Spring Boot MVC - EZ Store Notes

Overview:

This Spring Boot application serves as a backend for an e-commerce store (EZ Store), managing product data and providing RESTful APIs for operations such as retrieving, adding, updating, and deleting products. It also supports product image handling and search functionality.

Key Components:

Model: Product

- Represents a product entity stored in a PostgreSQL database.

- Annotations:

- @Entity: Marks the class as a JPA entity.

- @Id, @GeneratedValue: Defines the primary key with auto-increment.

- @Lob: Used for storing large binary data (product images).

- @JsonFormat: Specifies the format for the releaseDate field in JSON responses.

- Fields:

id, name, description, brand, price, category, releaseDate, productAvailable, stockQuantity, imageName, imageType, imageData.

Repository: ProductRepo

- Extends JpaRepository to provide CRUD operations.

- Custom Query: Searches across multiple fields (name, description,

brand, category) using a keyword.

Service: ProductService

- Handles business logic for the product.

- Methods:

- getAllProducts(), getProductById(), addProduct(), updateProduct(), deleteProduct(), searchProducts().

Controller: ProductController

- Exposes REST endpoints for product operations.

- Endpoints:

- GET /products, GET /product/{id}, GET /product/{productId}/image

- POST /product, PUT /product/{productId}, DELETE /product/{productId}

- GET /products/search?keyword={keyword}

Database Configuration:

Configured for PostgreSQL in application.properties:

- Database URL: jdbc:postgresql://localhost:5432/ECOM

- Username/Password: pratik_joshi/root

- Driver: org.postgresql.Driver

- JPA Settings: ddl-auto=update, database-platform=PostgreSQLDialect.

Features:

1. CRUD Operations: Create, Read, Update, and Delete products.

2. Product Images: Handle image uploads and retrieval using

MultipartFile.

3. Search Functionality: Supports case-insensitive searches across product attributes.

4. JSON Format: Ensures dates are formatted consistently in responses.

5. Cross-Origin Support: Allows access from front-end clients hosted on different domains.

Key Dependencies:

- Spring Boot Starter Dependencies: spring-boot-starter-web, spring-boot-starter-data-jpa.
- Database: PostgreSQL JDBC Driver.
- Lombok: Reduces boilerplate with annotations (@Data, @AllArgsConstructor, @NoArgsConstructor).
- Jackson: JSON serialization and deserialization.

Suggestions for Improvements:

1. Validation: Use @Valid annotations and validation constraints (@NotNull, @Size) on the Product fields.
2. Exception Handling: Add a @ControllerAdvice for centralized exception handling.
3. Pagination: Implement pagination for the getAllProducts and searchProducts endpoints.
4. Security: Integrate Spring Security to secure endpoints.
5. Testing: Add unit and integration tests for services and controllers.