

Tuần 6-7: Spring Boot – Spring Data Jpa – Thymeleaf Template Engine

1. Database **shoppingdb** với các table customers, products, orders, users, orderlines.

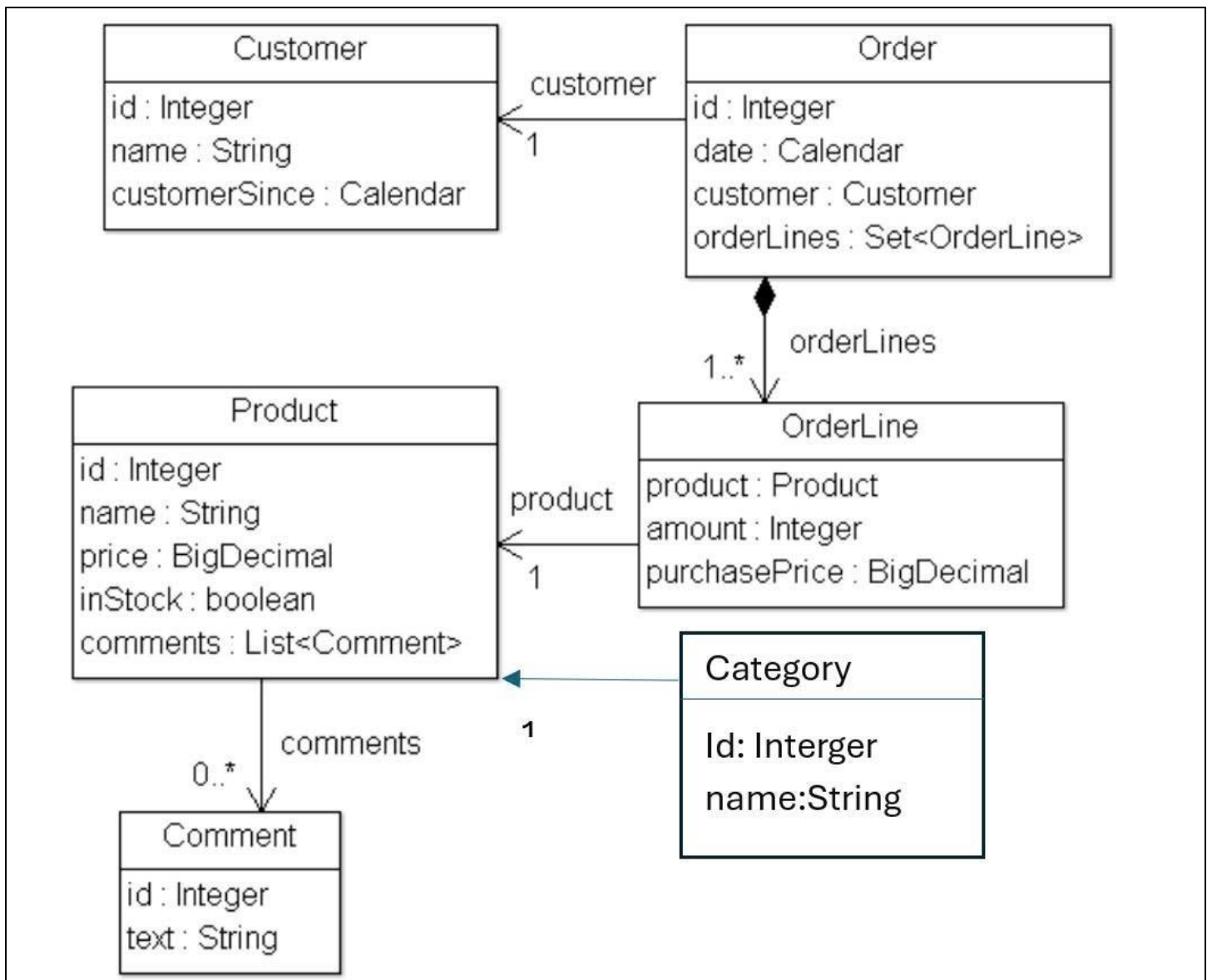
Tương ứng với Entities và các relationships được mô tả như sau:

- **Customer** ↔ **Order**: One-to-many (1 khách hàng có nhiều hóa đơn).
- **Order** ↔ **OrderLine**: One-to-many (1 hóa đơn có nhiều chi tiết hóa đơn).
- **Product** ↔ **OrderLine**: One-to-many (1 sản phẩm có nhiều chi tiết hóa đơn).
- **Product** ↔ **Comment**: One-to-many (1 sản phẩm có nhiều comment).

Chức năng chính cho ứng dụng cần thực thi:

- Theo dõi lịch sử mua hàng và lịch sử giao dịch của khách hàng.
- Quản lý hàng tồn kho và phản hồi sản phẩm.
- Ghi lại thông tin đơn hàng chi tiết, bao gồm giá cả và số lượng.
- Hỗ trợ hồ sơ người dùng để truy cập hoặc quản trị hệ thống. (User)

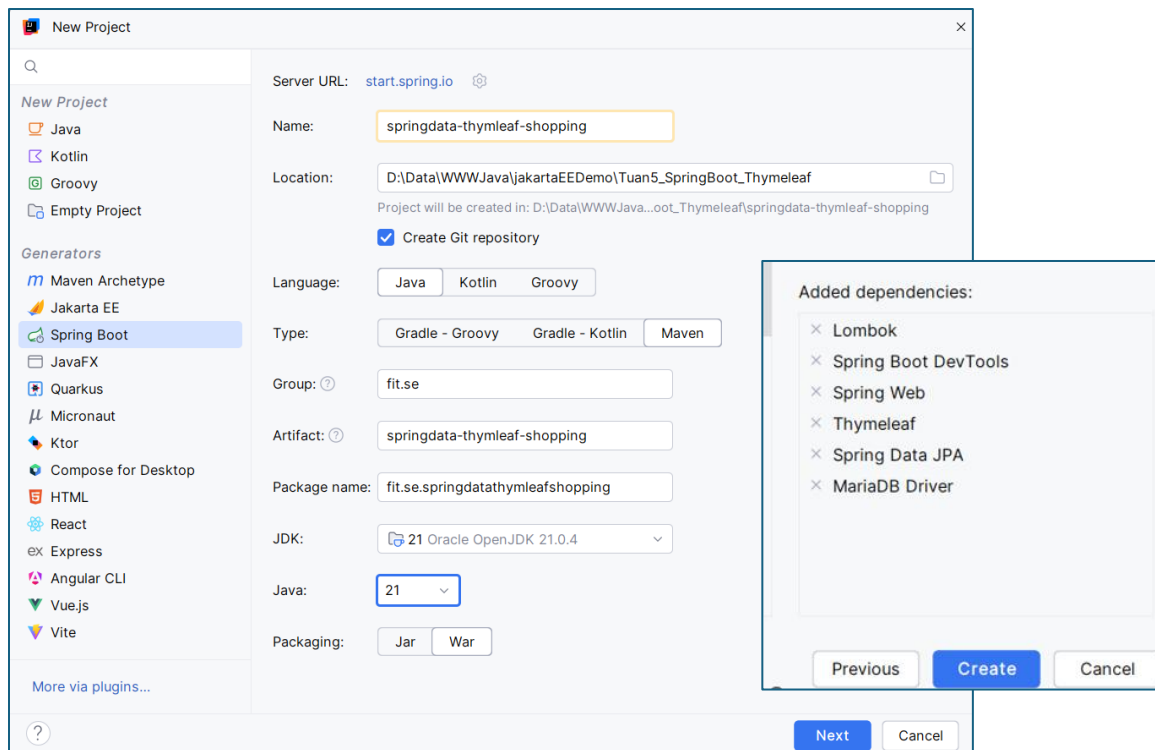
Lược đồ ERD như sau:



Đặc tả (BA)

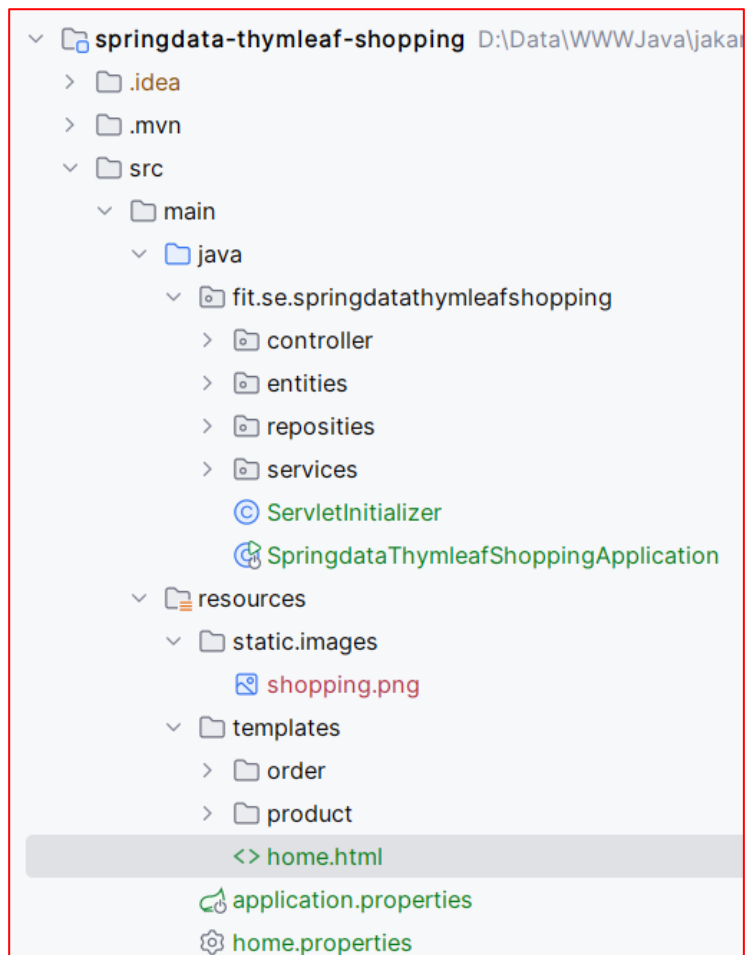
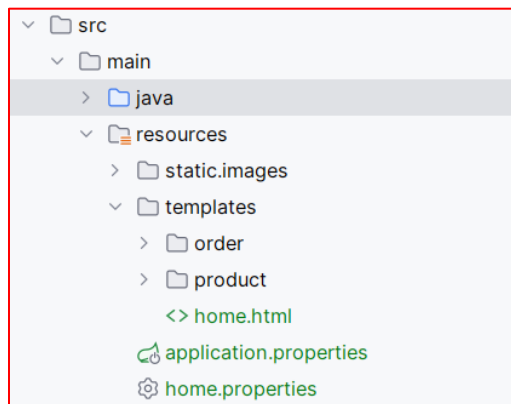
Actor: Admin, customer

2. IDE IntelliJ New Project Spring Boot:



Next → Chọn một số Dependencies cần thiết:

Cấu trúc Project như sau:



3. Config Spring Boot, mariadb, thymeleaf trong file **application.properties**

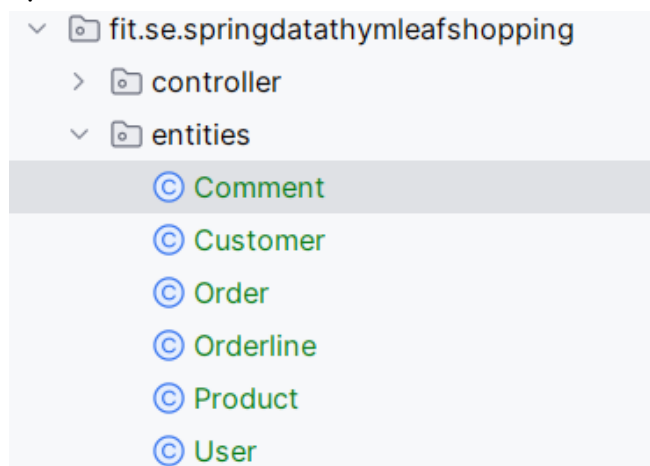
```
spring.application.name=springdata-thymleaf-shopping
server.port=8085
spring.devtools.add-properties=false
logging.level.web=debug
spring.messages.basename=home

# = Database Configuration
spring.datasource.url=jdbc:mariadb://localhost:3306/shoppingdb
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver

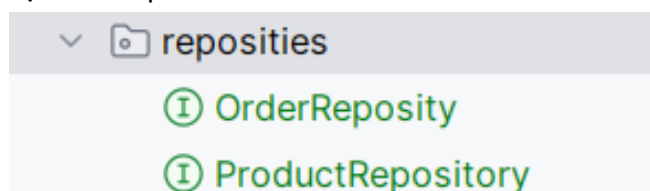
# = JPA / Hibernate
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.sql.init.mode=always
spring.sql.init.platform=mariadb
spring.jpa.database-platform=org.hibernate.dialect.MariaDBDialect
#spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect

# = thymeleaf Config
spring.thymeleaf.cache=false
spring.thymeleaf.prefix=classpath:/templates/
spring.thymeleaf.suffix=.html
```

4. Tạo Entities

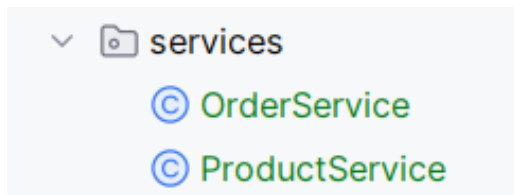


5. Tạo các repositories

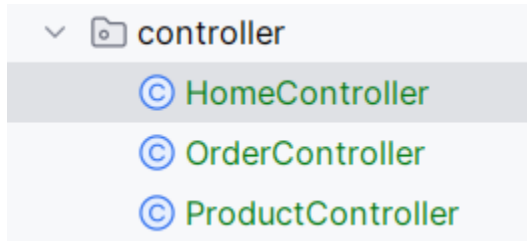


Các Repository extends JpaRepository (các phương thức truy cập theo Method Naming)

6. Tạo các services tương ứng



7. Tạo Controller truyền Model trả về cho các view là Thymeleaf Template



HomeController.java: trả về view home.html

Annotation @Controller, @RequestMapping cho view Thymeleaf.

Truyền qua view html thông qua Model

Các trang .html sử dụng Thymeleaf phải đúng cấu trúc thư mục cho phần view render về cho client đúng path được config thông qua instruction `spring.thymeleaf.prefix` trên application.properties

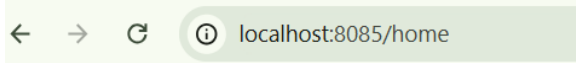
```
# = thymeleaf Config
spring.thymeleaf.cache=false
spring.thymeleaf.prefix=classpath:/templates/
spring.thymeleaf.suffix=.html
```

HomeController.java

```
1 package fit.se.springdatathymleafshopping.controller;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.ui.Model;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestMapping;
7
8 import java.time.LocalDate;
9 import java.util.Date;
10
11 @Controller
12 @RequestMapping("/home")
13 public class HomeController {
14
15     public HomeController() {
16         super();
17     }
18     @GetMapping
19     public String HomePage(Model model) {
20         LocalDate date = LocalDate.now();
21         String mess = "Welcome Thymeleaf";
22         model.addAttribute("message", mess);
23         model.addAttribute("date", date.toString());
24         return "home";
25     }
26 }
27
```

HomeController có annotation @Controller, @RequestMapping ("/home"): tương ứng với URI khi

trình duyệt:



Trong đó method `public String HomePage(Model model)`

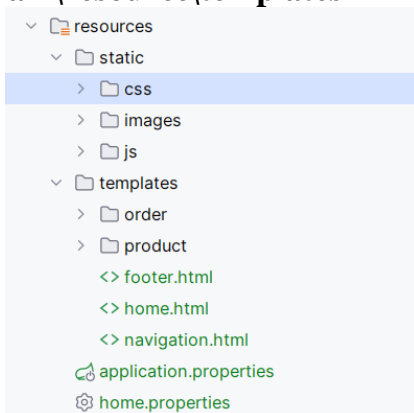
- @GetMapping không tham số nghĩa là mặc định URI như trên.
- Return về String "home" tương ứng với trang home.html.
- Method truyền Model và add 2 attribute `message`, `date`.

@GetMapping

```
public String HomePage(Model model)
{
    LocalDate date = LocalDate.now();
    String mess = "Welcome Thymeleaf";
    model.addAttribute("message", mess);
    model.addAttribute("date", date.toString());
    return "home";
}
```

8. Tạo View: sử dụng Thymeleaf template là các file html nhận dữ liệu thông qua Model mà Controller render về cho client (addAttribute)

Theo config thymeleaf trên thì tất cả các trang html phải thiết lập trong thư mục **main\resource\templates**



Trang templates\home.html sử dụng Thymeleaf template lấy attribute từ controller

home.html

```
1 <!DOCTYPE html>
2 <html lang="en" xmlns:th="http://www.thymeleaf.org">
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6 </head>
7 <body>
8
9 <p>
10
11
12 
13
14 <h1 th:text="${message}">mess</h1>
15
16 <h2 th:text="#{home.welcome}">welcome</h2>
17
18 <p th:text="${date}"></p>
```

```

19 <p>Please select an option</p>
20 <ol>
21     <li><a href="product/list.html" th:href="@{/product}">Product List</a></li>
22     <li><a href="order/list.html" th:href="@{/order}">Order List</a></li>
23 </ol>
24 </body>
25 </html>

```

Tương tự ProductController có annotation @Controller, @RequestMapping ("/product"): tương ứng với

URI khi trình duyệt: 

Trong đó có 2 phương thức:

- Lấy tất cả các Product (`public String showAllProducts(Model model)`), controller sử dụng model truyền qua view với giá trị "products" được lấy từ `productService.findAll()`; return "product/list": tương ứng với view thymeleaft trong `templates\product\list.html`
- Lấy 1 Product theo Id (`public String showProduct(@PathVariable int id, Model model)`), controller sử dụng model truyền qua view với giá trị "product" được lấy từ `productService.findById(id)`; return "product/productdetail": tương ứng với view thymeleaf trong `templates\product\productdetail.html`

ProductController.java

```

1  package fit.se.springdatathymleafshopping.controller;
2
3
4  import fit.se.springdatathymleafshopping.entities.Product;
5  import fit.se.springdatathymleafshopping.services.ProductService;
6  import org.springframework.beans.factory.annotation.Autowired;
7  import org.springframework.stereotype.Controller;
8  import org.springframework.web.bind.annotation.GetMapping;
9  import org.springframework.web.bind.annotation.PathVariable;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.ui.Model;
12
13
14 import java.util.List;
15
16 @Controller
17 @RequestMapping("/product")
18 public class ProductController {
19     @Autowired
20     private final ProductService productService;
21     public ProductController(ProductService productService) {
22         this.productService = productService;
23     }
24     @GetMapping
25     public String showAllProducts(Model model) {
26         List<Product> productlist = productService.findAll();
27         model.addAttribute("products", productlist);
28         return "product/list";
29     }
30     @GetMapping("/{id}")
31     public String showProduct(@PathVariable int id, Model model) {
32         Product product=productService.findById(id);

```

```

33         model.addAttribute("product", product);
34         return "product/productdetail";
35     }
36 }
37

```

Tương tự cho trang templates\product\list.html

list.html

```

1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3  <head>
4      <meta charset="UTF-8">
5      <title>Product List</title>
6      <style>
7          body {
8              font-family: Arial, sans-serif;
9              margin: 20px;
10         }
11
12         .product {
13             border: 1px solid #ccc;
14             padding: 15px;
15             margin-bottom: 20px;
16             border-radius: 8px;
17         }
18
19         .product h3 {
20             margin: 0;
21             color: #2a7;
22         }
23
24         .comments {
25             margin-top: 10px;
26             padding-left: 20px;
27         }
28
29         .comment {
30             border-left: 3px solid #2a7;
31             margin-bottom: 8px;
32             padding-left: 10px;
33             font-style: italic;
34         }
35     </style>
36 </head>
37 <body>
38 <h1>List Product</h1>
39
40 <div th:each="product : ${products}" class="product">
41     <a href="product/productdetail.html" th:href="@{'/product/' + ${product.id}}">
42         <input type="hidden" th:value="${product.id}" th:name="productId">
43         <h2 th:text="${product.name}">Product Name</h2>
44     </a>
45
46
47     <p><strong>Price:</strong> <span th:text="${product.price}">0</span> USD</p>
48
49     <div class="comments">
50         <h3>Comments:</h3>
51         <div th:if="${#lists.isEmpty(product.comments)}">
52             <p><em>Nothing</em></p>

```



```

53         </div>
54         <div th:each="c : ${product.comments}" class="comment">
55             <span th:text="${c.text}">Comment Content</span>
56         </div>
57     </div>
58 </div>
59
60 </body>
61 </html>
62

```

productdetail.html

```

1  <!DOCTYPE html>
2  <html lang="en" xmlns:th="http://www.thymeleaf.org">
3  <head>
4      <meta charset="UTF-8">
5      <title>Title</title>
6      <style>
7          .comments {
8              margin-top: 10px;
9              padding-left: 20px;
10         }
11
12         .comment {
13             border-left: 3px solid #2a7;
14             margin-bottom: 8px;
15             padding-left: 10px;
16             font-style: italic;
17         }
18     </style>
19 </head>
20 <body>
21     <h1>Product Detail</h1>
22     <p th:text="'Name: ' + ${product.name}"></p>
23     <p th:text="'Price: ' + ${product.price}"></p>
24     <div class="comments">
25         <h3>Comments:</h3>
26         <div th:if="${#lists.isEmpty(product.comments)}">
27             <p><em>Nothing</em></p>
28         </div>
29         <div th:each="c : ${product.comments}" class="comment">
30             <span th:text="${c.text}">Comment Content</span>
31         </div>
32     </div>
33
34 </body>
35 </html>

```

Tuần 8-9: Spring Security

Thiết lập 3 role cho hệ thống:

1. Guest: người dùng không có tài khoản, có chức năng Đăng ký tài khoản, xem danh sách Product.
2. Customer: người dùng có tài khoản, có chức năng Xem danh sách Product, mua Product, Lưu Order, tra cứu Order, OrderLine
3. Admin: Người quản trị hệ thống có đầy đủ các chức năng quản trị Customer, Product

Các bước thực hiện thiết lập spring Security:

Bước 1: Add Dependency

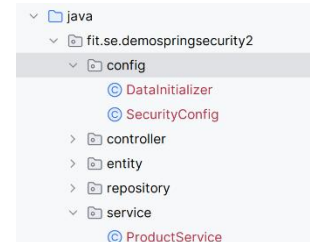
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Bước 2: Tạo **config** package in project

→ Tạo **SercurityConfig** class và add annotation

Code Gợi ý cho 2 role: ADMIN, CUSTOMER,..

Trong **SercurityConfig**, tạo a bean trả về **SecurityFilterChain** có tham số là **HttpSecurity** cho phép thực hiện các filter methods tương ứng với các Role



SecurityConfig.java

```
@Configuration
@EnableWebSecurity
@EnableMethodSecurity(prePostEnabled = true)
public class SercurityConfig {
    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public UserDetailsService userDetailsService() {
        UserDetails admin = User.builder()
            .username("admin")
            .password(passwordEncoder().encode("123"))
            .roles("ADMIN")
            .build();

        UserDetails customer = User.builder()
            .username("customer")
            .password(passwordEncoder().encode("111"))
            .roles("CUSTOMER")
            .build();

        return new InMemoryUserDetailsManager(admin, customer);
    }

    @Bean
    public SecurityFilterChain securityFilterChain (HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests(auth -> auth
                //1. Quy tắc mở
```

```

        .requestMatchers("/login", "/css/**", "/js/**", "/images/**",
"/test").permitAll()
        //2. Phân quyền

.requestMatchers("/product", "/product/detail/**").hasAnyRole("CUSTOMER", "ADMIN")
        //3. Đọc quyền
        .requestMatchers("/product/add", "/product/edit/",
"/product/update/").hasRole("ADMIN")
        //4. phần request còn lại phải chứng thực
        .anyRequest().authenticated()
    )
    .formLogin(form -> form
        //
        .loginPage("/login")
        .defaultSuccessUrl("/product")
        .permitAll()
    )
    .logout(logout -> logout
        .logoutUrl("/logout") // URL người dùng truy cập để đăng xuất
        (mặc định là POST)
        .logoutSuccessUrl("/login?logout=true") // Chuyển hướng sau khi
        đăng xuất thành công
        .invalidateHttpSession(true) // Hủy session
        .deleteCookies("JSESSIONID") // Xóa cookie session
        .permitAll() // Cho phép tất cả mọi người truy cập vào URL logout
    );

    return http.build();
}
}

```

Bước 3: Thiết lập các Mapping cho các Methods trong Controller với các role đã config:

Annotation cho role tương ứng:

Code gợi ý:

ProductController.java

```

@Controller
@RequestMapping("/product")

public class ProductController {

    @Autowired
    private ProductService productService;

    @Autowired
    private CategoryService categoryService;

    @GetMapping()
    public String showProducts(Model model) {

        model.addAttribute("products", productService.findAll());
        return "product/list";
    }

    @PreAuthorize("hasRole('ADMIN')")
    @GetMapping("/add")
    public String showAddForm(Model model) {
        model.addAttribute("product", new Product());
    }
}

```

```

        model.addAttribute("categories", categoryService.findAll());
        return "product/productform";
    }
    @PreAuthorize("hasRole('ADMIN')")
    @PostMapping("/save")
    public String saveProduct(@ModelAttribute Product product,
                              @RequestParam("fileImage") MultipartFile file
                              ) throws IOException {

        if (!file.isEmpty()) {
            // String fileName = file.getOriginalFilename();
            String fileName = UUID.randomUUID() + "_" + file.getOriginalFilename();
            // Path uploadDir = Paths.get("uploads/");
            Path uploadDir = Paths.get("src/main/resources/static/uploads/");

            Files.createDirectories(uploadDir);
            file.transferTo(uploadDir.resolve(fileName));
            product.setImage(fileName);
        }
        productService.save(product);
        return "redirect:/product";
    }

    // Show edit form
    @PreAuthorize("hasRole('ADMIN')")
    @GetMapping("/edit/{id}")
    public String showEditForm(@PathVariable("id") Long id, Model model) {
        Product product = productService.findById(id)
            .orElseThrow(() -> new IllegalArgumentException("Invalid product Id: " +
id));

        model.addAttribute("product", product);
        model.addAttribute("categories", categoryService.findAll());
        return "product/editproduct";
    }

    // Handle update
    @PreAuthorize("hasRole('ADMIN')")
    @PostMapping("/update")
    public String updateProduct(@ModelAttribute("product") Product product,
                                @RequestParam("fileImage") MultipartFile file) throws
IOException {

        // Handle new image upload (optional)
        if (!file.isEmpty()) {
            // String fileName = file.getOriginalFilename();
            String fileName = UUID.randomUUID() + "_" + file.getOriginalFilename();
            // Path uploadDir = Paths.get("uploads/");
            Path uploadDir = Paths.get("src/main/resources/static/uploads/");

            Files.createDirectories(uploadDir);
            file.transferTo(uploadDir.resolve(fileName));
            product.setImage(fileName);
        } else {
            // keep existing image
            Product existing = productService.findById(product.getId()).orElse(null);
            if (existing != null) {
                product.setImage(existing.getImage());
            }
        }

        productService.save(product);
    }

```

```

        return "redirect:/product";
    }

@GetMapping("/detail/{id}")
public String showProductDetails(@PathVariable("id") Long id, Model model) {
    Product p=productService.findById(id).orElse(null);
    model.addAttribute("product", p);
    return "product/productdetail";
}
}

```

Bước 4: Tại View thiết lập các role cần gọi các config cho các action:

Code gợi ý:

```

<div sec:authorize="hasRole('ADMIN')">
    <a th:href="@{/products/add}" class="btn btn-primary">
        <i class="fas fa-plus"></i> Add New Product
    </a>
</div>

```