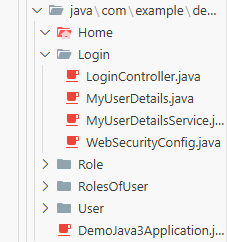
<https://www.codejava.net/frameworks/spring-boot/spring-boot-security-role-based-authorization-tutorial>



|  |  |
| --- | --- |
| MyUserDetails | |
| public class MyUserDetails      implements UserDetails  {      private Users user;      public MyUserDetails(Users user) {          this.user = user;      }      @Override      public Collection<? extends GrantedAuthority> getAuthorities() {            Set<RolesOfUser> listRou1 = this.user.getRolesOfUser();          List<SimpleGrantedAuthority> dsAuthorities = new ArrayList<>();          for(RolesOfUser rou1 : listRou1){              String nameRole1 = rou1.getRole().getName();              SimpleGrantedAuthority sgAuthority1 = new SimpleGrantedAuthority(nameRole1);                dsAuthorities.add(sgAuthority1);          }          return dsAuthorities;      }      @Override      public String getPassword() {          return user.getPassword();      }      @Override      public String getUsername() {          return user.getUsername();      }      @Override      public boolean isAccountNonExpired() {          return true;      }      @Override      public boolean isAccountNonLocked() {          return true;      }      @Override      public boolean isCredentialsNonExpired() {          return true;      }      @Override      public boolean isEnabled() {          return this.user.getEnabled();      }    } | |
| MyUserDetailsService |
| public class MyUserDetailsService      implements UserDetailsService  {      @Autowired      private UsersRepo usersRepo;      @Override      public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {          Users user1 = this.usersRepo.findByUsername(username).get(0);            if(user1 == null){              throw new UsernameNotFoundException("không tìm được user");          }          return new MyUserDetails(user1);      }    } |

|  |
| --- |
| WebSecurityConfig |
| @Configuration  public class WebSecurityConfig {        @Bean      UserDetailsService userDetailsService(){          return new MyUserDetailsService();      }      @Bean      BCryptPasswordEncoder passwordEncoder(){          return new BCryptPasswordEncoder();      }      @Bean      DaoAuthenticationProvider authenticationProvider(){          DaoAuthenticationProvider authProvider              = new DaoAuthenticationProvider(this.userDetailsService() );          authProvider.setPasswordEncoder(this.passwordEncoder());          return authProvider;      }      @Bean      SecurityFilterChain configure(HttpSecurity http) throws Exception {          http              .authorizeHttpRequests(                  auth -> auth                      .requestMatchers("/register").permitAll()                      .requestMatchers("/").hasAnyAuthority("CLIENT", "STAFF", "ADMIN")                      .requestMatchers("/create/\*\*", "/edit/\*\*").hasAnyAuthority("STAFF", "ADMIN")                      .requestMatchers("/delete/\*\*").hasAnyAuthority("ADMIN")                      .anyRequest().authenticated()              )              .formLogin(login-> login.permitAll())              .logout(logout-> logout.permitAll())              .exceptionHandling(eh-> eh.accessDeniedPage("/403"))              ;          return http.build();      }  } |

|  |
| --- |
| LoginController |
| @Controller  public class LoginController {      @Autowired      private UsersService usersService;      @Autowired      private RolesOfUserService rouService;      @Autowired      private RoleRepo roleRepo;      @GetMapping("/register")      public String getRegister      (          Model model      )      {          Users user1 = new Users();          model.addAttribute("users", user1);          return "Login/register";      }      @PostMapping("/register")      public String postRegister      (          @ModelAttribute("users") Users users      )      {          System.out.println("🟥"+ users.toString());            BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();          String encodedPassword = passwordEncoder.encode(users.getPassword());          users.setPassword(encodedPassword);          users.setEnabled(true);          // Bước 1 : save user          Users user1 = this.usersService.create(users);          // bước 2 : set role của user là client          Role clientRole = this.roleRepo.findByName("CLIENT").get(0);          RolesOfUserKey rouKey = new RolesOfUserKey(clientRole.getMaRole(), user1.getMaUser());          RolesOfUser rou = new RolesOfUser(rouKey, clientRole, user1);          rou=this.rouService.create(rou); // row 62 //          // user1.getRolesOfUser().add(rou);          // không cần save user vì       dòng thứ 62 đã      save user cùng với rou          // bước 2 : save user          // user1 = this.usersService.create(user1);          System.out.println("🟩"+ user1.toString());          return "redirect:/login";      }  } |