**SPRING SECURITY**

**Tổng quát**

* Flow:

Diagram

Description automatically generated

* Authentication và Authorization:
  + Authentication : Khi nói về authentication là ta nói về chức năng đăng nhập vào hệ thống. Authentication nghĩa là chúng ta có phải là người dùng của hệ thống hay không
  + Authorization : Khi nói về authorization ta nói về quyền hạn được phép làm gì? Ví dụ như user và admin. Bước đầu tiên họ phải authentication. Xác thực mình là user trong hệ thống. Tiếp đến tuỳ vào role của họ là admin hay user mà họ chỉ có quyền truy cập một số trang nhất định thuộc thẩm quyền của mình
* Để sử dụng spring security, chúng ta sử dụng dependency *spring-boot-starter-security*
  + Khi nhúng thư viện *spring-boot-starter-security* vào project, Spring Boot sử dụng cấu hình mặc định của class WebSecurityConfigurerAdapter để cấu hình cho phần authorization như sau:

A screen shot of a computer code

Description automatically generated

Mặc định Spring Security sẽ chặn hết tất cả các request, tự động generate login form và sử dụng http basic cho phần authentication. Chúng ta có thể override class WebSecurityConfigurerAdapter để thay đổi các cấu hình này

* + Username mặc định để chúng ta có thể đăng nhập vào là “user” và password là password được generate và in ra trong console log:



A screenshot of a login box

Description automatically generated



* + Chúng ta có thể thay đổi username và password mặc định này bằng cách cấu hình 2 property username và password trong tập tin application.properties. Lúc này default password sẽ không được generate nữa và chúng ta có thể sử dụng username và password mà mình đã khai báo ở trên để đăng nhập

A close-up of words

Description automatically generated

* + Tắt chế độ auto config mặc định của spring security:



**PasswordEncoder**

Spring Security khuyến nghị việc lưu mật khẩu của người dùng dưới dạng mã hóa. PasswordEncoder giúp chúng ta mã hóa và giải mã mật khẩu một cách an toàn. Hiện nay kiểu BRcypt được sử dụng phổ biến và bảo mật hơn md5

A computer screen shot of black text

Description automatically generated

**Custom config mặc định với** **Spring Security 5**

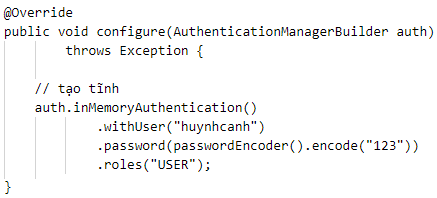
Ở phiên bản Spring Security 5, khi implement **WebSecurityConfigAdapter** chúng ta override 2 method ***configure*** nhận tham số truyền vào khác nhau:

A black text on a white background

Description automatically generated

Thứ nhất là ***configure***method với tham số là **AuthenticationManagerBuilder**. Ở đây chúng ta có thể khai báo thông tin của các user (user, password, role). Method này có liên quan đến authentication

Trường hợp tạo tĩnh:



Trường hợp tạo động:

UserEntity:

A screenshot of a computer program

Description automatically generated

**A screenshot of a computer code

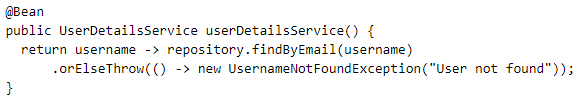
Description automatically generated**

UserRepository:

A black text on a white background

Description automatically generated

Tạo bean UserDetailsService:



Cấu hình method ***configure()***:

A close-up of a computer screen

Description automatically generated

Sử dụng **UserDetaiService** cho phép tạo ra chứng thực đơn giản với username và password. Nếu muốn chứng thực với nhiều field khác nhau chúng ta có thể sử dụng **AuthenticationProvider**

Mặc định, Spring Security sẽ tạo session cho mỗi người dùng sau khi họ đăng nhập thành công (*SessionCreationPolicy.IF\_REQUIRED*). Điều này có thể dẫn đến tốn tài nguyên bộ nhớ (RAM) khi có nhiều người dùng truy cập vào ứng dụng và mỗi người dùng có một session riêng. Cách giải quyết vấn đề này có thể là cấu hình Spring Security để sử dụng stateless authentication (*SessionCreationPolicy.STATELESS*). Khi sử dụng stateless authentication, không có session nào được lưu trữ trong bộ nhớ máy chủ và thông tin xác thực của người dùng được gửi đi trong mỗi yêu cầu dưới dạng token

Thứ hai là ***configure*** method với tham số là **HttpSecurity**. Ở đây chúng ta có thể cấu hình web security cho các HTTP request cụ thể. Method này có liên quan đến authorization

A computer screen shot of a program

Description automatically generated

Tương tự như /login, mặc định Spring cung cấp /logout để xóa session user ra khỏi hệ thống



A screenshot of a log out

Description automatically generated

* Custom login form mặc định của Spring Security:

*A white background with black text

Description automatically generated*

A computer screen shot of text

Description automatically generated

Successhandler cho phép xử lý logic sau khi xác thực thành công và trước khi trả về trang sau khi đăng nhập thành công

**Custom config mặc định với** **Spring Security 6**

**A black text on a white background

Description automatically generated**

Ở Spring Security 6, chúng ta có một số class mới thay thế như sau:

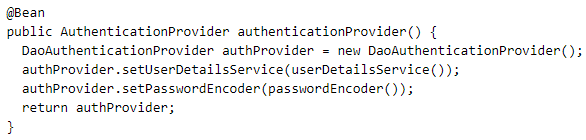
Đối với việc khai báo thông tin của các user liên quan đến authentication chúng ta sẽ sử dụng interface UserDetailsService của Spring Security để khởi tạo bean:

Trường hợp tạo user tĩnh:

**A screenshot of a computer program

Description automatically generated**

Trường hợp tạo user động:



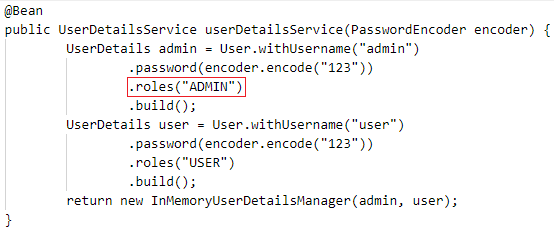
Đối với việc cấu hình web security cho các HTTP request, thì Spring Boot 3.0 đã tạo ra một class mới có tên là SecurityFilterChain để xử lý authorization:

**A screen shot of a computer code

Description automatically generated**

**Role, Authorize và prefix “ROLE\_”**

* Role (vai trò): Vai trò trong Spring Security là một nhóm các quyền có cùng mục tiêu hoặc chức năng, chẳng hạn như "ROLE\_USER" hoặc "ROLE\_ADMIN"
* Authorize (phân quyền): Phân quyền trong Spring Security xác định xem một người dùng cụ thể có quyền truy cập vào một tài nguyên hoặc chức năng cụ thể không. Ví dụ, "ROLE\_ADMIN" có thể có các quyền như "ADMIN\_ACCESS", "ADMIN\_UPDATE"
* Prefix "ROLE\_":
  + Spring Security sử dụng danh sách Authorities (có thể là các vai trò và quyền) để xác định quyền truy cập của người dùng
  + Để phân biệt vai trò (role) và quyền (permissions), Spring Security sử dụng tiền tố "ROLE\_" cho vai trò. Ví dụ, "ROLE\_USER" là một vai trò, trong khi "ADMIN\_ACCESS" là một quyền (permissions)
  + Tiền tố "ROLE\_" là một quy ước để nhận biết và xử lý vai trò một cách dễ dàng
  + Kiểm tra:



Check method ***roles()*** ta thấy Springboot đã tự động thêm prefix "ROLE\_":

A text on a white background

Description automatically generated

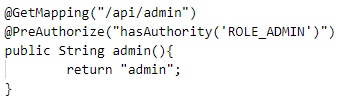
**Phân quyền với @PreAuthorize**

* Enables Spring Security Method Security:

A close-up of black text

Description automatically generated

* Sử dụng **@PreAuthorize** để phân quyền:



**Truy xuất các thông tin user login hiện tại**

A screenshot of a computer

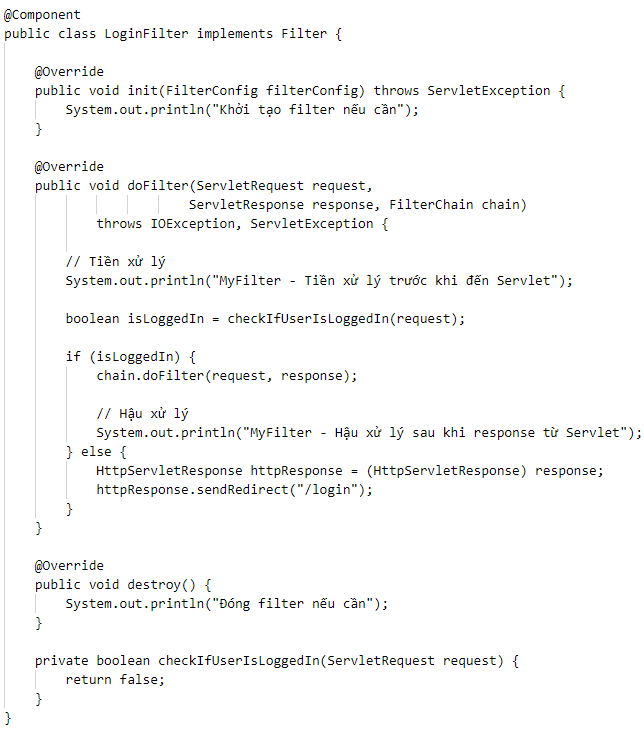
Description automatically generated

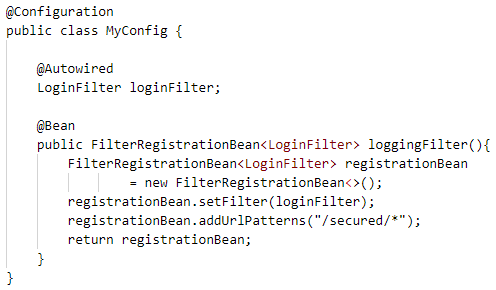
**Servlet Filter và Spring Interceptor**

Servlet Filter và Spring Interceptor là hai cách khác để thực hiện các nhiệm vụ liên quan đến bảo mật trong ứng dụng Spring ngoài việc sử dụng Spring Security

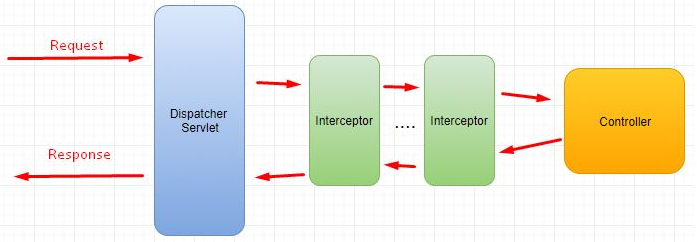
***Servlet Filter***

* Servlet Filter là một thành phần trong Servlet API của Java cho phép thực hiện các tác vụ tiền và hậu xử lý trước và sau khi request đến và response từ Servlet
* Demo:

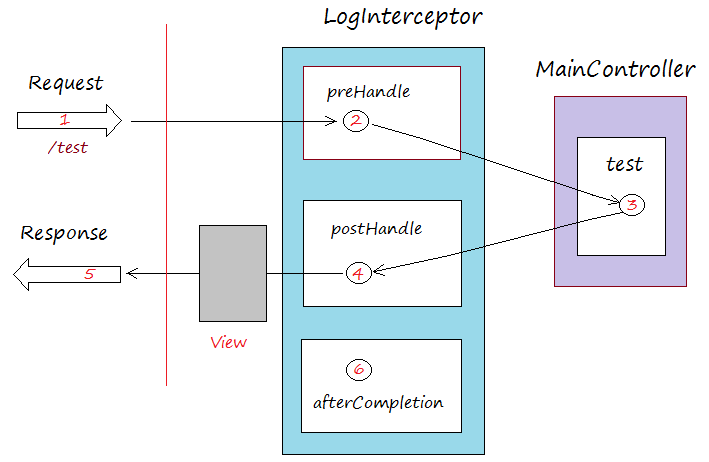




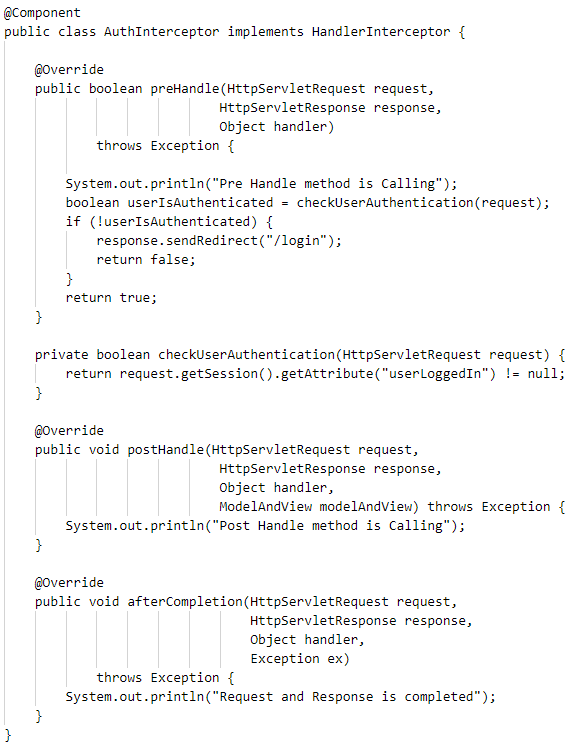
***Spring Interceptor***

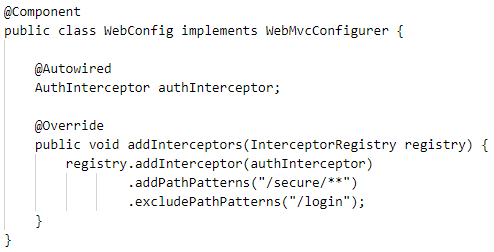


* Spring Interceptor là một phần của Spring Framework và thường được sử dụng trong Spring MVC. Interceptor cho phép chúng ta thực hiện các xử lý trước hoặc sau khi request và response đi qua các controller trong ứng dụng Spring. Spring Interceptor là một khái niệm khá giống với Servlet Filter, với mục tiêu thực hiện các thao tác tiền xử lý và hậu xử lý trên request và response
* Trong Spring, khi một request được gửi đến controller, trước khi request được xử lý bởi Controller, nó phải vượt qua các Interceptor (0 hoặc nhiều)
* Ba method bạn nên biết khi làm việc với Interceptor:



* + preHandle(): Được gọi trước khi request được gửi đến controller. Nó cho phép chúng ta thực hiện các xử lý tiền xử lý trước khi controller xử lý request. Nếu preHandle() trả về true, quá trình xử lý tiếp tục và controller được gọi. Nếu preHandle() trả về false, request sẽ bị chặn và controller sẽ không được gọi
  + postHandle(): Được gọi sau khi controller xử lý request và trước khi response được trả về cho client. Nó cho phép bạn thực hiện các xử lý hậu xử lý, sửa đổi dữ liệu response hoặc thêm thông tin vào model trước khi gửi response
  + afterCompletion(): Được gọi sau khi response đã được gửi đi. Nó cho phép bạn thực hiện các xử lý sau khi toàn bộ quá trình request-response đã hoàn tất, chẳng hạn như việc ghi log hoặc giải phóng tài nguyên
* Demo:





**SPRING OAUTH2**

* Flow login with github:

A diagram of a login with text

Description automatically generated

* Tạo application trên github:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a chat

Description automatically generated

* Config Spring Security Oauth2 Client:

Cấu hình file properties:

A screenshot of a computer program

Description automatically generated

Giá trị scope:

A screenshot of a computer

Description automatically generated

* Cấu hình security:

Cấu hình OAuth2User (Tương tự UserDetails):

A screenshot of a computer program

Description automatically generated

Cấu hình DefaultOAuth2UserService (Tương tự UserDetailsService):

A screenshot of a computer

Description automatically generated

Cấu hình OAuth2 security:

A screenshot of a computer screen

Description automatically generated

* Run server: Cần cho phép application truy cập vào account github hiện tại

**A screenshot of a computer

Description automatically generated**

A screenshot of a login form

Description automatically generated

A screenshot of a social media account

Description automatically generated

**CORS**

…………………………