Spring Data JPA and Hibernate - Solutions

# Exercise 1: Overview and Setup

1. Create Spring Boot Project using Spring Initializr:  
 - Project Name: EmployeeManagementSystem  
 - Dependencies: Spring Web, Spring Data JPA, H2 Database, Lombok  
  
2. application.properties:

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: Creating Entities

1. Department Entity:

java  
@Entity  
public class Department {  
 @Id @GeneratedValue  
 private Long id;  
 private String name;  
  
 @OneToMany(mappedBy = "department")  
 private List<Employee> employees;  
}  
  
  
2. Employee Entity:

java  
@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

1. EmployeeRepository.java:

java  
public interface EmployeeRepository extends JpaRepository<Employee, Long> {  
 List<Employee> findByName(String name);  
}  
  
  
2. DepartmentRepository.java:

java  
public interface DepartmentRepository extends JpaRepository<Department, Long> {}

# Exercise 4: Implementing CRUD Operations

1. EmployeeController.java:

java  
@RestController  
@RequestMapping("/employees")  
public class EmployeeController {  
 @Autowired private EmployeeRepository repository;  
  
 @PostMapping public Employee create(@RequestBody Employee e) { return repository.save(e); }  
 @GetMapping("/{id}") public Employee read(@PathVariable Long id) { return repository.findById(id).orElse(null); }  
 @PutMapping("/{id}") public Employee update(@PathVariable Long id, @RequestBody Employee e) {  
 e.setId(id); return repository.save(e);  
 }  
 @DeleteMapping("/{id}") public void delete(@PathVariable Long id) { repository.deleteById(id); }  
}  
  
  
2. DepartmentController is similar.

# Exercise 5: Defining Query Methods

1. Derived Query:

java  
List<Employee> findByDepartmentName(String deptName);  
  
  
2. Custom @Query:

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

3. Named Query:

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

# Exercise 6: Pagination and Sorting

1. Repository method with pagination:

java  
Page<Employee> findAll(Pageable pageable);  
  
  
2. Controller example:

java  
@GetMapping  
public Page<Employee> getAll(@RequestParam int page, @RequestParam int size) {  
 return repository.findAll(PageRequest.of(page, size, Sort.by("name")));  
}

# Exercise 7: Entity Auditing

1. Enable JPA Auditing in main class:

java  
@EnableJpaAuditing  
@SpringBootApplication  
public class App {}

2. Base Entity:

java  
@MappedSuperclass  
@EntityListeners(AuditingEntityListener.class)  
public abstract class Auditable {  
 @CreatedDate private LocalDateTime createdDate;  
 @LastModifiedDate private LocalDateTime modifiedDate;  
}  
  
  
3. Extend this base class in Employee and Department.

# Exercise 8: Creating Projections

1. Interface-based projection:

java  
public interface EmployeeNameView {  
 String getName();  
}  
  
  
2. Repository usage:  
java  
List<EmployeeNameView> findBy();

# Exercise 9: Customizing Data Source Configuration

1. application.properties:

properties  
spring.datasource.url=jdbc:mysql://localhost:3306/db1  
spring.datasource.username=root  
spring.datasource.password=pass  
  
spring.second-datasource.url=jdbc:mysql://localhost:3306/db2  
spring.second-datasource.username=root  
spring.second-datasource.password=pass  
  
  
2. Define configuration beans for each data source with @Primary on main one.

# Exercise 10: Hibernate-Specific Features

1. Hibernate annotations:

java  
@DynamicUpdate  
@SelectBeforeUpdate  
  
  
2. Batch processing config:

properties  
spring.jpa.properties.hibernate.jdbc.batch\_size=30  
spring.jpa.properties.hibernate.order\_inserts=true  
  
  
3. Use Session for bulk inserts if necessary.