# Tổng quan Spring Security

## Các khái niệm

1. Authorzation: Không phải ai cũng có thể vào trang web chúng ta và thêm sửa xóa tùy thích, chính vì thế cần có sự phân quyền truy cập ở đây. Để chỉ cho ai có quyền thêm thì thấy được trang thêm mới dữ liệu, ai không có quyền đó thì sẽ đi tới một trang lỗi. Ở đây người ta gọi nó là Authorzation (ủy quyền)
2. Authencation: Để biết ai với ai, ai vào hệ thống để làm việc thì cần phải có đăng nhập, đăng ký. Ở đây người ta gọi là Authencation (xác thực)

## Vì sao dùng Spring Security?

Với hai khái niệm trên người ta gọi chung là access-control (kiểm soát truy cập).

Vậy để kiểm soát truy cập có nhất thiết dùng spring security không?

Không! Ví dụ bạn có thể dùng Spring Integration để lắng nghe user đăng nhập. Vậy tại sao lại dùng Spring Security?

* Khả năng tích hợp: Spring Security tích hợp dễ dàng với các hệ thống doanh nghiệp đã có sẵn phân quyền. Ví dụ có thể dùng password đăng nhập máy tính để login vào trang quản trị.
* Dễ dàng sử dụng: Ví dụ khi đăng ký cần mã hóa mật khẩu. Nếu không dùng Spring Security thì các bạn gọi tớ một hàm tự xây dựng để mã hóa và đưa vào cơ sở dữ liệu. Với Spring Security thì không cần như vậy!

Đó là một vài ví dụ về tại sao lại sử dụng Spring Security. Bước tiếp chúng ta sẽ thực hiện Access-control một user với quyền hạn vào hệ thống mã chúng ta tự phân quyền, không phải hệ thống khác.

# Demo

## Step0: Add thư viện

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-security</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-thymeleaf</artifactId>  </dependency> |

## Step1: Tạo Các Entities

### User

|  |
| --- |
| **package** com.study.management.entities;  **import** java.util.Set;  **import** javax.persistence.CascadeType;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  **import** javax.persistence.JoinColumn;  **import** javax.persistence.JoinTable;  **import** javax.persistence.ManyToMany;  **import** javax.persistence.Table;  **import** org.hibernate.validator.constraints.~~Email~~;  **import** org.hibernate.validator.constraints.Length;  **import** org.hibernate.validator.constraints.~~NotEmpty~~;  **import** org.springframework.data.annotation.Transient;  @Entity  @Table(name = "user")  **public** **class** User {  @Id  @GeneratedValue(strategy = GenerationType.***AUTO***)  @Column(name = "user\_id")  **private** **int** id;    @Column(name = "email")  @~~Email~~(~~message~~ = "\*Please provide a valid Email")  @~~NotEmpty~~(~~message~~ = "\*Please provide an email")  **private** String email;    @Column(name = "password")  @Length(min = 5, message = "\*Your password must have at least 5 characters")  @~~NotEmpty~~(~~message~~ = "\*Please provide your password")  @Transient  **private** String password;    @Column(name = "name")  @~~NotEmpty~~(~~message~~ = "\*Please provide your name")  **private** String name;    @Column(name = "last\_name")  @~~NotEmpty~~(~~message~~ = "\*Please provide your last name")  **private** String lastName;    @Column(name = "active")  **private** **int** active;    @ManyToMany(cascade = CascadeType.***ALL***)  @JoinTable(name = "user\_role", joinColumns = @JoinColumn(name = "user\_id"), inverseJoinColumns = @JoinColumn(name = "role\_id"))  **private** Set<Role> roles;  **public** **int** getId() {  **return** id;  }  **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** String getPassword() {  **return** password;  }  **public** **void** setPassword(String password) {  **this**.password = password;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** String getLastName() {  **return** lastName;  }  **public** **void** setLastName(String lastName) {  **this**.lastName = lastName;  }  **public** String getEmail() {  **return** email;  }  **public** **void** setEmail(String email) {  **this**.email = email;  }  **public** **int** getActive() {  **return** active;  }  **public** **void** setActive(**int** active) {  **this**.active = active;  }  **public** Set<Role> getRoles() {  **return** roles;  }  **public** **void** setRoles(Set<Role> roles) {  **this**.roles = roles;  }  } |

### Role

|  |
| --- |
| **package** com.study.management.entities;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  **import** javax.persistence.Table;  @Entity  @Table(name = "role")  **public** **class** Role {  @Id  @GeneratedValue(strategy = GenerationType.***AUTO***)  @Column(name = "role\_id")  **private** **int** id;  @Column(name = "role")  **private** String role;  **public** **int** getId() {  **return** id;  }  **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** String getRole() {  **return** role;  }  **public** **void** setRole(String role) {  **this**.role = role;  }  } |

## Step 2: Cấu hình kết nối và cho phép tạo bảng tự động

|  |
| --- |
| # data source  spring.datasource.url=jdbc:mysql://localhost:3306/surveymanagement  spring.datasource.username=root  spring.datasource.password=  spring.jpa.hibernate.ddl-auto = update  spring.jpa.show-sql = true  spring.jpa.hibernate.naming-strategy = org.hibernate.cfg.ImprovedNamingStrategy  spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect  # ==============================================================  # = Spring Security / Queries for AuthenticationManagerBuilder  # ==============================================================  spring.queries.users-query=select email, password, active from user where email=?  spring.queries.roles-query=select u.email, r.role from user u inner join user\_role ur on(u.user\_id=ur.user\_id) inner join role r on(ur.role\_id=r.role\_id) where u.email=? |

1. spring.jpa.hibernate.ddl-auto = update cho phép tạo bảng
2. spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect hỗ trợ cho phép tạo SQL chính xác hơn với hệ quản trị CSDL
3. spring.jpa.hibernate.naming-strategy = org.hibernate.cfg.ImprovedNamingStrategy hỗ trợ việc tạo tên bảng và tên cột

Sau khi chạy ứng dụng nó sẽ tự động tao DB, khi đó chúng ra có thể chèn dữ liệu trong CSDL.

## Step 3: Tạo Repository

### Tạo cho User

|  |
| --- |
| **package** com.study.management.repository;  **import** org.springframework.data.jpa.repository.JpaRepository;  **import** org.springframework.stereotype.Repository;  **import** com.study.management.entities.User;  @Repository("userRepository")  **public** **interface** UserRepository **extends** JpaRepository<User, Long> {  User findByEmail(String email);  } |

### Tạo cho Role

|  |
| --- |
| **package** com.study.management.repository;  **import** org.springframework.data.jpa.repository.JpaRepository;  **import** org.springframework.stereotype.Repository;  **import** com.study.management.entities.Role;  @Repository("roleRepository")  **public** **interface** RoleRepository **extends** JpaRepository<Role, Integer> {    Role findByRole(String role);  } |

## Step 4: Tạo Service Layer

### Tạo Interface

|  |
| --- |
| **package** com.study.management.services;  **import** com.study.management.entities.User;  **public** **interface** UserService {  **public** User findUserByEmail(String email);  **public** **void** saveUser(User user);  } |

### Tạo Implementation

|  |
| --- |
| **package** com.study.management.services;  **import** java.util.Arrays;  **import** java.util.HashSet;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  **import** org.springframework.stereotype.Service;  **import** com.study.management.entities.Role;  **import** com.study.management.entities.User;  **import** com.study.management.repository.RoleRepository;  **import** com.study.management.repository.UserRepository;  @Service("userService")  **public** **class** UserServiceImpl **implements** UserService {  @Autowired  **private** UserRepository userRepository;  @Autowired  **private** RoleRepository roleRepository;  @Autowired  **private** BCryptPasswordEncoder bCryptPasswordEncoder;  @Override  **public** User findUserByEmail(String email) {  **return** userRepository.findByEmail(email);  }  @Override  **public** **void** saveUser(User user) {  user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));  user.setActive(1);  Role userRole = roleRepository.findByRole("ADMIN");  user.setRoles(**new** HashSet<Role>(Arrays.*asList*(userRole)));  userRepository.save(user);  }  } |

## Step 5: Tạo Authencation controller

|  |
| --- |
| **package** com.study.management.controller.web;  **import** javax.validation.Valid;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.security.core.Authentication;  **import** org.springframework.security.core.context.SecurityContextHolder;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.validation.BindingResult;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RequestMethod;  **import** org.springframework.web.servlet.ModelAndView;  **import** com.study.management.entities.User;  **import** com.study.management.services.UserService;  @Controller  **public** **class** LoginController {  @Autowired  **private** UserService userService;  @RequestMapping(value = { "/", "/login" }, method = RequestMethod.***GET***)  **public** ModelAndView login() {  ModelAndView modelAndView = **new** ModelAndView();  modelAndView.setViewName("login");  **return** modelAndView;  }  @RequestMapping(value = "/registration", method = RequestMethod.***GET***)  **public** ModelAndView registration() {  ModelAndView modelAndView = **new** ModelAndView();  User user = **new** User();  modelAndView.addObject("user", user);  modelAndView.setViewName("registration");  **return** modelAndView;  }  @RequestMapping(value = "/registration", method = RequestMethod.***POST***)  **public** ModelAndView createNewUser(@Valid User user, BindingResult bindingResult) {  ModelAndView modelAndView = **new** ModelAndView();  User userExists = userService.findUserByEmail(user.getEmail());  **if** (userExists != **null**) {  bindingResult.rejectValue("email", "error.user",  "There is already a user registered with the email provided");  }  **if** (bindingResult.hasErrors()) {  modelAndView.setViewName("registration");  } **else** {  userService.saveUser(user);  modelAndView.addObject("successMessage", "User has been registered successfully");  modelAndView.addObject("user", **new** User());  modelAndView.setViewName("registration");  }  **return** modelAndView;  }  @RequestMapping(value = "/admin/home", method = RequestMethod.***GET***)  **public** ModelAndView home() {  ModelAndView modelAndView = **new** ModelAndView();  Authentication auth = SecurityContextHolder.*getContext*().getAuthentication();  User user = userService.findUserByEmail(auth.getName());  modelAndView.addObject("userName",  "Welcome " + user.getName() + " " + user.getLastName() + " (" + user.getEmail() + ")");  modelAndView.addObject("adminMessage", "Content Available Only for Users with Admin Role");  modelAndView.setViewName("admin/home");  **return** modelAndView;  }  } |

## Step 6: Tạo Authorzation

|  |
| --- |
| **package** com.study.management.config;  **import** javax.sql.DataSource;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.beans.factory.annotation.Value;  **import** org.springframework.context.annotation.Configuration;  **import** org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  **import** org.springframework.security.config.annotation.web.builders.HttpSecurity;  **import** org.springframework.security.config.annotation.web.builders.WebSecurity;  **import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  **import** org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  **import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  **import** org.springframework.security.web.util.matcher.AntPathRequestMatcher;  @Configuration  @EnableWebSecurity  **public** **class** SecurityConfiguration **extends** WebSecurityConfigurerAdapter {  @Autowired  **private** BCryptPasswordEncoder bCryptPasswordEncoder;  @Autowired  **private** DataSource dataSource;  @Value("${spring.queries.users-query}")  **private** String usersQuery;  @Value("${spring.queries.roles-query}")  **private** String rolesQuery;  @Override  **protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception {  auth.jdbcAuthentication().usersByUsernameQuery(usersQuery).authoritiesByUsernameQuery(rolesQuery)  .dataSource(dataSource).passwordEncoder(bCryptPasswordEncoder);  }  @Override  **protected** **void** configure(HttpSecurity http) **throws** Exception {  http.authorizeRequests().antMatchers("/").permitAll().antMatchers("/login").permitAll()  .antMatchers("/registration").permitAll().antMatchers("/admin/\*\*").hasAuthority("ADMIN").anyRequest()  .authenticated().and().csrf().disable().formLogin().loginPage("/login").failureUrl("/login?error=true")  .defaultSuccessUrl("/admin/home").usernameParameter("email").passwordParameter("password").and()  .logout().logoutRequestMatcher(**new** AntPathRequestMatcher("/logout")).logoutSuccessUrl("/").and()  .exceptionHandling().accessDeniedPage("/access-denied");  }  @Override  **public** **void** configure(WebSecurity web) **throws** Exception {  web.ignoring().antMatchers("/resources/\*\*", "/static/\*\*", "/css/\*\*", "/js/\*\*", "/images/\*\*");  }  } |

## Step7: Tạo crypt Bean Mã hóa password

|  |
| --- |
| **package** com.study.management.config;  **import** org.springframework.context.annotation.Bean;  **import** org.springframework.context.annotation.Configuration;  **import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  **import** org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;  **import** org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  @Configuration  **public** **class** WebMvcConfig **implements** WebMvcConfigurer {  //public class WebMvcConfig extends WebMvcConfigurerAdapter {  @Bean  **public** BCryptPasswordEncoder passwordEncoder() {  BCryptPasswordEncoder bCryptPasswordEncoder = **new** BCryptPasswordEncoder();  **return** bCryptPasswordEncoder;  }  @Override  **public** **void** addResourceHandlers(ResourceHandlerRegistry registry) {  registry.addResourceHandler("/webjars/\*\*", "/images/\*\*", "/css/\*\*", "/js/\*\*").addResourceLocations(  "classpath:/META-INF/resources/webjars/", "classpath:/static/images/", "classpath:/static/css/",  "classpath:/static/js/");  }  } |

## Step8: Tạo View

### login.html

|  |
| --- |
| <!DOCTYPE html>  <html xmlns=*"http://www.w3.org/1999/xhtml"*  xmlns:th=*"http://www.thymeleaf.org"*>  <head>  <title>Spring Security Tutorial</title>    </head>  <body>  <form th:action=*"@{/registration}"* method=*"get"*>  <button class=*"btn btn-md btn-warning btn-block"* type=*"Submit"*>Go To Registration Page</button>  </form>    <div class=*"container"*>  <img th:src=*"@{/images/login.png}"* class=*"img-responsive center-block"* width=*"300"* height=*"300"* alt=*"Logo"* />  <form th:action=*"@{/login}"* method=*"POST"* class=*"form-signin"*>  <h3 class=*"form-signin-heading"* th:text=*"Welcome"*></h3>  <br/>    <input type=*"text"* id=*"email"* name=*"email"* th:placeholder=*"Email"*  class=*"form-control"* /> <br/>  <input type=*"password"* th:placeholder=*"Password"*  id=*"password"* name=*"password"* class=*"form-control"* /> <br />    <div align=*"center"* th:if=*"${param.error}"*>  <p style="font-size: *20*; color: *#FF1C19*;">Email or Password invalid, please verify</p>  </div>  <button class=*"btn btn-lg btn-primary btn-block"* name=*"Submit"* value=*"Login"* type=*"Submit"* th:text=*"Login"*></button>  </form>  </div>  </body>  </html> |

### registration.html

|  |
| --- |
| <!DOCTYPE html>  <html lang=*"en"* xmlns=*"http://www.w3.org/1999/xhtml"*  xmlns:th=*"http://www.thymeleaf.org"*>  <head>  <title>Registration Form</title>  </head>  <body>  <form th:action=*"@{/}"* method=*"get"*>  <button class=*"btn btn-md btn-warning btn-block"* type=*"Submit"*>Go  To Login Page</button>  </form>  <div class=*"container"*>  <div class=*"row"*>  <div class=*"col-md-6 col-md-offset-3"*>  <form autocomplete=*"off"* action=*"#"* th:action=*"@{/registration}"*  th:object=*"${user}"* method=*"post"* class=*"form-horizontal"*  role=*"form"*>  <h2>Registration Form</h2>  <div class=*"form-group"*>  <div class=*"col-sm-9"*>  <label th:if=*"${#fields.hasErrors('name')}"* th:errors=*"\*{name}"*  class=*"validation-message"*></label> <input type=*"text"*  th:field=*"\*{name}"* placeholder=*"Name"* class=*"form-control"* />  </div>  </div>  <div class=*"form-group"*>  <div class=*"col-sm-9"*>  <label th:if=*"${#fields.hasErrors('lastName')}"*  th:errors=*"\*{lastName}"* class=*"validation-message"*></label> <input  type=*"text"* th:field=*"\*{lastName}"* placeholder=*"Last Name"*  class=*"form-control"* />  </div>  </div>  <div class=*"form-group"*>  <div class=*"col-sm-9"*>  <input type=*"text"* th:field=*"\*{email}"* placeholder=*"Email"*  class=*"form-control"* /> <label  th:if=*"${#fields.hasErrors('email')}"* th:errors=*"\*{email}"*  class=*"validation-message"*></label>  </div>  </div>  <div class=*"form-group"*>  <div class=*"col-sm-9"*>  <input type=*"password"* th:field=*"\*{password}"*  placeholder=*"Password"* class=*"form-control"* /> <label  th:if=*"${#fields.hasErrors('password')}"* th:errors=*"\*{password}"*  class=*"validation-message"*></label>  </div>  </div>  <div class=*"form-group"*>  <div class=*"col-sm-9"*>  <button type=*"submit"* class=*"btn btn-primary btn-block"*>Register  User</button>  </div>  </div>  <span th:utext=*"${successMessage}"*></span>  </form>  </div>  </div>  </div>  </body>  </html> |