Phương thức countProduct(HttpServletRequest request, String searchQuery) này được sử dụng để đếm số lượng sản phẩm trong cơ sở dữ liệu dựa trên một chuỗi tìm kiếm searchQuery. Nó thực hiện các nhiệm vụ sau:

1. Kết nối đến cơ sở dữ liệu và tạo một PreparedStatement với câu lệnh SQL SELECT COUNT(\*) FROM product WHERE name LIKE ?. Câu lệnh này sẽ đếm số lượng bản ghi trong bảng product có tên chứa chuỗi tìm kiếm searchQuery.
2. Thực hiện truy vấn SQL với searchQuery được đặt vào tham số của PreparedStatement. Chuỗi tìm kiếm được bao quanh bởi % để tìm kiếm các chuỗi chứa searchQuery.
3. Lấy kết quả truy vấn và gán vào biến total, đại diện cho số lượng sản phẩm tìm thấy.
4. Tính toán số trang cần thiết để hiển thị tất cả sản phẩm tìm thấy, giả sử mỗi trang hiển thị 6 sản phẩm. Biến endPage sẽ chứa số trang cuối cùng cần hiển thị.
5. Đặt giá trị của endPage vào thuộc tính request.setAttribute("endPage", endPage) để có thể sử dụng trong JSP hoặc Servlet khác.

Phương thức này có thể được sử dụng trong một trang web bán hàng hoặc trang tìm kiếm sản phẩm. Nó giúp xác định số lượng sản phẩm tìm thấy dựa trên từ khóa tìm kiếm và tính toán số trang cần thiết để hiển thị tất cả kết quả. Thông tin này có thể được sử dụng để hiển thị phân trang hoặc thông báo số lượng kết quả tìm kiếm

Phương thức searchProduct(HttpServletRequest request, HttpServletResponse response) có chức năng tìm kiếm sản phẩm trong cơ sở dữ liệu dựa trên từ khóa tìm kiếm (searchQuery) và trả về danh sách các sản phẩm phù hợp. Phương thức này thực hiện các nhiệm vụ sau:

1. Khởi tạo một List rỗng productList để lưu trữ các sản phẩm tìm thấy.
2. Lấy searchQuery từ thuộc tính request.getAttribute("searchQuery").
3. Thêm ký tự % vào đầu và cuối searchQuery để tìm kiếm các sản phẩm chứa chuỗi này.
4. Gọi phương thức countProduct(request, searchQuery) để đếm số lượng sản phẩm phù hợp và tính toán số trang cần hiển thị.
5. Kết nối đến cơ sở dữ liệu và tạo một PreparedStatement với câu lệnh SQL SEARCH\_PRODUCT.
6. Gán searchQuery vào tham số thứ nhất của PreparedStatement.
7. Lấy tham số page từ request. Nếu không có tham số page, giá trị mặc định là 1.
8. Gán tham số thứ hai của PreparedStatement là giá trị (page - 1) \* 6, tương ứng với vị trí bản ghi đầu tiên cần hiển thị trên trang hiện tại (giả sử mỗi trang hiển thị 6 sản phẩm).
9. Thực hiện truy vấn SQL và lấy kết quả vào ResultSet.
10. Duyệt qua ResultSet và tạo đối tượng Product cho mỗi bản ghi. Thêm đối tượng Product vào productList.
11. Trả về productList chứa danh sách các sản phẩm phù hợp với searchQuery.

Phương thức này được sử dụng để tìm kiếm và hiển thị danh sách sản phẩm phù hợp với từ khóa tìm kiếm trên trang web. Nó kết hợp với phương thức countProduct để đếm số lượng sản phẩm và tính toán số trang cần hiển thị. Kết quả trả về là một danh sách các đối tượng Product có thể được sử dụng để hiển thị trong JSP hoặc xử lý tiếp theo

Phương thức getAllProduct(HttpServletRequest request, HttpServletResponse response) có chức năng lấy ra danh sách tất cả sản phẩm từ cơ sở dữ liệu. Nó thực hiện các bước như sau:

1. Đặt giá trị của thuộc tính searchQuery trong request thành một chuỗi rỗng "". Việc này có ý nghĩa là không có từ khóa tìm kiếm cụ thể, tương đương với yêu cầu lấy tất cả sản phẩm.
2. Gọi phương thức searchProduct(request, response). Đây có thể là phương thức đã được phân tích ở trước, chịu trách nhiệm tìm kiếm các sản phẩm dựa trên searchQuery và trả về danh sách các sản phẩm phù hợp. Vì searchQuery đã được đặt thành chuỗi rỗng "" ở bước 1, nên phương thức searchProduct sẽ tìm kiếm và trả về tất cả các sản phẩm trong cơ sở dữ liệu, thay vì chỉ tìm kiếm các sản phẩm phù hợp với một từ khóa cụ thể.
3. Kết quả trả về từ phương thức searchProduct(request, response) là một List<Product> chứa tất cả các sản phẩm được tìm thấy. Phương thức getAllProduct đơn giản trả về List<Product> này.

Tóm lại, phương thức getAllProduct có vai trò là gọi phương thức searchProduct với một searchQuery rỗng, từ đó lấy được danh sách tất cả các sản phẩm trong cơ sở dữ liệu. Nó cung cấp một cách đơn giản để lấy toàn bộ dữ liệu sản phẩm, có thể được sử dụng trong các trường hợp như hiển thị tất cả sản phẩm trên trang chủ, trang danh sách sản phẩm, hoặc các tính năng khác liên quan đến hiển thị toàn bộ dữ liệu sản phẩm

Phương thức filter(String filterType) này có chức năng lọc và trả về danh sách các sản phẩm trong cơ sở dữ liệu dựa trên loại (filterType) được chỉ định. Nó thực hiện các bước sau:

1. Khởi tạo một List<Product> rỗng productList để lưu trữ các sản phẩm sau khi lọc.
2. Kết nối đến cơ sở dữ liệu bằng phương thức getConnection().
3. Tạo một PreparedStatement với câu lệnh SQL FILTER\_PRODUCT. Câu lệnh SQL này có một tham số để lọc theo loại sản phẩm.
4. Gán giá trị filterType vào tham số thứ nhất của PreparedStatement.
5. Gọi phương thức executeQuery() để thực thi câu lệnh SQL và lấy ResultSet chứa kết quả truy vấn.
6. Duyệt qua ResultSet và tạo đối tượng Product cho mỗi bản ghi. Các thông tin như id, name, price, quantity, type, và image của sản phẩm được lấy từ ResultSet và sử dụng để khởi tạo đối tượng Product. Mỗi đối tượng Product được thêm vào productList.
7. Trả về productList chứa danh sách các sản phẩm phù hợp với filterType.

Ví dụ, nếu FILTER\_PRODUCT là câu lệnh SQL sau:

SELECT \* FROM product WHERE type = ?

Và filterType có giá trị "Laptop", thì phương thức filter("Laptop") sẽ trả về danh sách tất cả các sản phẩm có loại là "Laptop" trong cơ sở dữ liệu.

Lưu ý rằng phương thức filter có thể ném ra hai ngoại lệ:

* SQLException: Được ném ra khi có lỗi xảy ra trong quá trình truy vấn cơ sở dữ liệu.
* NullPointerException: Được ném ra nếu filterType là null, vì phương thức sẽ thất bại khi gán giá trị null cho tham số của PreparedStatement.

Phương thức này có thể được sử dụng trong các trang web bán hàng hoặc ứng dụng quản lý sản phẩm để lọc và hiển thị danh sách sản phẩm theo loại cụ thể. Nó cung cấp một cách để truy xuất dữ liệu sản phẩm dựa trên điều kiện lọc, làm tăng khả năng tùy chỉnh và trải nghiệm người dùng