**Exercise 1: Project Setup**

**Spring Boot Configuration (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**

// Department.java

@Entity

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "department")

private List<Employee> employees;

}

// Employee.java

@Entity

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**

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {}

@Repository

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

**Exercise 4: CRUD Operations**

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepo;

@PostMapping

public Employee save(@RequestBody Employee emp) {

return employeeRepo.save(emp);

}

@GetMapping("/{id}")

public Employee get(@PathVariable Long id) {

return employeeRepo.findById(id).orElseThrow();

}

// Similar for update and delete

}

**Exercise 5: Query Methods**

// Derived query method

List<Employee> findByNameContaining(String name);

// Custom query

@Query("SELECT e FROM Employee e WHERE e.email = ?1")

Employee findByEmail(String email);

// Named query

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

**Exercise 6: Pagination and Sorting**

@GetMapping("/employees")

public Page<Employee> getAll(

@RequestParam int page,

@RequestParam int size,

@RequestParam String sortBy

) {

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

}

**Exercise 7: Auditing**

1. Enable auditing:

@Configuration

@EnableJpaAuditing

public class JpaConfig {}

1. Base entity class:

@MappedSuperclass

@EntityListeners(AuditingEntityListener.class)

public abstract class Auditable {

@CreatedDate

private LocalDateTime createdDate;

@LastModifiedDate

private LocalDateTime modifiedDate;

}

**Exercise 8: Projections**

// Interface-based projection

public interface EmployeeNameView {

String getName();

}

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

List<EmployeeNameView> findAllNames();

You can also use constructor expressions for class-based projections.

**Exercise 9: Data Source Configuration**

**application.properties**

# Main DB

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

# Additional DBs (via config classes)

spring.second-datasource.url=...

**Exercise 10: Hibernate-Specific Features**

* **Annotations**:

@DynamicUpdate

@SelectBeforeUpdate

* **Batch processing**:

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

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

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