

# BÀI 6 XÂY DỰNG CHỨC NĂNG TẠO TÀI KHOẢN VÀ ĐĂNG NHẬP CHO NGƯỜI DÙNG

**Bài này giúp người học nắm được các nội dung sau:**

Thực hiện được chức năng đăng nhập và tạo tài khoản người dùng. Mục tiêu phát triển một chức năng đăng ký, an toàn và linh hoạt cho người dùng. Chức năng này sẽ cho phép người dùng tạo tài khoản mới, đặt mật khẩu và xác nhận mật khẩu, nhập thông tin cá nhân, và xác thực thông tin của họ. Điều này sẽ giúp chúng ta đảm bảo tính toàn vẹn và bảo mật thông tin cá nhân của người dùng.

## **Mô tả chức năng:**

Thêm chức năng đăng ký vào tạo tài khoản người dùng trên Spring Boot là một mục tiêu quan trọng để đáp ứng nhu cầu của người dùng và cải thiện trải nghiệm của họ khi sử dụng ứng dụng. Với chức năng đăng ký, người dùng có thể tạo tài khoản cá nhân và lưu trữ thông tin của mình trên hệ thống của chúng ta.

Thêm vào đó, việc thêm chức năng đăng ký sẽ giúp chúng ta thu thập thông tin từ người dùng. Chúng ta có thể sử dụng thông tin đăng ký của người dùng để tùy chỉnh trải nghiệm của họ, cung cấp các tính năng mới, và nâng cao chất lượng dịch vụ của chúng ta.

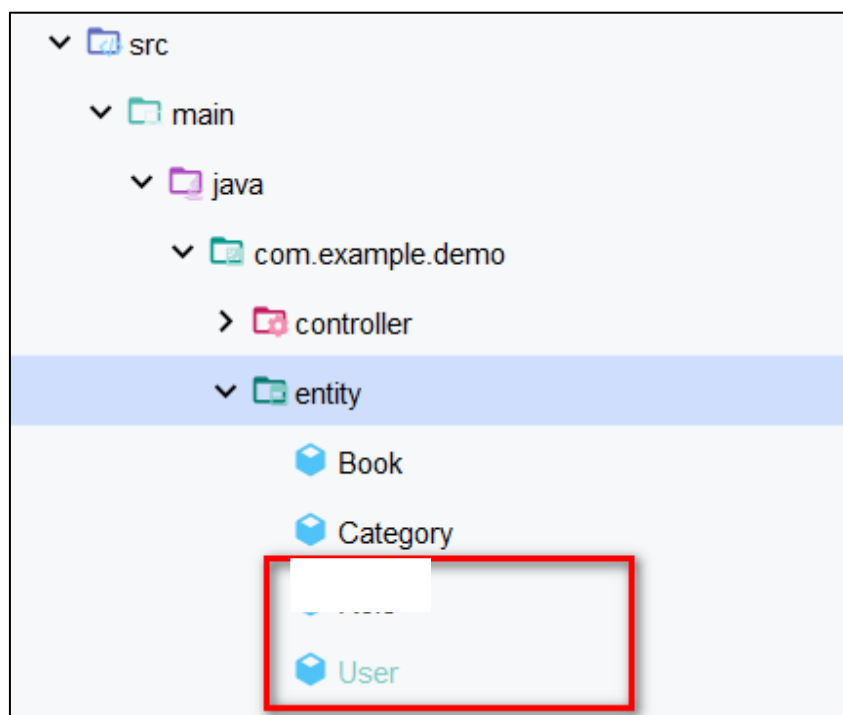
Vì vậy, mục tiêu của chúng ta khi thêm chức năng đăng ký vào tạo tài khoản người dùng trên Spring Boot là tạo ra một trải nghiệm dễ sử dụng và bảo mật cho người dùng, thu thập thông tin và cải thiện tính năng của ứng dụng của chúng ta, và đáp ứng nhu cầu của người dùng.

## 6.1 Xây dựng thêm table mới là User

Tạo class **User.java** trong thư mục **Entity** được đặt tại đường dẫn sau đây **src/main/java/com.example.demo/entity**

**User.java** lớp Java để xác định thông tin người dùng và quyền truy cập.

Lớp **User.java** được sử dụng để đại diện cho thông tin người dùng, bao gồm các thuộc tính như tên đăng nhập, mật khẩu, địa chỉ email và các thông tin cá nhân khác. Nó cũng có thể chứa các phương thức để xử lý các hoạt động liên quan đến người dùng như đăng ký, đăng nhập, cập nhật thông tin người dùng và quên mật khẩu.



Hình 6.1. Tạo 2 class User trong thư mục entity

Thêm nội dung code vào file **User.java** như bên dưới

```
User.java x
package com.example.demo.entity;

import com.example.demo.validator.annotation.ValidUsername;
import jakarta.persistence.*;
import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.Size;
import lombok.Data;

import java.util.ArrayList;
import java.util.List;

// Nguyễn Xuân Nhân *
@Data
@Entity
@Table(name = "user")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "username", length = 50, nullable = false, unique = true)
    @NotBlank(message = "Username is required")
    @Size(max = 50, message = "Username must be less than 50 characters")
    @ValidUsername
    private String username;

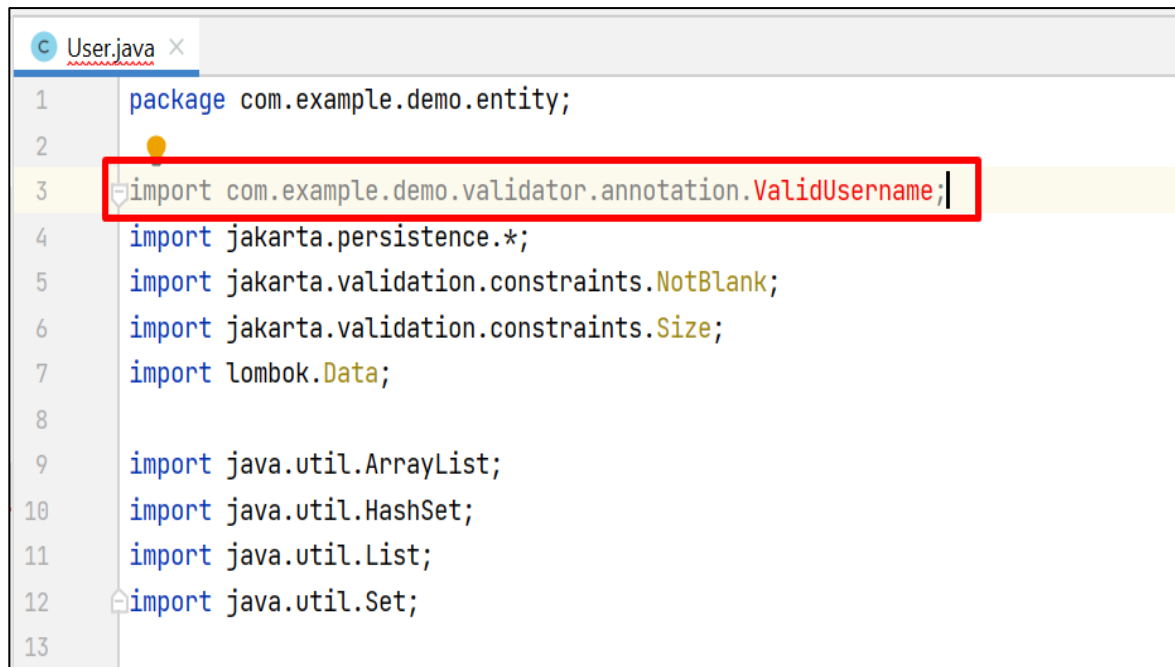
    @Column(name = "password", length = 250, nullable = false)
    @NotBlank(message = "Password is required")
    private String password;

    @Column(name = "email", length = 50)
    @Size(max = 50, message = "Email must be less than 50 characters")
    @Email(message = "Email should be valid")
    private String email;

    @Column(name = "name", length = 50, nullable = false)
    @Size(max = 50, message = "Your name must be less than 50 characters")
    @NotBlank(message = "Your name is required")
    private String name;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
    private List<Book> books = new ArrayList<>();
}
```

Hình 6.2. Thêm nội dung vào file User.java



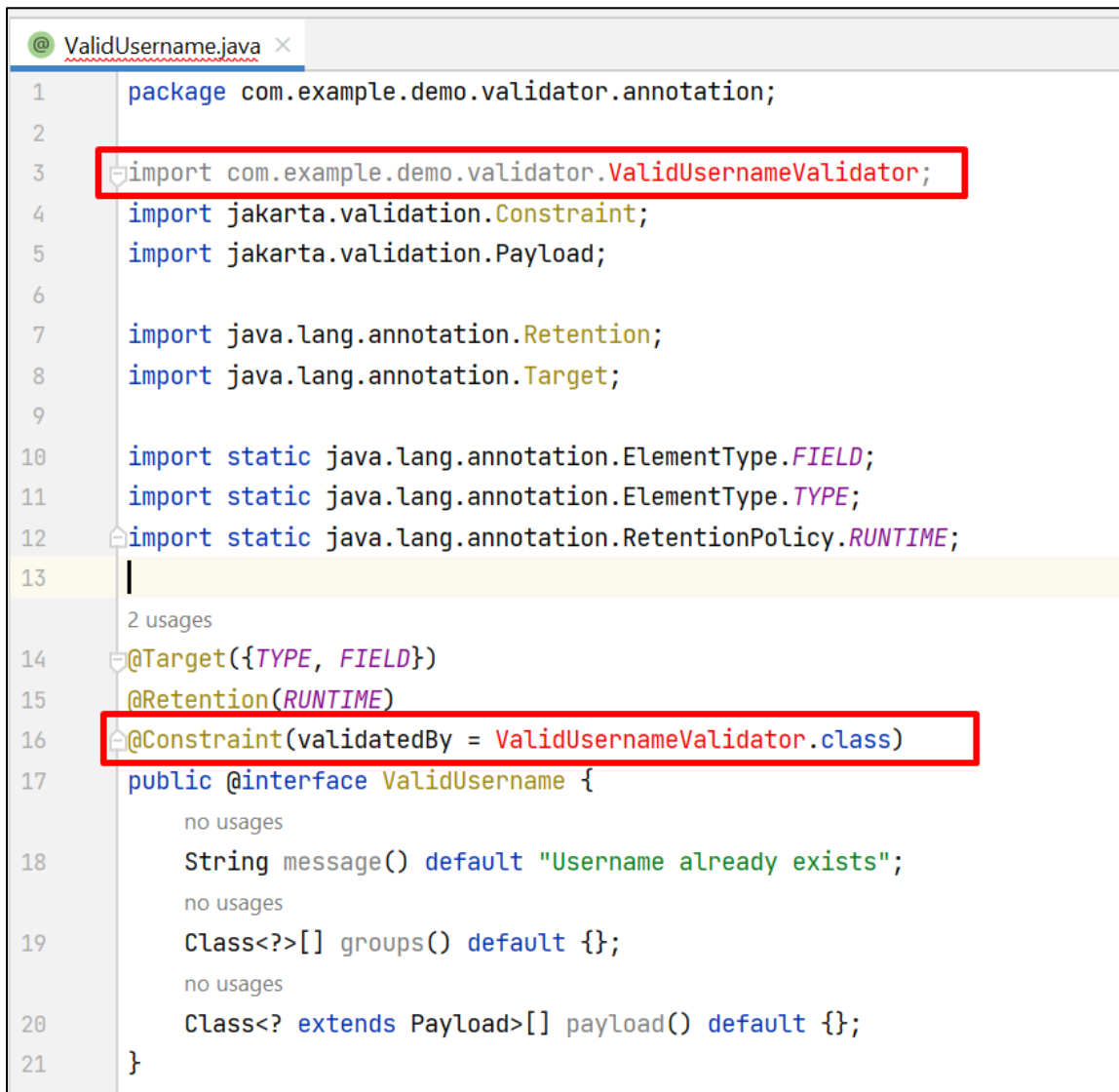
```
1 package com.example.demo.entity;  
2  
3 import com.example.demo.validator.annotation.ValidUsername;  
4 import jakarta.persistence.*;  
5 import jakarta.validation.constraints.NotBlank;  
6 import jakarta.validation.constraints.Size;  
7 import lombok.Data;  
8  
9 import java.util.ArrayList;  
10 import java.util.HashSet;  
11 import java.util.List;  
12 import java.util.Set;  
13
```

Hình 6.3. Thông báo lỗi chưa định nghĩa lớp ValidUsername

**ValidUsername** dùng để kiểm tra tính hợp lệ của tên người dùng trong ứng dụng. Có thể được định nghĩa là một chuỗi ký tự chỉ chứa các ký tự chữ cái, số và dấu gạch dưới, không được bắt đầu bằng số hoặc dấu gạch dưới, và không quá dài hoặc quá ngắn.

Tiến hành tạo thêm file tên là **ValidUsername.java** trong thư mục annotation theo đường dẫn sau:

**src/main/java/com/example/demo/validator/annotation**



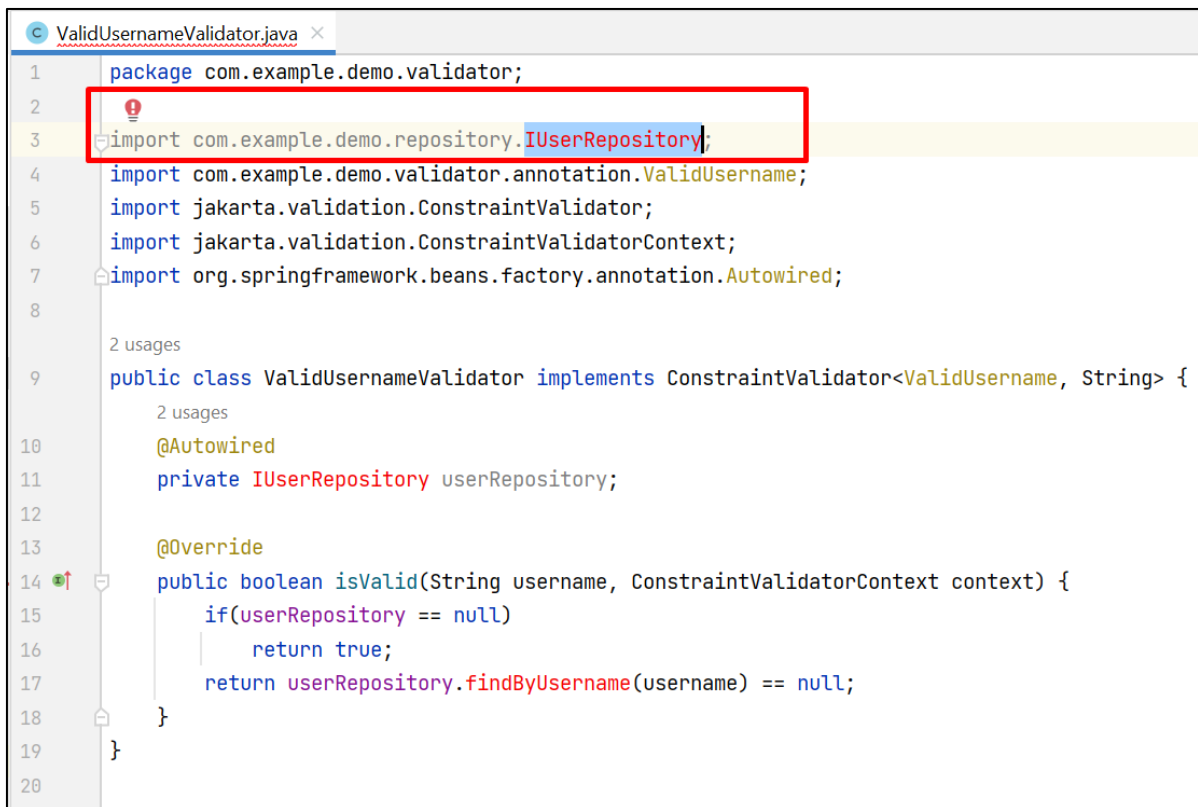
```
1 package com.example.demo.validator.annotation;
2
3 import com.example.demo.validator.ValidUsernameValidator;
4 import jakarta.validation.Constraint;
5 import jakarta.validation.Payload;
6
7 import java.lang.annotation.Retention;
8 import java.lang.annotation.Target;
9
10 import static java.lang.annotation.ElementType.FIELD;
11 import static java.lang.annotation.ElementType.TYPE;
12 import static java.lang.annotation.RetentionPolicy.RUNTIME;
13
14 @Target({TYPE, FIELD})
15 @Retention(RUNTIME)
16 @Constraint(validatedBy = ValidUsernameValidator.class)
17 public @interface ValidUsername {
18     String message() default "Username already exists";
19     Class<?>[] groups() default {};
20     Class<? extends Payload>[] payload() default {};
21 }
```

Hình 6.4. Thêm nội dung cho class ValidUsername.java

Trong đoạn code trong file **ValidUsername.java** có thông báo thiếu class **ValidUsernameValidator**.

Nên chúng ta tiến hành bổ sung thêm class **ValidUsernameValidator.java** vào thư mục **validator** theo đường dẫn sau:

**src/main/java/com/example/demo/validator**



Hình 6.5. Tạo thêm class ValidUsernameValidator.java

**IUserRepository** là một interface trong Java, thường được sử dụng trong kiến trúc phần mềm định hướng đối tượng (OOP) để truy xuất và lưu trữ dữ liệu người dùng.

Tại 1 file tên **IUserRepository.java** trong thư mục theo đường dẫn **src/main/java/com/example/demo/repository**

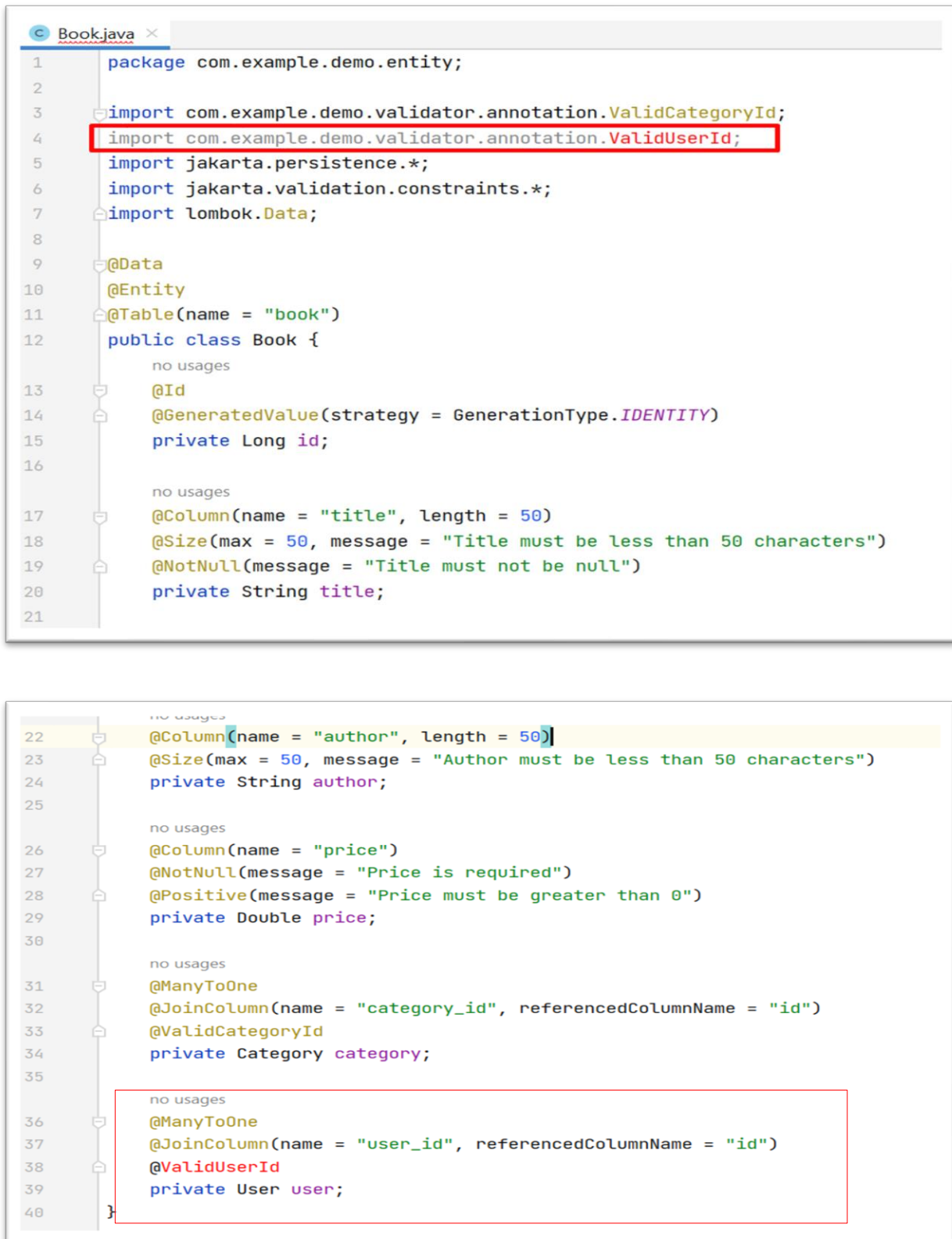


```
1 package com.example.demo.repository;
2
3 import com.example.demo.entity.User;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.data.jpa.repository.Query;
6 import org.springframework.stereotype.Repository;
7
8 @Repository
9 public interface IUserRepository extends JpaRepository<User, Long> {
10     @Query("SELECT u FROM User u WHERE u.username = ?1")
11     User findByUsername(String username);
12 }
```

Hình 6.6. Tạo 1 file tên *IuserRepository.java*

Tiến hành chỉnh sửa tiếp tại file Book trong thư mục **entity** theo thư mục **src/main/java/com/example/demo/entity**.

Cụ thể, thêm thuộc tính user vào lớp `com.example.demo.entity.Book` bởi vì trong lớp `com.example.demo.entity.User`, một thuộc tính `books` đã được định nghĩa với **mappedBy** là `user`. Điều này cho biết rằng, quan hệ giữa hai lớp này là quan hệ một-nhiều, trong đó một đối tượng `User` có thể có nhiều đối tượng `Book` tương ứng.



Hình 6.7. Thêm thuộc tính user vào Book



Tiếp theo ta cần thêm 1 file tên là **ValidUserId.java** vào thư mục annotation theo đường dẫn sau đây:

**src/main/java/com/example/demo/validator/annotation**

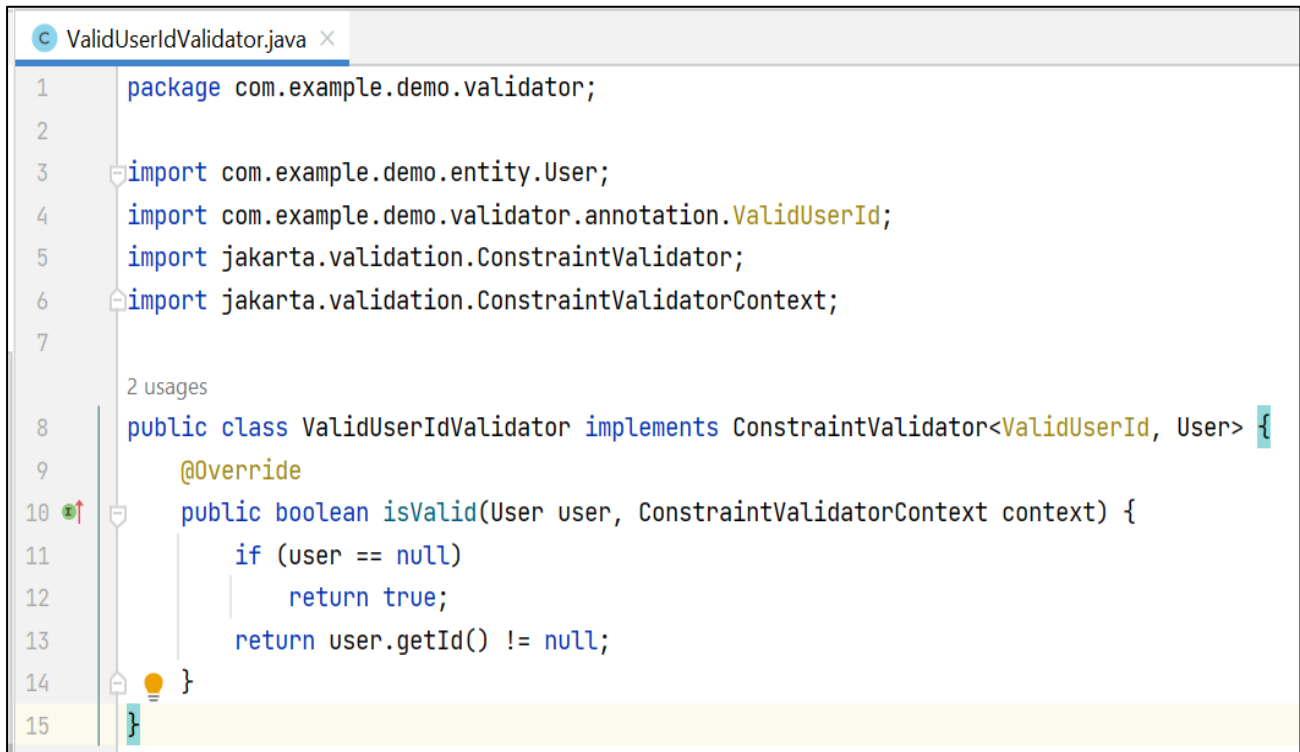
```

1  package com.example.demo.validator.annotation;
2
3  import com.example.demo.validator.ValidUserIdValidator;
4  import jakarta.validation.Constraint;
5  import jakarta.validation.Payload;
6
7  import java.lang.annotation.Documented;
8  import java.lang.annotation.Retention;
9  import java.lang.annotation.Target;
10
11  import static java.lang.annotation.ElementType.FIELD;
12  import static java.lang.annotation.ElementType.TYPE;
13  import static java.lang.annotation.RetentionPolicy.RUNTIME;
14
15  @Target({TYPE, FIELD})
16  @Retention(RUNTIME)
17  @Constraint(validatedBy = ValidUserIdValidator.class)
18  @Documented
19  public @interface ValidUserId {
20      String message() default "Invalid User ID";
21      Class<?>[] groups() default {};
22      Class<? extends Payload>[] payload() default {};
23  }

```

Hình 6.8. Thêm file ValidUserId.java

Tiếp tục, tạo thêm 1 file tên là **ValidUserIdValidator.java** tại thư mục validator **src/main/java/com/example/demo/validator**



```

1  package com.example.demo.validator;
2
3  import com.example.demo.entity.User;
4  import com.example.demo.validator.annotation.ValidUserId;
5  import jakarta.validation.ConstraintValidator;
6  import jakarta.validation.ConstraintValidatorContext;
7
8  2 usages
9  public class ValidUserIdValidator implements ConstraintValidator<ValidUserId, User> {
10     @Override
11     public boolean isValid(User user, ConstraintValidatorContext context) {
12         if (user == null)
13             return true;
14         return user.getId() != null;
15     }
16 }

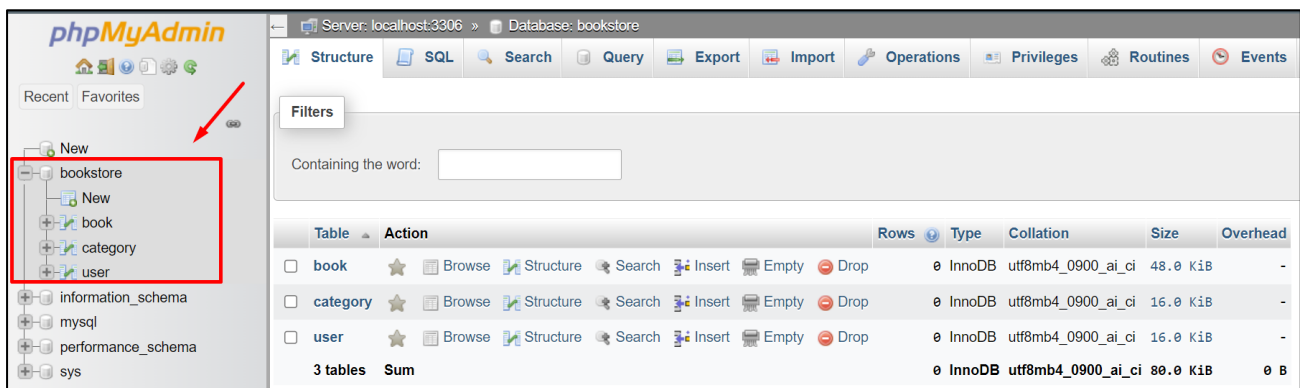
```

Hình 6.9. ValidUserIdValidator.java tại thư mục validator

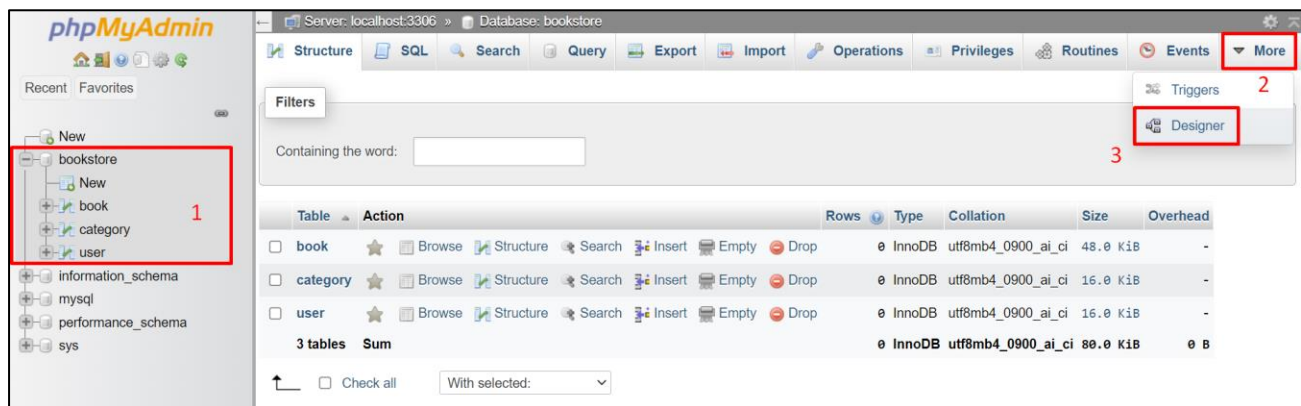
Kiểm tra lại các class và interface đã thêm, build lại project (nhấn Ctrl + F9)

Mở **phpMyAdmin** bằng cách truy cập <http://localhost/phpmyadmin/>

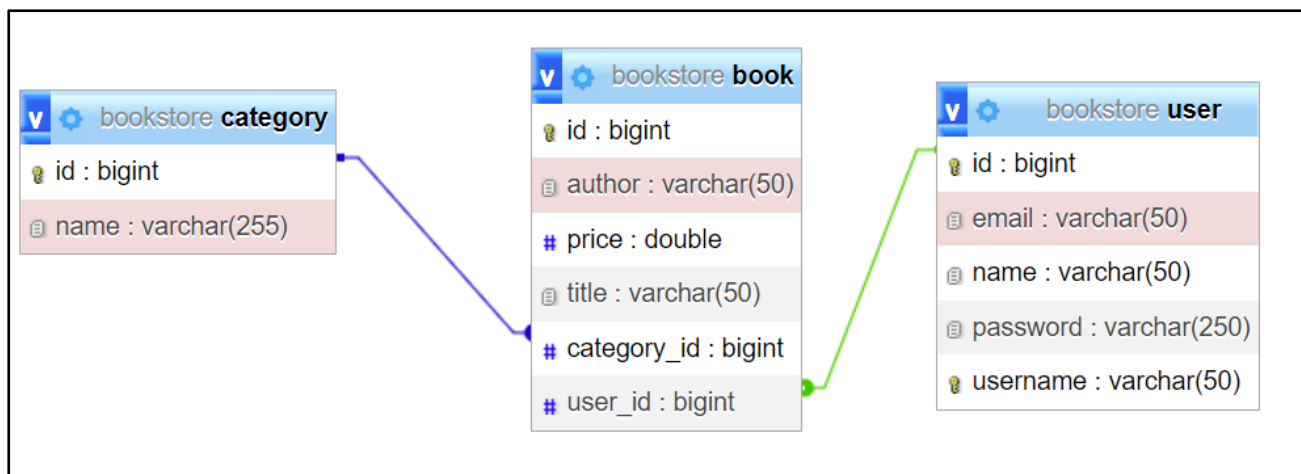
Kiểm tra Database sẽ xuất hiện thêm các table đã thêm tương ứng User.



Hình 6.10. Kiểm tra database sau khi thêm table User




Hình 6.11. Mở Designer và xem kết quả



Hình 6.12. Sơ đồ Designer của database bookstore

Tiếp theo, tạo thêm class mới tên là **UserService.java** trong thư mục services theo đường dẫn **src/main/java/com/example/demo/services**.

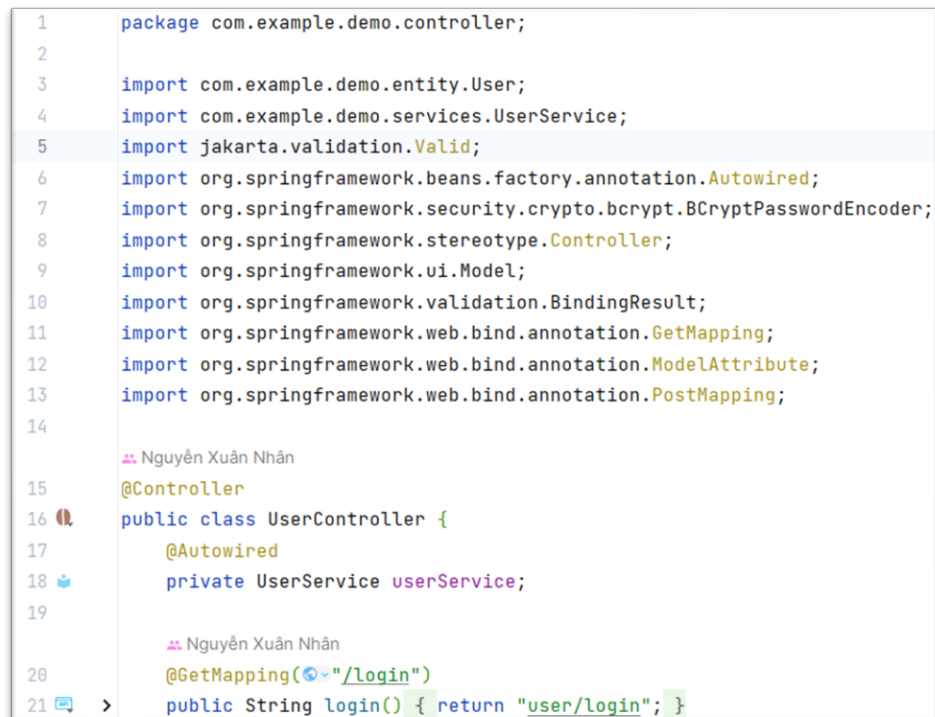
Lớp **UserService** có mục đích cung cấp các phương thức và chức năng để quản lý người dùng và các vai trò của họ trong hệ thống.



```
1 package com.example.demo.services;
2
3 import com.example.demo.entity.User;
4 import com.example.demo.repository.IUserRepository;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 @Service
9 public class UserService {
10     @Autowired
11     private IUserRepository userRepository;
12
13     public void save(User user) {
14         userRepository.save(user);
15     }
16 }
```

Hình 6.13. Thêm class lớp UserService

Tạo file **UserController.java** được đặt tại đường dẫn sau đây  
**src/main/java/com.example.demo/controller**



```
1 package com.example.demo.controller;
2
3 import com.example.demo.entity.User;
4 import com.example.demo.services.UserService;
5 import jakarta.validation.Valid;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
8 import org.springframework.stereotype.Controller;
9 import org.springframework.ui.Model;
10 import org.springframework.validation.BindingResult;
11 import org.springframework.web.bind.annotation.GetMapping;
12 import org.springframework.web.bind.annotation.ModelAttribute;
13 import org.springframework.web.bind.annotation.PostMapping;
14
15 @Controller
16 public class UserController {
17     @Autowired
18     private UserService userService;
19
20     @GetMapping("/login")
21     public String login() { return "user/login"; }
```

```

25  @GetMapping("/register")
26  @PostMapping("/register")
27  public String register(Model model) {
28      model.addAttribute("user", new User());
29      return "user/register";
30  }
31
32  @Valid @ModelAttribute("user") User user, BindingResult bindingResult, Model model) {
33      if (bindingResult.hasErrors()) {
34          bindingResult.getFieldErrors().forEach(error
35              → model.addAttribute("error.getField() + "_error", error.getDefaultMessage());
36          return "user/register";
37      }
38      user.setPassword(new BCryptPasswordEncoder().encode(user.getPassword()));
39      userService.save(user);
40      return "redirect:/login";
41  }
42  }

```

Hình 6.14. Tạo file UserController.java

Tạo thư mục **Utils** đặt tại **src/main/java/com.example.demo**

Tạo file **SecurityConfig.java** đặt tại **src/main/java/com.example.demo/Utils** dán đoạn code bên dưới vào.

**SecurityConfig** cung cấp các cấu hình bảo mật cho ứng dụng web. Cụ thể:

- Phương thức **userDetailsService()** trả về một CustomUserDetailsService, được sử dụng để cung cấp thông tin chi tiết về người dùng.
- Phương thức **passwordEncoder()** trả về một BCryptPasswordEncoder, được sử dụng để mã hóa mật khẩu của người dùng.
- Phương thức **authenticationProvider()** sẽ trả về một DaoAuthenticationProvider, được sử dụng để định nghĩa cách xác thực người dùng.
- Phương thức **securityFilterChain(HttpSecurity http)** trả về một SecurityFilterChain, được sử dụng để cấu hình các hành vi bảo mật trong ứng dụng.

```

package com.example.demo.utils;

import com.example.demo.services.CustomUserDetailsService;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

```

```
ty;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
@EnableMethodSecurity
public class SecurityConfig {

    @Bean
    public UserDetailsService userDetailsService() {
        return new CustomUserDetailService();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public DaoAuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider auth = new DaoAuthenticationProvider();
        auth.setUserDetailsService(userDetailsService());
        auth.setPasswordEncoder(passwordEncoder());
        return auth;
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
        return http.csrf().disable()
            .authorizeHttpRequests(auth -> auth
                .requestMatchers( "/css/**", "/js/**", "/", "/register",
"/error")
                .permitAll()
                .requestMatchers( "/books/edit", "/books/delete")
                .authenticated()
                .requestMatchers("/books", "/books/add")
                .authenticated()
                .anyRequest().authenticated()
            )
            .logout(logout -> logout.logoutUrl("/logout")
                .logoutSuccessUrl("/login")
                .deleteCookies("JSESSIONID")
                .invalidateHttpSession(true)
                .clearAuthentication(true)
                .permitAll()
            )
            .formLogin(formLogin -> formLogin.loginPage("/login")
                .loginProcessingUrl("/login")
                .defaultSuccessUrl("/")
                .permitAll()
            )
            .rememberMe(rememberMe -> rememberMe.key("uniqueAndSecret")
                .tokenValiditySeconds(86400)
            )
        );
    }
}
```

```
        .userService(userDetailsService())
    )
    .exceptionHandling(exceptionHandling ->
        exceptionHandling.accessDeniedPage("/403"))
    .build();
}
}
```



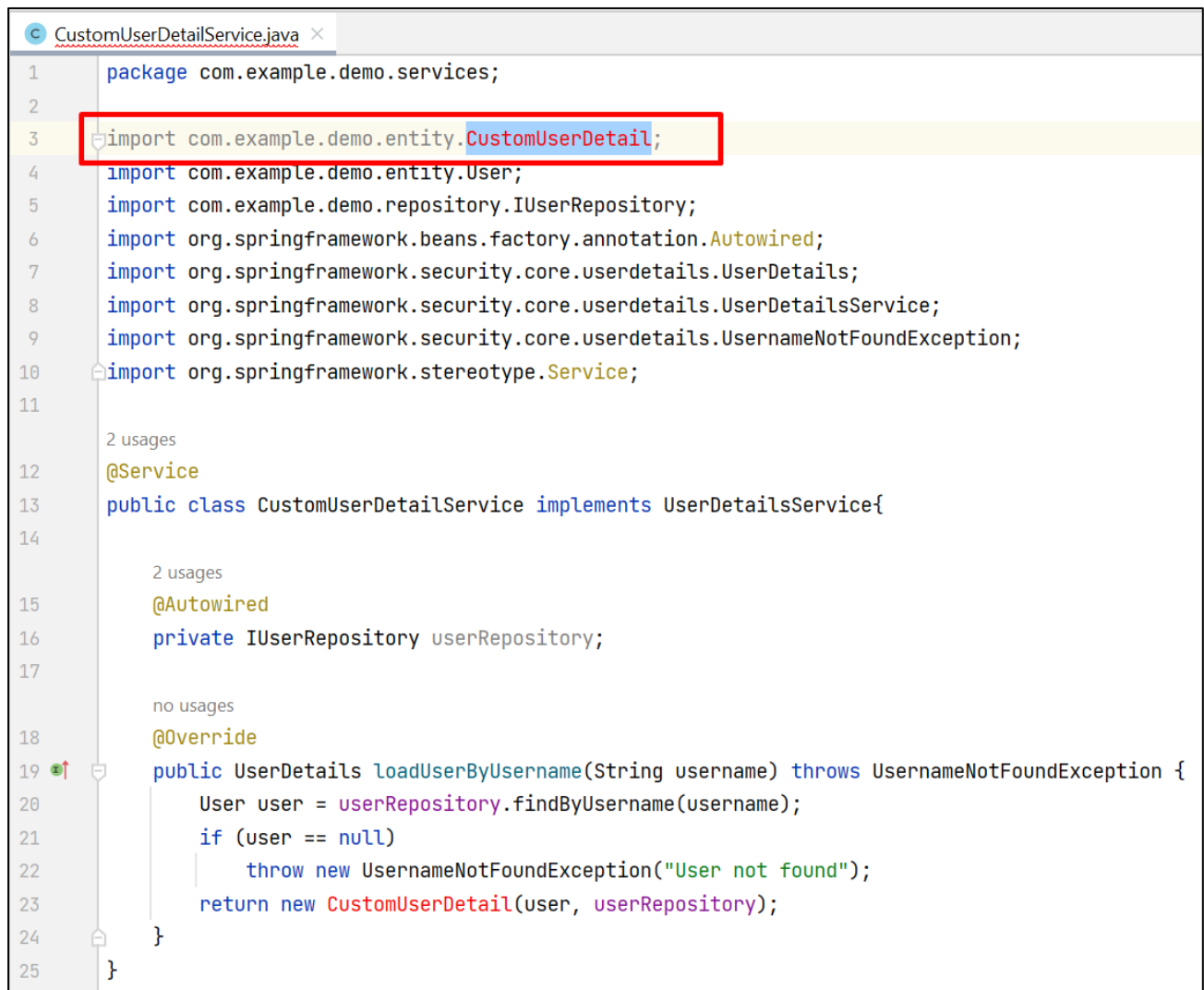
```
38 no usages
39 @Bean
40 @ public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
41     return http.csrf().disable()
42         .authorizeHttpRequests(auth -> auth
43             .requestMatchers(...patterns: "/css/**", "/js/**", "/", "/register", "/error") AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
44             .permitAll() AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
45             .requestMatchers(...patterns: "/books/edit", "/books/delete") AuthorizeHttpRequestsConfigurer<...>.Authenticated
46             .authenticated() AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
47             .requestMatchers(...patterns: "/books", "/books/add") AuthorizeHttpRequestsConfigurer<...>.AuthorizedUrl
48             .authenticated() AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
49             .anyRequest().authenticated()
50         )
51     }
```

Hình 6.15. Giải thích `authenticated()` là gì?

### Tìm hiểu và giải thích, `authenticated()` là gì ?

Tiếp theo, tạo file **CustomUserDetailsService.java** được đặt tại đường dẫn sau đây **src/main/java/com.example.demo/services**

Class **CustomUserDetailsService** là một implementation của interface **UserDetailsService**, cung cấp phương thức **loadUserByUsername** để tìm kiếm thông tin người dùng theo tên đăng nhập.



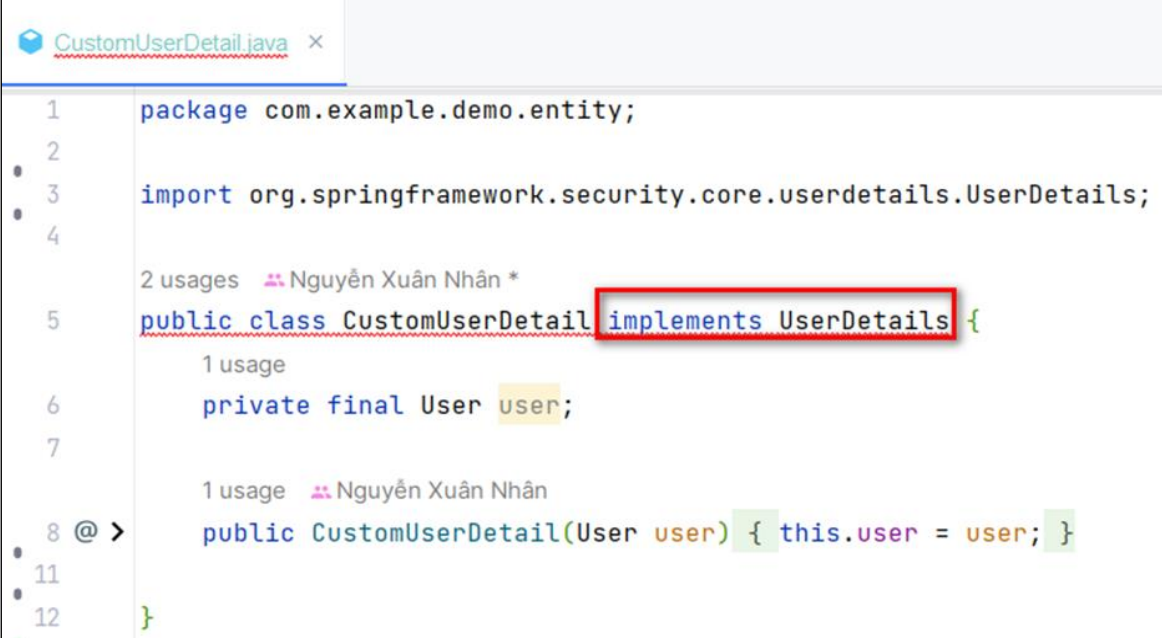
```
1 package com.example.demo.services;
2
3 import com.example.demo.entity.CustomUserDetail;
4 import com.example.demo.entity.User;
5 import com.example.demo.repository.IUserRepository;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.security.core.userdetails.UserDetails;
8 import org.springframework.security.core.userdetails.UserDetailsService;
9 import org.springframework.security.core.userdetails.UsernameNotFoundException;
10 import org.springframework.stereotype.Service;
11
12 2 usages
13 @Service
14 public class CustomUserDetailService implements UserDetailsService{
15
16 2 usages
17     @Autowired
18     private IUserRepository userRepository;
19
20 no usages
21 @Override
22 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
23     User user = userRepository.findByUsername(username);
24     if (user == null)
25         throw new UsernameNotFoundException("User not found");
26     return new CustomUserDetail(user, userRepository);
27 }
```

Hình 6.16. Tạo file CustomUserDetailService.java



Bên trên file **CustomUserDetailsService.java** ở dòng 3 có thông báo lỗi. Lý do là chưa add file **CustomUserDetail.java**

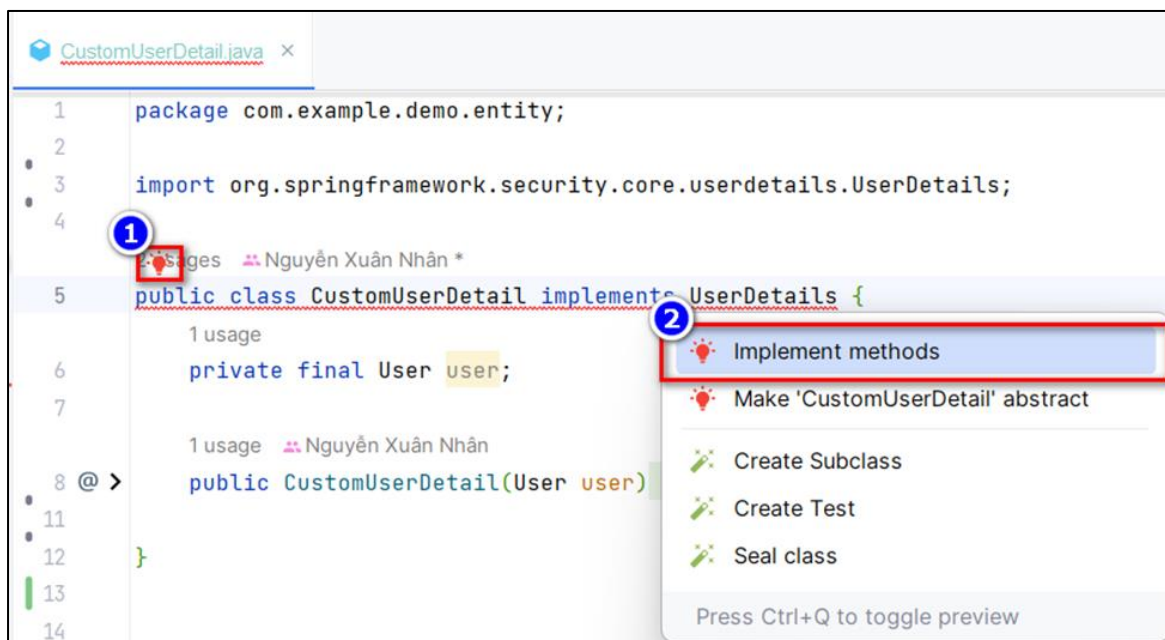
Tạo file **CustomUserDetail.java** được đặt tại đường dẫn sau đây  
**src/main/java/com.example.demo/entity**



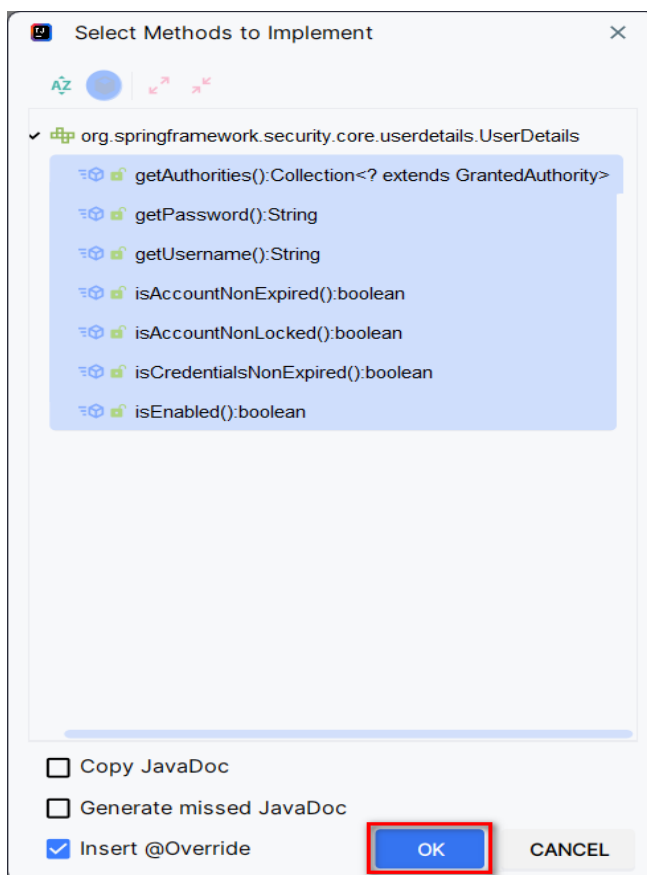
```
1 package com.example.demo.entity;
2
3 import org.springframework.security.core.userdetails.UserDetails;
4
5 public class CustomUserDetail implements UserDetails {
6     private final User user;
7
8     public CustomUserDetail(User user) { this.user = user; }
9
10 }
11
12
```

Hình 6.17. Tạo file CustomUserDetail.java

Nhấn tổ hợp phím **ALT + ENTER** hoặc nhấn vào **nút bóng đèn màu đỏ** để có thể implement các phương thức của **UserDetails**.



Hình 6.18. Implement methods



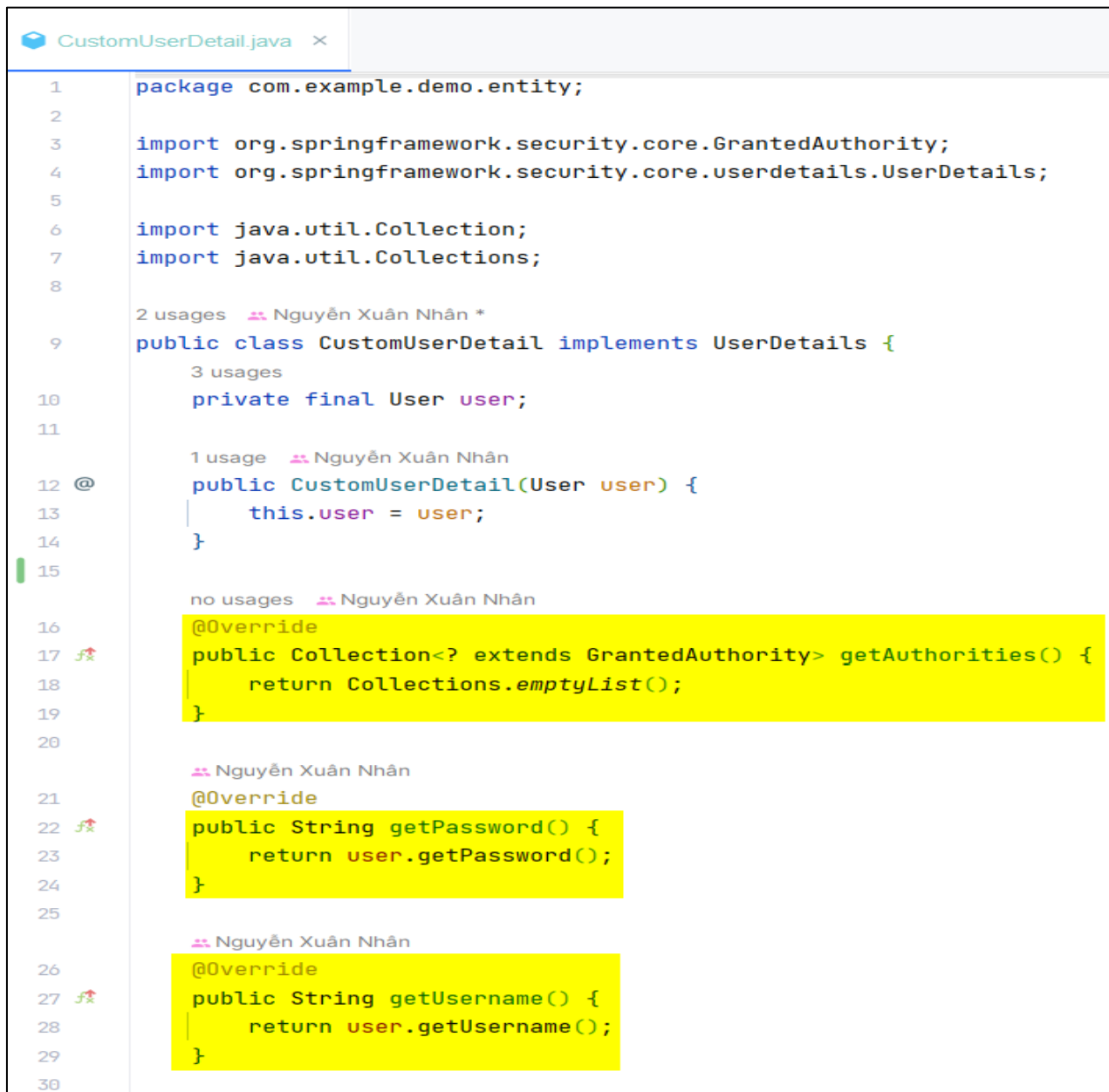
Hình 6.19. Implement các phương thức của UserDetails



```
1 package com.example.demo.entity;
2
3 import org.springframework.security.core.GrantedAuthority;
4 import org.springframework.security.core.userdetails.UserDetails;
5
6 import java.util.Collection;
7
8 public class CustomUserDetail implements UserDetails {
9     private final User user;
10
11     public CustomUserDetail(User user) { this.user = user; }
12
13     @Override
14     public Collection<? extends GrantedAuthority> getAuthorities() {
15         return null;
16     }
17
18     @Override
19     public String getPassword() {
20         return null;
21     }
22
23     @Override
24     public String getUsername() {
25         return null;
26     }
27
28     @Override
29     public boolean isAccountNonExpired() {
30         return false;
31     }
32
33     @Override
34     public boolean isAccountNonLocked() {
35         return false;
36     }
37
38     @Override
39     public boolean isCredentialsNonExpired() {
40         return false;
41     }
42
43     @Override
44     public boolean isEnabled() {
45         return false;
46     }
47 }
```

Hình 6.20. File CustomUserDetail.java

Tiến hành chỉnh sửa các phương thức CustomUserDetail.java theo như ảnh bên dưới.



```
1 package com.example.demo.entity;
2
3 import org.springframework.security.core.GrantedAuthority;
4 import org.springframework.security.core.userdetails.UserDetails;
5
6 import java.util.Collection;
7 import java.util.Collections;
8
9 2 usages  🧑 Nguyễn Xuân Nhân *
10 public class CustomUserDetail implements UserDetails {
11     3 usages
12     private final User user;
13
14     1 usage  🧑 Nguyễn Xuân Nhân
15     @
16     public CustomUserDetail(User user) {
17         this.user = user;
18     }
19
20     no usages  🧑 Nguyễn Xuân Nhân
21     @Override
22     public Collection<? extends GrantedAuthority> getAuthorities() {
23         return Collections.emptyList();
24     }
25
26     🧑 Nguyễn Xuân Nhân
27     @Override
28     public String getPassword() {
29         return user.getPassword();
30     }
31
32     🧑 Nguyễn Xuân Nhân
33     @Override
34     public String getUsername() {
35         return user.getUsername();
36     }
37 }
```

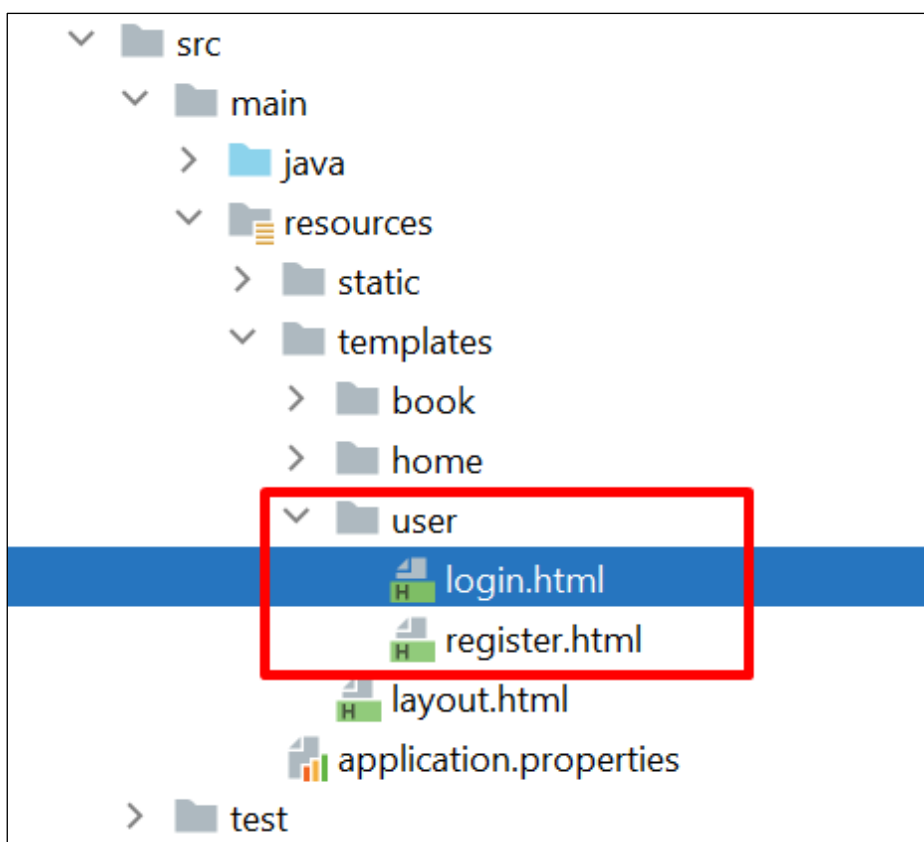
Hình 6.21. Chỉnh sửa các phương thức CustomUserDetail.java

```
31      @Override
32      public boolean isAccountNonExpired() {
33          return true;
34      }
35
36      no usages Nguyễn Xuân Nhân
37      @Override
38      public boolean isAccountNonLocked() {
39          return true;
40      }
41
42      no usages Nguyễn Xuân Nhân
43      @Override
44      public boolean isAccountNonLocked() {
45          return true;
46      }
47
48      no usages Nguyễn Xuân Nhân
49      @Override
50      public boolean isCredentialsNonExpired() {
51          return true;
52      }
53
54      Nguyễn Xuân Nhân
55      @Override
56      public boolean isEnabled() {
57          return true;
58      }
59  }
```

Hình 6.22. Chỉnh sửa các phương thức CustomUserDetail.java

## 6.2 Thiết kế giao diện Login và tạo tài khoản mới

- Tạo thư mục user tại đường dẫn ***src/main/java/com.example.demo/resources/templates***
- Tạo 2 file model login.html và register.html đặt tại thư mục ***src/main/java/com.example.demo/resources/templates/user***



Hình 6.23. Thêm 2 file login.html và register.html

Thêm nội dung code cho file **login.html**

```
login.html x
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Login</title>
8   <th:block th:replace="~{layout :: link-css}"></th:block>
9 </head>
10 <body>
11 <th:block th:replace="~{layout :: header}"></th:block>
12 <div class="container">
13   <div class="row justify-content-center">
14     <div class="col-lg-6 col-md-8">
15       <h3 class="card-title text-center">Login</h3>
16       <form th:action="@{/login}" method="post">
17         <fieldset>
18           <legend>Please login</legend>
19           <div th:if="${param.error}" class="alert alert-danger">
20             Invalid username and password.
21           </div>
22           <div th:if="${param.logout}" class="alert alert-success">
23             You have been logged out.
24           </div>
25           <div class="mb-3">
26             <label for="username" class="form-label">Username:</label>
27             <input type="text" required class="form-control" id="username" name="username">
28           </div>
29           <div class="mb-3">
30             <label for="password" class="form-label">Password:</label>
31             <input type="password" required class="form-control" id="password" name="password">
32           </div>
33           <div class="mb-3">
34             <input type="checkbox" name="remember-me" id="remember-me">
35             <label for="remember-me">Remember me</label>
36           </div>
37           <div class="d-grid gap-2 form-action">
38             <button type="submit" class="btn btn-primary">Login</button>
39             <a class="text-primary text-center" href="/register">Don't have account. Sign up?</a>
40           </div>
41         </fieldset>
42       </form>
43     </div>
44   </div>
45 </div>
46 <th:block th:replace="~{layout :: footer}"></th:block>
47 </body>
48 </html>
```

Hình 6.24. Thêm nội dung cho file login.html

Thêm nội dung code cho file **register.html**

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Sign Up</title>
8   <th:block th:replace="~{layout :: link-css}"></th:block>
9 </head>
10 <body>
11 <th:block th:replace="~{layout :: header}"></th:block>
12 <div class="container">
13   <div class="row justify-content-center">
14     <div class="col-lg-6 col-md-8">
15       <h3 class="card-title text-center">Sign Up</h3>
16       <form th:action="@{/register}" th:object="${user}" method="post">
17         <div class="mb-3">
18           <label for="name" class="form-label">First Name:</label><span class="text-danger">*</span>
19           <input type="text" class="form-control" th:field="${name}" id="name">
20           <span class="text-danger" th:if="${#fields.hasErrors('name')}" th:errors="${name}"></span>
21         </div>
22         <div class="mb-3">
23           <label for="email" class="form-label">Email:</label>
24           <input type="email" class="form-control" th:field="${email}" id="email">
25           <span class="text-danger" th:if="${#fields.hasErrors('email')}" th:errors="${email}"></span>
26         </div>
27         <div class="mb-3">
28           <label for="username" class="form-label">Username:</label><span class="text-danger">*</span>
29           <input type="text" class="form-control" id="username" th:field="${username}">
30           <span class="text-danger" th:if="${#fields.hasErrors('username')}" th:errors="${username}"></span>
31         </div>
32         <div class="mb-3">
33           <label for="password" class="form-label">Password:</label><span class="text-danger">*</span>
34           <input type="password" class="form-control" th:field="${password}" id="password">
35           <span class="text-danger" th:if="${#fields.hasErrors('password')}" th:errors="${password}"></span>
36         </div>
37         <div class="d-grid gap-2">
38           <button type="submit" class="btn btn-info">Sign up</button>
39         </div>
40       </form>
41     </div>
42   </div>
43 </div>
44 <th:block th:replace="~{layout :: footer}"></th:block>
45 </body>
46 </html>

```

Hình 6.25. Thêm nội dung cho file register.html

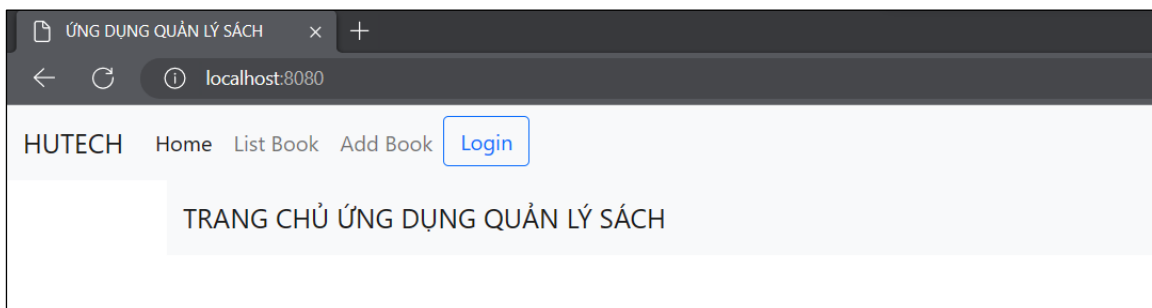


Chỉnh sửa file **layout.html**

```
1 <!DOCTYPE html>
2 <html
3     xmlns:th="http://www.thymeleaf.org"
4     xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4"
5     lang="en">
6 <head>
7     <meta charset="UTF-8">
8     <title>My App</title>
9     <link th:fragment="link-css" rel="stylesheet" th:href="@{/css/bootstrap.min.css}">
10 </head>
11 <body>
12 <header th:fragment="header">
13     <nav class="navbar navbar-expand-lg navbar-light bg-light">
14         <div class="container-fluid">
15             <a class="navbar-brand" href="/">HUTECH</a>
16             <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
17                 data-bs-target="#navbarSupportedContent"
18                 aria-controls="navbarSupportedContent"
19                 aria-expanded="false" aria-label="Toggle navigation">
20                 <span class="navbar-toggler-icon"></span>
21             </button>
22             <div class="collapse navbar-collapse" id="navbarSupportedContent">
23                 <ul class="navbar-nav me-auto mb-2 mb-lg-0">
24                     <li class="nav-item">
25                         <a class="nav-link active" aria-current="page" href="/">Home</a>
26                     </li>
27                     <li class="nav-item"><a class="nav-link" href="/books">List Book</a></li>
28                     <li class="nav-item"><a class="nav-link" href="/books/add">Add Book</a></li>
29                     <li sec:authorize="isAuthenticated()">
30                         <form th:action="@{/logout}" method="post">
31                             <button class="btn btn-outline-danger" type="submit">Logout</button>
32                         </form>
33                     </li>
34                     <li sec:authorize="!isAuthenticated()">
35                         <a class="btn btn-outline-primary" href="/login">Login</a>
36                     </li>
37                 </ul>
38             </div>
39         </div>
40     </nav>
41 </header>
42
43 <footer th:fragment="footer">
44     <script th:src="@{/js/bootstrap.min.js}"></script>
45 </footer>
46 </body>
47 </html>
```

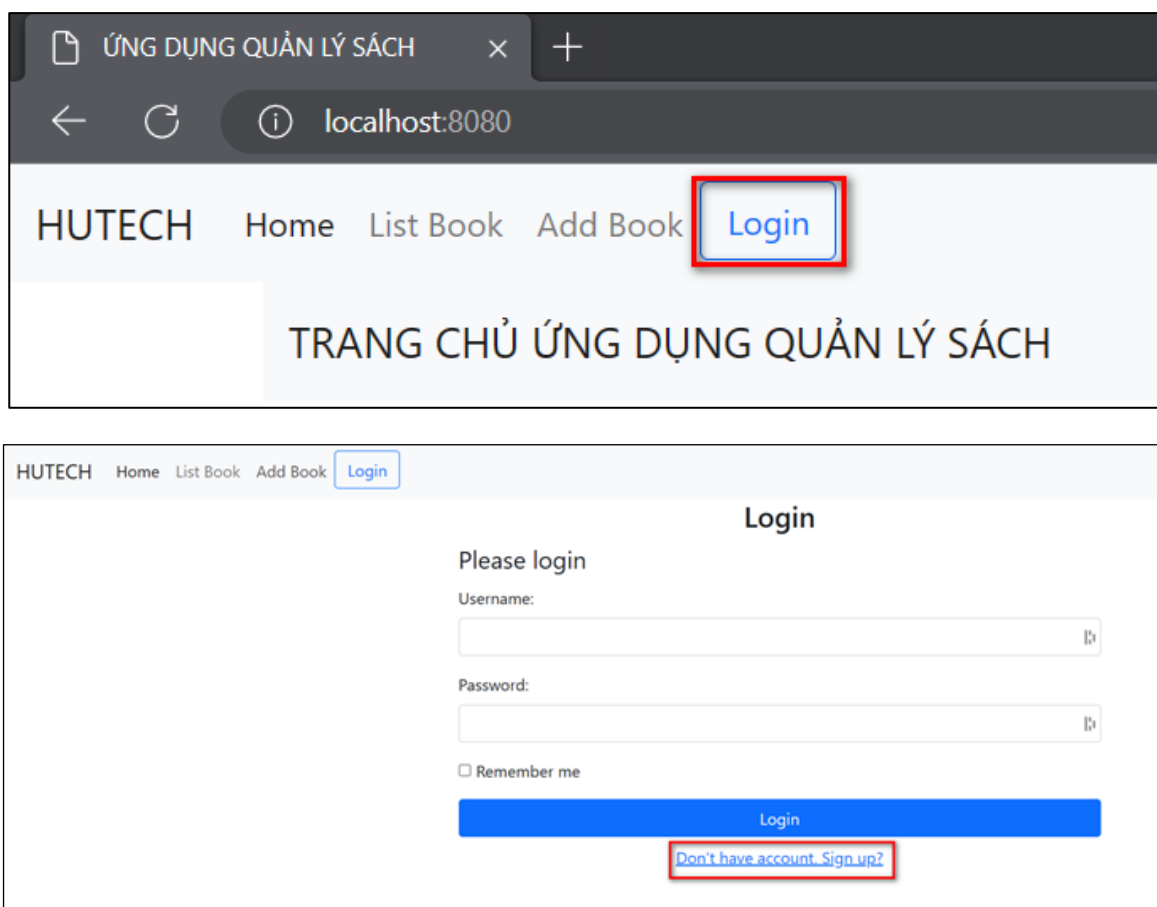
Hình 6.26. Chỉnh sửa lại file layout.html

**Build** và chạy ứng dụng tại địa chỉ **localhost:8080**




Hình 6.27. Giao diện web chưa đăng nhập

Kiểm tra bằng cách tạo tài khoản. Theo các bước sau:



Hình 6.28. Tạo tài khoản mới

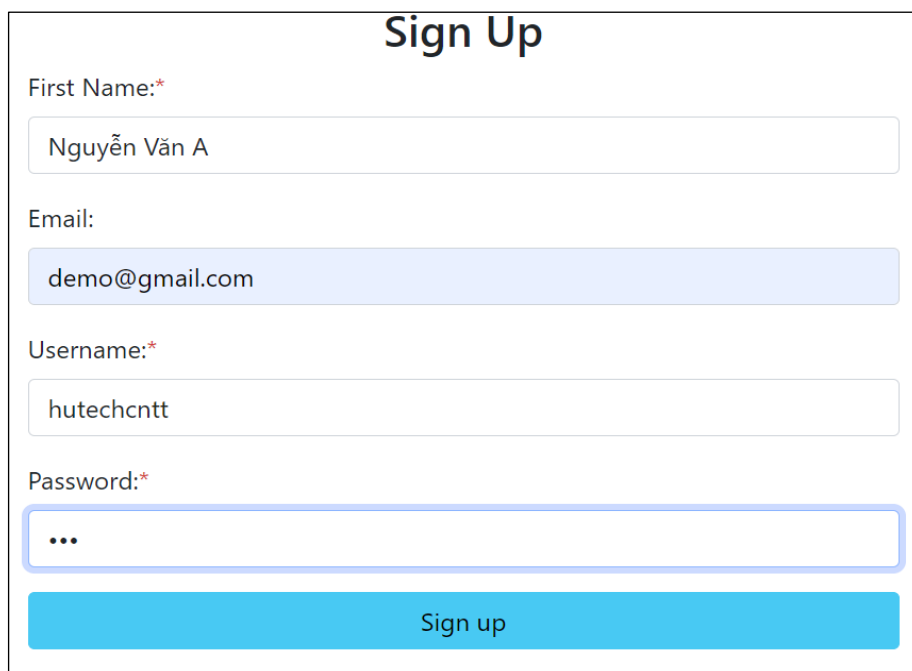
Kiểm tra **valid dữ liệu** bằng cách không nhập và ấn nút **sign up**



The image shows a 'Sign Up' form with four input fields: 'First Name:\*', 'Email:', 'Username:\*', and 'Password:\*'. Each field is empty, and below each field is a red error message: 'Your name is required', 'Email is required', 'Username is required', and 'Password is required'. At the bottom of the form is a blue button labeled 'Sign up'.

Hình 6.29. Kiểm tra valid dữ liệu khi không điền dữ liệu

Tiến hành đăng ký thử tài khoản làm mẫu:

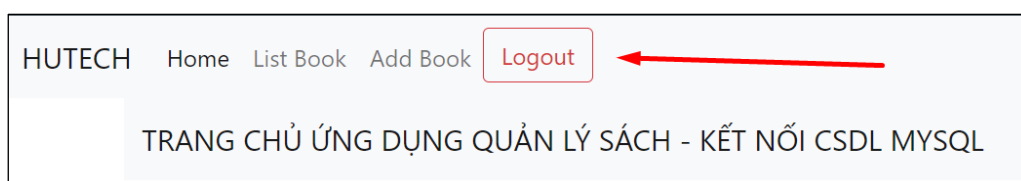


The image shows a 'Sign Up' form with four input fields: 'First Name:\*', 'Email:', 'Username:\*', and 'Password:\*'. The fields are filled with sample data: 'Nguyễn Văn A', 'demo@gmail.com', 'hutehcntt', and '...'. At the bottom of the form is a blue button labeled 'Sign up'.

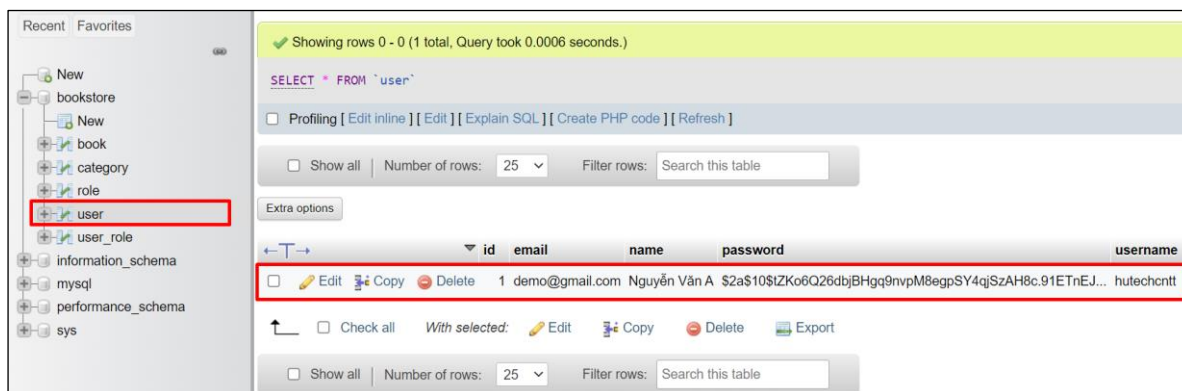
Hình 6.30. Đăng ký thử tài khoản mẫu

Hình 6.31. Đăng nhập bằng tài khoản vừa tạo

Đăng nhập thành công, nút **Login** chuyển sang trạng thái **Logout**.



Hình 6.32. Đăng nhập thành công, xuất hiện nút Logout



Hình 6.33. Kiểm tra thông tin tài khoản đã có trong Database

## TÓM TẮT

Trong Java Spring Boot, User và Role có vai trò quan trọng trong bảo mật ứng dụng và phân quyền người dùng.

User là đại diện cho người dùng trong hệ thống. Mỗi người dùng sẽ có một tài khoản riêng, được lưu trữ trong cơ sở dữ liệu. Thông tin người dùng bao gồm tên đăng nhập, mật khẩu đã được mã hóa, và các thuộc tính khác như tên, địa chỉ, email, số điện thoại, v.v.

Role là đại diện cho vai trò hoặc quyền hạn của người dùng trong hệ thống. Ví dụ, có thể có các vai trò như Admin, User, Manager, v.v. Mỗi vai trò sẽ được cấu hình để chỉ định các quyền hạn tương ứng với nó. Các quyền hạn này có thể được sử dụng để giới hạn truy cập vào các chức năng, tính năng hoặc tài nguyên khác nhau trong hệ thống.

Trong ứng dụng Spring Boot, thông tin người dùng và vai trò thường được lưu trữ trong cơ sở dữ liệu và được sử dụng bởi Spring Security để xác thực và phân quyền người dùng.

Trong Java Spring Boot, để tạo tài khoản và đăng nhập, bạn có thể sử dụng Spring Security để bảo mật ứng dụng và xác thực người dùng. Để làm điều này, bạn cần cấu hình SecurityConfig để chỉ định các bộ xác thực, phân quyền và các cấu hình khác cho ứng dụng.

## CÂU HỎI ÔN TẬP

1. Làm thế nào để thêm một người dùng mới vào hệ thống trong Java Spring Boot?
2. Làm thế nào để bảo mật thông tin người dùng trong Java Spring Boot, đảm bảo rằng mật khẩu được mã hóa và không bị đánh cắp hoặc lộ ra ngoài?