**Giải thích chi tiết đoạn mã**

Đoạn mã này là một controller trong ứng dụng Node.js dùng để quản lý việc hiển thị danh sách sản phẩm, chi tiết sản phẩm và xử lý dữ liệu liên quan đến danh mục, thương hiệu, và thẻ (tags).

**1. Khởi tạo Controller**

let controller = {};

* **controller** là một đối tượng chứa các phương thức xử lý logic liên quan đến yêu cầu HTTP.

**2. Hàm getData**

**Mục đích**

* Lấy dữ liệu liên quan đến danh mục (**categories**), thương hiệu (**brands**), và thẻ (**tags**) từ cơ sở dữ liệu.
* Gắn dữ liệu vào **res.locals** để sử dụng trong view.

**Phân tích**

let categories = await models.Category.findAll({

include: [{

model: models.Product

}]

});

res.locals.categories = categories;

* **models.Category.findAll**:
  + Lấy danh sách danh mục từ bảng Category và bao gồm cả danh sách sản phẩm liên quan (include: models.Product).
* **res.locals.categories**:
  + Gắn dữ liệu danh mục vào biến cục bộ res.locals để sử dụng trong các view.

Tương tự, logic lấy danh sách **brands** và **tags** cũng hoạt động giống như trên.

**3. Hàm show**

**Mục đích**

* Hiển thị danh sách sản phẩm dựa trên các tiêu chí lọc như danh mục, thương hiệu, thẻ, từ khóa, và sắp xếp.
* Xử lý phân trang.

**Phân tích**

**Xây dựng tùy chọn truy vấn (options)**

let options = {

attributes: ['id', 'name', 'imagePath', 'stars', 'price', 'oldPrice'],

where: {}

};

* **attributes**: Chỉ lấy các trường cần thiết từ bảng Product.
* **where**: Điều kiện lọc sẽ được thêm dựa trên các tham số truy vấn (**category**, **brand**, **tag**, **keyword**).

**Điều kiện lọc**

if (category > 0) {

options.where.categoryId = category;

}

if (brand > 0) {

options.where.brandId = brand;

}

if (tag > 0) {

options.include = [{

model: models.Tag,

where: { id: tag }

}];

}

if (keyword.trim() != '') {

options.where.name = {

[Op.iLike]: `%${keyword}%`

};

}

* **Danh mục và thương hiệu**:
  + Thêm điều kiện vào where nếu người dùng chọn một danh mục hoặc thương hiệu cụ thể.
* **Thẻ**:
  + Sử dụng include để lọc sản phẩm có liên quan đến thẻ cụ thể.
* **Từ khóa**:
  + Sử dụng Op.iLike để tìm kiếm sản phẩm dựa trên từ khóa (không phân biệt chữ hoa/chữ thường).

**Sắp xếp sản phẩm**

switch(sort) {

case 'newest':

options.order = [['createdAt', 'DESC']];

break;

case 'popular':

options.order = [['stars', 'DESC']];

break;

default:

options.order = [['price', 'ASC']];

break;

}

* Sắp xếp theo:
  + **Mới nhất**: createdAt giảm dần.
  + **Phổ biến nhất**: stars giảm dần.
  + **Giá thấp nhất** (mặc định): price tăng dần.

**Phân trang**

const limit = 6;

options.offset = limit \* (page - 1);

let { rows, count } = await models.Product.findAndCountAll(options);

* **limit**: Số sản phẩm hiển thị trên mỗi trang.
* **offset**: Xác định vị trí bắt đầu lấy dữ liệu dựa trên trang hiện tại.
* **findAndCountAll**:
  + Lấy danh sách sản phẩm (rows) và tổng số lượng sản phẩm (count).

**Dữ liệu phân trang**

res.locals.pagination = {

page: page,

limit: limit,

totalRows: count,

queryParams: req.query

};

* Lưu thông tin cần thiết để hiển thị phân trang.

**Hiển thị view**

res.locals.products = rows;

res.render('product-list');

* Truyền danh sách sản phẩm vào view **product-list**.

**4. Hàm showDetails**

**Mục đích**

* Hiển thị thông tin chi tiết của một sản phẩm.
* Lấy danh sách sản phẩm liên quan dựa trên thẻ.

**Phân tích**

let product = await models.Product.findOne({

attributes: ['id', 'name', 'stars', 'oldPrice', 'price', 'summary', 'description', 'specification'],

where: { id },

include: [

{ model: models.Image, attributes: ['name', 'imagePath'] },

{

model: models.Review,

attributes: ['id', 'review', 'stars', 'createdAt'],

include: [{ model: models.User, attributes: ['firstName', 'lastName'] }]

},

{ model: models.Tag, attributes: ['id'] }

]

});

res.locals.product = product;

* Lấy thông tin sản phẩm chi tiết từ bảng Product.
* Bao gồm:
  + **Hình ảnh**: Từ bảng Image.
  + **Đánh giá**: Từ bảng Review và thông tin người dùng liên quan (User).
  + **Thẻ**: Từ bảng Tag.

**Sản phẩm liên quan**

let tagIds = [];

product.Tags.forEach(tag => tagIds.push(tag.id));

let relatedProducts = await models.Product.findAll({

attributes: ['id', 'name', 'imagePath', 'oldPrice', 'price'],

include: [{

model: models.Tag,

attributes: ['id'],

where: { id: { [Op.in]: tagIds } }

}],

limit: 10

});

res.locals.relatedProducts = relatedProducts;

* Dựa trên các thẻ của sản phẩm hiện tại (tagIds), lấy danh sách sản phẩm liên quan.
* **Op.in**: Kiểm tra nếu id của thẻ nằm trong danh sách tagIds.

**Hiển thị view**

res.render('product-detail');

* Truyền dữ liệu vào view **product-detail**.

**5. Hàm removeParam**

**Mục đích**

* Loại bỏ một tham số truy vấn ra khỏi URL.

**Phân tích**

function removeParam(key, sourceURL) {

var rtn = sourceURL.split("?")[0],

param,

params\_arr = [],

queryString = (sourceURL.indexOf("?") !== -1) ? sourceURL.split("?")[1] : "";

if (queryString !== "") {

params\_arr = queryString.split("&");

for (var i = params\_arr.length - 1; i >= 0; i--) {

param = params\_arr[i].split("=")[0];

if (param === key) {

params\_arr.splice(i, 1);

}

}

if (params\_arr.length) rtn = rtn + "?" + params\_arr.join("&");

}

return rtn;

}

* Chia URL thành phần path và query string.
* Xóa tham số cần loại bỏ ra khỏi query string.
* Kết hợp lại thành URL mới.

**6. Xuất Controller**

module.exports = controller;

* Xuất controller để sử dụng trong các route.