông tin lỗi

**try**:  
 some\_func()  
**except**:  
 app.logger.error('Error’s Content!')

* 1. File log phải được đặt tại thư mục an toàn ngoài thư mục web.
  2. Không log lại các dữ liệu nhạy cảm (thông tin người dùng, session id, thông tin hệ thống).
  3. Giới hạn người dùng cho phép truy cập file log.

1. **Sử dụng framework, lib (third-party components)**
   1. Loại các thành phần, lib không cần thiết.
   2. Sử dụng phiên bản mới nhất của framework tại thời điểm phát triển ứng dụng.
      * Truy cập trang chủ của framework: <http://flask.pocoo.org/>, tải package tại đường dẫn ***Last Version***
      * Hoặc: pip install flask
   3. Thường xuyên cập nhật các bản vá lỗi cho framework.

Theo dõi thông tin cập nhật của framework tại <http://flask.pocoo.org/docs/0.10/changelog/>.

* 1. Tắt chế độ development của framework khi triển khai ứng dụng thực tế.

# Cấu hình trong file config.py

DEBUG = False

# Hoặc

app.run(host='0.0.0.0', port=8888, debug=False)

1. **Xử lý bussiness logic**

Xử lý business logic phụ vào từng ứng dụng nhưng lưu ý:

* 1. Lập trình viên phải nắm rõ được toàn bộ luồng nghiệp vụ của ứng dụng, phải xây dựng các cây testcase cho từng nghiệp vụ để tránh bỏ sót các trường hợp có thể xảy ra.
  2. Các chức năng quan trọng (ví dụ chuyển khoản ngân hàng), sử dụng lock hoặc các hình thức tương đương để tránh lỗi race condition.