Phát triển hệ thống thông tin doanh nghiệp (ESDC)

Bài tập lớn: Tìm hiểu Spring Security

Huỳnh Văn Duy - 51703006
Trương Nhật Hoàng – 51703092
Trần Quốc Lĩnh - 51703124

Tóm tắt

Bài báo cáo này trình bày các cơ sở lý thuyết căn bản của Srping securiy và hướng dẫn xây dựng một ứng ụng web đơn giản có sử dụng Srping security.

Spring Security là gì?

Spring security là một framework cho phép lập trình viên áp đặt các hạn chế về bảo mật dựa trên nền Spring-framework thông qua các thành phần của Java Enterprise Edition (Java EE). Đơn giản hơn, nó là một thư viện mà lập trình viên có thể tự do sử dụng, tùy chỉnh, mở rộng để phù hợp với nhu cầu. Spring Security cũng gắn liền với Spring Web MVC. Tác dụng chính của nó là thực hiện xác thực (authentication) và ủy quyền (authorization) ở mức Web request cũng như ở lời gọi hàm. Ưu điểm lớn nhất của framework này là dù nó rất mạnh mẽ nhưng khả năng tùy biến cũng rất cao. Tuy phải tuân theo quy ước của Spring về cấu hình, các lập trình viên cũng có thể lựa chọn các thành phần đã có sẵn hoặc thay đổi nó theo nhu cầu của mình.

Authentication có hai quy trình để xác định danh tính của user: nhận dạng (identification) và xác minh (verification). Sau đó một principal được tạo ra, principal có thể là người, hoặc thiết bị, hoặc hệ thống nào đó có thể thực hiện các thao tác trên ứng dụng.

Authorization xác định phạm vi quyền hạn của user đối với những tài nguyên bị hạn chế. Nó mang lại sự đảm bảo rằng user chỉ có thể truy cập vào các tài nguyện được hệ thống cho phép. Dù là trên ứng dụng web nào, điều này đều thực hiện thông qua bảo mật URL (Sau khi xem quá trình xây dựng một ứng dụng web bên dưới sẽ dễ hiểu hơn). Spring cung cấp một bộ lọc để đảm bảo tính bảo mật cho ứng dụng.

Các thành phần chính

SecurityContext là interface cốt lõi của Spring Security, lưu trữ tất cả các chi tiết liên quan đến bảo mật trong ứng dụng. Khi chúng ta kích hoạt Spring Security trong ứng dụng thì SecurityContext cũng sẽ được kích hoạt theo.

SecurityContextHolder lưu trữ security context hiện tại của ứng dụng, bao gồm thông tin chi tiết của principal đang tương tác. Đây cũng là thành phần trung gian để nhà phát triển truy cập vào SecurityContext thay vì truy cập trực tiếp.

UserDetails là một interface đại diện cho một principal nhưng theo một cách mở rộng và cụ thể hơn. UserDetails gồm các method sau:

- `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

Khi muốn mở rộng thêm các thông tin, phải sử dụng lớp `CustomUserDetails implements org.springframework.security.userdetails.UserDetails`

UserDetailsService là một interface có duy nhất một phương thức

```
UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;
```

Tham số truyền vào là username của người dùng. Hệ thống sẽ dùng nó để tìm kiếm trong cơ sở dữ liệu.

GrantedAuthority là một đối tượng chứa các quyền được cấp cho principal, được trả lại sau khi gọi hàm `getAuthorities()`.

Tính năng

- Linh hoạt và toàn diện trong cả Authentication và Authorization
- Có khả năng chống lại các kiểu tấn công session fixation, clickjacking, cross site request forgery,…
- Có thể tính hợp với Servlet API
- Tùy chọn sử dụng Spring Web MVC
- …

Sử dụng Spring Security ở đâu? Khi nào? Và tại sao chọn Spring Security?

Ở đâu? Như đã trình bày ở trên, Spring Security dùng cho việc authentication và authorization. Nên chúng ta có thể đặt nó ở phần chức năng đăng nhập của hệ thống hoặc ở những vị trí có hạn chế truy cập tài nguyên. Spring Security cũng có thể tích hợp với Lightweight Directory Access Protocol (LDAP).

Khi nào? Dựa vào chức năng mà ta có thể dùng Spring Security khi muốn ẩn đi những liên kết hoặc chức năng đặc biệt

Tại sao? Spring Security được tạo ra để lấp đầy lỗ trống khổng lồ của thư viện Java bên thứ ba. Những tiêu chí như Java Authentication và Authorization Service (JAAS) hay Java EE Security cũng có nhiều đơn vị hỗ trợ nhưng Spring Security cung cấp mọi thứ mà nhà phát triển cần để thực hiện bảo mật cho ứng dụng một cách ngắn gọn và hợp lý. Ngoài ra, Spring Security còn cho phép tích hợp theo hướng out-of-the-box với hệ thống authentication của nhiều doanh nghiệp. Do đó, nó có thể thích ứng với hầu hết các trường hợp mà không phải bỏ quá nhiều công sức.

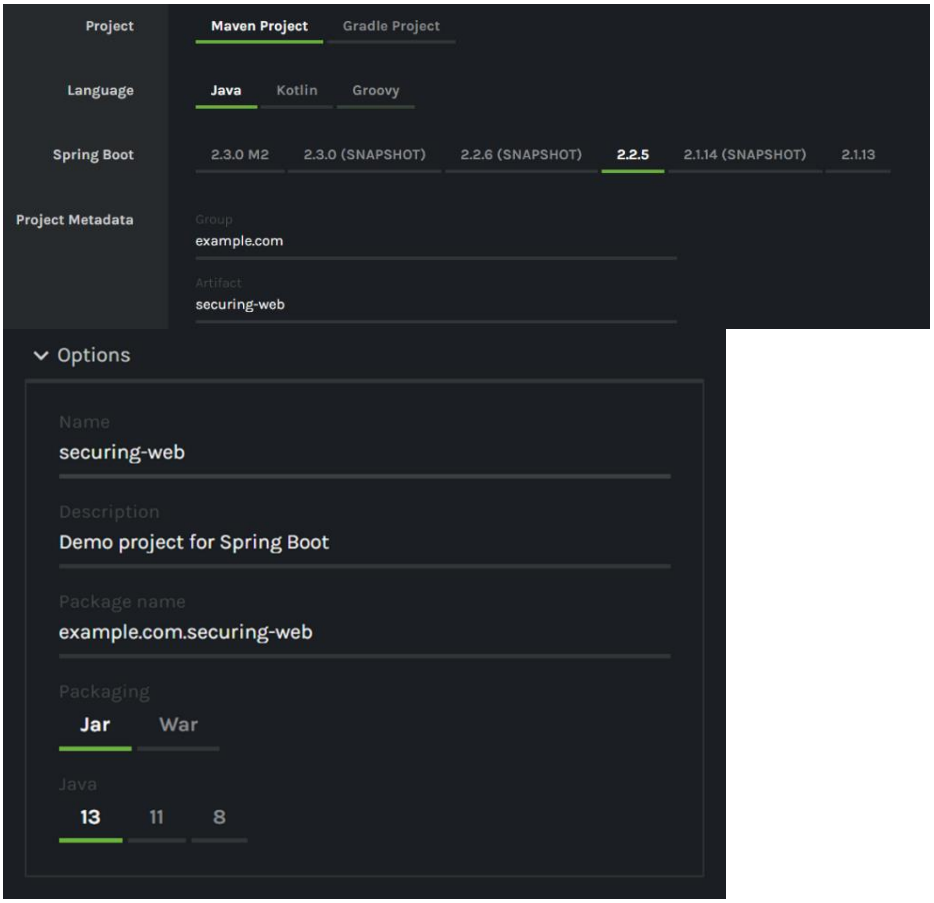
Xây dựng ứng dụng web đơn giản với Spring Security

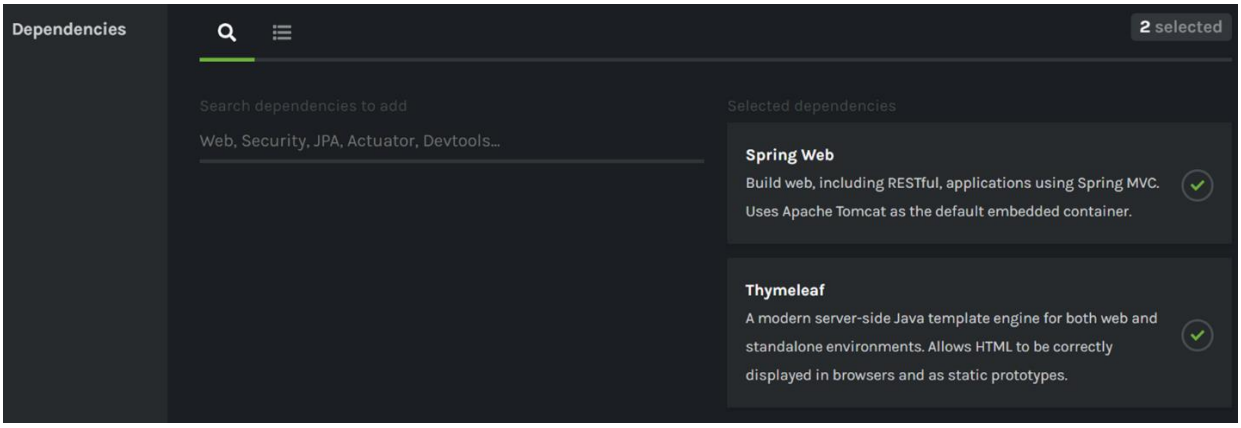
Chuẩn bị:

- JDK 1.8 hoặc các phiên bản mới hơn
- Công cụ Gradle 4+ hoặc Maven 3.2+

Bắt đầu project với Spring Initializr

Đầu tiên truy cập vào https://start.spring.io/ và lựa chọn các mục phù hợp với hệ thống. (Hướng dẫn này dựa vào hệ thống của người viết)





Khi đã hoàn tất các lựa chọn, ấn vào nút generate hoặc ctr + enter để tải project về máy. Kế đó hãy giải nén thư mục vừa tải về và thực hiện theo các bước bên dưới.

Tạo một trang web không bảo mật

Tạo hai tập tin HTML (home.html và hello.html) theo đường dẫn

- src/main/resources/templates/home.html
- src/main/resources/templates/hello.html

Bên trong home.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org"
xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
    <head>
        <title>Spring Security Example</title>
    </head>
    <body>
        <h1>Welcome!</h1>
        <p>Click <a th:href="@{/hello}">here</a> to see a greeting.</p>
    </body>
</html>
```

Bên trong hello.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org"
xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
    <head>
        <title>Hello World!</title>
    </head>
    <body>
        <h1>Hello world!</h1>
    </body>
</html>
```

Tạo thêm một file MvcConfig.java như bên dưới vào đường dẫn src/main/java/com/example/securingweb/MvcConfig.java

```
package com.example.securingweb;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
@Configuration public class MvcConfig implements WebMvcConfigurer {
    public void addViewControllers(ViewControllerRegistry registry) {
            registry.addViewController("/home").setViewName("home");
            registry.addViewController("/").setViewName("home");
            registry.addViewController("/hello").setViewName("hello");
            registry.addViewController("/login").setViewName("login");
    }
}
```

Lúc này chúng ta đã có thể mở trang web lên và chạy thử, tuy nhiên lúc này trang web vẫn chưa được bảo mật.

Cài đặt Spring Security

Thêm hai entry như bên dưới vào file pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>
```

Tạo thêm một file login.html và chỉnh sửa lại trang hello.html một chút.
Bên trong login.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org"
xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
   <head> <title>Spring Security Example </title> </head>
   <body>
      <div th:if="${param.error}"> Invalid username and password. </div>
      <div th:if="${param.logout}"> You have been logged out. </div>
      <form th:action="@{/login}" method="post">
      <div><label> User Name : <input type="text" name="username"/> </label></div>
      <div><label> Password: <input type="password" name="password"/> </label></div>
      <div><input type="submit" value="Sign In"/></div>
      </form>
   </body>
</html>
```

Sửa lại hello.html như sau

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org"
xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
  <head> <title>Hello World!</title> </head>
  <body>
    <h1 th:inline="text">Hello [[${#httpServletRequest.remoteUser}]]!</h1>
    <form th:action="@{/logout}" method="post">
      <input type="submit" value="Sign Out"/>
    </form>
  </body>
</html>
```

Thêm hai tập tin nữa (WebSecurityConfig.java và SecuringWebApplication.java) vào đường dẫn src/main/java/com/example/securingweb/

Bên trong WebSecurityConfig.java

```
package com.example.securingweb;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAda
pter;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;

@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
        @Override
        protected void configure(HttpSecurity http) throws Exception {
                http
                        .authorizeRequests()
                                .antMatchers("/", "/home").permitAll()
                                .anyRequest().authenticated()
                                .and()
                        .formLogin()
                                .loginPage("/login")
                                .permitAll()
                                .and()
                        .logout()
                                .permitAll();
```

```
	}

	@Bean
	@Override
	public UserDetailsService userDetailsService() {
		UserDetails user =
			User.withDefaultPasswordEncoder()
				.username("user")
				.password("password")
				.roles("USER")
				.build();

		return new InMemoryUserDetailsManager(user);
	}
}
```

Bên trong SecuringWebApplication.java

```
package com.example.securingweb;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SecuringWebApplication {

	public static void main(String[] args) throws Throwable {
		SpringApplication.run(SecuringWebApplication.class, args);
	}
}
```

Xây dựng web có bảo mật

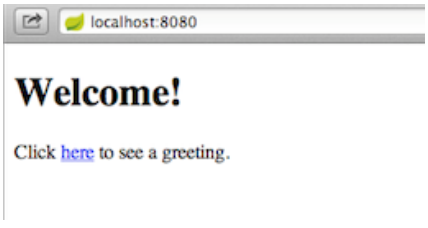
Tạo file jar để xây dựng hệ thống với dòng lệnh như sau:

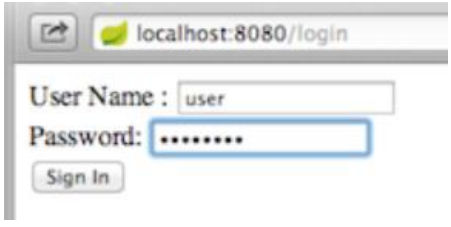
```
mvnw spring-boot:run
mvnw clean package
```

Sau đó chạy câu lệnh này để xây dựng bảo mật cho trang web

```
java –jar target/.jar
```

Cuối cùng mở trình duyệt với đường dẫn http://localhost:8080/ để xem thành quả





Ưu điểm và nhược điểm

Ưu điểm

- Dễ dàng phát triển các ứng dụng Spring với Java hoặc Groovy
- Giảm thời gian triển khai và tăng hiệu suất
- Có sẵn các đoạn code mẫu, chú thích và cấu hình XML
- Dễ dàng tích hợp Spring Boot Application (Spring JDBC, Spring ORM, Spring Data, Spring Security etc) với Spring Ecosystem
- Giảm công sức của nhà phát triển nhờ có "Opinionated Defaults Configuration"
- Cung cấp rất nhiều plugin và các máy chủ HTTP như Tomcat, Jetty,… để dễ dàng phát triển và kiểm tra ứng dụng.
- Nó cũng cung cấp công cụ giao diện dòng lệnh (Command Line Interface) để phát triển và kiểm tra Spring Boot Application (Java hay Groovy) từ cửa sổ command prompt một cách tiện lợi và nhanh chóng.

Nhược điểm: Sẽ rất khó khăn và tốn thời gian để chuyển đổi các dự án trên nên Spring Framework cũ sang các ứng dụng Spring Boot. Nó chỉ thích hợp trên các dự án Spring mới.

Tổng kết

Spring Security là một framework mạnh mẽ nhưng tương đối phức tạp, đòi hỏi người sử dụng phải dành nhiều thời gian nghiên cứu và tìm hiểu cách vận hành nó. Bài báo cáo này chỉ mới giới thiệu một nhỏ trong danh sách chức năng dày đặc của Spring Security.

Tài liệu tham khảo

[1] https://spring.io/projects/spring-security
[2] https://en.wikipedia.org/wiki/Spring_Security
[3] https://www.youtube.com/watch?v=sm-8qfMWEV8
[4] Mick Knutson, Robert Winch, Peter Mularien - Spring Security-Packt Publishing (2017)
[5] https://kipalog.com/posts/Huong-dan-lap-trinh-Spring-Security
[6] https://www.developer.com/java/ent/what-is-spring-security.html
[7] https://spring.io/guides/gs/securing-web/