

## MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- ✓ Xây dựng được các DAO làm việc với CSDL
- ✓ Xây dựng được các trang ứng dụng quản lý dữ liệu CRUD
- ✓ Sắp xếp dữ liệu truy vấn tùy biến theo tham số
- ✓ Truy vấn phân trang dữ liệu

## PHẦN I

### Bài 1 (2 điểm)

Xây dựng DAO làm việc với CSDL J5Shop.sql (tải về từ tài nguyên học liệu) và thực hiện theo hướng dẫn sau:

- Chạy WebShop.sql để tạo CSDL WebShop
- Khai báo bổ sung thư viện cần thiết cho dự án

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>com.microsoft.sqlserver</groupId>
  <artifactId>mssql-jdbc</artifactId>
  <scope>runtime</scope>
</dependency>
```

- Khai báo kết nối CSDL

```
spring.datasource.url=jdbc:sqlserver://localhost;database=WebShop
spring.datasource.username=sa
spring.datasource.password=*****
spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
spring.jpa.hibernate.dialect=org.hibernate.dialect.SQLServer2012Dialect
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto = none
```

- Tạo các lớp Entity ánh xạ với CSDL

○ Category

```
@Data
@Entity
@Table(name = "Categories")
public class Category implements Serializable{
    @Id
    String id;
    String name;
    @OneToMany(mappedBy = "category")
    List<Product> products;
}
```

○ Product

```
@Data
@Entity @Table(name = "Products")
public class Product implements Serializable{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Integer id;
    String name;
    String image;
    Double price;
    @Temporal(TemporalType.DATE)
    @Column(name = "Createdate")
    Date createDate = new Date();
    Boolean available;
    @ManyToOne @JoinColumn(name = "Categoryid")
    Category category;
    @OneToMany(mappedBy = "product")
    List<OrderDetail> orderDetails;
}
```

○ Account

```
@Data
@Entity
@Table(name = "Accounts")
public class Account implements Serializable{
```

```
@Id
String username;
String password;
String fullname;
String email;
String photo;
boolean activated;
boolean admin;
@OneToMany(mappedBy = "account")
List<Order> orders;
}
```

○ Order

```
@Data
@Entity
@Table(name = "Orders")
public class Order implements Serializable{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;
    String address;
    @Temporal(TemporalType.DATE)
    @Column(name = "Createdate")
    Date createDate = new Date();
    @ManyToOne @JoinColumn(name = "Username")
    Account account;
    @OneToMany(mappedBy = "order")
    List<OrderDetail> orderDetails;
}
```

○ OrderDetail

```
@Data
@Entity
@Table(name = "Orderdetails")
public class OrderDetail implements Serializable{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;
```

```
Double price;  
Integer quantity;  
@ManyToOne @JoinColumn(name = "Productid")  
Product product;  
@ManyToOne @JoinColumn(name = "Orderid")  
Order order;  
}
```

- Tạo các interface DAO để làm việc với CSDL

```
public interface CategoryDAO extends JpaRepository<Category, String>{}  
public interface ProductDAO extends JpaRepository<Product, Integer>{}  
public interface AccountDAO extends JpaRepository<Account, String>{}  
public interface OrderDAO extends JpaRepository<Order, Long>{}  
public interface OrderDetailDAO extends JpaRepository<OrderDetail, Long>{}  

```

## Bài 2 (2 điểm)

Xây dựng ứng dụng CRUD quản lý loại hàng (Category) như hình sau:

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/category/in...'. The page title is 'CATEGORY MANAGEMENT'. Below the title, there are two input fields: 'Category Id?' and 'Category Name?'. Below these fields are three buttons: 'Create', 'Update', and a link 'Delete Reset'. Below the buttons is a table with three columns: 'Id', 'Name', and an empty column. The table contains seven rows of data:

Id	Name	
1000	Đồng hồ đeo tay	<a href="#">Edit</a>
1001	Máy tính xách tay	<a href="#">Edit</a>
1002	Máy ảnh	<a href="#">Edit</a>
1003	Điện thoại	<a href="#">Edit</a>
1004	Nước hoa	<a href="#">Edit</a>
1005	Nữ trang	<a href="#">Edit</a>
1006	Nón thời trang	<a href="#">Edit</a>
1007	Túi xách du lịch	<a href="#">Edit</a>

Hướng dẫn:

Để xây dựng ứng dụng CRUD quản lý loại hàng như trên, cần tạo các thành phần sau:

### 1. CategoryController

Chuẩn bị dữ liệu để hiển thị lên giao diện và xử lý các hành động tương tác của người sử dụng.

Sau đây là địa chỉ URL (@/category/index) chuẩn bị dữ liệu

```
@Controller
public class CategoryController {
    @Autowired
    CategoryDAO dao; // làm việc với bảng Categories
```

```
@RequestMapping("/category/index")
public String index(Model model) {
    Category item = new Category();
    model.addAttribute("item", item);
    List<Category> items = dao.findAll();
    model.addAttribute("items", items);
    return "category/index";
}
```

Trong đó:

- Biến **item**: buộc lên form
- Biến **items**: hiển thị lên bảng

## 2. View index.html, \_form.html và \_table.html

- Index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Category Management</title>
</head>
<body>
    <h3>CATEGORY MANAGEMENT</h3>
    <th:block th:replace="@{/category/_form}" />
    <hr>
    <th:block th:replace="@{/category/_table}" />
</body>
</html>
```

- \_form.html

```
<form action="/category/index" th:object="${item}">
    <input th:field="*{id}" placeholder="Category Id?"/>
    <input th:field="*{name}" placeholder="Category Name?"/>
    <hr>
    <button formaction="/category/create">Create</button>
    <button formaction="/category/update">Update</button>
```

```
<a th:href="@{/category/delete/{item.id}}">Delete</a>
<a href="/category/index">Reset</a>
</form>
```

- `_table.html`

```
<table border="1" style="width:100%">
<tr>
  <th>Id</th>
  <th>Name</th>
  <th></th>
</tr>
<th:block th:each="item: ${items}">
  <tr>
    <td th:text="${item.id}"></td>
    <td th:text="${item.name}"></td>
    <td>
      <a th:href="@{/category/edit/{item.id}}">Edit</a>
    </td>
  </tr>
</th:block>
```

Tổng hợp các hoạt động trên các trang html tương tác với phía server gồm

```
<button formaction="/category/create">Create</button>
<button formaction="/category/update">Update</button>
<a th:href="@{/category/delete/{item.id}}">Delete</a>
<a href="/category/index">Reset</a>
<a th:href="@{/category/edit/{item.id}}">Edit</a>
```

### 3. Hoàn thiện CategoryController để xử lý các hành động Edit, Create, Update và Delete.

Bổ sung vào CategoryController các phương thức và ánh xạ với các địa chỉ URL có trên giao diện để xử lý các request của người sử dụng như sau

```
@RequestMapping("/category/edit/{id}")
public String edit(Model model, @PathVariable("id") String id) {
  Category item = dao.findById(id).get();
  model.addAttribute("item", item);
  List<Category> items = dao.findAll();
```

```
        model.addAttribute("items", items);
        return "category/index";
    }

    @RequestMapping("/category/create")
    public String create(Category item) {
        dao.save(item);
        return "redirect:/category/index";
    }

    @RequestMapping("/category/update")
    public String update(Category item) {
        dao.save(item);
        return "redirect:/category/edit/" + item.getId();
    }

    @RequestMapping("/category/delete/{id}")
    public String create(@PathVariable("id") String id) {
        dao.deleteById(id);
        return "redirect:/category/index";
    }
}
```

### Bài 3 (1 điểm)

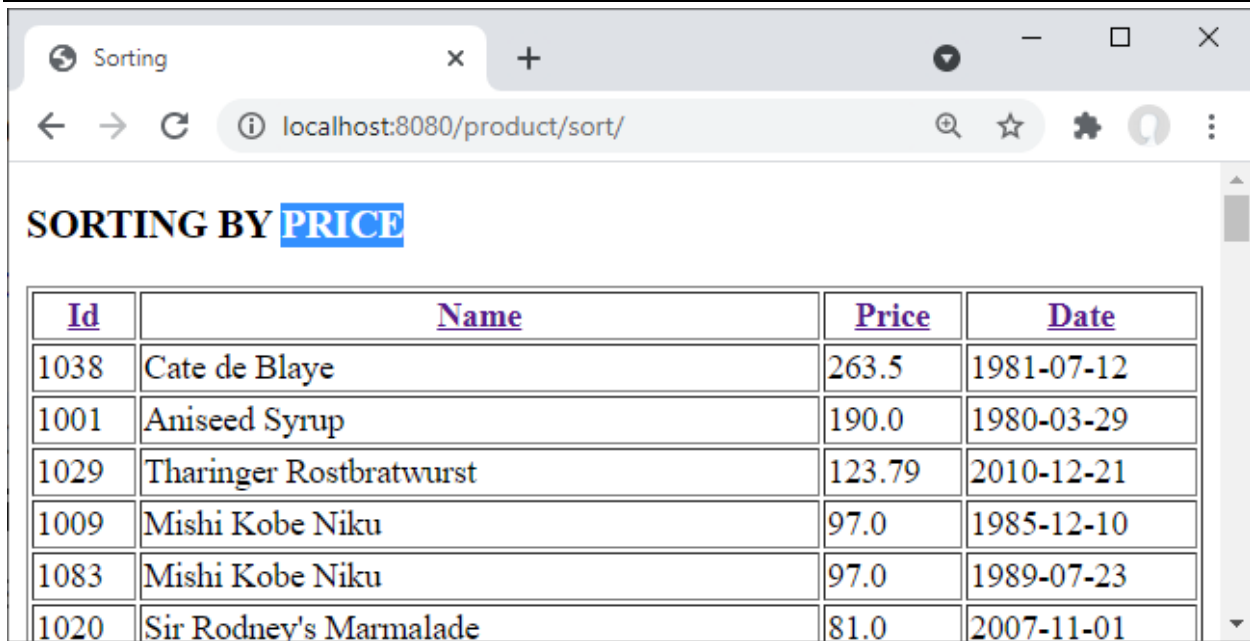
Giảng viên cho thêm quản lý CRUD của một table khác trong CSDL J5Shop

## PHẦN II

### Bài 3 (2 điểm)

Xây dựng trang web cho phép sắp xếp giảm dần theo cột được chọn trên header của table như hình sau, khởi đầu thì sắp theo giá.





<u><a href="#">Id</a></u>	<u><a href="#">Name</a></u>	<u><a href="#">Price</a></u>	<u><a href="#">Date</a></u>
1038	Cate de Blaye	263.5	1981-07-12
1001	Aniseed Syrup	190.0	1980-03-29
1029	Tharinger Rostbratwurst	123.79	2010-12-21
1009	Mishi Kobe Niku	97.0	1985-12-10
1083	Mishi Kobe Niku	97.0	1989-07-23
1020	Sir Rodney's Marmalade	81.0	2007-11-01

## 1. ProductController

```
@Controller
public class ProductController {
    @Autowired
    ProductDAO dao;

    @RequestMapping("/product/sort")
    public String sort(Model model) {
        List<Product> items = dao.findAll(); // truy vấn tất cả
        model.addAttribute("items", items);
        return "product/sort";
    }
}
```

## 2. View sort.html hiển thị các sản phẩm

```
<h3>SORTING BY <th:block th:text="{param.field}"/></h3>
<table border="1" style="width:100%">
<tr>
    <th><a href="/product/sort?field=id">Id</a></th>
    <th><a href="/product/sort?field=name">Name</a></th>
    <th><a href="/product/sort?field=price">Price</a></th>
    <th><a href="/product/sort?field=createDate">Date</a></th>
```

```
</tr>
<th:block th:each="item: ${items}">
    <tr>
        <td th:text="${item.id}"></td>
        <td th:text="${item.name}"></td>
        <td th:text="${item.price}"></td>
        <td th:text="${item.createDate}"></td>
    </tr>
</th:block>
</table>
```

Trang web trên đây chưa được sắp xếp, chỉ hiển thị tất cả. Mong muốn khi click vào các liên kết sau sẽ tiến hành sắp xếp theo giá trị của tham số **field**

```
<a href="/product/sort?field=id">Id</a>
<a href="/product/sort?field=name">Name</a>
<a href="/product/sort?field=price">Price</a>
<a href="/product/sort?field=createDate">Date</a>
```

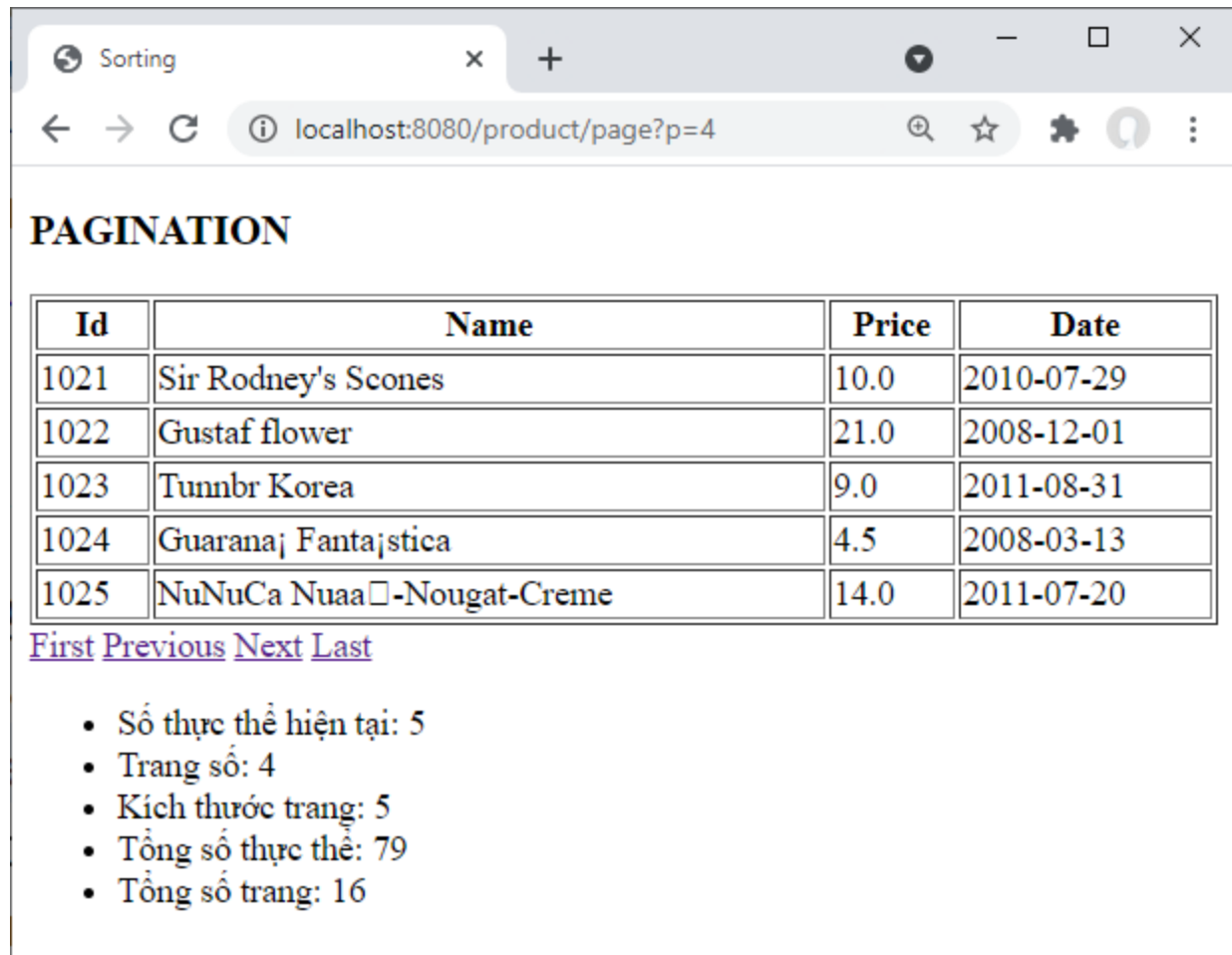
### 3. Bổ sung thêm mã sắp xếp

```
@RequestMapping("/product/sort")
public String sort(Model model,
    @RequestParam("field") Optional<String> field) {
    Sort sort = Sort.by(Direction.DESC, field.orElse("price"));
    model.addAttribute("field", field.orElse("price").toUpperCase());
    List<Product> items = dao.findAll(sort);
    model.addAttribute("items", items);
    return "product/sort";
}
```

- Optional là vì mới chạy chưa chọn cột, trong trường hợp này sẽ sắp xếp theo price.
- Chú ý: **findAll(sort)**

### Bài 4 (2 điểm)

Xây dựng trang web cho phép phân trang (mỗi trang 5 sản phẩm) như hình sau:



**PAGINATION**

Id	Name	Price	Date
1021	Sir Rodney's Scones	10.0	2010-07-29
1022	Gustaf flower	21.0	2008-12-01
1023	Tunnbr Korea	9.0	2011-08-31
1024	Guarana  Fanta stica	4.5	2008-03-13
1025	NuNuCa Nuaa□-Nougat-Creme	14.0	2011-07-20

[First](#) [Previous](#) [Next](#) [Last](#)

- Số thực thể hiện tại: 5
- Trang số: 4
- Kích thước trang: 5
- Tổng số thực thể: 79
- Tổng số trang: 16

## 1. ProductController

Mã của paginate() truy xuất các sản phẩm của trang thứ 3, mỗi trang 5 sản phẩm

```
@RequestMapping("/product/page")
public String paginate(Model model) {
    Pageable pageable = PageRequest.of(2, 5);
    Page<Product> page = dao.findAll(pageable);
    model.addAttribute("page", page);
    return "product/page";
}
```

## 2. View page.html hiển thị các sản phẩm của trang và thông tin phân trang

```
<h3>PAGINATION</h3>
<table border="1" style="width:100%">
<tr>
```

```

<th>Id</th>
<th>Name</th>
<th>Price</th>
<th>Date</th>
</tr>
<th:block th:each="item: ${page.content}">
    <tr>
        <td th:text="${item.id}"></td>
        <td th:text="${item.name}"></td>
        <td th:text="${item.price}"></td>
        <td th:text="${item.createDate}"></td>
    </tr>
</th:block>
</table>
<a th:href="@{/product/page?p=0}">First</a>
<a th:href="@{/product/page?p=${page.number - 1}}">Previous</a>
<a th:href="@{/product/page?p=${page.number + 1}}">Next</a>
<a th:href="@{/product/page?p=${page.totalPages - 1}}">Last</a>
<ul>
    <li>Số thực thể hiện tại: [[${page.numberOfElements}]]</li>
    <li>Trang số: [[${page.number}]]</li>
    <li>Kích thước trang: [[${page.size}]]</li>
    <li>Tổng số thực thể: [[${page.totalElements}]]</li>
    <li>Tổng số trang: [[${page.totalPages}]]</li>
</ul>

```

Hiệu chỉnh ProductController để khi click vào các liên kết điều hướng thì truy vấn trang theo tham số **p**

```

@RequestMapping("/product/page")
public String paginate(Model model,
    @RequestParam("p") Optional<Integer> p) {
    Pageable pageable = PageRequest.of(p.orElse(0), 5);
    Page<Product> page = dao.findAll(pageable);
    model.addAttribute("page", page);
    return "product/page";
}

```

**Bài 5 (1 điểm)**

Giảng viên cho thêm kết hợp phân trang và sắp xếp