|  |
| --- |
| Register.html |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>register</title>  </head>  <body>      <form th:action="@{/register}" method="post" th:object="${user1}">          <div>              <label>username</label>              <input type="text" th:field="\*{username}" placeholder="username" required="true" />          </div>          <div>              <label>password</label>              <input type="password" th:field="\*{password}" placeholder="password" required="true"/>          </div>          <div>              <label>email</label>              <input type="email" th:field="\*{email}" placeholder="email" required="true"/>          </div>          <div>              <label>sodienthoai</label>              <input type="tel" th:field="\*{sodienthoai}" placeholder="sodienthoai" required="true"/>          </div>          <div>              <button type="submit">register</button>          </div>      </form>  </body>  </html> |

|  |
| --- |
| MyUserDetails |
| public class MyUserDetails      implements UserDetails  {      private User user;      RolesOfUserService rouService;      RoleService roleService;      public MyUserDetails(User user) {          this.user = user;      }        public MyUserDetails(User user, RolesOfUserService rouService, RoleService roleService) {          this.user = user;          this.rouService = rouService;          this.roleService = roleService;      }      @Override      public Collection<? extends GrantedAuthority> getAuthorities() {          List<RolesOfUser> listRou1 = this.rouService.findByIdUser(this.user.getId());            List<SimpleGrantedAuthority> dsAuthorities = new ArrayList<>();          for(RolesOfUser rou1 : listRou1){              Role role1 = this.roleService.findById(rou1.getRouKey().getRoleId());              String nameOfRole1 = role1.getName();              SimpleGrantedAuthority sgAuth1 = new SimpleGrantedAuthority(nameOfRole1);              dsAuthorities.add(sgAuth1);          }          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 user.getEnabled();      }    } |

|  |
| --- |
| MyUserDetailsService |
| @Service  public class MyUserDetailsService      implements UserDetailsService  {      @Autowired      private UserService userService;      @Autowired      RolesOfUserService rouService;      @Autowired      RoleService roleService;      @Override      public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {          User user1 = this.userService.findByUsername(username);            if(user1==null){              throw new UsernameNotFoundException("không tìm được user");          }          return new MyUserDetails(user1, rouService, roleService);      }  } |

|  |
| --- |
| 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").anonymous()                  .requestMatchers("/").authenticated()                  .requestMatchers("/create/\*\*", "/edit/\*\*").hasAnyAuthority("NHANVIEN", "ADMIN")                  .requestMatchers("/delete/\*\*").hasAnyAuthority("ADMIN")                  .anyRequest().authenticated()          );            http.formLogin(              login-> login.permitAll()          );            http.logout(              logout-> logout.permitAll()          );          http.exceptionHandling(              eh-> eh.accessDeniedPage("/403")          );            return http.build();      }  } |

|  |
| --- |
| LoginController |
| @Controller  public class LoginController {      private BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();      @Autowired      private UserService usersService;      @Autowired      private RoleService roleService;      @Autowired      private RolesOfUserService rouService;      @Autowired      private KhachHangService khachHangService;      @GetMapping("/register")      public String getAtRegister(Model model){          User user = new User();          model.addAttribute("user1", user);            return "Login/register";      }      @PostMapping("/register")      public String postAtRegister      (          @ModelAttribute("user1") User user      )      {          System.out.println(user.toString());          // ma hoa mat khau          String encodedPassword = this.passwordEncoder.encode(user.getPassword());          user.setPassword(encodedPassword);          user.setEnabled(true);          // tạo user          user= this.usersService.create(user);          // tìm id của role client          Role clientRole = this.roleService.findByName("CLIENT");          // save role client của user          RolesOfUserKey rouKey = new RolesOfUserKey(user.getId(), clientRole.getId());          RolesOfUser rou = new RolesOfUser(rouKey);          rou = this.rouService.create(rou);          // save khách hàng của user          KhachHang kh = new KhachHang(user.getId());          kh = this.khachHangService.create(kh);          return "redirect:/login";      }  } |

|  |
| --- |
|  |