Contact List Management Project - Simplified Overview

Overview:

The Contact List Management project is a Spring Boot-based web application that allows users to create, update, delete, and retrieve contact details. It is built with Spring Boot 3.x, leveraging Spring Data JPA to interact with a MySQL database.

Key Features:

1. Spring Boot Project Setup:

- The project is set up using Spring Boot 3.x, with Jakarta EE namespaces (jakarta.persistence) for persistence-related operations.
- Dependencies: Spring Data JPA, MySQL connector, Spring Boot Starter Web, Lombok, Spring Boot Starter Validation.

2. Entity Definition (Contact):

- Created a Contact entity to represent the contact information, including fields like id, name, email, phoneNumber, and address. Mapped using JPA annotations to interact with the database.

3. Repository Layer:

- A JPA repository was created for the Contact entity, providing CRUD operations through ContactRepository that extends JpaRepository<Contact, Long>.

4. Service Layer:

- Developed a service layer to encapsulate the business logic, including handling contact CRUD operations, data validation, and interaction with the repository.
- Includes additional methods to apply business rules to the contact data.

5. Controller Layer (REST API):

- Developed RESTful endpoints using @RestController to manage contact details. These endpoints include:

- GET /contacts: Fetch all contacts.

- POST /contacts: Create a new contact.

- PUT /contacts/{id}: Update a contact.

- DELETE /contacts/{id}: Delete a contact.

6. Form Validation:

- Applied validation for fields such as email, phoneNumber, and name. Annotations like @NotNull, @Email, and @Size ensure that the input data is valid.

- Error handling was implemented to return meaningful error messages for invalid data.

7.Global Exception Handling:

- Implemented global exception handling using @ControllerAdvice and @ExceptionHandler to handle errors gracefully and return standardized error responses.

8.Swagger Integration:

- Swagger/OpenAPI documentation was added to provide an interactive API documentation interface, making it easier to test and visualize API endpoints.

- http://localhost:8787/swagger-ui/index.html