Simplified Inventory Management System Implement Using JWT Token.

Type: Projects

Skill: Java

JWT AuthenticationRole-based Access ControlSpring Boot DevelopmentUser Registration

Medium

You are tasked with developing a Spring Boot-based web application for a Simplified Inventory Management System that uses JSON Web Token (JWT) authentication for secure access. The system will allow users to register, log in, view available products, add new products, update product details, and manage product stock levels. Only authenticated users can perform certain actions, and only authorized users (admins) can add or update products.

**Functional Requirements:**

1. User Registration and Authentication

* Endpoint: /users/register
* Method: @PostMapping
* Description: This endpoint allows new users to register by providing their username, email, password, and role.
* The password should be encrypted using BCryptPasswordEncoder.
* The user can have roles such as ADMIN or USER.
* Input: JSON request body with the following fields:
* username: Unique identifier for the user
* email: The user's email address
* password: The user's password
* role: The role of the user (either "ADMIN" or "USER")
* Output: A confirmation message with the saved user object, including a unique user ID.

2. User Login (Authentication)

* Endpoint: /users/login
* Method: @PostMapping
* Description: This endpoint allows users to log in by providing their username and password.
* On successful login, generate and return a JWT token to be used for subsequent requests.
* If login fails, return HTTP status 401 (Unauthorized).
* Input: JSON request body with the following fields:
* username: The username of the user
* password: The password of the user
* Output:
* A token field containing the generated JWT token.
* 401 Unauthorized if login fails.

You can use already provided dto classes for user registration and login.

Please find provided classes in the following package: src/main/java/com/wecp/inventory\_management\_system\_jwt/dto/

3. View Products (For All Authenticated Users)

* Endpoint: /products
* Method: @GetMapping
* Description: This endpoint allows users to view a list of all available products in the inventory.
* This endpoint is accessible to all authenticated users, but users must provide a valid JWT token in the request header.
* Returns product details like product name, description, price, and stock level.
* Input: No input required; just the JWT token in the Authorization header.
* Output: A list of available products with details (name, description, price, stock).
* 401 Unauthorized if the user does not provide a valid JWT token.

4. Add New Product (Admin Only)

* Endpoint: /products
* Method: @PostMapping
* Description: This endpoint allows only admin users to add new products to the inventory.
* Requires the user to be authenticated and have the ADMIN role.
* Adds a new product with details such as name, description, price, and initial stock quantity.
* Input: JSON request body with the following fields:
* name: The name of the product
* description: A brief description of the product
* price: The price of the product
* stock: The initial stock level of the product
* Output: The created product details along with a success message.
* 403 Forbidden if a non-admin user attempts to add a product.

5. Update Product Details (Admin Only)

* Endpoint: /products/{productId}
* Method: @PutMapping
* Description: This endpoint allows admin users to update product details, such as price or stock levels.
* Requires the user to be authenticated and have the ADMIN role.
* Admin can update the price, description, and stock level of a product.
* Input: JSON request body with the fields to be updated:
* price: The updated price of the product (optional)
* stock: The updated stock level of the product (optional)
* description: The updated description (optional)
* name: The updated name of the product (optional)
* Output: The updated product details along with a success message.
* 403 Forbidden if a non-admin user attempts to update the product.

6. Delete Product (Admin Only)

* Endpoint: /products/{productId}
* Method: @DeleteMapping
* Description: This endpoint allows admin users to delete a product from the inventory.
* Requires the user to be authenticated and have the ADMIN role.
* Input: No input required except the productId as a path parameter.
* Output: A success message confirming the deletion of the product.
* 403 Forbidden if a non-admin user attempts to delete a product.

7. View Product Details

* Endpoint: /products/{productId}
* Method: @GetMapping
* Description: This endpoint allows users to view detailed information about a specific product.
* Accessible to all authenticated users with a valid JWT token.
* Input: Path parameter:
* productId: The ID of the product whose details are to be viewed.
* Output: Product details (name, description, price, stock).
* 401 Unauthorized if the user does not provide a valid JWT token.

**Entity Design:**

User Entity

Attributes:

* id: Long (auto-generated unique ID)
* username: String (unique username)
* email: String (user's email address)
* password: String (user's encrypted password)
* role: String (either "ADMIN" or "USER")

Relationships:

* Many-to-many relationship with products (if needed, such as products viewed by users).

Product Entity

Attributes:

* id: Long (auto-generated unique ID)
* name: String (name of the product)
* description: String (description of the product)
* price: double (price of the product)
* stock: int (current stock level)

Relationships:

* Many-to-many relationship with users (if needed, such as products purchased by users).

**Technical Notes:**

Checkin Permission: Check permission w.r.t authority of the user.

* Use hasAuthority() method to check the role of the user. For example hasAuthority("ADMIN") or hasAuthority("USER").

Password Encoding:

* Use BCryptPasswordEncoder to securely encode the user's password during registration and store it in the database.

JWT Token Generation:

* After a successful login, generate a JWT token using a library like jjwt and return it to the user. The token should be used for subsequent API requests.

JWT Token Validation:

* Secure endpoints such as product addition, update, and deletion with JWT validation to ensure that only authenticated users (with valid JWT tokens) can access these resources.

Roles & Permissions:

* Use role-based access control where only ADMIN users can manage products (create, update, delete), while USER-role users can only view products.

Error Handling:

* Return appropriate HTTP status codes:
* 401 Unauthorized for failed authentication or invalid JWT tokens.
* 403 Forbidden for unauthorized attempts to perform actions restricted to admins.
* 404 Not Found for non-existing products.
* 400 Bad Request for invalid input data.

**Note:**

* You need to write API in a.http file to use it via Restclient.