**Exercise 1: Employee Management System**

# H2 Database Configuration

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

# JPA Properties

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

spring.jpa.show-sql=true # Optional: Show SQL queries in the console

spring.h2.console.enabled=true # Optional: Enable H2 web console at /h2-console

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

1)package com.example.employeemanagementsystem.entity;

import jakarta.persistence.\*;

import lombok.Data;

import lombok.NoArgsConstructor;

@Entity

@Table(name = "employees")

@Data

@NoArgsConstructor

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false)

private String name

@Column(nullable = false, unique = true)

private String email;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "department\_id", nullable = false)

private Department department;

}

2)package com.example.employeemanagementsystem.entity;

import jakarta.persistence.\*;

import lombok.Data;

import lombok.NoArgsConstructor;

import java.util.List;

@Entity

@Table(name = "departments")

@Data

@NoArgsConstructor

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false, unique = true)

private String name;

@OneToMany(mappedBy = "department", cascade = CascadeType.ALL, orphanRemoval = true)

private List<Employee> employees;

}

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

1)package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

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

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// Derived query method to find employees by department name

List<Employee> findByDepartmentName(String departmentName);

// Derived query method to find employees by name

List<Employee> findByNameContaining(String name);

}

2)package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Department;

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

import org.springframework.stereotype.Repository;

@Repository

public interface DepartmentRepository extends JpaRepository<Department, Long> {

// Derived query method to find a department by its name

Department findByName(String name);

}

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

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 java.util.List;

import java.util.Optional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

// Create a new employee

public Employee saveEmployee(Employee employee) {

return employeeRepository.save(employee);

}

// Get all employees

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

// Get an employee by ID

public Optional<Employee> getEmployeeById(Long id) {

return employeeRepository.findById(id);

}

// Update an existing employee

public Employee updateEmployee(Employee employee) {

return employeeRepository.save(employee);

}

// Delete an employee by ID

public void deleteEmployee(Long id) {

employeeRepository.deleteById(id);

}

}

package com.example.employeemanagementsystem.service;

import com.example.employeemanagementsystem.entity.Department;

import com.example.employeemanagementsystem.repository.DepartmentRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

public class DepartmentService {

@Autowired

private DepartmentRepository departmentRepository;

// Create a new department

public Department saveDepartment(Department department) {

return departmentRepository.save(department);

}

// Get all departments

public List<Department> getAllDepartments() {

return departmentRepository.findAll();

}

// Get a department by ID

public Optional<Department> getDepartmentById(Long id) {

return departmentRepository.findById(id);

}

// Update an existing department

public Department updateDepartment(Department department) {

return departmentRepository.save(department);

}

// Delete a department by ID

public void deleteDepartment(Long id) {

departmentRepository.deleteById(id);

}

}

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.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

// Create a new employee

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

return employeeService.saveEmployee(employee);

}

// Get all employees

@GetMapping

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

// Get an employee by ID

@GetMapping("/{id}")

public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id) {

return employeeService.getEmployeeById(id)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

// Update an existing employee

@PutMapping("/{id}")

public ResponseEntity<Employee> updateEmployee(@PathVariable Long id, @RequestBody Employee employeeDetails) {

return employeeService.getEmployeeById(id)

.map(employee -> {

employee.setName(employeeDetails.getName());

employee.setEmail(employeeDetails.getEmail());

employee.setDepartment(employeeDetails.getDepartment());

return ResponseEntity.ok(employeeService.saveEmployee(employee));

})

.orElse(ResponseEntity.notFound().build());

}

// Delete an employee by ID

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteEmployee(@PathVariable Long id) {

return employeeService.getEmployeeById(id)

.map(employee -> {

employeeService.deleteEmployee(id);

return ResponseEntity.ok().build();

})

.orElse(ResponseEntity.notFound().build());

}

}

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.entity.Department;

import com.example.employeemanagementsystem.service.DepartmentService;

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

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/api/departments")

public class DepartmentController {

@Autowired

private DepartmentService departmentService;

// Create a new department

@PostMapping

public Department createDepartment(@RequestBody Department department) {

return departmentService.saveDepartment(department);

}

// Get all departments

@GetMapping

public List<Department> getAllDepartments() {

return departmentService.getAllDepartments();

}

// Get a department by ID

@GetMapping("/{id}")

public ResponseEntity<Department> getDepartmentById(@PathVariable Long id) {

return departmentService.getDepartmentById(id)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

// Update an existing department

@PutMapping("/{id}")

public ResponseEntity<Department> updateDepartment(@PathVariable Long id, @RequestBody Department departmentDetails) {

return departmentService.getDepartmentById(id)

.map(department -> {

department.setName(departmentDetails.getName());

return ResponseEntity.ok(departmentService.saveDepartment(department));

})

.orElse(ResponseEntity.notFound().build());

}

// Delete a department by ID

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteDepartment(@PathVariable Long id) {

return departmentService.getDepartmentById(id)

.map(department -> {

departmentService.deleteDepartment(id);

return ResponseEntity.ok().build();

})

.orElse(ResponseEntity.notFound().build());

}

}

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

List<Employee> findByDepartmentName(String departmentName);

List<Department> findByNameContaining(String namePart);

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

List<Employee> searchByName(@Param("name") String name);

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

List<Department> findByEmployeeCountGreaterThan(@Param("employeeCount") int employeeCount);

@Entity

@NamedQueries({

@NamedQuery(

name = "Employee.findByDepartment",

query = "SELECT e FROM Employee e WHERE e.department.name = :departmentName"

)

})

public class Employee {

// Fields, constructors, getters, setters...

}

@Query(name = "Employee.findByDepartment")

List<Employee> findByDepartmentUsingNamedQuery(@Param("departmentName") String departmentName);

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

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;

// Other methods...

// Get paginated list of employees

public Page<Employee> getAllEmployees(Pageable pageable) {

return employeeRepository.findAll(pageable);

}

}

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;

import org.springframework.stereotype.Repository;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// Find all employees with pagination

Page<Employee> findAll(Pageable pageable);

}

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.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

// Other methods...

// Get paginated list of employees

@GetMapping

public Page<Employee> getAllEmployees(Pageable pageable) {

return employeeService.getAllEmployees(pageable);

}

}

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.data.domain.Sort;

import org.springframework.stereotype.Service;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

// Other methods...

// Get paginated and sorted list of employees

public Page<Employee> getAllEmployees(Pageable pageable, Sort sort) {

return employeeRepository.findAll(Pageable.ofSize(pageable.getPageSize()).withPage(pageable.getPageNumber()).withSort(sort));

}

}

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;

import org.springframework.stereotype.Repository;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// Find all employees with pagination and sorting

Page<Employee> findAll(Pageable pageable);

}

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.data.domain.Sort;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

// Get paginated and sorted list of employees

@GetMapping

public Page<Employee> getAllEmployees(Pageable pageable,

@RequestParam(defaultValue = "id") String sortBy,

@RequestParam(defaultValue = "asc") String sortDir) {

Sort sort = Sort.by(sortBy);

sort = sortDir.equalsIgnoreCase("asc") ? sort.ascending() : sort.descending();

return employeeService.getAllEmployees(pageable, sort);

}

}

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

package com.example.employeemanagementsystem;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

@SpringBootApplication

@EnableJpaAuditing

public class EmployeeManagementSystemApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementSystemApplication.class, args);

}

}

package com.example.employeemanagementsystem.entity;

import lombok.Getter;

import lombok.Setter;

import org.springframework.data.annotation.CreatedDate;

import org.springframework.data.annotation.LastModifiedDate;

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

import javax.persistence.\*;

import java.time.LocalDateTime;

@Entity

@EntityListeners(AuditingEntityListener.class)

@Getter

@Setter

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

@CreatedDate

@Column(nullable = false, updatable = false)

private LocalDateTime createdDate;

@LastModifiedDate

private LocalDateTime lastModifiedDate;

// Other fields, constructors, getters, and setters...

}

package com.example.employeemanagementsystem.entity;

import lombok.Getter;

import lombok.Setter;

import org.springframework.data.annotation.CreatedDate;

import org.springframework.data.annotation.LastModifiedDate;

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

import javax.persistence.\*;

import java.time.LocalDateTime;

import java.util.List;

@Entity

@EntityListeners(AuditingEntityListener.class)

@Getter

@Setter

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "department")

private List<Employee> employees;

@CreatedDate

@Column(nullable = false, updatable = false)

private LocalDateTime createdDate;

@LastModifiedDate

private LocalDateTime lastModifiedDate;

// Other fields, constructors, getters, and setters...

}

package com.example.employeemanagementsystem.entity;

import lombok.Getter;

import lombok.Setter;

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 javax.persistence.\*;

import java.time.LocalDateTime;

@Entity

@EntityListeners(AuditingEntityListener.class)

@Getter

@Setter

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

@CreatedDate

@Column(nullable = false, updatable = false)

private LocalDateTime createdDate;

@LastModifiedDate

private LocalDateTime lastModifiedDate;

@CreatedBy

private String createdBy;

@LastModifiedBy

private String lastModifiedBy;

// Other fields, constructors, getters, and setters...

}

package com.example.employeemanagementsystem.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.data.domain.AuditorAware;

import java.util.Optional;

@Configuration

public class AuditorAwareImpl implements AuditorAware<String> {

@Override

public Optional<String> getCurrentAuditor() {

// Return the current user (in a real application, retrieve this from the security context)

return Optional.of("System"); // Placeholder for the current user

}

}

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

package com.example.employeemanagementsystem.projection;

public interface EmployeeNameProjection {

String getName();

String getEmail();

}

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.projection.EmployeeNameProjection;

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

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// Fetch only name and email of all employees

List<EmployeeNameProjection> findAllBy();

}

package com.example.employeemanagementsystem.dto;

public class DepartmentDto {

private String name;

private long employeeCount;

public DepartmentDto(String name, long employeeCount) {

this.name = name;

this.employeeCount = employeeCount;

}

// Getters and setters...

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public long getEmployeeCount() {

return employeeCount;

}

public void setEmployeeCount(long employeeCount) {

this.employeeCount = employeeCount;

}

}

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.dto.DepartmentDto;

import com.example.employeemanagementsystem.entity.Department;

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

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

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface DepartmentRepository extends JpaRepository<Department, Long> {

// Fetch department name and employee count

@Query("SELECT new com.example.employeemanagementsystem.dto.DepartmentDto(d.name, COUNT(e)) " +

"FROM Department d LEFT JOIN d.employees e GROUP BY d.name")

List<DepartmentDto> findDepartmentEmployeeCounts();

}

package com.example.employeemanagementsystem.projection;

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

public interface EmployeeDetailsProjection {

String getName();

@Value("#{target.name + ' (' + target.email + ')'}")

String getNameWithEmail();

}

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.projection.EmployeeDetailsProjection;

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

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// Fetch name and a custom string combining name and email

List<EmployeeDetailsProjection> findAllBy();

}

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

# H2 Database configuration

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

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.h2.console.enabled=true

# MySQL Database configuration for production

spring.datasource.url=jdbc:mysql://localhost:3306/employeedb

spring.datasource.username=root

spring.datasource.password=yourpassword

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.database-platform=org.hibernate.dialect.MySQL5Dialect

package com.example.employeemanagementsystem.config;

import org.springframework.boot.autoconfigure.orm.jpa.JpaProperties;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.boot.jdbc.DataSourceBuilder;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Primary;

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

import org.springframework.orm.jpa.JpaTransactionManager;

import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;

import org.springframework.transaction.PlatformTransactionManager;

import javax.sql.DataSource;

@Configuration

@EnableJpaRepositories(

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

entityManagerFactoryRef = "employeeEntityManagerFactory",

transactionManagerRef = "employeeTransactionManager"

)

public class EmployeeDataSourceConfig {

@Primary

@Bean(name = "employeeDataSource")

@ConfigurationProperties(prefix = "spring.datasource.employee")

public DataSource employeeDataSource() {

return DataSourceBuilder.create().build();

}

@Primary

@Bean(name = "employeeEntityManagerFactory")

public LocalContainerEntityManagerFactoryBean employeeEntityManagerFactory(

JpaProperties jpaProperties) {

LocalContainerEntityManagerFactoryBean factoryBean = new LocalContainerEntityManagerFactoryBean();

factoryBean.setDataSource(employeeDataSource());

factoryBean.setPackagesToScan("com.example.employeemanagementsystem.entity.employee");

factoryBean.setJpaProperties(jpaProperties.getProperties());

return factoryBean;

}

@Primary

@Bean(name = "employeeTransactionManager")

public PlatformTransactionManager employeeTransactionManager() {

return new JpaTransactionManager(employeeEntityManagerFactory().getObject());

}

}

package com.example.employeemanagementsystem.config;

import org.springframework.boot.autoconfigure.orm.jpa.JpaProperties;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.boot.jdbc.DataSourceBuilder;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

import org.springframework.orm.jpa.JpaTransactionManager;

import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;

import org.springframework.transaction.PlatformTransactionManager;

import javax.sql.DataSource;

@Configuration

@EnableJpaRepositories(

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

entityManagerFactoryRef = "departmentEntityManagerFactory",

transactionManagerRef = "departmentTransactionManager"

)

public class DepartmentDataSourceConfig {

@Bean(name = "departmentDataSource")

@ConfigurationProperties(prefix = "spring.datasource.department")

public DataSource departmentDataSource() {

return DataSourceBuilder.create().build();

}

@Bean(name = "departmentEntityManagerFactory")

public LocalContainerEntityManagerFactoryBean departmentEntityManagerFactory(

JpaProperties jpaProperties) {

LocalContainerEntityManagerFactoryBean factoryBean = new LocalContainerEntityManagerFactoryBean();

factoryBean.setDataSource(departmentDataSource());

factoryBean.setPackagesToScan("com.example.employeemanagementsystem.entity.department");

factoryBean.setJpaProperties(jpaProperties.getProperties());

return factoryBean;

}

@Bean(name = "departmentTransactionManager")

public PlatformTransactionManager departmentTransactionManager() {

return new JpaTransactionManager(departmentEntityManagerFactory().getObject());

}

}

# Employee Data Source

spring.datasource.employee.url=jdbc:h2:mem:employeedb

spring.datasource.employee.username=sa

spring.datasource.employee.password=password

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

# Department Data Source

spring.datasource.department.url=jdbc:h2:mem:departmentdb

spring.datasource.department.username=sa

spring.datasource.department.password=password

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

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

package com.example.employeemanagementsystem.entity;

import org.hibernate.annotations.NaturalId;

import org.hibernate.annotations.Formula;

import javax.persistence.\*;

@Entity

@Table(name = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@NaturalId

@Column(unique = true, nullable = false)

private String email;

@Column(nullable = false)

private String name;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "department\_id")

private Department department;

// Calculated property using @Formula

@Formula("(select count(e.id) from employees e where e.department\_id = department\_id)")

private int departmentEmployeeCount;

}

package com.example.employeemanagementsystem.entity;

import org.hibernate.annotations.Cache;

import org.hibernate.annotations.CacheConcurrencyStrategy;

import javax.persistence.\*;

import java.util.HashSet;

import java.util.Set;

@Entity

@Table(name = "departments")

@Cache(usage = CacheConcurrencyStrategy.READ\_WRITE)

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false)

private String name;

@OneToMany(mappedBy = "department", cascade = CascadeType.ALL, orphanRemoval = true)

private Set<Employee> employees = new HashSet<>();

// Getters and setters...

}

# Hibernate Dialect for H2 Database

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

# Enable second-level cache and query cache

spring.jpa.properties.hibernate.cache.use\_second\_level\_cache=true

spring.jpa.properties.hibernate.cache.use\_query\_cache=true

spring.jpa.properties.hibernate.cache.region.factory\_class=org.hibernate.cache.jcache.JCacheRegionFactory

spring.jpa.properties.javax.cache.provider=org.ehcache.jsr107.EhcacheCachingProvider

# Show SQL statements (optional for debugging)

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

# Enable Hibernate batch processing

spring.jpa.properties.hibernate.jdbc.batch\_size=20

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

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

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

package com.example.employeemanagementsystem.service;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.repository.EmployeeRepository;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service

public class EmployeeService {

private final EmployeeRepository employeeRepository;

public EmployeeService(EmployeeRepository employeeRepository) {

this.employeeRepository = employeeRepository;

}

@Transactional

public void saveAllEmployees(List<Employee> employees) {

for (int i = 0; i < employees.size(); i++) {

employeeRepository.save(employees.get(i));

if (i % 20 == 0) { // Flush and clear session after every 20 inserts

employeeRepository.flush();

employeeRepository.clear();

}

}

}

}