**SPRING DATA JPA AND HIBERNATE**

**Exercise 1: Employee Management System - Overview and Setup**

**// pom.xml**

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.5</version>

<relativePath/>

</parent>

<groupId>com.example</groupId>

<artifactId>EmployeeManagementSystem</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>EmployeeManagementSystem</name>

<description>Employee Management Spring Boot App</description>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

**// application.properties**

# H2 In-memory DB config

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

# JPA Hibernate

spring.jpa.hibernate.ddl-auto=update

# H2 Console

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

**// EmployeeManagementSystemApplication.java**

package com.example.employeemanagementsystem;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class EmployeeManagementSystemApplication {

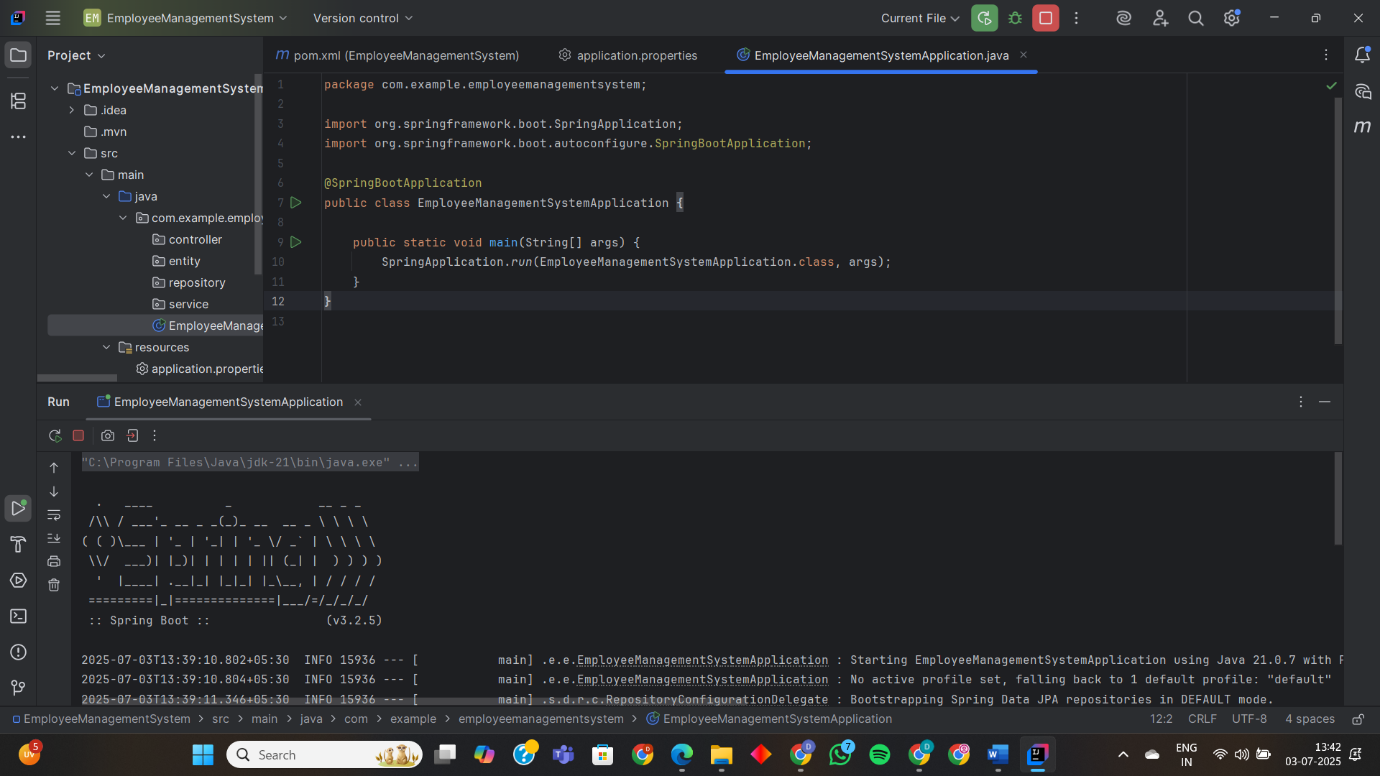
public static void main(String[] args) {

SpringApplication.run(EmployeeManagementSystemApplication.class, args);

}

}

**Output:**



**Exercise 2: Employee Management System - Creating Entities**

**// Department.java**

package com.example.employeemanagementsystem.entity;

import jakarta.persistence.\*;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

import java.util.ArrayList;

import java.util.List;

@Entity

@Table(name = "departments")

@Getter @Setter @NoArgsConstructor

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false, unique = true, length = 100)

private String name;

@OneToMany(mappedBy = "department",

cascade = CascadeType.ALL,

orphanRemoval = true,

fetch = FetchType.LAZY)

private List<Employee> employees = new ArrayList<>();

// Convenience helper

public void addEmployee(Employee employee) {

employees.add(employee);

employee.setDepartment(this);

}

}

**// Employee.java**

package com.example.employeemanagementsystem.entity;

import jakarta.persistence.\*;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Entity

@Table(name = "employees",

uniqueConstraints = @UniqueConstraint(columnNames = "email"))

@Getter @Setter @NoArgsConstructor

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false, length = 120)

private String name;

@Column(nullable = false, length = 150)

private String email;

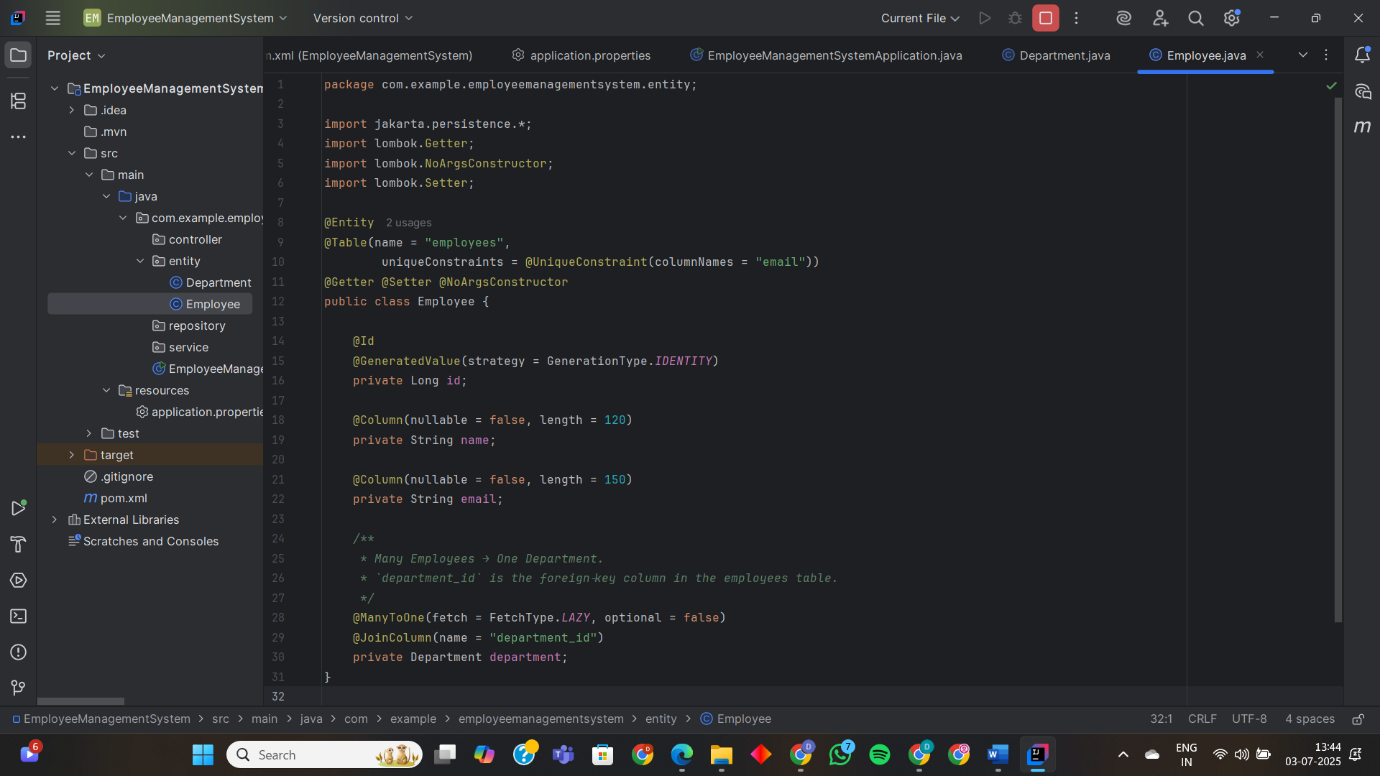
@ManyToOne(fetch = FetchType.LAZY, optional = false)

@JoinColumn(name = "department\_id")

private Department department;

}

**Output:**



**Exercise 3: Employee Management System - Creating Repositories**

**// DepartmentRepository.java**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Department;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface DepartmentRepository extends JpaRepository<Department, Long> {

Optional<Department> findByName(String name);

boolean existsByName(String name);

}

**// EmployeeRepository.java**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import java.util.List;

import java.util.Optional;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

Optional<Employee> findByEmail(String email);

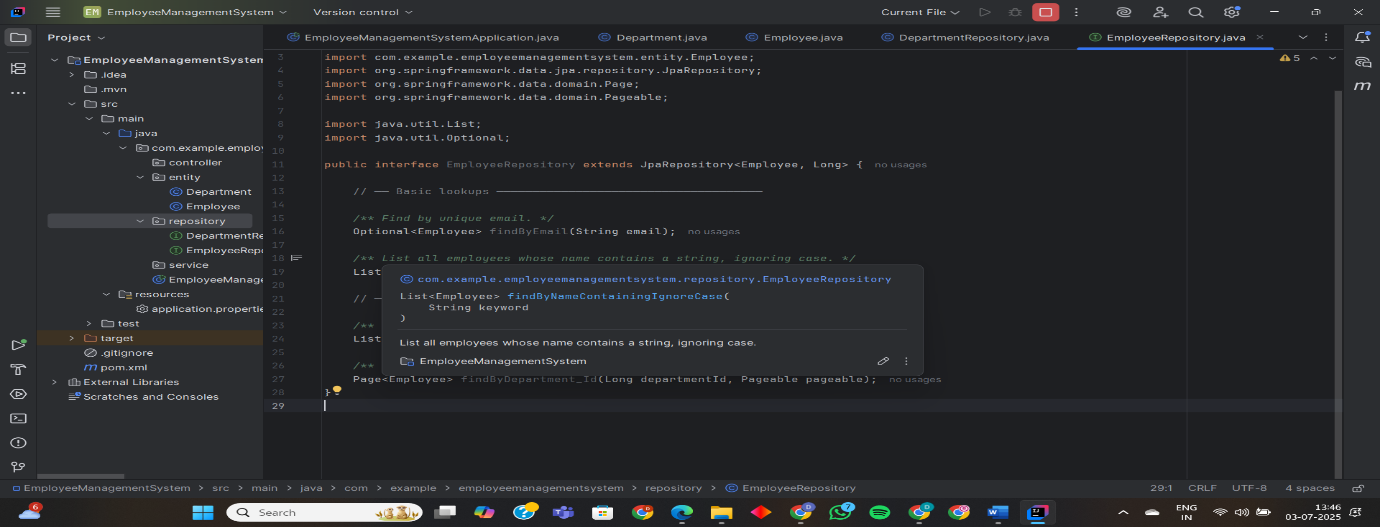
List<Employee> findByNameContainingIgnoreCase(String keyword);

List<Employee> findByDepartment\_Id(Long departmentId);

Page<Employee> findByDepartment\_Id(Long departmentId, Pageable pageable);

}

**Output:**



**Exercise 4: Employee Management System - Implementing CRUD Operations**

**// DepartmentDTO.java**

package com.example.employeemanagementsystem.dto;

import lombok.Data;

@Data

public class DepartmentDTO {

private Long id;

private String name;

}

**// EmployeeDTO.java**

package com.example.employeemanagementsystem.dto;

import lombok.Data;

@Data

public class EmployeeDTO {

private Long id;

private String name;

private String email;

private Long departmentId; // reference only, not nested entity

}

**// DepartmentService.java**

package com.example.employeemanagementsystem.service;

import com.example.employeemanagementsystem.dto.DepartmentDTO;

import com.example.employeemanagementsystem.entity.Department;

import com.example.employeemanagementsystem.repository.DepartmentRepository;

import jakarta.transaction.Transactional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

@Transactional

public class DepartmentService {

@Autowired

private DepartmentRepository departmentRepository;

public DepartmentDTO save(DepartmentDTO dto) {

Department dep = new Department();

dep.setName(dto.getName());

Department saved = departmentRepository.save(dep);

dto.setId(saved.getId());

return dto;

}

public List<Department> findAll() {

return departmentRepository.findAll();

}

public Department get(Long id) {

return departmentRepository.findById(id)

.orElseThrow(() -> new RuntimeException("Department not found"));

}

public DepartmentDTO update(Long id, DepartmentDTO dto) {

Department dep = get(id);

dep.setName(dto.getName());

return dto;

}

public void delete(Long id) {

departmentRepository.deleteById(id);

}

}

**// EmployeeService.java**

package com.example.employeemanagementsystem.service;

import com.example.employeemanagementsystem.dto.EmployeeDTO;

import com.example.employeemanagementsystem.entity.Department;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.repository.DepartmentRepository;

import com.example.employeemanagementsystem.repository.EmployeeRepository;

import jakarta.transaction.Transactional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.stereotype.Service;

@Service

@Transactional

public class EmployeeService {

@Autowired private EmployeeRepository employeeRepository;

@Autowired private DepartmentRepository departmentRepository;

public EmployeeDTO save(EmployeeDTO dto) {

Department dep = departmentRepository.findById(dto.getDepartmentId())

.orElseThrow(() -> new RuntimeException("Department not found"));

Employee emp = new Employee();

emp.setName(dto.getName());

emp.setEmail(dto.getEmail());

emp.setDepartment(dep);

Employee saved = employeeRepository.save(emp);

dto.setId(saved.getId());

return dto;

}

public Page<Employee> findAll(Pageable pageable) {

return employeeRepository.findAll(pageable);

}

public Employee get(Long id) {

return employeeRepository.findById(id)

.orElseThrow(() -> new RuntimeException("Employee not found"));

}

public EmployeeDTO update(Long id, EmployeeDTO dto) {

Employee emp = get(id);

emp.setName(dto.getName());

emp.setEmail(dto.getEmail());

if (dto.getDepartmentId() != null &&

!dto.getDepartmentId().equals(emp.getDepartment().getId())) {

Department dep = departmentRepository.findById(dto.getDepartmentId())

.orElseThrow(() -> new RuntimeException("Department not found"));

emp.setDepartment(dep);

}

return dto;

}

public void delete(Long id) {

employeeRepository.deleteById(id);

}

}

**// DepartmentController.java**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.dto.DepartmentDTO;

import com.example.employeemanagementsystem.entity.Department;

import com.example.employeemanagementsystem.service.DepartmentService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/departments")

public class DepartmentController {

@Autowired private DepartmentService departmentService;

@PostMapping

public DepartmentDTO create(@RequestBody DepartmentDTO dto) {

return departmentService.save(dto);

}

@GetMapping

public List<Department> list() {

return departmentService.findAll();

}

@GetMapping("/{id}")

public Department get(@PathVariable Long id) {

return departmentService.get(id);

}

@PutMapping("/{id}")

public DepartmentDTO update(@PathVariable Long id, @RequestBody DepartmentDTO dto) {

return departmentService.update(id, dto);

}

@DeleteMapping("/{id}")

public void delete(@PathVariable Long id) {

departmentService.delete(id);

}

}

**// EmployeeController.java**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.dto.EmployeeDTO;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.service.EmployeeService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired private EmployeeService employeeService;

@PostMapping

public EmployeeDTO create(@RequestBody EmployeeDTO dto) {

return employeeService.save(dto);

}

@GetMapping

public Page<Employee> list(Pageable pageable) {

return employeeService.findAll(pageable);

}

@GetMapping("/{id}")

public Employee get(@PathVariable Long id) {

return employeeService.get(id);

}

@PutMapping("/{id}")

public EmployeeDTO update(@PathVariable Long id, @RequestBody EmployeeDTO dto) {

return employeeService.update(id, dto);

}

@DeleteMapping("/{id}")

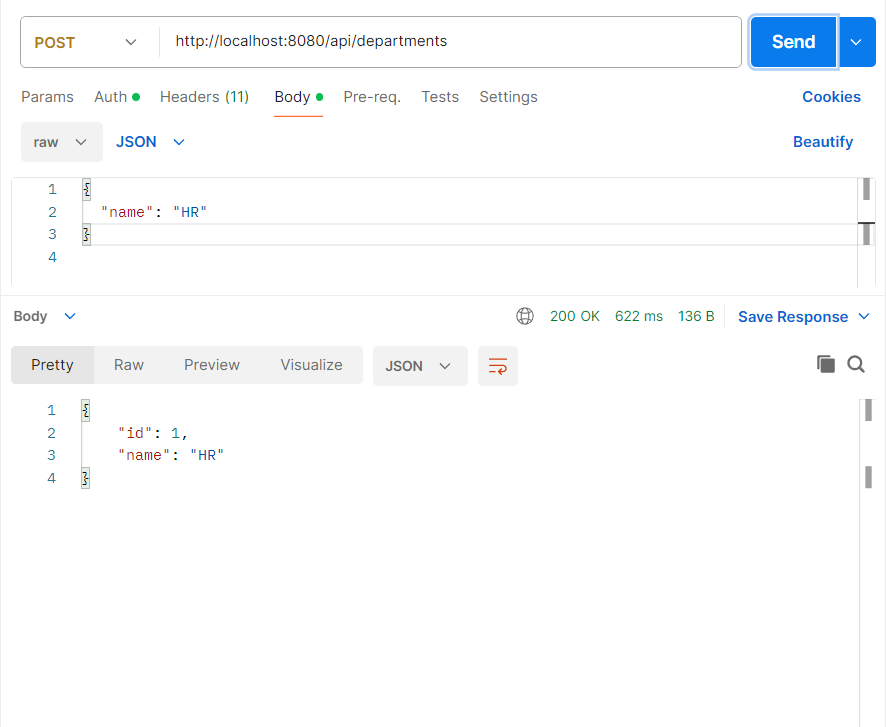
public void delete(@PathVariable Long id) {

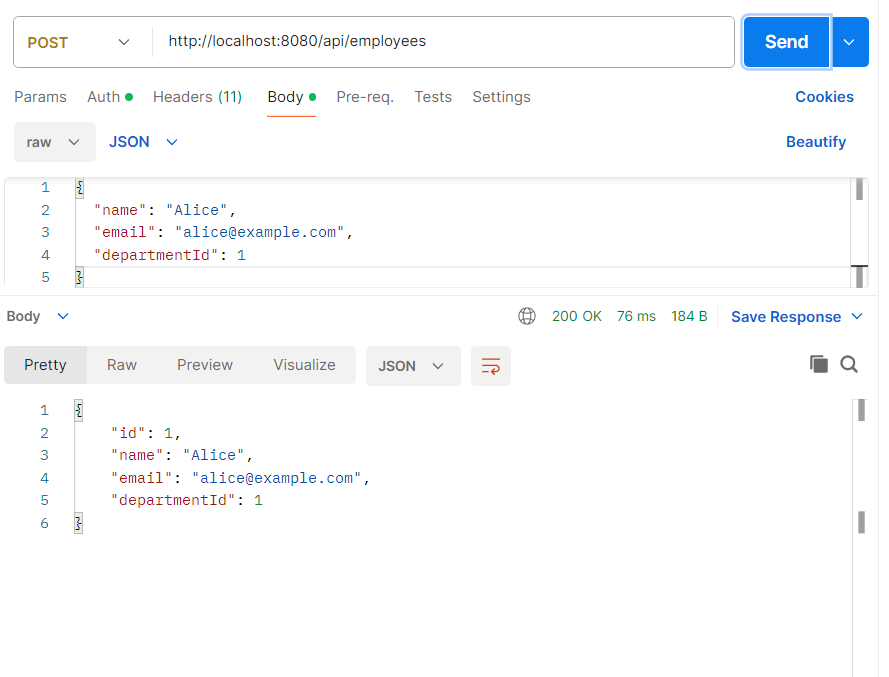
employeeService.delete(id);

}

}

Output:





**Exercise 5: Employee Management System - Defining Query Methods**

**// Employee.java**

package com.example.employeemanagementsystem.entity;

import jakarta.persistence.\*;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Entity

@Table(name = "employees")

@NamedQuery(

name = "Employee.findByEmailPrefix",

query = "SELECT e FROM Employee e WHERE e.email LIKE CONCAT(:prefix, '%')"

)

@Getter @Setter @NoArgsConstructor

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false, length = 120)

private String name;

@Column(nullable = false, length = 150, unique = true)

private String email;

@ManyToOne(fetch = FetchType.LAZY, optional = false)

@JoinColumn(name = "department\_id")

private Department department;

}

**// Department.java**

package com.example.employeemanagementsystem.entity;

import jakarta.persistence.\*;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

import java.util.ArrayList;

import java.util.List;

@Entity

@Table(name = "departments")

@Getter @Setter @NoArgsConstructor

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false, unique = true, length = 100)

private String name;

@OneToMany(mappedBy = "department",

cascade = CascadeType.ALL,

orphanRemoval = true,

fetch = FetchType.LAZY)

private List<Employee> employees = new ArrayList<>();

}

**// EmployeeRepository.java**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import java.util.List;

import java.util.Optional;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

Optional<Employee> findByEmail(String email);

List<Employee> findByNameContainingIgnoreCase(String keyword);

List<Employee> findByDepartment\_Name(String departmentName);

@Query("SELECT e FROM Employee e WHERE e.email LIKE %:domain")

List<Employee> findByEmailDomain(@Param("domain") String domain);

@Query(value = "SELECT \* FROM employees WHERE name ILIKE :letter || '%'",

nativeQuery = true)

List<Employee> findByNameStartingWithNative(@Param("letter") String letter);

@Query(name = "Employee.findByEmailPrefix")

List<Employee> findByEmailPrefix(@Param("prefix") String prefix);

}

**// DepartmentRepository.java**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Department;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import java.util.Optional;

public interface DepartmentRepository extends JpaRepository<Department, Long> {

Optional<Department> findByName(String name);

boolean existsByName(String name);

@Query("SELECT d FROM Department d WHERE SIZE(d.employees) >= :min")

java.util.List<Department> findWithMinEmployees(int min);

}

**Exercise 6: Employee Management System - Implementing Pagination and Sorting**

**// EmployeeRepository.java**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.data.jpa.repository.JpaRepository;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

java.util.Optional<Employee> findByEmail(String email);

java.util.List<Employee> findByDepartment\_Name(String departmentName);

Page<Employee> findByDepartment\_Id(Long departmentId, Pageable pageable);

}

**// EmployeeService.java**

package com.example.employeemanagementsystem.service;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.repository.EmployeeRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.stereotype.Service;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public Page<Employee> getAll(Pageable pageable) {

return employeeRepository.findAll(pageable);

}

public Page<Employee> getByDepartment(Long deptId, Pageable pageable) {

return employeeRepository.findByDepartment\_Id(deptId, pageable);

}

}

**// EmployeeController.java**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.service.EmployeeService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@GetMapping

public Page<Employee> list(Pageable pageable) {

return employeeService.getAll(pageable);

}

@GetMapping("/department/{deptId}")

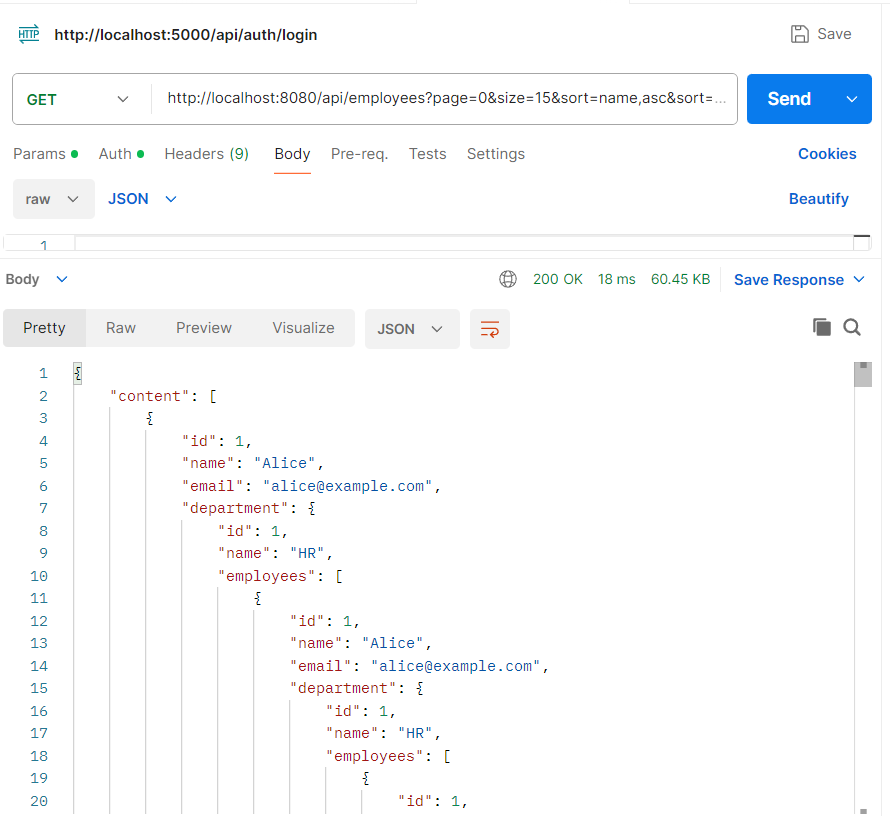
public Page<Employee> listByDept(@PathVariable Long deptId, Pageable pageable) {

return employeeService.getByDepartment(deptId, pageable);

}

}

**Output:**



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

**// EmployeeManagementSystemApplication.java**

package com.example.employeemanagementsystem;

import com.example.employeemanagementsystem.entity.Department;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.repository.DepartmentRepository;

import com.example.employeemanagementsystem.repository.EmployeeRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

import org.springframework.data.jpa.repository.config.EnableJpaAuditing;

@SpringBootApplication

@EnableJpaAuditing // ← turns on auditing

public class EmployeeManagementSystemApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementSystemApplication.class, args);

}

// quick demo: insert one department & employee, then print audit columns

@Bean

CommandLineRunner demo(DepartmentRepository depRepo,

EmployeeRepository empRepo) {

return args -> {

Department hr = depRepo.save(new Department("HR"));

Employee alice = new Employee("Alice", "alice@example.com", hr);

empRepo.save(alice);

System.out.println(depRepo.findById(hr.getId()).orElseThrow());

System.out.println(empRepo.findById(alice.getId()).orElseThrow());

};

}

}

**// AuditingConfig.java**

package com.example.employeemanagementsystem.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.data.domain.AuditorAware;

import java.util.Optional;

@Configuration

public class AuditingConfig {

@Bean

public AuditorAware<String> auditorAware() {

return () -> Optional.of("system");

}

}

**// Department.java**

package com.example.employeemanagementsystem.entity;

import jakarta.persistence.\*;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.ToString;

import org.springframework.data.annotation.CreatedBy;

import org.springframework.data.annotation.CreatedDate;

import org.springframework.data.annotation.LastModifiedBy;

import org.springframework.data.annotation.LastModifiedDate;

import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import java.time.Instant;

@Entity

@Table(name = "departments")

@EntityListeners(AuditingEntityListener.class)

@Getter @ToString @NoArgsConstructor

public class Department {

public Department(String name) { this.name = name; }

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@CreatedBy @Column(updatable = false) private String createdBy;

@CreatedDate @Column(updatable = false) private Instant createdDate;

@LastModifiedBy private String lastModifiedBy;

@LastModifiedDate private Instant lastModifiedDate;

}

**// Employee.java**

package com.example.employeemanagementsystem.entity;

import jakarta.persistence.\*;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.ToString;

import org.springframework.data.annotation.CreatedBy;

import org.springframework.data.annotation.CreatedDate;

import org.springframework.data.annotation.LastModifiedBy;

import org.springframework.data.annotation.LastModifiedDate;

import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import java.time.Instant;

@Entity

@Table(name = "employees")

@EntityListeners(AuditingEntityListener.class)

@Getter @ToString @NoArgsConstructor

public class Employee {

public Employee(String name, String email, Department department) {

this.name = name;

this.email = email;

this.department = department;

}

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

@ManyToOne(fetch = FetchType.EAGER, optional = false)

@JoinColumn(name = "department\_id")

private Department department;

@CreatedBy @Column(updatable = false) private String createdBy;

@CreatedDate @Column(updatable = false) private Instant createdDate;

@LastModifiedBy private String lastModifiedBy;

@LastModifiedDate private Instant lastModifiedDate;

}

**// DepartmentRepository.java**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Department;

import org.springframework.data.jpa.repository.JpaRepository;

public interface DepartmentRepository extends JpaRepository<Department, Long> { }

**// EmployeeRepository.java**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

import org.springframework.data.jpa.repository.JpaRepository;

public interface EmployeeRepository extends JpaRepository<Employee, Long> { }

**// application.properties**

# src/main/resources/application.properties

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

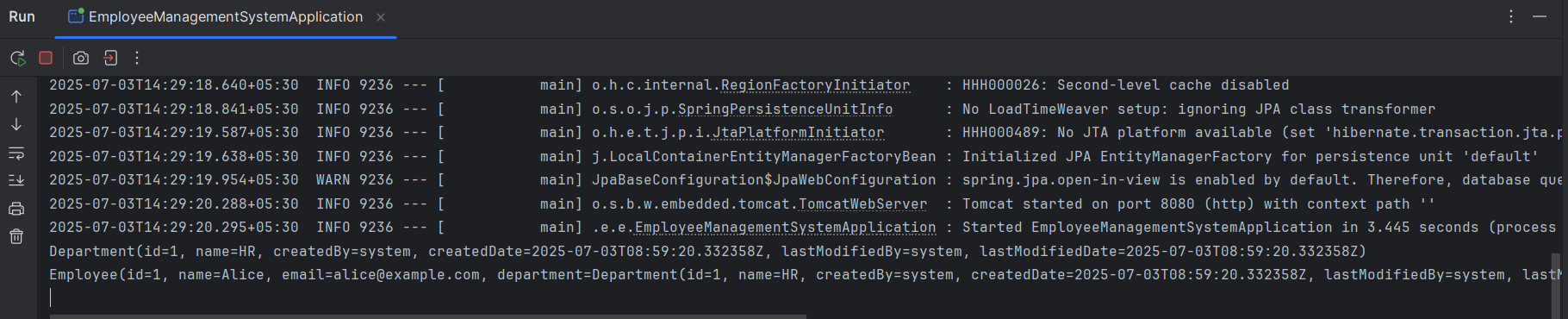
spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true

**Output:**



**Exercise 8: Employee Management System - Creating Projections**

**// Department.java**

package com.example.employeemanagementsystem.entity;

import jakarta.persistence.\*;

import lombok.\*;

@Entity

@Getter @Setter

@NoArgsConstructor @AllArgsConstructor

@ToString

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

}  
  
**// Employee.java**

package com.example.employeemanagementsystem.entity;

import jakarta.persistence.\*;

import lombok.\*;

@Entity

@Getter @Setter

@NoArgsConstructor @AllArgsConstructor

@ToString

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "department\_id")

private Department department;

}  
  
// EmployeeNameEmailProjection

package com.example.employeemanagementsystem.projection;

public interface EmployeeNameEmailProjection {

String getName();

String getEmail();

}

**// EmployeeSummary.java**

package com.example.employeemanagementsystem.projection;

public class EmployeeSummary {

private String name;

private String departmentName;

public EmployeeSummary(String name, String departmentName) {

this.name = name;

this.departmentName = departmentName;

}

public String getName() {

return name;

}

public String getDepartmentName() {

return departmentName;

}

@Override

public String toString() {

return "EmployeeSummary{name='" + name + "', departmentName='" + departmentName + "'}";

}

}

**// EmployeeRepository.java**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.projection.EmployeeNameEmailProjection;

import com.example.employeemanagementsystem.projection.EmployeeSummary;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// Interface-based projection

List<EmployeeNameEmailProjection> findBy();

// Class-based projection using constructor expression

@Query("SELECT new com.example.employeemanagementsystem.projection.EmployeeSummary(e.name, e.department.name) FROM Employee e")

List<EmployeeSummary> findEmployeeSummaries();

}

**// EmployeeManagementSystemApplication.java**

package com.example.employeemanagementsystem;

import com.example.employeemanagementsystem.entity.Department;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.projection.EmployeeNameEmailProjection;

import com.example.employeemanagementsystem.projection.EmployeeSummary;

import com.example.employeemanagementsystem.repository.DepartmentRepository;

import com.example.employeemanagementsystem.repository.EmployeeRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class EmployeeManagementSystemApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementSystemApplication.class, args);

}

@Bean

CommandLineRunner demo(EmployeeRepository employeeRepo, DepartmentRepository departmentRepo) {

return args -> {

Department d1 = departmentRepo.save(new Department(null, "HR"));

Department d2 = departmentRepo.save(new Department(null, "IT"));

employeeRepo.save(new Employee(null, "Alice", "alice@example.com", d1));

employeeRepo.save(new Employee(null, "Bob", "bob@example.com", d2));

System.out.println("=== Interface-based Projection ===");

for (EmployeeNameEmailProjection e : employeeRepo.findBy()) {

System.out.println("Name: " + e.getName() + ", Email: " + e.getEmail());

}

System.out.println("=== Class-based Projection ===");

for (EmployeeSummary s : employeeRepo.findEmployeeSummaries()) {

System.out.println(s);

}

};

}

}

**// DepartmentRepository.java**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Department;

import org.springframework.data.jpa.repository.JpaRepository;

public interface DepartmentRepository extends JpaRepository<Department, Long> {

}

**// application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

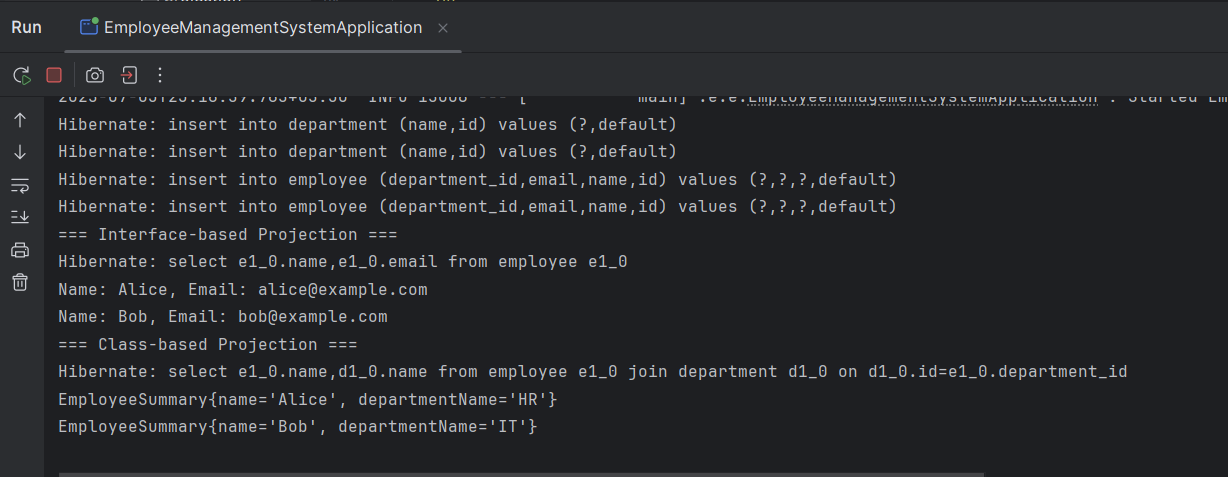
spring.datasource.password=password

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.jpa.show-sql=true

**Output:**



**Exercise 9: Employee Management System - Customizing Data Source Configuration**

**1. Spring Boot Auto-Configuration:**

**// application.properties**

# Customized DataSource Configuration

spring.datasource.url=jdbc:h2:mem:primarydb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.show-sql=true

spring.h2.console.enabled=true

**2. Externalizing Configuration:**

**// application.properties**

# Primary Data Source

spring.datasource.primary.url=jdbc:h2:mem:primarydb

spring.datasource.primary.driver-class-name=org.h2.Driver

spring.datasource.primary.username=sa

spring.datasource.primary.password=password

# Secondary Data Source

spring.datasource.secondary.url=jdbc:h2:mem:secondarydb

spring.datasource.secondary.driver-class-name=org.h2.Driver

spring.datasource.secondary.username=sa

spring.datasource.secondary.password=password

# Disable Spring Boot default datasource auto-config

spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration

spring.jpa.show-sql=true

spring.h2.console.enabled=true

**// PrimaryDataSourceConfig.java**

@Configuration

@EnableTransactionManagement

@EnableJpaRepositories(

basePackages = "com.example.employeemanagementsystem.primary.repository",

entityManagerFactoryRef = "primaryEntityManagerFactory",

transactionManagerRef = "primaryTransactionManager"

)

public class PrimaryDataSourceConfig {

@Bean

@Primary

@ConfigurationProperties("spring.datasource.primary")

public DataSourceProperties primaryDataSourceProperties() {

return new DataSourceProperties();

}

@Bean

@Primary

public DataSource primaryDataSource() {

return primaryDataSourceProperties().initializeDataSourceBuilder().build();

}

@Bean

@Primary

public LocalContainerEntityManagerFactoryBean primaryEntityManagerFactory(

EntityManagerFactoryBuilder builder) {

return builder

.dataSource(primaryDataSource())

.packages("com.example.employeemanagementsystem.primary.entity")

.persistenceUnit("primary")

.build();

}

@Bean

@Primary

public PlatformTransactionManager primaryTransactionManager(

@Qualifier("primaryEntityManagerFactory") EntityManagerFactory emf) {

return new JpaTransactionManager(emf);

}

}

**// SecondaryDataSourceConfig.java**

@Configuration

@EnableTransactionManagement

@EnableJpaRepositories(

basePackages = "com.example.employeemanagementsystem.secondary.repository",

entityManagerFactoryRef = "secondaryEntityManagerFactory",

transactionManagerRef = "secondaryTransactionManager"

)

public class SecondaryDataSourceConfig {

@Bean

@ConfigurationProperties("spring.datasource.secondary")

public DataSourceProperties secondaryDataSourceProperties() {

return new DataSourceProperties();

}

@Bean

public DataSource secondaryDataSource() {

return secondaryDataSourceProperties().initializeDataSourceBuilder().build();

}

@Bean

public LocalContainerEntityManagerFactoryBean secondaryEntityManagerFactory(

EntityManagerFactoryBuilder builder) {

return builder

.dataSource(secondaryDataSource())

.packages("com.example.employeemanagementsystem.secondary.entity")

.persistenceUnit("secondary")

.build();

}

@Bean

public PlatformTransactionManager secondaryTransactionManager(

@Qualifier("secondaryEntityManagerFactory") EntityManagerFactory emf) {

return new JpaTransactionManager(emf);

}

}

**// primary/entity/Employee.java**

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

}

**// secondary/entity/AuditLog.java**

@Entity

public class AuditLog {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String action;

private LocalDateTime timestamp;

}

**// EmployeeManagementSystemApplication.java**

// src/main/java/com/example/employeemanagementsystem/EmployeeManagementSystemApplication.java

package com.example.employeemanagementsystem;

import com.example.employeemanagementsystem.primary.entity.Employee;

import com.example.employeemanagementsystem.primary.repository.EmployeeRepository;

import com.example.employeemanagementsystem.secondary.entity.AuditLog;

import com.example.employeemanagementsystem.secondary.repository.AuditLogRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

import java.time.LocalDateTime;

@SpringBootApplication

public class EmployeeManagementSystemApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementSystemApplication.class, args);

}

@Bean

CommandLineRunner runner(EmployeeRepository employeeRepo,

AuditLogRepository auditRepo) {

return args -> {

Employee emp = employeeRepo.save(new Employee(null, "Alice"));

auditRepo.save(new AuditLog(null, "CREATED\_EMPLOYEE\_" + emp.getId(),

LocalDateTime.now()));

System.out.println("Employees : " + employeeRepo.findAll());

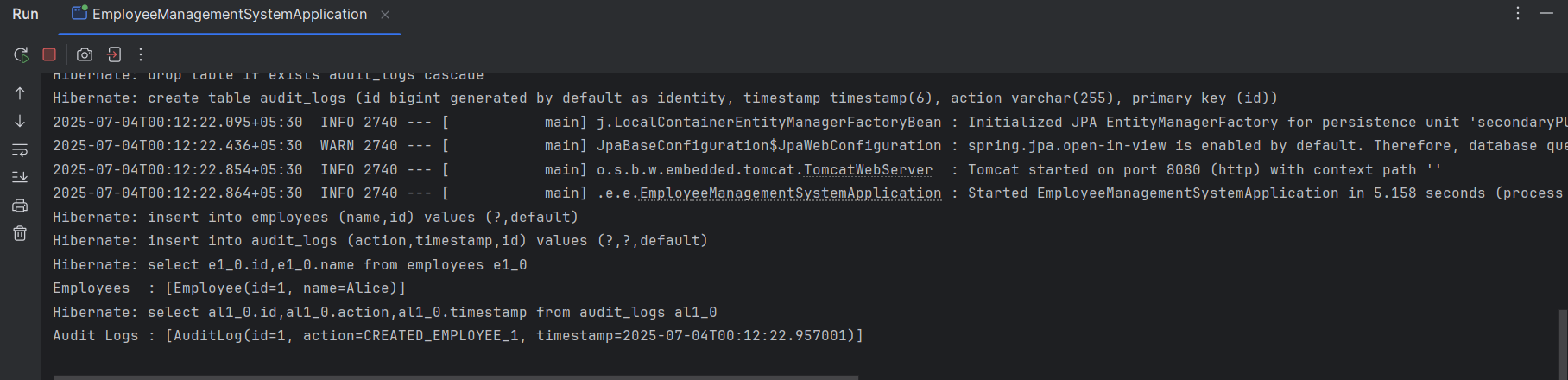
System.out.println("Audit Logs : " + auditRepo.findAll());

};

}

}

**Output:**



**Exercise 10: Employee Management System - Hibernate-Specific Features**

**// src/main/java/com/example/employeemanagementsystem/entity/Employee.java**

package com.example.employeemanagementsystem.entity;

import jakarta.persistence.\*;

import lombok.\*;

import org.hibernate.annotations.\*;

import java.time.Instant;

@Entity

@Table(name = "employees")

@DynamicInsert // ← Hibernate-specific: only non‑null cols on INSERT

@DynamicUpdate // ← Hibernate-specific: only changed cols on UPDATE

@BatchSize(size = 50) // ← fetch 50 rows per round trip when lazy-loading

@Getter @Setter @NoArgsConstructor @AllArgsConstructor @ToString

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@Column(unique = true)

private String email;

@CreationTimestamp // filled only once

@Column(updatable = false)

private Instant createdAt;

@UpdateTimestamp // updated each UPDATE

private Instant updatedAt;

}

**// src/main/java/com/example/employeemanagementsystem/repository/EmployeeRepository.java**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

import org.springframework.data.jpa.repository.JpaRepository;

public interface EmployeeRepository extends JpaRepository<Employee, Long> { }

**// src/main/java/com/example/employeemanagementsystem/service/BatchEmployeeService.java**

package com.example.employeemanagementsystem.service;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.repository.EmployeeRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.ArrayList;

import java.util.List;

@Service

public class BatchEmployeeService {

@Autowired

private EmployeeRepository repo;

@Transactional

public void saveBulk(int n) {

List<Employee> batch = new ArrayList<>(n);

for (int i = 1; i <= n; i++) {

batch.add(new Employee(null, "User " + i, "user" + i + "@example.com", null, null));

if (i % 50 == 0) { // flush & clear every 50 rows

repo.saveAll(batch);

repo.flush();

batch.clear();

}

}

if (!batch.isEmpty()) {

repo.saveAll(batch);

repo.flush();

}

}

}

**// src/main/java/com/example/employeemanagementsystem/EmployeeManagementSystemApplication.java**

package com.example.employeemanagementsystem;

import com.example.employeemanagementsystem.service.BatchEmployeeService;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class EmployeeManagementSystemApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementSystemApplication.class, args);

}

@Bean

CommandLineRunner runBatch(BatchEmployeeService batchService) {

return args -> batchService.saveBulk(10\_000); // insert 10k rows

}

}

**# src/main/resources/application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect

spring.jpa.properties.hibernate.batch\_size=50

spring.jpa.properties.hibernate.order\_inserts=true

spring.jpa.properties.hibernate.order\_updates=true

spring.jpa.properties.hibernate.jdbc.batch\_versioned\_data=true

spring.jpa.properties.hibernate.generate\_statistics=true # optional – logs metrics

# Schema creation for quick demo

spring.jpa.hibernate.ddl-auto=create

spring.jpa.show-sql=true

logging.level.org.hibernate.SQL=DEBUG

logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE

**Output:**

