

## MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng củng cố lại kiến thức

- ✓ Sử dụng @Query để thực hiện truy vấn tùy biến theo JPQL
- ✓ Truy vấn tổng hợp dữ liệu
- ✓ Kết hợp JPQL với sắp xếp và phân trang
- ✓ Sử dụng DSL để truy vấn dữ liệu, kết hợp với sắp xếp và phân trang

## PHẦN I

### Bài 1 (2 điểm)

Xây dựng trang tìm kiếm sản phẩm theo khoảng giá như hình sau

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/product/search'. The page title is 'SEARCH BY PRICE'. Below the title, there are two input fields for price range: the first contains '5' and the second contains '8'. To the right of these fields is a 'Search' button. Below the input fields is a table with the following data:

Id	Name	Price	Date
1013	Konbu	6.0	2002-07-01
1052	Filo Mix	7.0	2001-05-20
1054	Tourtiaire	7.45	2009-10-07
1075	RhanbrAu Klosterbier	7.75	1982-10-31

Để thực hiện theo yêu cầu trên, cần tạo và viết mã cho các thành phần cần thiết sau đây:

#### 1. ProductController

```
@RequestMapping("/product/search")
public String search(Model model,
    @RequestParam("min") Optional<Double> min,
```

```
@RequestParam("max") Optional<Double> max) {
    double minPrice = min.orElse(Double.MIN_VALUE);
    double maxPrice = max.orElse(Double.MAX_VALUE);
    List<Product> items = dao.findByPrice(minPrice, maxPrice);
    model.addAttribute("items", items);
    return "product/search";
}
```

- Nhận các tham số min và max
- Gọi phương thức findByPrice() của ProductDAO.

## 2. ProductDAO

```
public interface ProductDAO extends JpaRepository<Product, Integer>{
    @Query("FROM Product o WHERE o.price BETWEEN ?1 AND ?2")
    List<Product> findByPrice(double minPrice, double maxPrice);
}
```

- Sử dụng @Query(JPQL) để truy vấn dữ liệu
- Chú ý vị trí các tham số tương ứng với vị trí đối số của phương thức truy vấn

## 3. View search.html

```
<form action="/product/search" method="post">
    <input name="min" th:value="${param.min}" placeholder="Min Price?">
    <input name="max" th:value="${param.max}" placeholder="Max Price?">
    <button>Search</button>
</form>
...
<th:block var="item" items="${items}">
    <tr>
        <td th:text="${item.id}"></td>
        <td th:text="${item.name}"></td>
        <td th:text="${item.price}"></td>
        <td th:text="${item.createDate}"></td>
    </tr>
</th:block>
...
```

## Bài 2 (2 điểm)

Xây dựng trang web tìm kiếm sản phẩm theo tên và kết hợp với phân trang các sản phẩm như hình sau:

**SEARCH & PAGE**

an

Id	Name	Price	Date
1001	Aniseed Syrup	190.0	1980-03-29
1002	Change	19.0	1982-12-18
1003	Aniseed Syrup	10.0	1973-06-14
1004	Chef Anton's Cajun Seasoning	22.0	1976-03-10
1005	Chef Anton's Gumbo Mix	21.35	1978-12-06

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

- Số thực thể hiện tại: 5
- Trang số: 0
- Kích thước trang: 5
- Tổng số thực thể: 21
- Tổng số trang: 5

Để thực hiện yêu cầu trên, cần phải viết mã cho các thành phần sau đây:

### 1. ProductController

```
@Autowired
SessionService session;

@RequestMapping("/product/search-and-page")
public String searchAndPage(Model model,
    @RequestParam("keywords") Optional<String> kw,
    @RequestParam("p") Optional<Integer> p) {
```

```
String kwords = kw.orElse(session.get("keywords", ""));
session.set("keywords", kwords);
Pageable pageable = PageRequest.of(p.orElse(0), 5);
Page<Product> page = dao.findByKeywords("%"+kwords+"%", pageable);
model.addAttribute("page", page);
return "product/search-and-page";
}
```

- Nhận từ khóa tìm kiếm keywords và trang số p thông qua tham số
- Ghi nhớ từ khóa vào session để duy trì trong quá trình phân trang
- Gọi findByKeywords() của ProductDAO để truy vấn trang theo từ khóa

## 2. ProductDAO

```
public interface ProductDAO extends JpaRepository<Product, Integer>{
    @Query("FROM Product o WHERE o.name LIKE ?1")
    Page<Product> findByKeywords(String keywords, Pageable pageable);
}
```

- Chú ý: Pageable và Sort nên là đối số cuối cùng để tránh nhầm lẫn với các vị trí tham số trong JPQL

## 3. View search-and-page.html

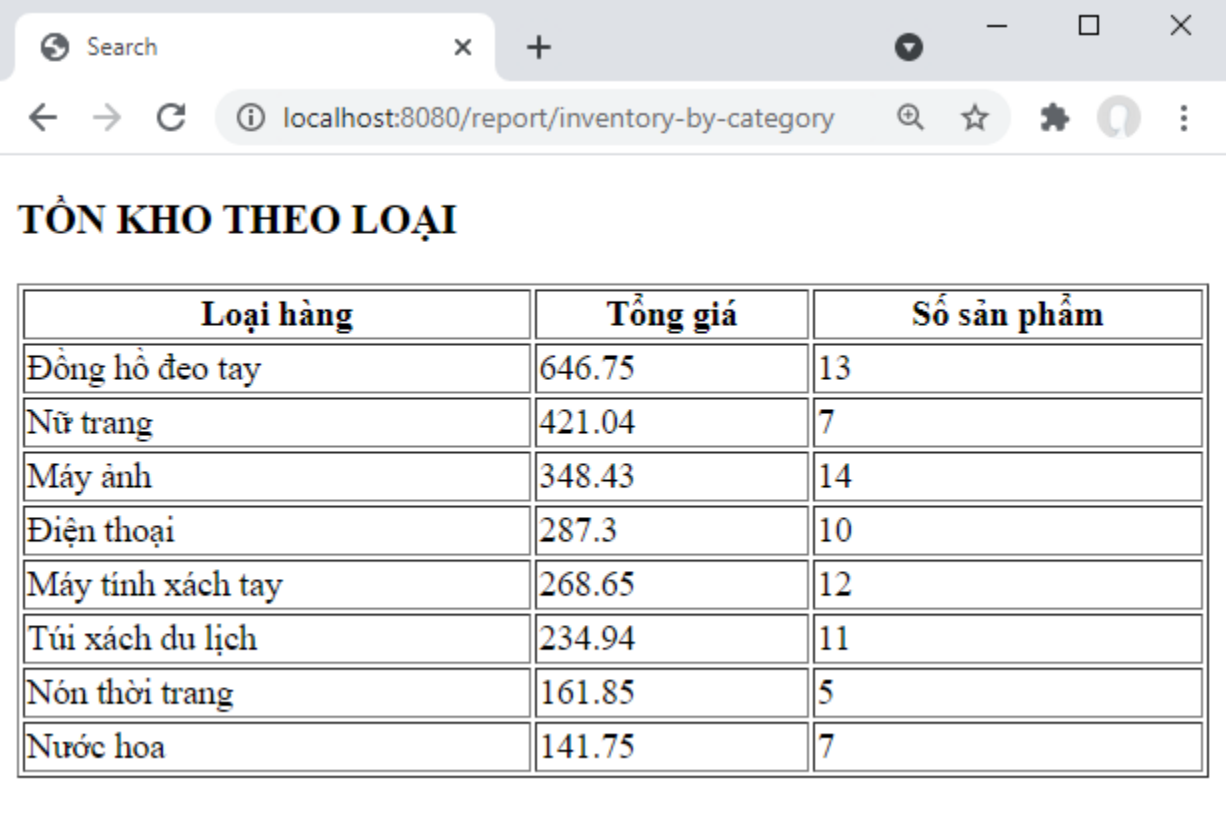
```
<form action="/product/search-and-page" method="post">
    <input name="keywords" th:value="${keywords}">
    <button>Search</button>
</form>
<table border="1" style="width:100%">
<th:block th:each="item: ${page.content}">
...
</th:block>
</table>
<a th:href="@{/product/search-and-page?p=0}">First</a>
<a th:href="@{/product/search-and-page?p=${page.number - 1}}">Previous</a>
<a th:href="@{/product/search-and-page?p=${page.number + 1}}">Next</a>
<a th:href="@{/product/search-and-page?p=${page.totalPages - 1}}">Last</a>
<ul>
    <li>Số thực thể hiện tại: [[${page.numberOfElements}]]</li>
</ul>
```

```
<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>
```

- Chú ý: các thuộc tính của Page giúp chúng ta triển khai việc phân trang một cách dễ dàng và thuận lợi.

### Bài 3 (1 điểm)

Xây dựng trang web cho phép thực hiện tổng hợp dữ liệu tồn kho như hình sau:



Loại hàng	Tổng giá	Số sản phẩm
Đồng hồ đeo tay	646.75	13
Nữ trang	421.04	7
Máy ảnh	348.43	14
Điện thoại	287.3	10
Máy tính xách tay	268.65	12
Túi xách du lịch	234.94	11
Nón thời trang	161.85	5
Nước hoa	141.75	7

Để thực hiện trang web theo yêu cầu thì cần viết mã cho các thành phần sau đây:

#### 1. ReportController

```
@RequestMapping("/report/inventory-by-category")
public String inventory(Model model) {
    List<Report> items = dao.getInventoryByCategory();
    model.addAttribute("items", items);
}
```

```
return "report/inventory-by-category";
}
```

- Gọi phương thức **getInventoryByCategory()** của ProductDAO để lấy dữ liệu tổng hợp
- Cũng cần tạo một entity class **Report** để chứa dữ liệu

## 2. ProductDAO

```
@Query("SELECT o.category AS group, sum(o.price) AS sum, count(o) AS count "
      + " FROM Product o "
      + " GROUP BY o.category"
      + " ORDER BY sum(o.price) DESC")
List<Report> getInventoryByCategory();
```

- Do Report không ánh xạ đến một table nào nên chúng ta cần phải new Report() và truyền các đối số phù hợp để tạo đối tượng từ dữ liệu truy vấn được.

## 3. Report

```
public interface Report {
    Serializable getGroup();
    Double getSum();
    Long getCount();
}
```

- Lưu ý: Để group by được thì biểu thức trong group by phải Serializable

## 4. View inventory-by-category.html

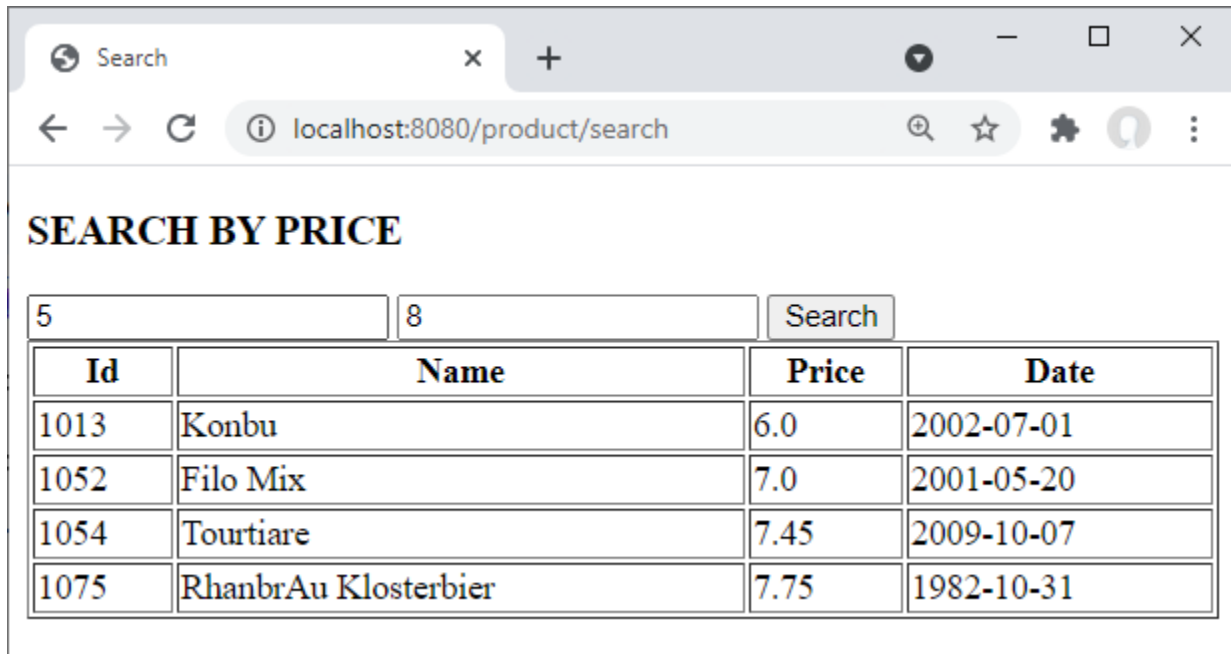
```
<table border="1" style="width:100%">
<tr>
    <th>Loại hàng</th>
    <th>Tổng giá</th>
    <th>Số sản phẩm</th>
</tr>
<th:block th:each="item, ${items}">
    <tr>
        <td th:text="${item.group.name}"></td>
        <td th:text="${item.sum}"></td>
        <td th:text="${item.count}"></td>
    </tr>
</th:block>
```

```
</th:block>
</table>
```

## PHẦN II

### Bài 4 (2 điểm)

Hãy xây dựng trang web tìm kiếm như bài 1 (hình sau) nhưng sử dụng DSL để viết phương thức truy vấn thay cho @Query(JPQL)



Id	Name	Price	Date
1013	Konbu	6.0	2002-07-01
1052	Filo Mix	7.0	2001-05-20
1054	Tourtiare	7.45	2009-10-07
1075	RhanbrAu Klosterbier	7.75	1982-10-31

Hướng dẫn: Trong ProductDAO thay đoạn mã sau

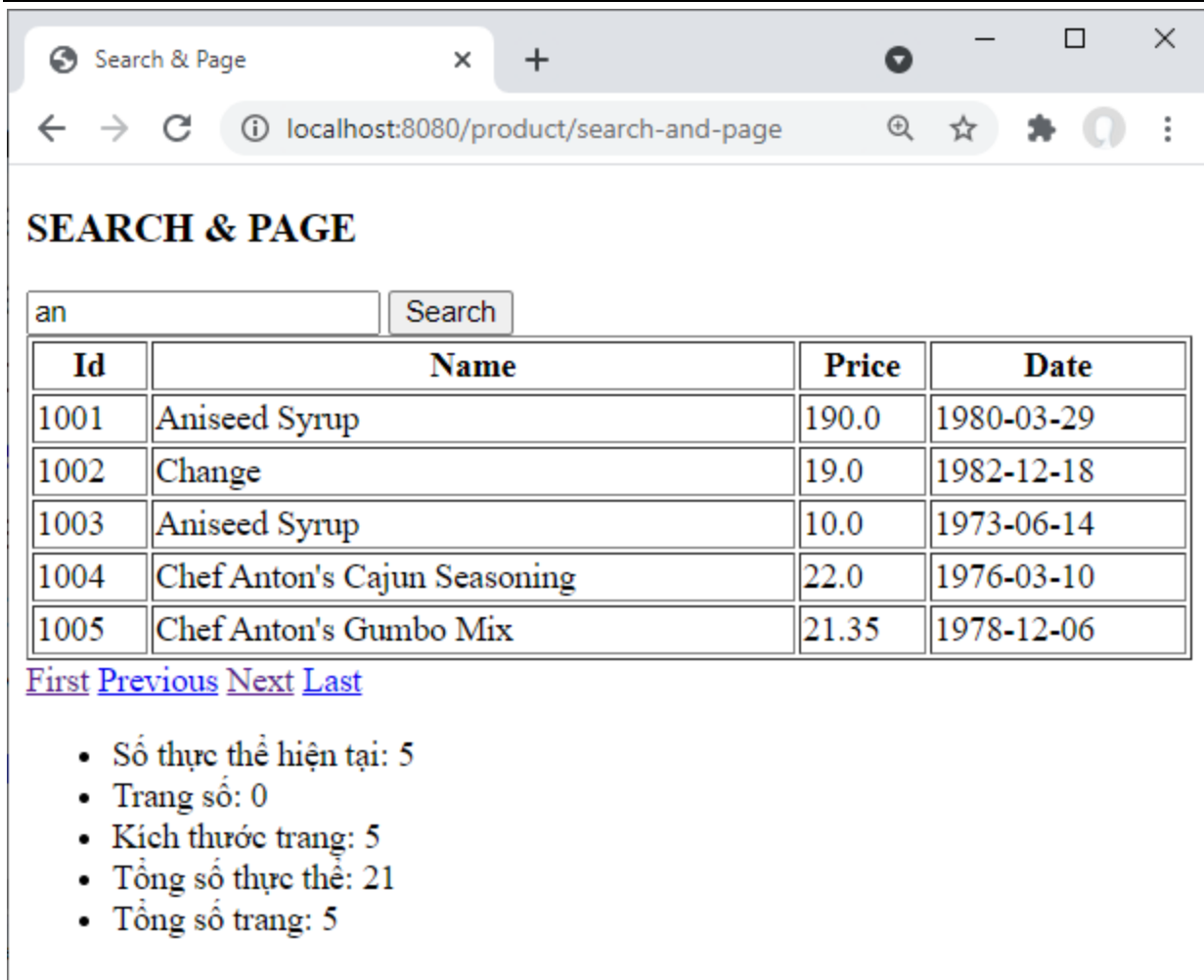
```
@Query("FROM Product o WHERE o.price BETWEEN ?1 AND ?2")
List<Product> findByPrice(double minPrice, double maxPrice);
```

Bằng đoạn mã sau

```
List<Product> findByPriceBetween(double minPrice, double maxPrice);
```

### Bài 5 (2 điểm)

Hãy xây dựng trang web tìm kiếm theo từ khóa kết hợp với phân trang sản phẩm như bài 2 (hình sau) nhưng sử dụng DSL để viết phương thức truy vấn thay cho @Query(JPQL)



**SEARCH & PAGE**

an Search

Id	Name	Price	Date
1001	Aniseed Syrup	190.0	1980-03-29
1002	Change	19.0	1982-12-18
1003	Aniseed Syrup	10.0	1973-06-14
1004	Chef Anton's Cajun Seasoning	22.0	1976-03-10
1005	Chef Anton's Gumbo Mix	21.35	1978-12-06

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

- Số thực thể hiện tại: 5
- Trang số: 0
- Kích thước trang: 5
- Tổng số thực thể: 21
- Tổng số trang: 5

Hướng dẫn: Trong ProductDAO thay đoạn mã sau

```
@Query("FROM Product o WHERE o.name LIKE ?1")
Page<Product> findByKeywords(String keywords, Pageable pageable);
```

Bằng đoạn mã sau

```
Page<Product> findAllByNameLike(String keywords, Pageable pageable);
```

Chú ý: với DSL, các phương thức truy vấn có phân trang và tìm kiếm nên sử dụng `findAllBy_` thay vì `findBy_`

## Bài 6 (1 điểm)

Giảng viên cho thêm