# SPRING\_DATA\_JPA

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

###### EmployeeManagementSystemApplication.java

package com.ems;

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);

}

}

entity/Department.java

package com.ems.entity;

import jakarta.persistence.\*; import lombok.\*;

import java.util.List;

@Entity @Data

@NoArgsConstructor @AllArgsConstructor public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private Integer id;

private String name;

@OneToMany(mappedBy = "department", cascade = CascadeType.ALL) private List<Employee> employees;

}

# entity/Employee.java

package com.ems.entity;

import jakarta.persistence.\*; import lombok.\*;

@Entity @Data

@NoArgsConstructor @AllArgsConstructor public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private Integer id;

private String name; private Double salary;

@ManyToOne

@JoinColumn(name = "department\_id") private Department department;

}

#### repository/EmployeeRepository.java

package com.ems.repository;

import com.ems.entity.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

repository/DepartmentRepository.java

package com.ems.repository; import com.ems.entity.Department;

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

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

service/EmployeeService.java

package com.ems.service;

import com.ems.entity.Employee;

import com.ems.repository.EmployeeRepository;

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

import java.util.List;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public List<Employee> getAllEmployees() { return employeeRepository.findAll();

}

public Employee saveEmployee(Employee employee) { return employeeRepository.save(employee);

}

}

#### controller/EmployeeController.java

package com.ems.controller;

import com.ems.entity.Employee;

import com.ems.service.EmployeeService;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/employees") public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@GetMapping

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) { return employeeService.saveEmployee(employee);

}

}

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

Department.java

package com.ems.entity;

import jakarta.persistence.\*; import lombok.\*;

import java.util.List;

@Entity

@Table(name = "department") @Data

@NoArgsConstructor @AllArgsConstructor public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private Integer id;

private String name;

// One department can have many employees

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

private List<Employee> employees;

}

#### Employee.java

package com.ems.entity;

import jakarta.persistence.\*; import lombok.\*;

@Entity

@Table(name = "employee") @Data

@NoArgsConstructor @AllArgsConstructor public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private Integer id;

private String name;

private String email;

// Many employees can belong to one department @ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

}

#### Exercise 3: Creating Repositories for the Employee Management System

EmployeeRepository.java

package com.ems.repository;

import com.ems.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, Integer> {

// Derived query methods

List<Employee> findByName(String name);

List<Employee> findByDepartmentId(Integer departmentId); List<Employee> findByEmailContaining(String keyword);

}

#### DepartmentRepository.java

package com.ems.repository;

import com.ems.entity.Department;

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

import org.springframework.stereotype.Repository;

@Repository

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

// Derived query methods

Department findByName(String name);

}

#### Exercise 4: Implementing CRUD Operations for the

Employee Management System

##### EmployeeService.java

package com.ems.service;

import com.ems.entity.Employee;

import com.ems.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;

public Employee createEmployee(Employee employee) { return employeeRepository.save(employee);

}

public List<Employee> getAllEmployees() { return employeeRepository.findAll();

}

public Optional<Employee> getEmployeeById(int id) { return employeeRepository.findById(id);

}

public Employee updateEmployee(int id, Employee updatedEmployee) { updatedEmployee.setId(id);

return employeeRepository.save(updatedEmployee);

}

public void deleteEmployee(int id) { employeeRepository.deleteById(id);

}

}

#### DepartmentService.java

package com.ems.service;

import com.ems.entity.Department;

import com.ems.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;

public Department createDepartment(Department department) { return departmentRepository.save(department);

}

public List<Department> getAllDepartments() { return departmentRepository.findAll();

}

public Optional<Department> getDepartmentById(int id) {

return departmentRepository.findById(id);

}

public Department updateDepartment(int id, Department updatedDepartment) { updatedDepartment.setId(id);

return departmentRepository.save(updatedDepartment);

}

public void deleteDepartment(int id) {

departmentRepository.deleteById(id);

}

}

## EmployeeController.java

package com.ems.controller;

import com.ems.entity.Employee;

import com.ems.service.EmployeeService;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/employees") public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) { return employeeService.createEmployee(employee);

}

@GetMapping

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

@GetMapping("/{id}")

public Employee getEmployeeById(@PathVariable int id) { return employeeService.getEmployeeById(id)

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

}

@PutMapping("/{id}")

public Employee updateEmployee(@PathVariable int id, @RequestBody Employee employee) {

return employeeService.updateEmployee(id, employee);

}

@DeleteMapping("/{id}")

public void deleteEmployee(@PathVariable int id) { employeeService.deleteEmployee(id);

}

}

#### DepartmentController.java

package com.ems.controller;

import com.ems.entity.Department;

import com.ems.service.DepartmentService;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/departments") public class DepartmentController {

@Autowired

private DepartmentService departmentService;

@PostMapping

public Department createDepartment(@RequestBody Department department) { return departmentService.createDepartment(department);

}

@GetMapping

public List<Department> getAllDepartments() {

return departmentService.getAllDepartments();

}

@GetMapping("/{id}")

public Department getDepartmentById(@PathVariable int id) { return departmentService.getDepartmentById(id)

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

}

@PutMapping("/{id}")

public Department updateDepartment(@PathVariable int id, @RequestBody Department department) {

return departmentService.updateDepartment(id, department);

}

@DeleteMapping("/{id}")

public void deleteDepartment(@PathVariable int id) { departmentService.deleteDepartment(id);

}

}

#### Exercise 5: Deflning Query Methods

###### EmployeeRepository.java

package com.ems.repository; import com.ems.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, Integer> {

// Derived Query Methods

List<Employee> findByName(String name);

List<Employee> findByDepartmentName(String departmentName); List<Employee> findByEmailContaining(String keyword);

}

1. Custom Queries Using @Query

java

CopyEdit

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

@Query("SELECT e FROM Employee e WHERE e.email LIKE %:keyword%") List<Employee> searchByEmailKeyword(@Param("keyword") String keyword);

@Query("SELECT e FROM Employee e WHERE e.department.name = :deptName") List<Employee> getEmployeesByDepartment(@Param("deptName") String deptName);

}

#### Named Queries

Employee.java

package com.ems.entity;

import jakarta.persistence.\*;

import lombok.\*;

@Entity

@Table(name = "employee") @Data

@NoArgsConstructor @AllArgsConstructor @NamedQueries({

@NamedQuery(name = "Employee.findByEmail",

query = "SELECT e FROM Employee e WHERE e.email = :email"), @NamedQuery(name = "Employee.findByDeptId",

query = "SELECT e FROM Employee e WHERE e.department.id = :deptId")

})

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private int id;

private String name; private String email;

@ManyToOne

@JoinColumn(name = "department\_id") private Department department;

}

#### Exercise 6: Implementing Pagination and Sorting

###### EmployeeRepository.java:

package com.ems.repository; import com.ems.entity.Employee;

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

import org.springframework.data.domain.Page; import org.springframework.data.domain.Pageable; import org.springframework.stereotype.Repository;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

Page<Employee> findAll(Pageable pageable);

Page<Employee> findByDepartmentName(String deptName, Pageable pageable);

}

#### EmployeeService.java

package com.ems.service;

import com.ems.entity.Employee;

import com.ems.repository.EmployeeRepository;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.data.domain.\*;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public Page<Employee> getPaginatedEmployees(int page, int size, String sortField, String sortDir) {

Sort sort = sortDir.equalsIgnoreCase("asc") ?

Sort.by(sortField).ascending() : Sort.by(sortField).descending();

Pageable pageable = PageRequest.of(page, size, sort); return employeeRepository.findAll(pageable);

}

public Page<Employee> getEmployeesByDepartment(String deptName, int page, int size, String sortField, String sortDir) {

Sort sort = sortDir.equalsIgnoreCase("asc") ?

Sort.by(sortField).ascending() :

Sort.by(sortField).descending();

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

return employeeRepository.findByDepartmentName(deptName, pageable);

}

}

#### EmployeeController.java

package com.ems.controller;

import com.ems.entity.Employee;

import com.ems.service.EmployeeService;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.data.domain.Page;

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

@RestController

@RequestMapping("/employees") public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@GetMapping("/paginated")

public Page<Employee> getPaginatedEmployees( @RequestParam(defaultValue = "0") int page, @RequestParam(defaultValue = "5") int size,

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

) {

return employeeService.getPaginatedEmployees(page, size, sortBy, sortDir);

}

@GetMapping("/by-department")

public Page<Employee> getEmployeesByDepartment( @RequestParam String dept,

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

@RequestParam(defaultValue = "5") int size,

@RequestParam(defaultValue = "name") String sortBy, @RequestParam(defaultValue = "asc") String sortDir

) {

return employeeService.getEmployeesByDepartment(dept, page, size, sortBy, sortDir);

}

}

# Output

{

"content": [

{

"id": 1,

"name": "Alice",

"email": "[alice@example.com](mailto:alice@example.com)", "department": {

"id": 101,

"name": "HR"

}

},

...

],

"totalPages": 3,

"totalElements": 9, "last": false,

"size": 3,

"number": 0, "sort": { ... }, "first": true,

"numberOfElements": 3, "empty": false

}

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

##### EmployeeManagementSystemApplication.java

package com.ems;

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);

}

}

Auditable.java

package com.ems.audit;

import jakarta.persistence.\*; import lombok.Getter; import lombok.Setter;

import org.springframework.data.annotation.\*;

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

import java.time.LocalDateTime;

@MappedSuperclass

@EntityListeners(AuditingEntityListener.class) @Getter

@Setter

public abstract class Auditable {

@CreatedBy

@Column(updatable = false) protected String createdBy;

@CreatedDate

@Column(updatable = false)

protected LocalDateTime createdDate;

@LastModifiedBy

protected String lastModifiedBy;

@LastModifiedDate

protected LocalDateTime lastModifiedDate;

}

#### Employee.java

package com.ems.entity;

import com.ems.audit.Auditable; import jakarta.persistence.\*; import lombok.\*;

@Entity @Data

@NoArgsConstructor @AllArgsConstructor

public class Employee extends Auditable {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private int id;

private String name;

private String email;

@ManyToOne

@JoinColumn(name = "department\_id") private Department department;

}

#### Department.java

package com.ems.entity;

import com.ems.audit.Auditable; import jakarta.persistence.\*; import lombok.\*;

import java.util.List;

@Entity @Data

@NoArgsConstructor @AllArgsConstructor

public class Department extends Auditable {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private int id;

private String name;

@OneToMany(mappedBy = "department") private List<Employee> employees;

}

#### AuditorAwareConflg.java

package com.ems.config;

import org.springframework.context.annotation.Configuration; import org.springframework.context.annotation.Bean;

import org.springframework.data.domain.AuditorAware;

import java.util.Optional;

@Configuration

public class AuditorAwareConfig {

@Bean

public AuditorAware<String> auditorProvider() {

return () -> Optional.of("system\_user"); // Replace with actual user logic

}

}

## output

{

"id": 1,

"name": "John",

"email": ["john@example.com](mailto:john@example.com)", "department": {

"id": 1,

"name": "IT"

},

"createdBy": "system\_user",

"createdDate": "2025-07-05T10:12:33",

"lastModifiedBy": "system\_user", "lastModifiedDate": "2025-07-05T10:12:33"

}

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

#### EmployeeNameAndEmail.java

package com.ems.projection;

public interface EmployeeNameAndEmail { String getName();

String getEmail();

}

##### EmployeeRepository.java

package com.ems.repository;

import com.ems.entity.Employee;

import com.ems.projection.EmployeeNameAndEmail;

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

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

// Interface-based projection

@Query("SELECT e FROM Employee e") List<EmployeeNameAndEmail> findAllNameAndEmail();

}

#### EmployeeDTO.java

package com.ems.dto;

import lombok.AllArgsConstructor; import lombok.Data;

@Data

@AllArgsConstructor public class EmployeeDTO {

private String name;

private String departmentName;

}

##### EmployeeRepository.java

package com.ems.repository;

import com.ems.dto.EmployeeDTO; import com.ems.entity.Employee;

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

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

@Query("SELECT new com.ems.dto.EmployeeDTO(e.name, e.department.name) FROM Employee e")

List<EmployeeDTO> fetchNameAndDeptName();

}

#### EmployeeService .java

package com.ems.service;

import com.ems.dto.EmployeeDTO;

import com.ems.projection.EmployeeNameAndEmail; import com.ems.repository.EmployeeRepository; import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class EmployeeService {

private final EmployeeRepository repository;

public EmployeeService(EmployeeRepository repository) { this.repository = repository;

}

public List<EmployeeNameAndEmail> getNameAndEmailOnly() { return repository.findAllNameAndEmail();

}

public List<EmployeeDTO> getNameAndDeptName() { return repository.fetchNameAndDeptName();

}

}

#### EmployeeController.java

package com.ems.controller;

import com.ems.dto.EmployeeDTO;

import com.ems.projection.EmployeeNameAndEmail; import com.ems.service.EmployeeService;

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

import java.util.List;

@RestController

public class EmployeeController {

private final EmployeeService service;

public EmployeeController(EmployeeService service) { this.service = service;

}

@GetMapping("/employees/projection/name-email")

public List<EmployeeNameAndEmail> getNamesAndEmails() { return service.getNameAndEmailOnly();

}

@GetMapping("/employees/projection/name-department")

public List<EmployeeDTO> getNamesAndDepartments() { return service.getNameAndDeptName();

}

}

Exercise 9: Employee Management System - Customizing Data Source Conflguration

### LoggingDataSourceConflg.java

package com.ems.config;

import javax.sql.DataSource;

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

import org.springframework.boot.autoconfigure.jdbc.DataSourceProperties; import org.springframework.boot.context.properties.ConfigurationProperties; import org.springframework.context.annotation.\*;

@Configuration

public class LoggingDataSourceConfig {

@Bean

@ConfigurationProperties("app.datasource.logging")

public DataSourceProperties loggingDataSourceProperties() { return new DataSourceProperties();

}

@Bean(name = "loggingDataSource") public DataSource loggingDataSource() {

return loggingDataSourceProperties().initializeDataSourceBuilder().build();

}

}

Exercise 10: Employee Management System - Hibernate- Speciflc Features

## Employee .java

package com.ems.entity;

import jakarta.persistence.\*; import org.hibernate.annotations.\*;

import java.time.LocalDate;

@Entity

@Table(name = "employee")

@BatchSize(size = 10) // Batch fetch size for collections or associations public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private Integer id;

private String name;

private String email;

@ManyToOne(fetch = FetchType.LAZY) @JoinColumn(name = "department\_id")

private Department department;

@Column(name = "salary") private Double salary;

@Type(type = "org.hibernate.type.LocalDateType") // Hibernate-specific type for LocalDate private LocalDate joiningDate;

@Formula("(SELECT COUNT(\*) FROM employee e2 WHERE e2.department\_id = department\_id)")

private int departmentEmployeeCount; // Computed column - employees in the same department

// getters and setters

}

#### EmployeeService.java

@Service

public class EmployeeService {

@PersistenceContext

private EntityManager entityManager;

@Transactional

public void saveEmployeesBatch(List<Employee> employees) { int batchSize = 20;

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

entityManager.persist(employees.get(i)); if (i > 0 && i % batchSize == 0) {

entityManager.flush();

entityManager.clear();

}

}

entityManager.flush(); entityManager.clear();

}

}