RIKEI Academy
where the dream begins

Bài 14
Bảo mật ứng dụng Web

Module: BOOTCAMP WEB-BACKEND DEVELOPMENT

RIKEI Academy
where the dream begins

Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài "I18N"

Tóm tắt lại các phần đã học từ bài "I18N"

Thảo luận

Nguy cơ mất an toàn thông tin

Cơ chế bảo mật ứng dụng

Các cơ chế xác thực

Nguy cơ mất an toàn thông tin

- Phân quyền và xác thực người dùng
- Tấn công bởi các phần mềm độc hại
- Mất, hỏng, sửa đổi thông tin
- Mất an toàn thông tin do sử dụng Email, mạng xã hội
- Mất an toàn thông tin đối với Website

Cơ chế bảo mật ứng dụng

- Xác thực (Authentication)
- Phân quyền (Authorization)
- Phân quyền hệ thống tệp tin
- Quản lý tài khoản
- Sử dụng SSL để truyền dữ liệu
- Mã hoáFirewalls

Authentication

- Authentication (xác thực) là một hành động nhằm thiết lập hoặc chứng thực một thông điệp hoặc đối tượng nào đó là đáng tin cậy. Điều này có nghĩa là chúng ta có thể tin tưởng vào những lời khai báo hoặc thông điệp mà đối tượng ấy đưa ra.
- Xác thực cũng còn có nghĩa khác là công nhận nguồn gốc của đối tượng. Để tiến hành xác thực thì thông thường người ta sẽ sử dụng nhiều nhân tố khác nhau để chứng minh.
- Ví dụ: Khi chúng ta trình diện giấy chứng minh hợp thức của mình với một nhân viên thu ngân của ngân hàng, đầu tiên chúng ta được nhân viên thu ngân chứng thực. => Quá trình xác thực chúng ta đáng tin cậy.

Authorization

RIKEI Academy® where the dream begins

- Authorization (Cấp quyền/phân quyền) là quá trình xác định xem một người dùng có quyền truy cập một tài nguyên cụ thể để thực hiện một số hành động hay không.
- Authorization còn được hiểu là sự phân quyền người dùng để họ có quyền truy cập và sử dụng những tính năng mà ứng dụng cung cấp.
- Ví dụ:
  - Sau khi được nhân viên ngân hàng chứng minh chúng ta đáng tin cậy. Chúng ta được cấp quyền để truy cập những thông tin về tài khoản trong ngân hàng của mình. Đương nhiên, chúng ta sẽ không được cấp quyền để truy cập các tài khoản mà chúng ta không sở hữu => Quá trình cấp quyền truy cập.
- Việc cấp quyền không thể được tiến hành mà không có sự xác thựcđi kèm.

Các cơ chế xác thực

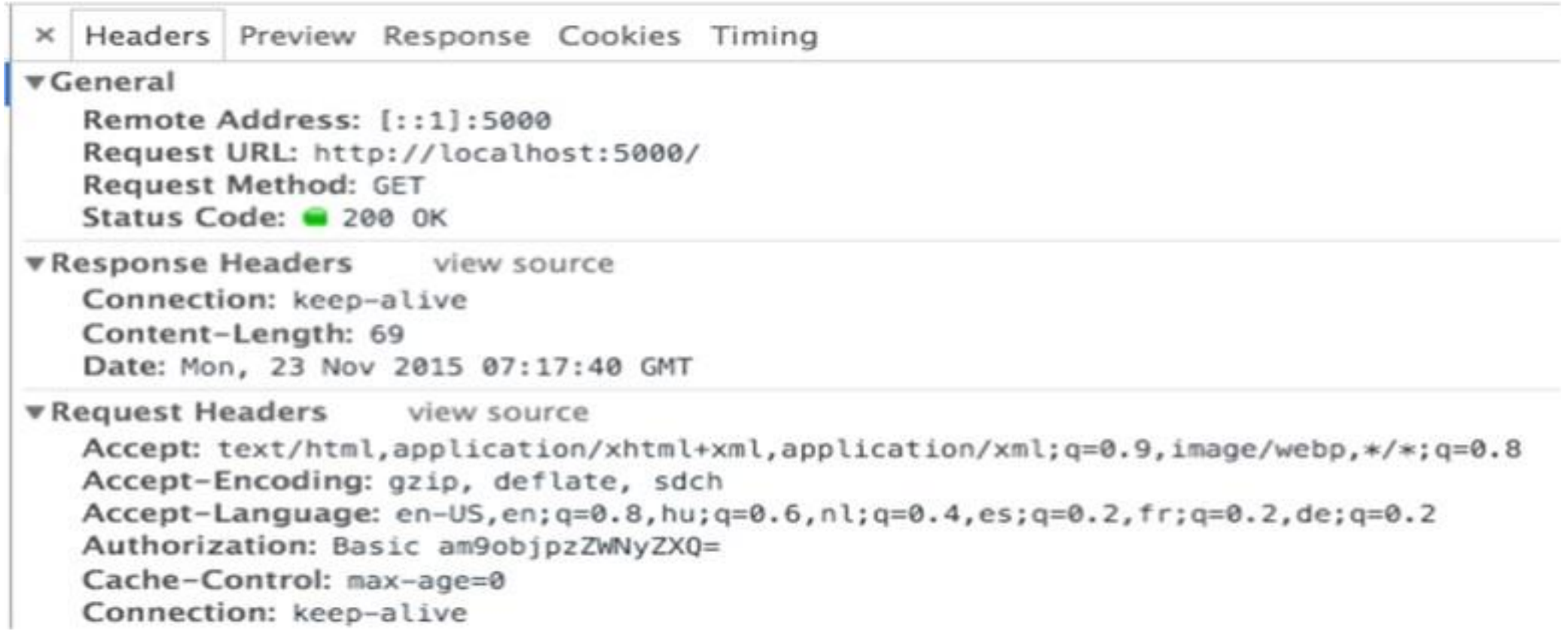
- HTTP Basic
- Cookies
- Tockens
- Signature
- One-time password

HTTP Basic

- Xác thực HTTP Basic là một phương thức để client cung cấp username và password khi thực hiện yêu cầu.
- Đây là cách đơn giản nhất có thể để thực thi kiểm soát truy cập vì nó không yêu cầu cookie, session hoặc bất kỳ thứ gì khác.
- Để sử dụng HTTP Basic, client phải gửi tiêu đề Cấp phép (Authorization) cùng với mọi yêu cầu mà nó thực hiện. Username và password không được mã hóa, nhưng được xây dựng theo cách:
  - Username và password được nối vào một chuỗi duy nhất: username:password
  - Chuỗi này được mã hóa với Base64
  - Từ khóa Basic được đặt trước giá trị được mã hóa này

Ví dụ quan sát thấy trong Chrome

RIKEI Academy
where the dream begins



Hạn chế HTTP Basic

- username và password được gửi cùng với mọi yêu cầu, có khả năng hiển thị chúng - ngay cả khi được gửi qua kết nối an toàn.
- Được kết nối với SSL/TLS, nếu trang web sử dụng mã hóa yếu hoặc kẻ tấn công có thể phá vỡ nó, tên người dùng và mật khẩu sẽ được hiển thị ngay lập tức.
- Không có cách nào để đăng xuất người dùng bằng cách sử dụng xác thực cơ bản.
- Yêu cầu thay đổi mật khẩu thường xuyên.

Cookies

- Để sử dụng cookie cho mục đích xác thực, có một vài nguyên tắc chính cần tuân theo:
  - Luôn sử dụng HttpOnly cookies
    - Để giảm thiểu khả năng tấn công XSS luôn sử dụng cờ HttpOnly khi thiết lập cookie. Bằng cách này, họ sẽ không hiển thị trong document.cookies.
  - Luôn sử dụng cookie đã ký
    - Với cookie đã ký, máy chủ có thể biết cookie có được khách hàng sửa đổi hay không.

Ví dụ trong Chrome

- Xem máy chủ thiết lập cookie

▼General
Remote Address: 185.63.147.10:443
Request URL: https://www.linkedin.com/nhome/?trk=hb_signin
Request Method: GET
Status Code: 200 OK

▼Response Headers
cache-control: no-cache, no-store
content-encoding: gzip
content-type: text/html; charset=utf-8
date: Mon, 23 Nov 2015 07:25:35 GMT
expires: Thu, 01 Jan 1970 00:00:00 GMT
pragma: no-cache
server: Play
set-cookie: lidc="b=TB16:g=272:u=1:i=1448263535:t=1448349780:s=AQHtWo_H6eKMTC-ysMUDV4m_j19p94Ha"; Expires=Tue, 24 Nov 2015 07:23:00 GMT; domain=.linkedin.com; Path=/

- Sau đó, tất cả các yêu cầu sử dụng tập hợp cookie cho miền đã cho:

Request Headers
:host: www.linkedin.com
:method: GET
:path: /home?trk=nav_responsive_tab_home
:scheme: https
:version: HTTP/1.1
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
accept-encoding: gzip, deflate, sdch
accept-language: en-US,en;q=0.8,hu;q=0.6,nl;q=0.4,es;q=0.2,fr;q=0.2,de;q=0.2
cookie: bcookie="v=2&8b44d748-8d82-46a4-8577-36c00172794b"; bscookie="v=1&201511230722494b3c36f8-d94e-4796-8b49-a5371e6804d1AQEWRbrseJHuyZDHtFipryvvTHp0Sh7o"; L1e=108b50b8; _gat=1; L1c=506114f5; visit="v=1&M"; _ga=GA1.2.96221765.1448263372; oz_props_fetch_size1_undefined=undefined; wutan=i4YA1rjunkpRvR+nkTLFo1Tc8VA9+AOYr0i6eMU+s84=; sl="v=1&i-LQe"; li_at=AQEDARuYmFkFnETwAAABUTM7yS8AAAFRM6mmL04AHZ3tRDqVEG7yio6z6WHQZWKXW0KcCuaX8VpT1gUf8RaKq1YzYoXazo47r3u4tP28mIy5bTs3GLWoym5Xg9XQpqeT3lwdy3Ps5pLNICgTwOjCsoDa; JSESSIONID="ajax:8745093687238714270"; liap=true; lidc="b=TB16:g=272:u=1:i=1448263557:t=1448349780:s=AQH_zUY2E78NFlUbZOHCzlfOatKxlxxy"; RT=s=1448263559420&r=https%3A%2F%2Fwww.linkedin.com%2Fnhome%2F%3Ftrk%3Dhb_signin; share_setting=PUBLIC; _lipt=0_1bmJJGSdPtMK4q9DLkNulXXW5-iWZ3vLKJSmIXmUDL9otamnhKTL_Tp4xsiTWowWeXz07fMN_z7blHodRWXvgr9M72nA7ucghFT4sQG3E6fD3sF9zVmVHVynVYd9lijIAtUYh3Vz8BfPe0oLeRwdL8; lang="v=2&lang=en-us"; sdsc=1%3A1SZM1shxDNbLt36wZwCgPgvN58iw%3D

Hạn chế sử dụng Cookie

- Cần phải nỗ lực nhiều hơn để giảm thiểu các cuộc tấn công CSRF
- Không tương thích với REST - vì nó đưa vào một trạng thái thành một giao thức không trạng thái

Tokens

- JWT (JSON Web Token) là 1 tiêu chuẩn mở (RFC 7519) định nghĩa cách thức truyền tin an toàn giữa các thành viên bằng 1 đối tượng JSON. Thông tin này có thể được xác thực và đánh dấu tin cậy nhờ vào "chữ ký" của nó. Phần chữ ký của JWT sẽ được mã hóa lại bằng **HMAC** hoặc **RSA**.
- JWT bao gồm ba phần:
  - Tiêu đề, chứa loại mã thông báo và thuật toán băm
  - Tải trọng, chứa các xác nhận quyền sở hữu
  - Chữ ký, có thể được tính như sau nếu bạn chọn HMAC SHA256:

    HMACSHA256(base64UrlEncode(header) + **"."** + base64UrlEncode(payload), secret)
- Nhược điểm
  - Cần phải nỗ lực nhiều hơn để giảm thiểu các cuộc tấn công XSS.

Signatures

- Chữ ký số khóa công khai là mô hình sử dụng các kỹ thuật mật mã để gắn với mỗi người sử dụng một cặp khóa công khai - bí mật và qua đó có thể ký các văn bản điện tử cũng như trao đổi các thông tin mật. Khóa công khai thường được phân phối thông qua chứng thực khóa công khai.
- Quá trình sử dụng chữ ký số bao gồm 2 quá trình: tạo chữ ký và kiểm tra chữ ký.

One-Time Passwords

- Thuật toán One-Time passwords tạo mật khẩu một lần với bí mật dùng chung và thời gian hoặc bộ đếm hiện tại:
  - Thuật toán Time-based One-time Password dựa trên thời gian hiện tại.
  - Thuật toán HMAC-based One-time Password dựa trên bộ đếm.
- Các phương thức này được sử dụng trong các ứng dụng tận dụng xác thực hai yếu tố: người dùng nhập username và password, sau đó cả máy chủ lẫn máy khách đều tạo mật khẩu một lần.

Certificate (Chứng thực)

- Hệ thống chứng thực là một hạ tầng an ninh mạng được xây dựng trên một hạ tầng cơ sở khóa công khai (PKI) cung cấp các giải pháp đảm bảo an toàn cho các hoạt động (gọi chung là giao dịch) thông qua mạng.
- Hệ thống chứng thực cung cấp các dịch vụ đảm bảo an toàn cho các giao dịch thông qua mạng. Các dịch vụ cơ bản mà một hệ thống chứng thực cung cấp bao gồm:
  - Dịch vụ xác thực: nhằm xác định xem ai đang giao dịch với mình.
  - Dịch vụ bảo mật: đảm bảo tính bí mật của thông tin, người không có thẩm quyền không thể đọc được nội dung của thông tin.
  - Dịch vụ toàn vẹn: khẳng định thông tin có bị thay đổi hay không.
  - Dịch vụ chống chối bỏ: cung cấp các bằng chứng chống lại việc chối bỏ một hành động đã thực hiện hay đã diễn ra
  - Như vậy sử dụng hệ thống chứng thực sẽ đảm bảo, bí mật, toàn vẹn cho thông tin được truyền qua mạng, xác thực được người dùng và chống chối bỏ các hành động hay sự kiện đã xảy ra.

Certificate (Chứng thực)

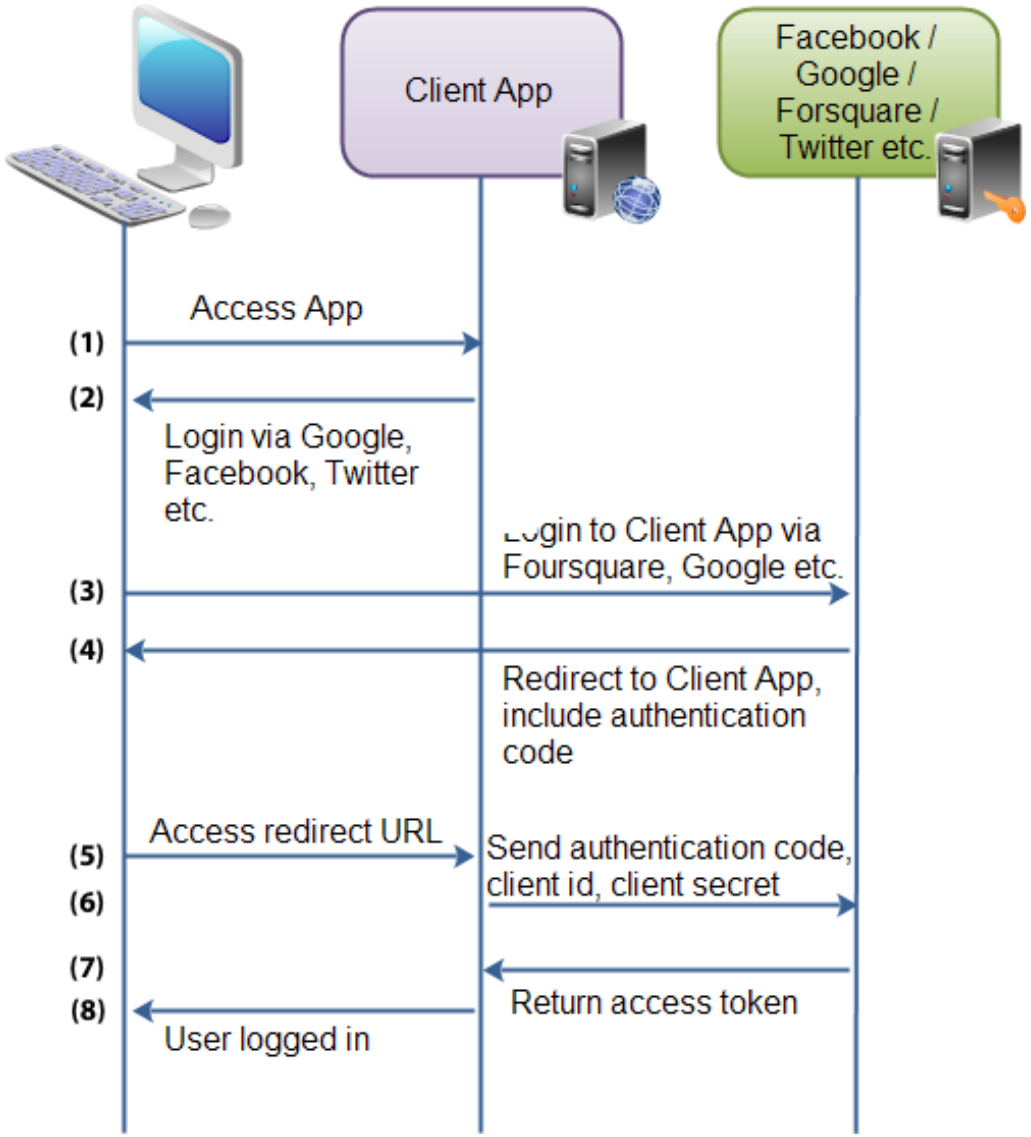
- Hệ thống chứng thực gồm 2 thành phần:
  - Thành phần thực hiện các nhiệm vụ về quản lý chứng thư số như: đăng ký và phát hành, thu hồi … chứng thư số.
  - Thành phần thực hiện chức năng xác định xem một chứng thư số có hợp lệ hay không
- Cơ quan chứng thực (Certification Authority - CA) có thẩm quyền cấp phát, thu hồi, quản lý chứng thư số cho các thực thể thực hiện các giao dịch an toàn. Cơ quan chứng thực là một thành phần chính của hệ thống chứng thực.
- Một số ứng dụng hệ thống chứng thực
  - Thuế điện tử (E-Tax Filing)
  - Thanh toán trực tuyến (E-Payment)
  - Bảo mật email
  - …

OAuth2

- OAuth là viết tắt của Open Authorization là một ột giao thức mở nhằm cung cấp cách thức đơn giản và tiêu chuẩncho các thao tác xác thực an toàn trong các ứng dụng trên web, mobile & desktop.
- **OAuth** đã ra mắt thế hệ thứ 2, gọi là **OAuth2**, là 1 phiên bản nâng cấp hoàn thiện hơn của **OAuth** đời đầu tiên.

OAuth2

- Cơ chế của OAuth



RIKEI Academy
where the dream begins

Thảo luận

Bảo mật trong Spring

Bảo mật trong Spring

- Spring Security là một dự án nổi bật trong hệ sinh thái Spring, cung cấp các dịch vụ bảo mật toàn diện cho các ứng dụng doanh nghiệp có nền tảng Java EE.
- Spring Security là một framework tập trung vào việc cung cấp khả năng xác thực (authentication) và phân quyền (authorization) cho ứng dụng Java.
  - **Authentication**: là tiến trình thiết lập một principal. Principal có thể hiểu là một người, hoặc một thiết bị, hoặc một hệ thống nào đó có thể thực hiện một hành động trong ứng dụng.
  - **Authorization** hay **Access-control**: là tiến trình quyết định xem một principal có được phép thực hiện một hành động trong ứng dụng hay không. Trước khi diễn tiến tới Authorization, principal cần phải được thiết lập bởi Authentication.

Các thành phần trong Spring Security (1)

- **SecurityContext** là interface, lưu trữ tất cả các chi tiết liên qua đến bảo mật trong ứng dụng. Khi kích hoạt Spring Security trong ứng dụng thì SecurityContext cũng sẽ được kích hoạt theo.
- Sử dụng lớp **SecurityContextHolder** để truy cập vào SecurityContext. Lớp này lưu trữ security context hiện tại của ứng dungnj, gồm chi tiết của principal đang tương tác với ứng dụng.
- Đối tượng **Authentication** biểu diễn thông tin của principal.

Các thành phần trong Spring Security (2)

- Ví dụ: Lấy username của principal đã được xác thực.

```
Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();

if (principal instanceof UserDetails) {
   userName = ((UserDetails)principal).getUsername();
} else {
   userName = principal.toString();
}
```

- Principal ở đây đơn giản là một đối tượng và được ép kiểu sang UserDetails

Các thành phần trong Spring Security (3)

- **UserDetails** là một interface, đại diện cho một principal. Gồm các phương thức sau:

Tên phương thức	Mô tả
getAuthorities()	Trả về danh sách các quyền của người dùng
getPassword()	Trả về password đã dùng trong qúa trình xác thực
getUsername()	Trả về username đã dùng trong qúa trình xác thực
isAccountNonExpired()	Trả về true nếu tài khoản của người dùng chưa hết hạn
isAccountNonLocked()	Trả về true nếu người dùng chưa bị khóa
isCredentialsNonExpired()	Trả về true nếu chứng thực (mật khẩu) của người dùng chưa hết hạn
isEnabled()	Trả về true nếu người dùng đã được kích hoạt

Các thành phần trong Spring Security (4)

- Phương thức getAuthorities() trả về một tập hợp các đối tượng GrantedAuthority. GrantedAuthority là một quyền được ban cho principal.
- Các quyền đều có tiền tốt là ROLE_, ví dụ: ROLE_ADMIN, ROLE_USER, ROLE_ADMIN, ROLE_DBA

Triển khai cơ chế bảo mật sử dụng password(1)

Cấu hình dependency trong file build.gradle

```
compile group: 'org.springframework.security', name: 'spring-security-web', version: '5.4.2'
compile group: 'org.springframework.security', name: 'spring-security-config', version: '5.4.2'
compile group: 'org.springframework.security', name: 'spring-security-taglibs', version: '5.4.2'
compile group: 'org.thymeleaf.extras', name: 'thymeleaf-extras-springsecurity5', version: '3.0.4.RELEASE'
```

Triển khai cơ chế bảo mật sử dụng password(2)

- Để kích hoạt Spring Security, tạo lớp kế thừa lớp WebSecurityConfigurerAdapter

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends
  WebSecurityConfigurerAdapter { @Override
    protected void configure(HttpSecurity http) throws
                                Exception {
  http.authorizeRequests().antMatchers("/").authenticated()
          .and()
          .formLogin();
  }
  @Override
  protected void configure(AuthenticationManagerBuilder auth) throws
    Exception {
    auth.inMemoryAuthentication()
          .withUser("user").password("{noop}12345").roles("USER")
          .and()
          .withUser("admin").password("{noop}12345").roles("ADMIN");
  }
}
```

authenticated() yêu cầu đăng nhập để truy cập

permitAll() cho phép tất cả các user đều được phép truy cập

Triển khai cơ chế bảo mật sử dụng password(2)

- WebSecurityConfig tạo ra một bộ lọc Servlet được gọi là springSecurityFilterChain, chịu trách nhiệm bảo vệ URL, xác nhận username và password, chuyển hướng tới form đăng nhập... trong ứng dụng.
- Phương thức configure(AuthenticationManagerBuilder auth) cấu hình xác thực bộ nhớ với thông tin đăng nhập và vai trò của người dùng.
- Phương thức configure(HttpSecurity http) cấu hình bảo mật dựa trên web cho tất cả các yêu cầu HTTP. Theo mặc định, nó sẽ được áp dụng cho tất cả các yêu cầu, nhưng có thể bị hạn chế bằng cách sử dụng requestMatcher() hoặc các phương thức tương tự khác.

Triển khai cơ chế bảo mật sử dụng password(3)

- Đăng ký springSecurityFilterChain sử dụng lớp AbstractSecurityWebApplicationInitializer

```
package com.codegym.security;

import org.springframework.security.web.context.AbstractSecurityWebApplicationInitializer;

public class SecurityInit extends AbstractSecurityWebApplicationInitializer {
}
```

- Cấu hình này chỉ đăng ký bộ lọc springSecurityFilterChain cho mỗi URL cho ứng dụng của bạn.

Triển khai cơ chế bảo mật sử dụng password(4)

- Đảm bảo SecurityInit được tải trong AppInit

```
package com.codegym.configuration;
import com.codegym.security.SecurityConfig;
import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;
public class AppInit extends AbstractAnnotationConfigDispatcherServletInitializer {
    @Override
    protected Class<?>[] getRootConfigClasses() {
        return new Class[]{SecurityConfig.class};
    }
    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class[]{AppConfig.class};
    }
    @Override
    protected String[] getServletMappings() {
        return new String[]{"/"};
    }
}
```

Triển khai cơ chế bảo mật sử dụng password(4)

- Spring Security sẽ tự động tạo form đăng nhập nếu ta không định nghĩa cho nó.
- Trong Spring Security, trang xử lý submit form đăng nhập mặc định là /login.
- Sau khi đăng nhập một đối tượng user sẽ được lưu trong session, đối tượng user này sẽ gồm các thông tin như username, password, các quyền.

Triển khai cơ chế phân quyền dựa trên vai trò

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
   @Override
   protected void configure(HttpSecurity http) throws Exception {
      http.authorizeRequests().antMatchers("/").authenticated().and()
            .authorizeRequests().antMatchers("/user**").hasRole("USER")
            .and()
            .authorizeRequests().antMatchers("/admin**").hasRole("ADMIN")
            .and()
            .formLogin();
   }
   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception {
      auth.inMemoryAuthentication()
            .withUser("user").password("{noop}12345").roles("USER")
            .and()
            .withUser("admin").password("{noop}12345").roles("ADMIN");
   }
}
```

Chỉ cho phép các user có GrantedAuthority là ROLE_roleName mới được phép truy cập

Tên quyền do người dùng tự đặt

Demo

Triển khai cơ chế bảo mật

RIKEI Academy
where the dream begins

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: Automated Testing