**POM.XML**

<project xmlns="http://maven.apache.org/POM/4.0.0" ...>

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>EmployeeManagementSystem</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

</dependencies>

</project>

**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

**Exercise 2: Entities**

**Department.java**

package com.example.EmployeeManagementSystem.model;

import jakarta.persistence.\*;

import lombok.\*;

import java.util.List;

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

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

private List<Employee> employees;

}

**Employee.java**

package com.example.EmployeeManagementSystem.model;

import jakarta.persistence.\*;

import lombok.\*;

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

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;

}

**Exercise 3: Repositories**

**DepartmentRepository.java**

package com.example.EmployeeManagementSystem.repository;

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

import com.example.EmployeeManagementSystem.model.Department;

public interface DepartmentRepository extends JpaRepository<Department, Long> {

}

**EmployeeRepository.java**

package com.example.EmployeeManagementSystem.repository;

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

import com.example.EmployeeManagementSystem.model.Employee;

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

List<Employee> findByNameContaining(String name);

}

**Exercise 4: CRUD Controllers**

**DepartmentController.java**

package com.example.EmployeeManagementSystem.controller;

import com.example.EmployeeManagementSystem.model.Department;

import com.example.EmployeeManagementSystem.repository.DepartmentRepository;

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 DepartmentRepository departmentRepository;

@GetMapping

public List<Department> getAll() {

return departmentRepository.findAll();

}

@PostMapping

public Department create(@RequestBody Department department) {

return departmentRepository.save(department);

}

@PutMapping("/{id}")

public Department update(@PathVariable Long id, @RequestBody Department dept) {

dept.setId(id);

return departmentRepository.save(dept);

}

@DeleteMapping("/{id}")

public void delete(@PathVariable Long id) {

departmentRepository.deleteById(id);

}

}

**EmployeeController.java**

package com.example.EmployeeManagementSystem.controller;

import com.example.EmployeeManagementSystem.model.Employee;

import com.example.EmployeeManagementSystem.repository.EmployeeRepository;

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 EmployeeRepository employeeRepository;

@GetMapping

public List<Employee> getAll() {

return employeeRepository.findAll();

}

@PostMapping

public Employee create(@RequestBody Employee employee) {

return employeeRepository.save(employee);

}

@PutMapping("/{id}")

public Employee update(@PathVariable Long id, @RequestBody Employee employee) {

employee.setId(id);

return employeeRepository.save(employee);

}

@DeleteMapping("/{id}")

public void delete(@PathVariable Long id) {

employeeRepository.deleteById(id);

}

}

**Exercise 5: Query Methods**

**EmployeeRepository.java:**

List<Employee> findByEmailContaining(String email);

List<Employee> findByDepartmentName(String name);

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

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

**Exercise 6: Pagination & Sorting**

EmployeeController.java:

@GetMapping("/page")

public Page<Employee> getPaginated(

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

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

@RequestParam(defaultValue = "name") String sortBy

) {

return employeeRepository.findAll(PageRequest.of(page, size, Sort.by(sortBy)));

}

**Exercise 7: Entity Auditing**

<dependency>

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

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

</dependency>

**Auditable.java:**

@MappedSuperclass

@EntityListeners(AuditingEntityListener.class)

@Getter

@Setter

public abstract class Auditable {

@CreatedDate

private LocalDateTime createdDate;

@LastModifiedDate

private LocalDateTime modifiedDate;

}

Enable Auditing:

java

Copy code

@SpringBootApplication

@EnableJpaAuditing

public class EmployeeManagementSystemApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementSystemApplication.class, args);

}

}

**Exercise 8: Projections**

public interface EmployeeNameProjection {

String getName();

}

**EmployeeRepository.java:**

@Query("SELECT e.name as name FROM Employee e")

List<EmployeeNameProjection> fetchOnlyNames();

Exercise 9: Data Source Customization

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

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

spring.datasource.secondary.username=sa

spring.datasource.secondary.password=password

**Exercise 10: Hibernate-Specific Features**

@Entity

@DynamicInsert

@BatchSize(size = 25)

public class Employee {

...

}

application.properties

spring.jpa.properties.hibernate.jdbc.batch\_size=30

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