**Exercise 7: Employee Management System – Enabling Entity Auditing**

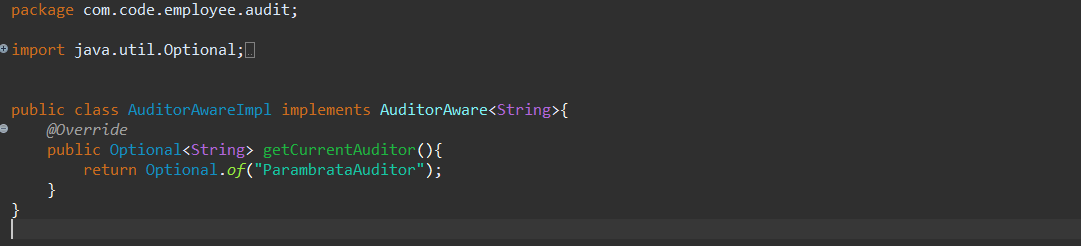
Business Scenario:

Implement auditing to track the creation and modification of employees and departments.

**1. Entity Auditing:**

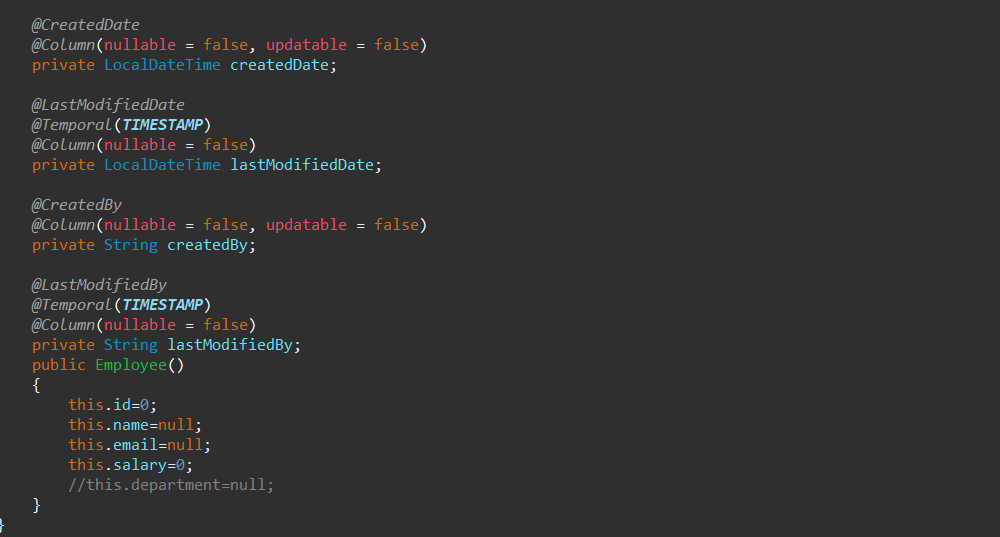
I enabled auditing in your application by configuring auditing properties.

I used annotations like **@CreatedBy, @LastModifiedBy, @CreatedDate**, and **@LastModifiedDate**.

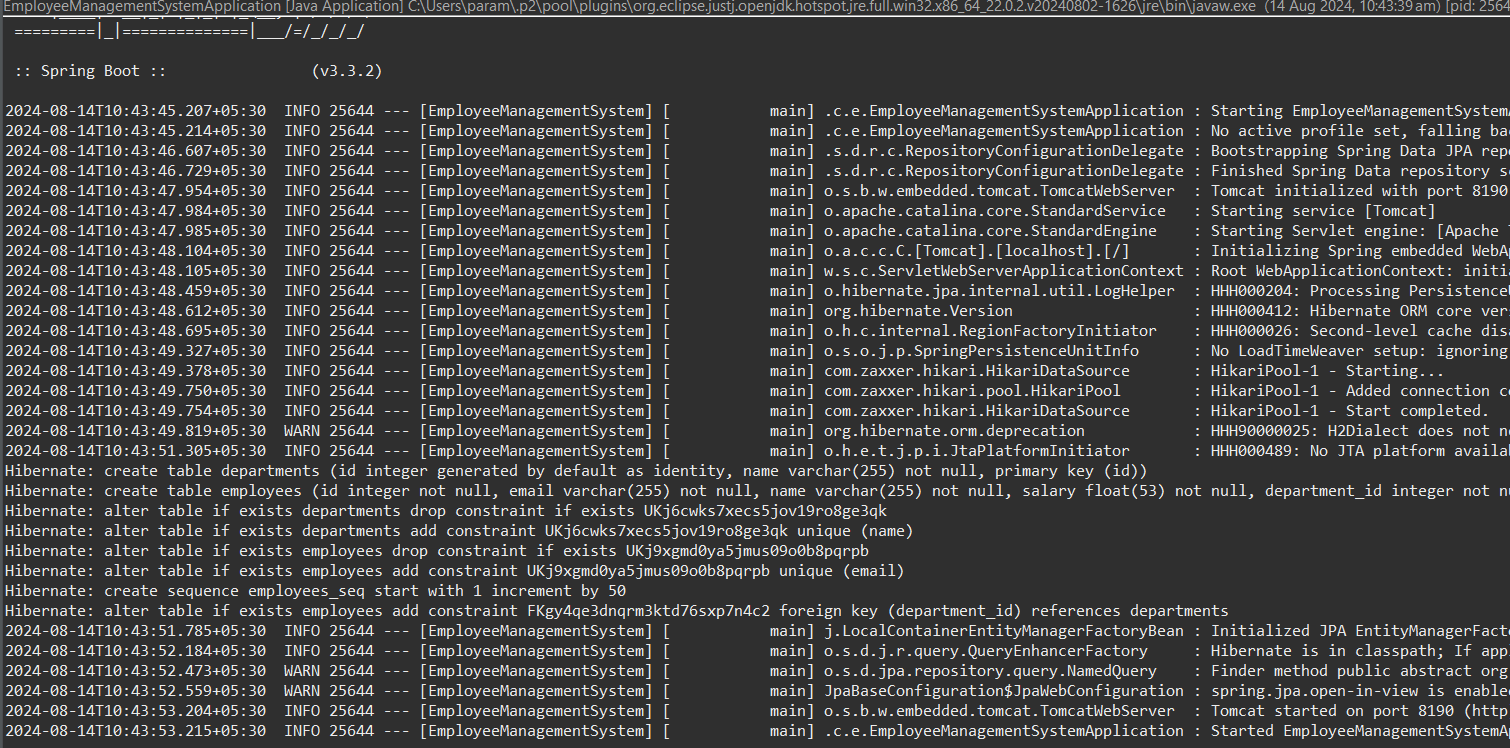


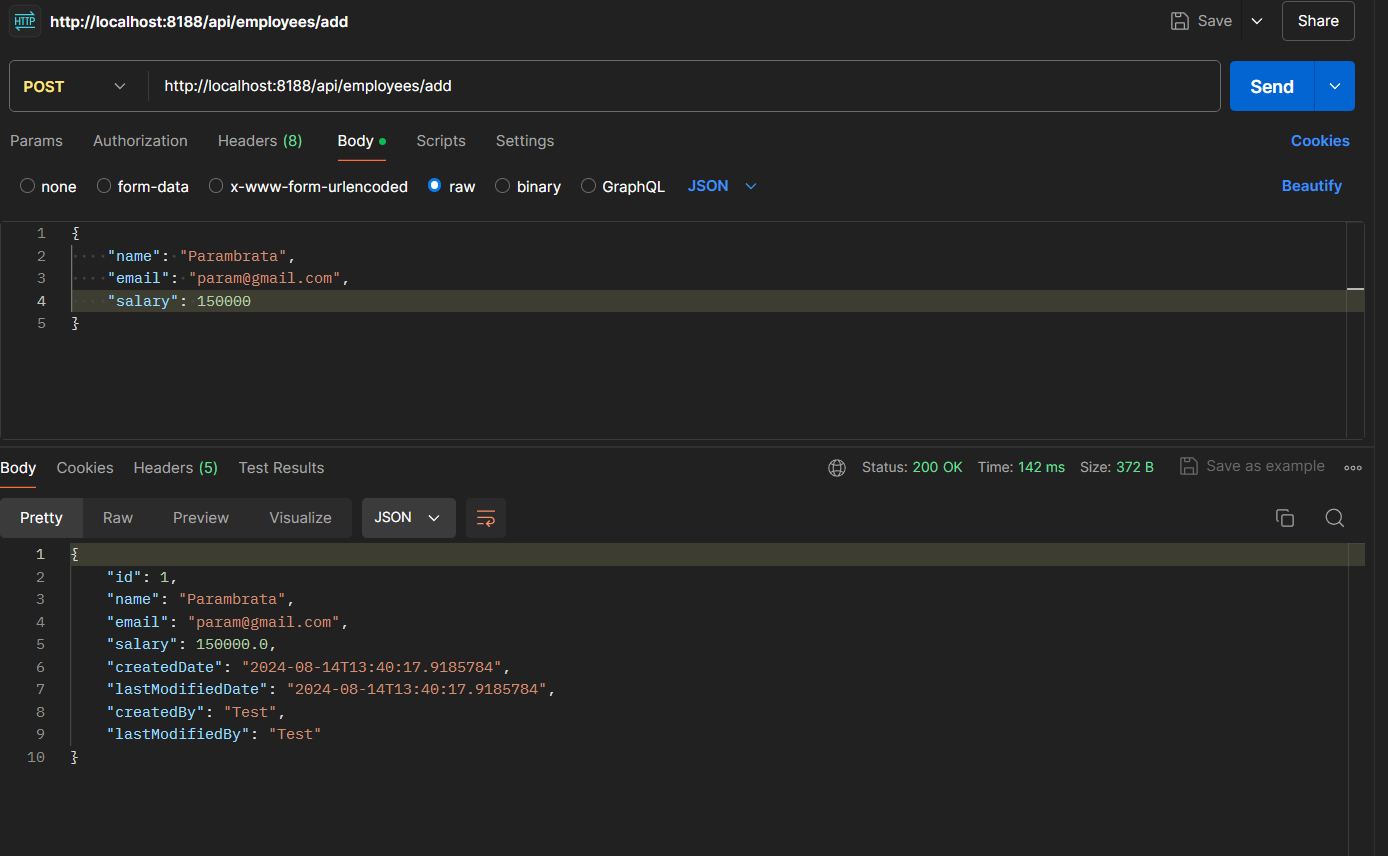


**Employee.class**

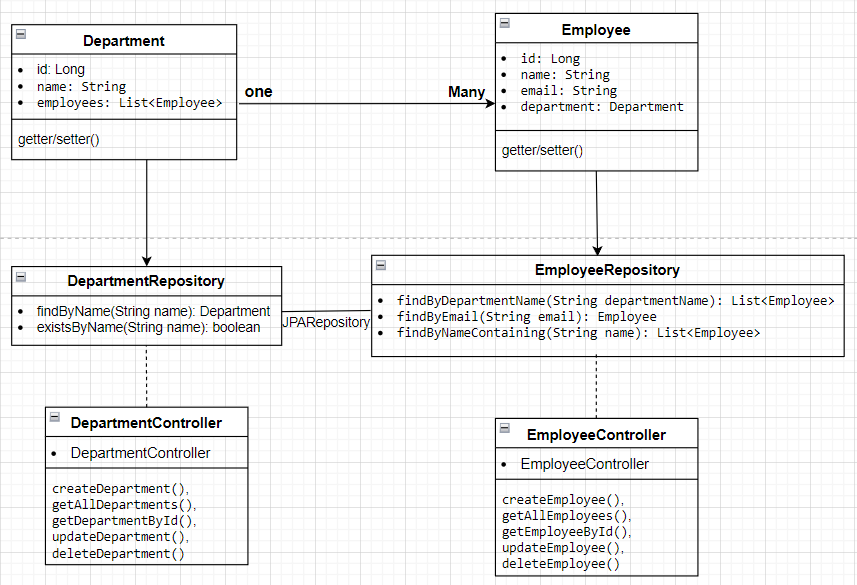
****

**Output:**

****



**Class Diagram:**

****

** Department Class:**

* **Attributes: Represents the id, name, and a list of Employee objects.**
* **Relationship: Has a one-to-many relationship with Employee.**

** Employee Class:**

* **Attributes: Represents the id, name, email, and a reference to the Department object.**
* **Relationship: Each Employee belongs to a single Department.**

** EmployeeRepository Interface:**

* **Methods:**
  + **findByDepartmentName(String departmentName): List<Employee>**
  + **findByEmail(String email): Employee**
  + **findByNameContaining(String name): List<Employee>**

** DepartmentRepository Interface:**

* **Methods:**
  + **findByName(String name): Department**
  + **existsByName(String name): Boolean**

** EmployeeController Class:**

* **Methods:**
* **createEmployee()**
* **getAllEmployees()**
* **getEmployeeById()**
* **updateEmployee()**
* **deleteEmployee().**

** DepartmentController Class:**

* **Methods**
* **createDepartment()**
* **getAllDepartments()**
* **getDepartmentById()**
* **updateDepartment()**
* **deleteDepartment()**

**Analysis:**

** Enable Auditing: Configure auditing in your Spring Boot application using @EnableJpaAuditing.**

** Annotate Entities: Use @CreatedBy, @LastModifiedBy, @CreatedDate, and @LastModifiedDate annotations in your entities.**

** Implement AuditorAware: Provide the current user for auditing purposes.**

** Test and Verify: Create and modify entities to ensure auditing is working as expected.**

**Relationship:**

* A Department can have many Employees (OneToMany relationship).
* An Employee belongs to one Department (ManyToOne relationship).
* Department to DepartmentRepository
* Employee to EmployeeRepository