Spring Data JPA and Hibernate - Solved Exercises

# Exercise 1: Employee Management System - Overview and Setup

Create a Spring Boot project using Spring Initializr:  
- Name: EmployeeManagementSystem  
- Dependencies: Spring Web, Spring Data JPA, H2 Database, Lombok  
  
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

# Exercise 2: Creating Entities

@Entity  
public class Department {  
 @Id @GeneratedValue  
 private Long id;  
 private String name;  
 @OneToMany(mappedBy = "department")  
 private List<Employee> employees;  
}  
  
@Entity  
public class Employee {  
 @Id @GeneratedValue  
 private Long id;  
 private String name;  
 private String email;  
 @ManyToOne  
 @JoinColumn(name = "department\_id")  
 private Department department;  
}

# Exercise 3: Creating Repositories

@Repository  
public interface DepartmentRepository extends JpaRepository<Department, Long> {}  
  
@Repository  
public interface EmployeeRepository extends JpaRepository<Employee, Long> {}

# Exercise 4: Implementing CRUD Operations

@RestController  
@RequestMapping("/employees")  
public class EmployeeController {  
 @Autowired private EmployeeRepository repository;  
  
 @GetMapping public List<Employee> getAll() { return repository.findAll(); }  
 @PostMapping public Employee create(@RequestBody Employee emp) { return repository.save(emp); }  
 @PutMapping("/{id}") public Employee update(@PathVariable Long id, @RequestBody Employee emp) {  
 emp.setId(id);  
 return repository.save(emp);  
 }  
 @DeleteMapping("/{id}") public void delete(@PathVariable Long id) { repository.deleteById(id); }  
}

# Exercise 5: Defining Query Methods

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

# Exercise 6: Pagination and Sorting

@GetMapping("/paged")  
public Page<Employee> getPaged(Pageable pageable) {  
 return repository.findAll(pageable);  
}

# Exercise 7: Enabling Entity Auditing

@EnableJpaAuditing  
  
@EntityListeners(AuditingEntityListener.class)  
public abstract class Auditable {  
 @CreatedDate private Instant createdDate;  
 @LastModifiedDate private Instant lastModifiedDate;  
}  
  
@Entity  
public class Employee extends Auditable { ... }

# Exercise 8: Creating Projections

public interface EmployeeNameView {  
 String getName();  
}  
  
@Query("SELECT new com.example.EmployeeSummary(e.name, e.email) FROM Employee e")  
List<EmployeeSummary> findEmployeeSummaries();

# Exercise 9: Customizing Data Source Configuration

application.properties:  
spring.datasource.url=jdbc:mysql://localhost:3306/db1  
spring.datasource.username=root  
  
# Define beans for multiple datasources using @Primary and @Qualifier

# Exercise 10: Hibernate-Specific Features

@BatchSize(size = 25)  
@DynamicUpdate  
@SelectBeforeUpdate  
  
spring.jpa.properties.hibernate.jdbc.batch\_size=30  
spring.jpa.properties.hibernate.order\_inserts=true  
spring.jpa.properties.hibernate.order\_updates=true