**Exercise 1: Employee Managemnet System Overview and Setup**

<dependencies>

<!-- Spring Data JPA -->

<dependency>

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

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

</dependency>

<!-- H2 Database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Lombok -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<!-- Test Dependencies -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

**Database Configuration**

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

**Hibernate Configuration**

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

spring.jpa.show-sql=true

spring.h2.console.enabled=true

mvn spring-boot:run

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

**Department Entity**

import javax.persistence.\*;

import java.util.List;

@Entity

@Table(name = "departments")

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;

public Department() {}

public Department(String name) {

this.name = name;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public List<Employee> getEmployees() {

return employees;

}

public void setEmployees(List<Employee> employees) {

this.employees = employees;

}

}

**Employee Entity**

import javax.persistence.\*;

@Entity

@Table(name = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false)

private String name;

@Column(nullable = false, unique = true)

String email;

@ManyToOne

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

private Department department;

public Employee() {}

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

this.name = name;

this.email = email;

this.department = department;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public Department getDepartment() {

return department;

}

public void setDepartment(Department department) {

this.department = department;

}

}

**Exercise 3: Creating Repositories**

**Employee Repository Interface**

package com.example.employeemanagement.repository;

import com.example.employeemanagement.entity.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

**Department Repository Interface**

package com.example.employeemanagement.repository

import com.example.employeemanagement.entity.Department;

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

public interface DepartmentRepository extends JpaRepository<Department, Long> {

}

**Employee Repository**

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

List<Employee> findByFirstName(String firstName);

List<Employee> findByLastNameAndDepartmentName(String lastName, String departmentName); List<Employee> findBySalaryGreaterThan(Double salary);

}

import java.util.Optional;

**Department Repository**

public interface DepartmentRepository extends JpaRepository<Department, Long> {

Optional<Department> findByName(String name);

List<Department> findByEmployeeCountGreaterThan(int count);

}

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

import org.springframework.stereotype.Service;

import com.example.employeemanagement.entity.Employee;

import com.example.employeemanagement.repository.EmployeeRepository;

import java.util.List;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

public Employee getEmployeeById(Long id) {

return employeeRepository.findById(id).orElse(null);

}

public Employee saveEmployee(Employee employee) {

return employeeRepository.save(employee);

}

public void deleteEmployee(Long id) {

employeeRepository.deleteById(id);

}

}

**Exercise 4:Implementing CRUD Operations**

**Employee Service:**

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

import org.springframework.stereotype.Service;

import com.example.employeemanagement.entity.Employee;

import com.example.employeemanagement.repository.EmployeeRepository;

import java.util.List;

import java.util.Optional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public Employee saveEmployee(Employee employee) {

return employeeRepository.save(employee);

}

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

public Optional<Employee> getEmployeeById(Long id) {

return employeeRepository.findById(id);

}

public Employee updateEmployee(Employee employee) {

return employeeRepository.save(employee);

}

public void deleteEmployee(Long id) {

employeeRepository.deleteById(id);

}

}

**Department Service**

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

import org.springframework.stereotype.Service;

import com.example.employeemanagement.entity.Department;

import com.example.employeemanagement.repository.DepartmentRepository;

import java.util.List;

import java.util.Optional;

@Service

public class DepartmentService {

@Autowired

private DepartmentRepository departmentRepository;

public Department saveDepartment(Department department) {

return departmentRepository.save(department);

}

public List<Department> getAllDepartments() {

return departmentRepository.findAll();

}

public Optional<Department> getDepartmentById(Long id) {

return departmentRepository.findById(id);

}

public Department updateDepartment(Department department) {

return departmentRepository.save(department);

}

public void deleteDepartment(Long id) {

departmentRepository.deleteById(id);

}

}

**Employee Controller**

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

import org.springframework.http.ResponseEntity;

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

import com.example.employeemanagement.entity.Employee;

import com.example.employeemanagement.service.EmployeeService;

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

return employeeService.saveEmployee(employee);

}

@GetMapping

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

@GetMapping("/{id}")

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

Optional<Employee> employee = employeeService.getEmployeeById(id);

return employee.map(ResponseEntity::ok).orElseGet(() -> ResponseEntity.notFound().build());

}

@PutMapping("/{id}")

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

Optional<Employee> employee = employeeService.getEmployeeById(id);

if (employee.isPresent()) {

Employee updatedEmployee = employee.get();

updatedEmployee.setFirstName(employeeDetails.getFirstName());

updatedEmployee.setLastName(employeeDetails.getLastName());

updatedEmployee.setSalary(employeeDetails.getSalary());

updatedEmployee.setDepartment(employeeDetails.getDepartment());

return ResponseEntity.ok(employeeService.updateEmployee(updatedEmployee));

} else {

return ResponseEntity.notFound().build();

}

}

@DeleteMapping("/{id}")

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

employeeService.deleteEmployee(id);

return ResponseEntity.noContent().build();

}

}

**Depatment Controller**

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

import org.springframework.http.ResponseEntity;

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

import com.example.employeemanagement.entity.Department;

import com.example.employeemanagement.service.DepartmentService;

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/departments")

public class DepartmentController {

@Autowired

private DepartmentService departmentService;

@PostMapping

public Department createDepartment(@RequestBody Department department) {

return departmentService.saveDepartment(department);

**Exercise 6:Implementing pagination and Sorting**

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String firstName;

private String lastName;

private String email;

private String department;

}

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public Page<Employee> getAllEmployees(Pageable pageable) {

return employeeRepository.findAll(pageable);

}

}

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@GetMapping

public ResponseEntity<Page<Employee>> getAllEmployees(

@RequestParam(defaultValue = "0") int page,

@RequestParam(defaultValue = "10") int size,

@RequestParam(defaultValue = "id,asc") String[] sort

) {

Sort.Direction direction = sort[1].equalsIgnoreCase("desc") ? Sort.Direction.DESC : Sort.Direction.ASC;

Sort sortOrder = Sort.by(direction, sort[0]);

Pageable pageable = PageRequest.of(page, size, sortOrder);

Page<Employee> employees = employeeService.getAllEmployees(pageable);

return new ResponseEntity<>(employees, HttpStatus.OK);

}

}

Test Pagnitation: ’/api/employees?page=0&size=5’

Test Sorting : ’/api/employees?page=0&size=5&sort=lastName,desc’

**Exercise 7:Enabling Entity Auditing**

**Employee entity Class:**

import javax.persistence.Entity;

@Entity

public class Employee extends Auditable<String> {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String firstName;

private String lastName;

private String email;

private String department;

}

import javax.persistence.Entity;

@Entity

public class Department extends Auditable<String> {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

}

**Auditable Base Class**

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.EntityListeners;

import javax.persistence.MappedSuperclass;

import java.time.LocalDateTime;

@MappedSuperclass

@EntityListeners(AuditingEntityListener.class)

public abstract class Auditable<U> {

@CreatedBy

protected U createdBy;

@CreatedDate

protected LocalDateTime createdDate;

@LastModifiedBy

protected U lastModifiedBy;

@LastModifiedDate

protected LocalDateTime lastModifiedDate;

}

**Exercise 8:Creating Projections**

**Interface Based Projection**

public interface EmployeeNameProjection {

String getFirstName();

String getLastName();

}

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

List<EmployeeNameProjection> findAllByDepartment(String department);

}

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public List<EmployeeNameProjection> getEmployeeNamesByDepartment(String department) {

return employeeRepository.findAllByDepartment(department);

}

}

**Class Based Projection**

public class EmployeeDetailsDTO {

private String fullName;

private String email;

public EmployeeDetailsDTO(String firstName, String lastName, String email) {

this.fullName = firstName + " " + lastName;

this.email = email;

}

}

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

@Query("SELECT new com.example.EmployeeDetailsDTO(e.firstName, e.lastName, e.email) FROM Employee e WHERE e.department = :department")

List<EmployeeDetailsDTO> findEmployeeDetailsByDepartment(@Param("department") String department);

}

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public List<EmployeeDetailsDTO> getEmployeeDetailsByDepartment(String department) {

return employeeRepository.findEmployeeDetailsByDepartment(department);

}

}

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

List<EmployeeSummaryProjection> findByDepartment(String department);

}

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public List<EmployeeSummaryProjection> getEmployeeSummariesByDepartment(String department) {

return employeeRepository.findByDepartment(department);

}

}

**Exercise 9:Customize Data Source Configuration**

**Data Sources**

Primary Data Source

spring.datasource.primary.url=jdbc:mysql://localhost:3306/employee\_db

spring.datasource.primary.username=root

spring.datasource.primary.password=password

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

spring.datasource.primary.hikari.maximum-pool-size=10

Secondary Data Source

spring.datasource.secondary.url=jdbc:mysql://localhost:3306/department\_db

spring.datasource.secondary.username=root

spring.datasource.secondary.password=password

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

spring.datasource.secondary.hikari.maximum-pool-size=10

**Configuration Class**

@Configuration

@EnableTransactionManagement

@EnableJpaRepositories(

basePackages = "com.example.employee.repository.primary", // Repository package for primary DB

entityManagerFactoryRef = "primaryEntityManagerFactory",

transactionManagerRef = "primaryTransactionManager"

)

public class PrimaryDataSourceConfig {

@Primary

@Bean(name = "primaryDataSource")

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

public DataSource dataSource() {

return DataSourceBuilder.create().build();

}

@Primary

@Bean(name = "primaryEntityManagerFactory")

public LocalContainerEntityManagerFactoryBean primaryEntityManagerFactory(EntityManagerFactoryBuilder builder,

@Qualifier("primaryDataSource") DataSource dataSource) {

return builder

.dataSource(dataSource)

.packages("com.example.employee.model.primary") // Model package for primary DB

.persistenceUnit("primary")

.build();

}

@Primary

@Bean(name = "primaryTransactionManager")

public PlatformTransactionManager primaryTransactionManager(

@Qualifier("primaryEntityManagerFactory") EntityManagerFactory entityManagerFactory) {

return new JpaTransactionManager(entityManagerFactory);

}

}

@Configuration

@EnableTransactionManagement

@EnableJpaRepositories(

basePackages = "com.example.employee.repository.secondary", // Repository package for secondary DB

entityManagerFactoryRef = "secondaryEntityManagerFactory",

transactionManagerRef = "secondaryTransactionManager"

)

public class SecondaryDataSourceConfig {

@Bean(name = "secondaryDataSource")

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

public DataSource dataSource() {

return DataSourceBuilder.create().build();

}

@Bean(name = "secondaryEntityManagerFactory")

public LocalContainerEntityManagerFactoryBean secondaryEntityManagerFactory(EntityManagerFactoryBuilder builder,

@Qualifier("secondaryDataSource") DataSource dataSource) {

return builder

.dataSource(dataSource)

.packages("com.example.employee.model.secondary") // Model package for secondary DB

.persistenceUnit("secondary")

.build();

}

@Bean(name = "secondaryTransactionManager")

public PlatformTransactionManager secondaryTransactionManager(

@Qualifier("secondaryEntityManagerFactory") EntityManagerFactory entityManagerFactory) {

return new JpaTransactionManager(entityManagerFactory);

}

}

**Controller Layer Implementation**

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

return employeeService.saveEmployee(employee);

}

@PostMapping("/audit")

public AuditLog createAuditLog(@RequestBody AuditLog auditLog) {

return employeeService.saveAuditLog(auditLog);

}

}

**Exercise 10:Enhance Employee Management System using Hibernate Features**

**NaturalId**

import org.hibernate.annotations.NaturalId;

@Entity

@Table(name = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@NaturalId

@Column(unique = true, nullable = false)

private String employeeCode;

private String name;

private String position;

}

**Batch SIze**

import org.hibernate.annotations.BatchSize;

@Entity

@Table(name = "departments")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "department", fetch = FetchType.LAZY)

@BatchSize(size = 10)

private List<Employee> employees;

}

**Immutable**

import org.hibernate.annotations.Immutable;

@Entity

@Immutable

@Table(name = "audit\_logs")

public class AuditLog {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String action;

private LocalDateTime timestamp;

}

Hibernate Dialect for MySQL

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

Other Hibernate properties

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

spring.jpa.properties.hibernate.use\_sql\_comments=true

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.cache.use\_second\_level\_cache=true

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

**Batch Insert or Delete**

import org.springframework.transaction.annotation.Transactional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void batchInsertEmployees(List<Employee> employees) {

int batchSize = 20;

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

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

if (i % batchSize == 0 && i > 0) {

employeeRepository.flush();

employeeRepository.clear();

}

}

}

}

**Batch Delete**

@Service

public class EmployeeService {

@Autowired

private EntityManager entityManager;

@Transactional

public void batchDeleteEmployees(List<Long> employeeIds) {

Query query = entityManager.createQuery("DELETE FROM Employee e WHERE e.id IN :ids");

query.setParameter("ids", employeeIds);

query.executeUpdate();

}

}