Spring Boot-based web application for a simplified Employee Management System

Type: Projects

Skill: Java

Data ValidationError HandlingRESTful ServicesSpring BootSpring Data JPA

Easy

Develop a Spring Boot-based web application for a simplified Employee Management System that allows users to perform CRUD (Create, Read, Update, Delete) operations on employee records.

Implement RESTful endpoints to manage employee data using Spring Boot's powerful capabilities.

**Functional Requirements:**

* **Employee Entity:**
* Create an Employee class with attributes such as id, firstName, lastName, email, and salary.
* Use Spring Data JPA to persist and retrieve employee records.
* **Employee Retrieval:**
* Implement a RESTful endpoint (/employees) with a @GetMapping annotation to retrieve a list of all existing employees.
* **Individual Employee Retrieval:**
* Extend the functionality to retrieve details of a specific employee.Implement a RESTful endpoint (/employees/{employeeId}) with a @GetMapping annotation to retrieve details of a specific employee identified by their employeeId.
* **Employee Creation:**
* Allow users to create new employee records. Implement a RESTful endpoint (/employees) with a @PostMapping annotation to handle the creation of a new employee. The creation request should include the employee's firstName, lastName, email, and salary.
* **Employee Update:**
* Allow users to update existing employee records.Implement a RESTful endpoint (/employees/{employeeId}) with a @PutMapping annotation to handle the update of an existing employee record. The update request should include the updated information for the employee.
* **Employee Deletion:**
* Allow users to delete existing employee records. Implement a RESTful endpoint (/employees/{employeeId}) with a @DeleteMapping annotation to handle the deletion of an existing employee record.
* **Input:**
* **Employee Creation:** Input fields for firstName, lastName, email, and salary.
* **Employee Update:** Input fields for updating firstName, lastName, email, and salary.
* **Output:**
* **Employee Retrieval:** Display a list of all existing employees.
* **Individual Employee Retrieval:** Display the details of a specific employee.
* **Employee Creation:**
* Created Employee with Confirmation message upon successful creation of a new employee. Error message if there are issues with the employee creation process.
* **Employee Update:**
* Updated Employee with Confirmation message upon successful update of an existing employee. Error message if there are issues with the employee update process.
* **Employee Deletion:**
* Confirmation message upon successful deletion of an existing employee. Error message if there are issues with the employee deletion process.

**Notes:**

* Implement proper error handling and validation to ensure data integrity and user-friendly interactions.
* You need to throw EmployeeNotFoundException if the employee with the given id is not found for retrieval, update, or deletion operations.

**You need to complete the following files:**

* *./src/main/java/com/wecp/simplified\_emp\_management\_system/model/Employee.java*
* *./src/main/java/com/wecp/simplified\_emp\_management\_system/controller/EmployeeController.java*
* *./src/main/java/com/wecp/simplified\_emp\_management\_system/exception/EmployeeNotFoundException.java*
* *./src/main/java/com/wecp/simplified\_emp\_management\_system/repository/EmployeeRepository.java*
* *./src/main/java/com/wecp/simplified\_emp\_management\_system/service/EmployeeServiceImpl.java*