Spring Boot – Spring Data Jpa – Thymeleaf Template Engine

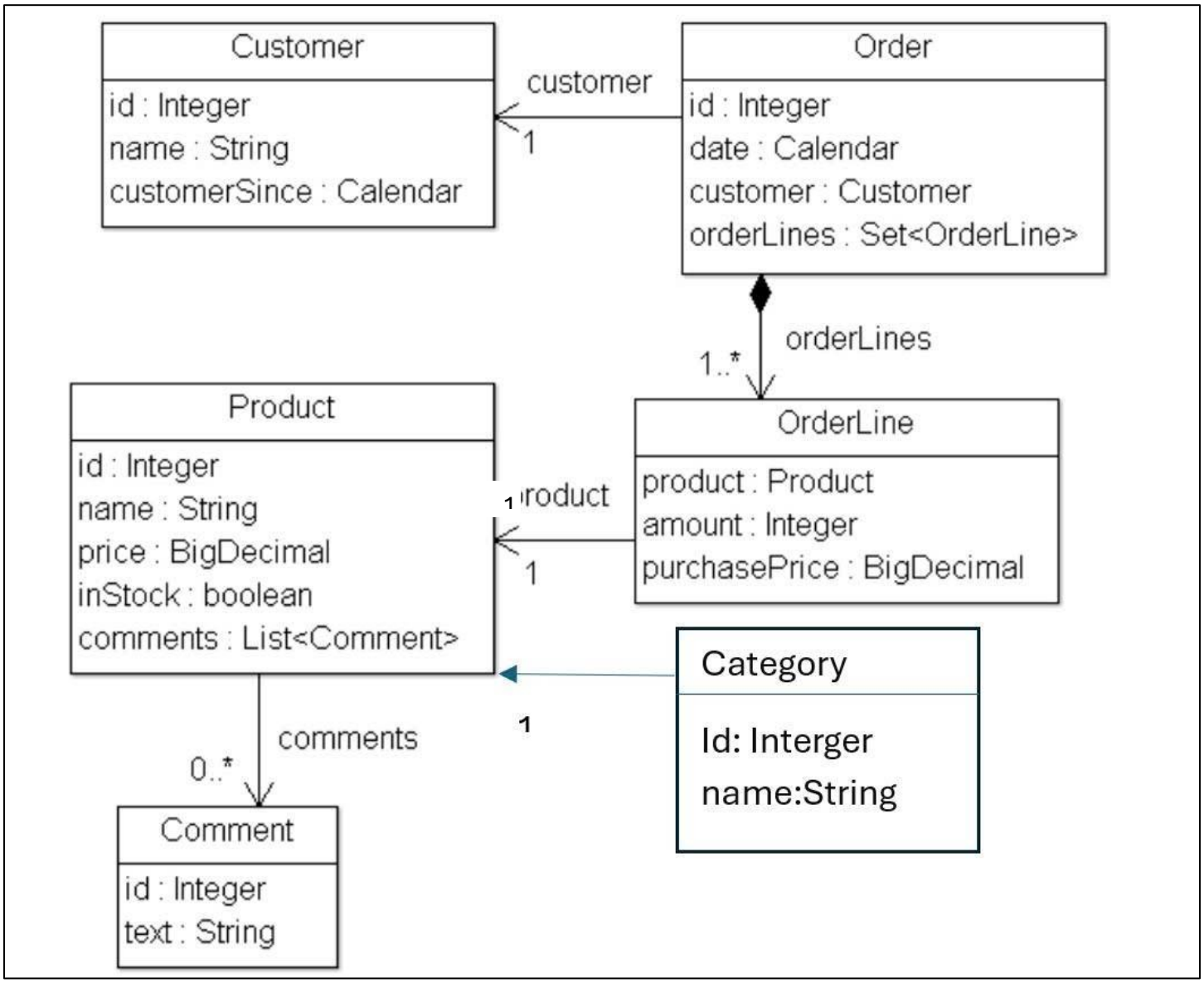
1. Database **shoppingdb** với các table customers, products, orders, users, orderlines.
   Tương ứng với Entities và các relationships được mô tả như sau:

- **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)

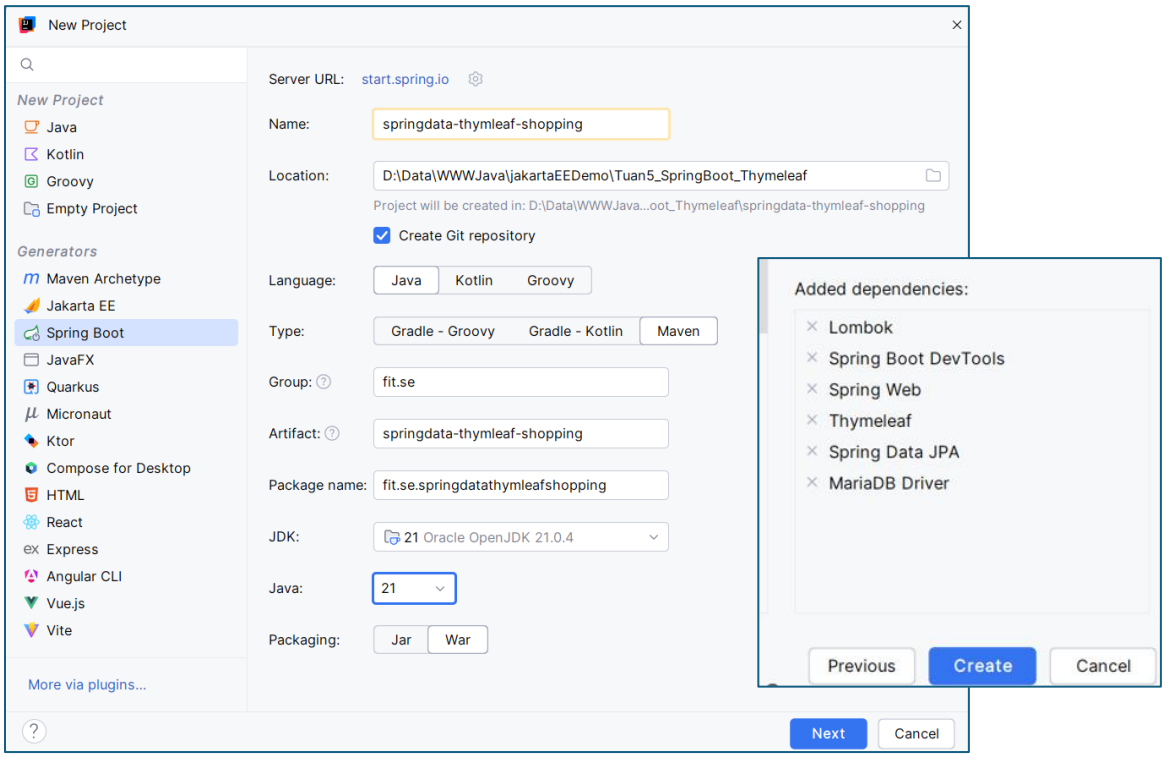
Lược đồ ERD như sau:



Đặc tả (BA)

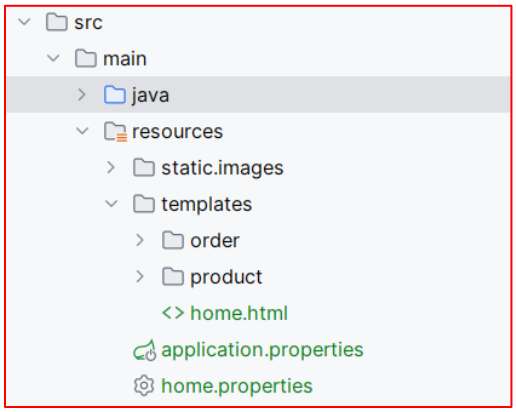
Actor: Admin, customer

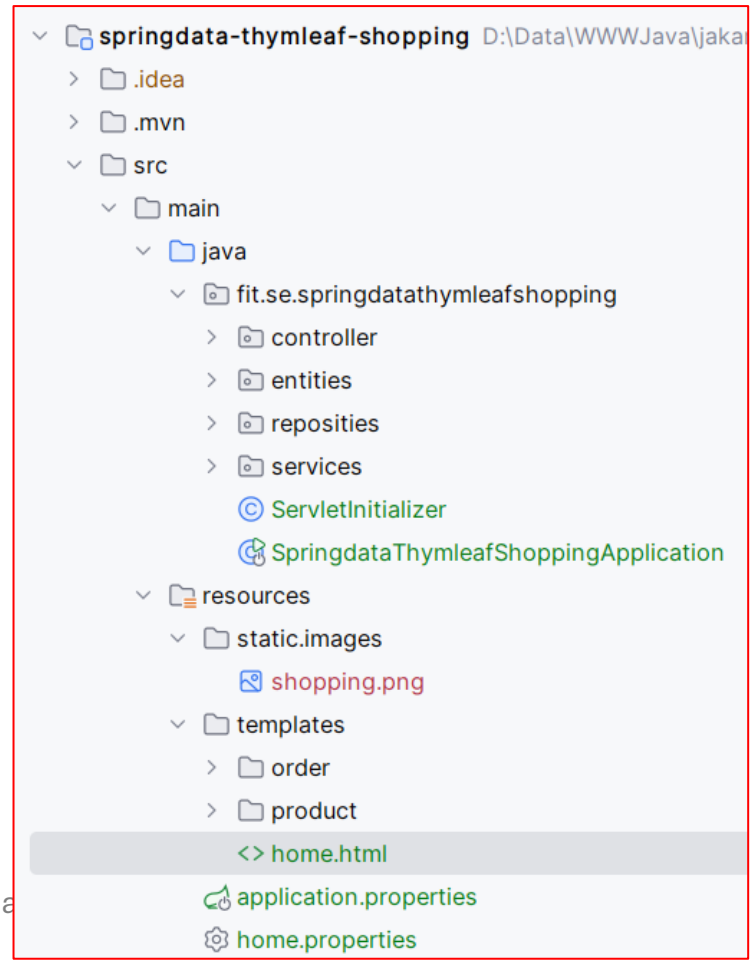
2. IDE InteliJ New Project Spring Boot:



Next → Chọn một số Dependencies cần thiết:

Cấu trúc Project như sau:

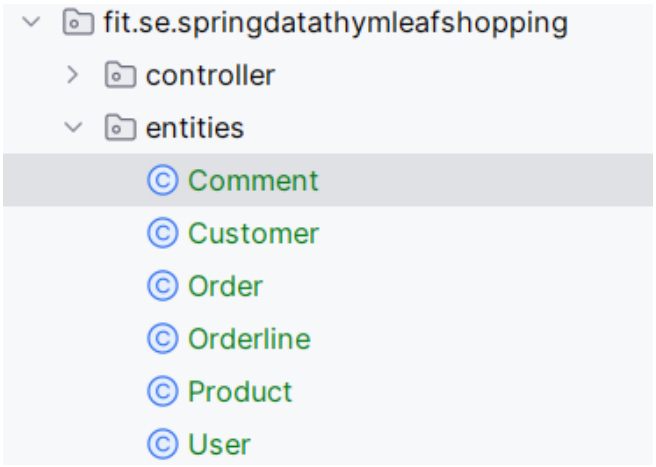




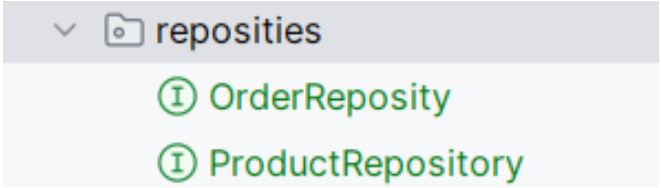
3. Config Spring Boot, mariadb, thymeleaf trong file **application.properties**

```
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.MariaD
BDialect

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

4. Tạo Entities

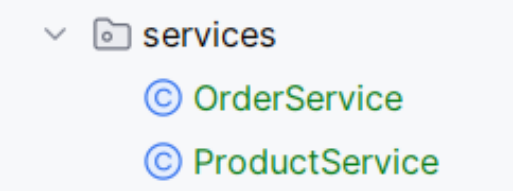


5. Tạo các repositories

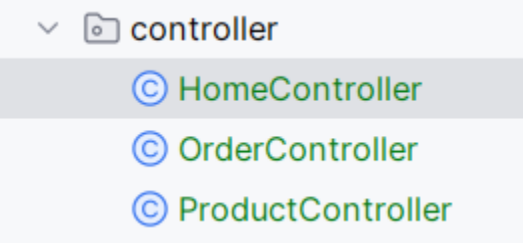


Các Repository extends JpaRepository (các phương thức truy cập theo Method Naming)

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.

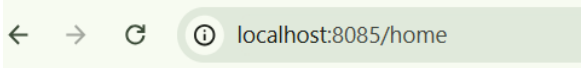
Truyền qua view html thông qua Model

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
```

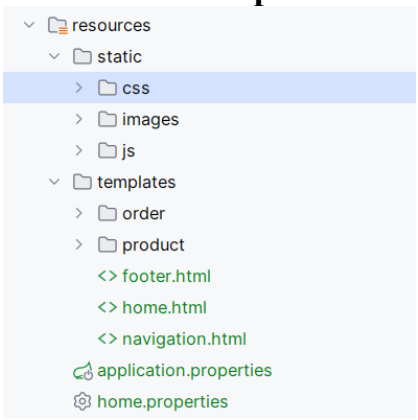
HomeController có annotation @Controller, @RequestMapping (“/home”): tương ứng với URI khi trình duyệt: 

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

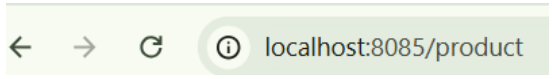


Trang templates\home.html sử dụng Thymeleaf template lấy attribute từ controller

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  <img src="../../images/shopping.png" th:src="@{/images/shopping.png}"/>
13
14  <h1 th:text="${message}">mess</h1>
15
16  <h2 th:text="#{home.welcome}">welcome</h2>
17
18  <p th:text="${date}"></p>
```

```
<p>Please select an option</p>
<ol>
    <li><a href="product/list.html" th:href="@{/product}">Product List</a></li>
    <li><a href="order/list.html" th:href="@{/order}">Order List</a></li>
</ol>
</body>
</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

```
package fit.se.springdatathymleafshopping.controller;


import fit.se.springdatathymleafshopping.entities.Product;
import fit.se.springdatathymleafshopping.services.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.ui.Model;


import java.util.List;

@Controller
@RequestMapping("/product")
public class ProductController {
    @Autowired
    private final ProductService productService;
    public ProductController(ProductService productService) {
        this.productService = productService;
    }
    @GetMapping
    public String showAllProducts(Model model) {
        List<Product> productlist = productService.findAll();
        model.addAttribute("products", productlist);
        return "product/list";
    }
    @GetMapping("/{id}")
    public String showProduct(@PathVariable int id, Model model) {
        Product product=productService.findById(id);
```

```
            model.addAttribute("product", product);
            return "product/productdetail";
        }
    }

```

Tương tự cho trang templates\product\list.html

list.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Product List</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }

        .product {
            border: 1px solid #ccc;
            padding: 15px;
            margin-bottom: 20px;
            border-radius: 8px;
        }

        .product h3 {
            margin: 0;
            color: #2a7;
        }

        .comments {
            margin-top: 10px;
            padding-left: 20px;
        }

        .comment {
            border-left: 3px solid #2a7;
            margin-bottom: 8px;
            padding-left: 10px;
            font-style: italic;
        }
    </style>
</head>
<body>
<h1>List Product</h1>

<div th:each="product : ${products}" class="product">
    <a href="product/productdetail.html" th:href="@{'/product/' + ${product.id}}">
        <input type="hidden" th:value="${product.id}" th:name="productId">
        <h2 th:text="${product.name}">Product Name</h2>
    </a>


    <p><strong>Price:</strong> <span th:text="${product.price}">0</span> USD</p>

    <div class="comments">
        <h3>Comments:</h3>
        <div th:if="${#lists.isEmpty(product.comments)}">
            <p><em>Nothing</em></p>
```

```
        </div>
        <div th:each="c : ${product.comments}" class="comment">
            <span th:text="${c.text}">Comment Content</span>
        </div>
    </div>
</div>

</body>
</html>
```

productdetail.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        .comments {
            margin-top: 10px;
            padding-left: 20px;
        }

        .comment {
            border-left: 3px solid #2a7;
            margin-bottom: 8px;
            padding-left: 10px;
            font-style: italic;
        }
    </style>
</head>
<body>
<h1>Product Detail</h1>
<p th:text="'Name: ' + ${product.name}"></p>
<p th:text="'Price: ' + ${product.price}"></p>
<div class="comments">
    <h3>Comments:</h3>
    <div th:if="${#lists.isEmpty(product.comments)}">
        <p><em>Nothing</em></p>
    </div>
    <div th:each="c : ${product.comments}" class="comment">
        <span th:text="${c.text}">Comment Content</span>
    </div>
</div>

</body>
</html>
```