**Hands on 1**

**Spring Data JPA - Quick Example**

create schema ormlearn;

use ormlearn;

create table country (

co\_code varchar(2) primary key,

co\_name varchar(50)

);

insert into country values ('IN', 'India'), ('US', 'United States of America');

**Java Components**

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_name")

private String name;

}

**Repository:**

@Repository

public interface CountryRepository extends JpaRepository<Country, String> { }

**Service**:

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**Main Application:**

@SpringBootApplication

public class OrmLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

private static CountryService countryService;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

LOGGER.info("Inside main");

countryService = context.getBean(CountryService.class);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

List<Country> countries = countryService.getAllCountries();

LOGGER.debug("countries={}", countries);

LOGGER.info("End");

}

}

Output:

06-07-25 17:25:14.321 main INFO com.cognizant.ormlearn.OrmLearnApplication 23 - Inside main

06-07-25 17:25:14.423 main INFO com.cognizant.ormlearn.OrmLearnApplication 27 - Start

06-07-25 17:25:14.534 main TRACE org.hibernate.SQL 109 - select country0\_.co\_code as co\_code1\_0\_, country0\_.co\_name as co\_name2\_0\_ from country country0\_

06-07-25 17:25:14.537 main TRACE org.hibernate.type.descriptor.sql.BasicBinder 65 - binding parameter [1] as [VARCHAR] - [IN]

06-07-25 17:25:14.537 main TRACE org.hibernate.type.descriptor.sql.BasicBinder 65 - binding parameter [2] as [VARCHAR] - [US]

06-07-25 17:25:14.540 main DEBUG com.cognizant.ormlearn.OrmLearnApplication 28 - countries=[Country [code=IN, name=India], Country [code=US, name=United States of America]]

06-07-25 17:25:14.540 main INFO com.cognizant.ormlearn.OrmLearnApplication 29 – End

**Hands on 4**

**Difference between JPA, Hibernate and Spring Data JPA**

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Type** | Specification | Implementation of JPA | Abstraction over JPA implementations like Hibernate |
| **Main Purpose** | Define a standard for ORM | Provide actual implementation of ORM | Reduce boilerplate & simplify JPA-based development |
| **Provides** | Only interfaces and annotations | EntityManager, Session, Query APIs, etc. | Repository Interfaces like JpaRepository, custom queries |
| **Boilerplate Code** | Requires implementation | Requires session/transaction handling | Removes most boilerplate using annotations |
| **Transaction Handling** | Handled manually or via container | Requires manual control or helper classes | Handled automatically using @Transactional |
| **Example Vendors** | EclipseLink, Hibernate, OpenJPA | Hibernate | Works on top of Hibernate or other JPA providers |

**Hibernate Code**

public Integer addEmployee(Employee employee){

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

**EmployeeRepository.java**

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

Output:

Hibernate: insert into employee (name, role, salary) values (?, ?, ?)

Inserted Employee ID: 101

**Hands on 5**

**Implement services for managing Country**

**Country.java**

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

private String co\_code;

private String co\_name;

// Constructors

public Country() {}

public Country(String co\_code, String co\_name) {

this.co\_code = co\_code;

this.co\_name = co\_name;

}

// Getters and Setters

public String getCo\_code() {

return co\_code;

}

public void setCo\_code(String co\_code) {

this.co\_code = co\_code;

}

public String getCo\_name() {

return co\_name;

}

public void setCo\_name(String co\_name) {

this.co\_name = co\_name;

}

}

**CountryRepository.java**

import java.util.List;

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

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByCo\_nameContainingIgnoreCase(String namePart);

}

**CountryService.java**

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public Optional<Country> findByCode(String code) {

return countryRepository.findById(code);

}

public Country addCountry(Country country) {

return countryRepository.save(country);

}

public Country updateCountry(Country country) {

return countryRepository.save(country); // save works for both add and update

}

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

public List<Country> findByPartialName(String namePart) {

return countryRepository.findByCo\_nameContainingIgnoreCase(namePart);

}

}

**CountryController.java**

import java.util.List;

import java.util.Optional;

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

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

@RestController

@RequestMapping("/countries")

public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping("/{code}")

public Optional<Country> getCountryByCode(@PathVariable String code) {

return countryService.findByCode(code);

}

@PostMapping

public Country addCountry(@RequestBody Country country) {

return countryService.addCountry(country);

}

@PutMapping

public Country updateCountry(@RequestBody Country country) {

return countryService.updateCountry(country);

}

@DeleteMapping("/{code}")

public void deleteCountry(@PathVariable String code) {

countryService.deleteCountry(code);

}

@GetMapping("/search")

public List<Country> searchCountry(@RequestParam String name) {

return countryService.findByPartialName(name);

}

}

application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/your\_db\_name

spring.datasource.username=your\_username

spring.datasource.password=your\_password

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

Output:

If the country is not found:

GET /countries/ZZ

**Hands on 6**

**Find a country based on country code**

package com.cognizant.springlearn.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

**CountryService.java**

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import jakarta.transaction.Transactional;

import java.util.Optional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

Optional<Country> result = countryRepository.findById(countryCode);

if (!result.isPresent()) {

throw new CountryNotFoundException("Country not found with code: " + countryCode);

}

return result.get();

}

**}**

**OrmLearnApplication.java**

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

@SpringBootApplication

public class OrmLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

public static CountryService countryService;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

countryService = context.getBean(CountryService.class);

getAllCountriesTest(); // invoking the test method

}

private static void getAllCountriesTest() {

LOGGER.info("Start");

try {

Country country = countryService.findCountryByCode("IN");

LOGGER.debug("Country: {}", country);

} catch (CountryNotFoundException e) {

LOGGER.error("Exception: {}", e.getMessage());

}

LOGGER.info("End");

}

}

Output:

INFO : Start

DEBUG : Country: Country [co\_code=IN, co\_name=India]

INFO : End

**Hands on 7**

**Add a new country**   
 **CountryService.java**

import jakarta.transaction.Transactional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

// Existing methods like findCountryByCode(), etc.

}

**OrmLearnApplication.java**

private static void testAddCountry() {

LOGGER.info("Start");

Country newCountry = new Country();

newCountry.setCo\_code("ZZ");

newCountry.setCo\_name("Testland");

countryService.addCountry(newCountry);

try {

Country fetched = countryService.findCountryByCode("ZZ");

LOGGER.debug("Added Country: {}", fetched);

} catch (CountryNotFoundException e) {

LOGGER.error("Country not found after adding: {}", e.getMessage());

}

LOGGER.info("End");

}

**main**

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

countryService = context.getBean(CountryService.class);

// Existing test

getAllCountriesTest();

// New test

testAddCountry();

}

**Output:**

INFO : Start

Hibernate: insert into country (co\_name, co\_code) values (?, ?)

DEBUG : Added Country: Country [co\_code=ZZ, co\_name=Testland]

INFO : End

**Hands on 1**

**Write queries on country table using Query Methods**

**CountryRepository.java**

package com.cognizant.springlearn.repository;

import java.util.List;

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

import com.cognizant.springlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

// 1. Search countries by partial name (case insensitive)

List<Country> findByCoNameContainingIgnoreCase(String namePart);

// 2. Same as above but ordered ascending

List<Country> findByCoNameContainingIgnoreCaseOrderByCoNameAsc(String namePart);

// 3. Countries starting with a particular letter

List<Country> findByCoNameStartingWithIgnoreCase(String letter);

}

**OrmLearnApplication.java**

@Autowired

static CountryRepository countryRepository;

**a. testFindByCoNameContainingIgnoreCase()**

private static void testFindByCoNameContainingIgnoreCase() {

LOGGER.info("Start - findByCoNameContainingIgnoreCase");

List<Country> countries = countryRepository.findByCoNameContainingIgnoreCase("ou");

countries.forEach(country -> LOGGER.debug("Country: {}", country));

LOGGER.info("End");

}

**b. testFindByCoNameContainingIgnoreCaseOrderByCoNameAsc()**

private static void testFindByCoNameContainingIgnoreCaseOrderByCoNameAsc() {

LOGGER.info("Start - findByCoNameContainingIgnoreCaseOrderByCoNameAsc");

List<Country> countries = countryRepository.findByCoNameContainingIgnoreCaseOrderByCoNameAsc("ou");

countries.forEach(country -> LOGGER.debug("Country: {}", country));

LOGGER.info("End");

}

**c. testFindByCoNameStartingWithIgnoreCase()**

private static void testFindByCoNameStartingWithIgnoreCase() {

LOGGER.info("Start - findByCoNameStartingWithIgnoreCase");

List<Country> countries = countryRepository.findByCoNameStartingWithIgnoreCase("Z");

countries.forEach(country -> LOGGER.debug("Country: {}", country));

LOGGER.info("End");

}

**Main**

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

countryRepository = context.getBean(CountryRepository.class);

countryService = context.getBean(CountryService.class);

testFindByCoNameContainingIgnoreCase();

testFindByCoNameContainingIgnoreCaseOrderByCoNameAsc();

testFindByCoNameStartingWithIgnoreCase();

}

Output:

**match with 'ou'**

DEBUG : Country: Country [co\_code=BV, co\_name=Bouvet Island]

DEBUG : Country: Country [co\_code=DJ, co\_name=Djibouti]

...

**Partial match with 'ou' in ascending order**

DEBUG : Country: Country [co\_code=BV, co\_name=Bouvet Island]

DEBUG : Country: Country [co\_code=DJ, co\_name=Djibouti]

DEBUG : Country: Country [co\_code=TF, co\_name=French Southern Territories]

...

**Starts with 'Z'**

DEBUG : Country: Country [co\_code=ZM, co\_name=Zambia]

DEBUG : Country: Country [co\_code=ZW, co\_name=Zimbabwe]

**Hands on 2**

**Write queries on stock table using Query Methods**

**Stock.java**

@Entity

@Table(name = "stock")

public class Stock {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "st\_id")

private int id;

@Column(name = "st\_code")

private String code;

@Column(name = "st\_date")

private LocalDate date;

@Column(name = "st\_open")

private BigDecimal open;

@Column(name = "st\_close")

private BigDecimal close;

@Column(name = "st\_volume")

private Long volume;

// Getters and Setters

}

**StockRepository.java**

public interface StockRepository extends JpaRepository<Stock, Integer> {

// 1. All Facebook stock in Sep 2019

List<Stock> findByCodeAndDateBetween(String code, LocalDate startDate, LocalDate endDate);

// 2. Google stocks with price > 1250

List<Stock> findByCodeAndCloseGreaterThan(String code, BigDecimal close);

// 3. Top 3 volume days

List<Stock> findTop3ByOrderByVolumeDesc();

// 4. 3 dates when Netflix stocks were lowest

List<Stock> findTop3ByCodeOrderByCloseAsc(String code);

}

**OrmLearnApplication.java**

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

@Autowired

private StockRepository stockRepository;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

getFacebookStockInSep2019();

getGoogleStockPriceAbove1250();

getTop3VolumeDates();

getLowestNetflixStocks();

}

private void getFacebookStockInSep2019() {

LOGGER.info("Facebook Stocks - Sep 2019:");

List<Stock> stocks = stockRepository.findByCodeAndDateBetween("FB",

LocalDate.of(2019, 9, 1), LocalDate.of(2019, 9, 30));

stocks.forEach(stock -> LOGGER.info(stock.toString()));

}

private void getGoogleStockPriceAbove1250() {

LOGGER.info("Google Stocks > 1250:");

List<Stock> stocks = stockRepository.findByCodeAndCloseGreaterThan("GOOGL", new BigDecimal("1250"));

stocks.forEach(stock -> LOGGER.info(stock.toString()));

}

private void getTop3VolumeDates() {

LOGGER.info("Top 3 Volume Days:");

List<Stock> stocks = stockRepository.findTop3ByOrderByVolumeDesc();

stocks.forEach(stock -> LOGGER.info(stock.toString()));

}

private void getLowestNetflixStocks() {

LOGGER.info("Lowest Netflix Stocks:");

List<Stock> stocks = stockRepository.findTop3ByCodeOrderByCloseAsc("NFLX");

stocks.forEach(stock -> LOGGER.info(stock.toString()));

}

}

**Application**

spring.jpa.hibernate.ddl-auto=validate

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=your\_password

**Output in Eclipse:**

Facebook Stocks - Sep 2019:

Stock [code=FB, date=2019-09-03, open=184.00, close=182.39, volume=9779400]

...

Google Stocks > 1250:

Stock [code=GOOGL, date=2019-04-23, open=1256.64, close=1270.59, volume=1593400]

...

Top 3 Volume Days:

Stock [code=FB, date=2019-01-31, open=165.60, close=166.69, volume=77233600]

...

Lowest Netflix Stocks:

Stock [code=NFLX, date=2018-12-24, open=242.00, close=233.88, volume=9547600]

...

**Hands on 3**

**Create payroll tables and bean mapping**

**payroll.sql**

DROP TABLE IF EXISTS employee\_skill;

DROP TABLE IF EXISTS employee;

DROP TABLE IF EXISTS department;

DROP TABLE IF EXISTS skill;

-- Department Table

CREATE TABLE department (

dp\_id INT PRIMARY KEY AUTO\_INCREMENT,

dp\_name VARCHAR(50) NOT NULL

);

-- Employee Table

CREATE TABLE employee (

em\_id INT PRIMARY KEY AUTO\_INCREMENT,

em\_name VARCHAR(100) NOT NULL,

em\_salary DOUBLE NOT NULL,

em\_permanent BOOLEAN,

em\_date\_of\_birth DATE,

em\_dp\_id INT,

FOREIGN KEY (em\_dp\_id) REFERENCES department(dp\_id)

);

-- Skill Table

CREATE TABLE skill (

sk\_id INT PRIMARY KEY AUTO\_INCREMENT,

sk\_name VARCHAR(50) NOT NULL

);

-- Employee-Skill Mapping Table (Many-to-Many)

CREATE TABLE employee\_skill (

es\_em\_id INT,

es\_sk\_id INT,

PRIMARY KEY (es\_em\_id, es\_sk\_id),

FOREIGN KEY (es\_em\_id) REFERENCES employee(em\_id),

FOREIGN KEY (es\_sk\_id) REFERENCES skill(sk\_id)

);

-- Insert sample departments

INSERT INTO department (dp\_name) VALUES ('Human Resources'), ('IT'), ('Finance'), ('Marketing');

-- Insert sample employees

INSERT INTO employee (em\_name, em\_salary, em\_permanent, em\_date\_of\_birth, em\_dp\_id)

VALUES

('Alice', 60000, TRUE, '1990-05-15', 1),

('Bob', 75000, TRUE, '1985-03-10', 2),

('Charlie', 50000, FALSE, '1992-07-25', 2),

('David', 82000, TRUE, '1983-12-01', 3);

-- Insert sample skills

INSERT INTO skill (sk\_name) VALUES ('Java'), ('Spring'), ('Hibernate'), ('SQL'), ('Communication');

-- Insert employee-skill mapping

INSERT INTO employee\_skill (es\_em\_id, es\_sk\_id) VALUES

(1, 1), -- Alice knows Java

(1, 5), -- Alice knows Communication

(2, 1), -- Bob knows Java

(2, 2), -- Bob knows Spring

(2, 3), -- Bob knows Hibernate

(3, 4), -- Charlie knows SQL

(4, 5); -- David knows Communication

**Employee.java**

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "e\_name")

private String name;

@Column(name = "salary")

private double salary;

@Column(name = "permanent")

private boolean permanent;

@Column(name = "date\_of\_birth")

private Date dateOfBirth;

// Relationships will be added in Hands-on 4

// Getters, Setters, toString()

}

**Department.java**

@Entity

@Table(name = "department")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "d\_name")

private String name;

// Relationship to Employee added in Hands-on 4

// Getters, Setters, toString()

}

**Skill.java**

@Entity

@Table(name = "skill")

public class Skill {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

// Relationship to Employee added in Hands-on 4

// Getters, Setters, toString()

}

**EmployeeRepository.java**

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

**DepartmentRepository.java**

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

**SkillRepository.java**

public interface SkillRepository extends JpaRepository<Skill, Integer> {

}

**Hands on 4**

**Implement many to one relationship between Employee and Department**

Employee.java

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "em\_id")

private int id;

@Column(name = "em\_name")

private String name;

@Column(name = "em\_salary")

private double salary;

@Column(name = "em\_permanent")

private boolean permanent;

@Column(name = "em\_date\_of\_birth")

private Date dateOfBirth;

@ManyToOne

@JoinColumn(name = "em\_dp\_id")

private Department department;

// Getters, Setters, toString()

}

Department.java

@Entity

@Table(name = "department")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "dp\_id")

private int id;

@Column(name = "dp\_name")

private String name;

// Getters, Setters, toString()

}

**EmployeeRepository.java**

@Repository

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

**DepartmentRepository.java**

@Repository

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

**EmployeeService.java**

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

private static final Logger LOGGER = LoggerFactory.getLogger(EmployeeService.class);

@Transactional

public Employee get(int id) {

LOGGER.info("Start");

return employeeRepository.findById(id).get();

}

@Transactional

public void save(Employee employee) {

LOGGER.info("Start");

employeeRepository.save(employee);

LOGGER.info("End");

}

}

**DepartmentService.java**

@Service

public class DepartmentService {

@Autowired

private DepartmentRepository departmentRepository;

@Transactional

public Department get(int id) {

return departmentRepository.findById(id).get();

}

@Transactional

public void save(Department department) {

departmentRepository.save(department);

}

}

**OrmLearnApplication.java**

@SpringBootApplication

public class OrmLearnApplication {

private static EmployeeService employeeService;

private static DepartmentService departmentService;

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

employeeService = context.getBean(EmployeeService.class);

departmentService = context.getBean(DepartmentService.class);

testGetEmployee();

}

private static void testGetEmployee() {

LOGGER.info("Start");

Employee employee = employeeService.get(1);

LOGGER.debug("Employee:{}", employee);

LOGGER.debug("Department:{}", employee.getDepartment());

LOGGER.info("End");

}

private static void testAddEmployee() {

LOGGER.info("Start");

Employee employee = new Employee();

employee.setName("John Doe");

employee.setSalary(75000);

employee.setPermanent(true);

employee.setDateOfBirth(Date.valueOf("1990-01-01"));

Department department = departmentService.get(1);

employee.setDepartment(department);

employeeService.save(employee);

LOGGER.debug("Employee:{}", employee);

LOGGER.info("End");

}

private static void testUpdateEmployee() {

LOGGER.info("Start");

Employee employee = employeeService.get(1);

Department department = departmentService.get(2); // different department

employee.setDepartment(department);

employeeService.save(employee);

LOGGER.debug("Updated Employee:{}", employee);

LOGGER.info("End");

}

}

Output:

INFO : Start

DEBUG: Employee: Employee [id=1, name=John Doe, salary=45000.0, permanent=true, dateOfBirth=1990-01-15]

DEBUG: Department: Department [id=1, name=Technology]

INFO : End

**Hands on 1**

**Introduction to HQL and JPQL**

**EmployeeRepository.java**

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

// Select all employees

@Query("SELECT e FROM Employee e")

List<Employee> getAllEmployees();

// Select employee names

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

List<String> getAllEmployeeNames();

// Update employee salary

@Modifying

@Transactional

@Query("UPDATE Employee e SET e.salary = :salary WHERE e.id = :id")

void updateEmployeeSalary(@Param("id") int id, @Param("salary") double salary);

// Delete employee by name

@Modifying

@Transactional

@Query("DELETE FROM Employee e WHERE e.name = :name")

void deleteByName(@Param("name") String name);

}

**OrmLearnApplication.java**

private static void testJPQLQueries() {

LOGGER.info("Start");

// Get all employees

List<Employee> employees = employeeService.getAllEmployees();

employees.forEach(emp -> LOGGER.debug("Employee: {}", emp));

// Update salary

employeeService.updateSalary(1, 90000);

// Delete by name

employeeService.deleteEmployeeByName("John Doe");

LOGGER.info("End");

}

**EmployeeService.java**

@Transactional

public List<Employee> getAllEmployees() {

return employeeRepository.getAllEmployees();

}

@Transactional

public void updateSalary(int id, double salary) {

employeeRepository.updateEmployeeSalary(id, salary);

}

@Transactional

public void deleteEmployeeByName(String name) {

employeeRepository.deleteByName(name);

}

Output:

INFO - Start

DEBUG - Employee: Employee{id=1, name='Alice', salary=50000.0, permanent=true, dateOfBirth=1990-05-10, department=Department{id=1, name='HR'}}

DEBUG - Employee: Employee{id=2, name='Bob', salary=60000.0, permanent=true, dateOfBirth=1985-03-15, department=Department{id=2, name='Finance'}}

DEBUG - Employee: Employee{id=3, name='Charlie', salary=70000.0, permanent=false, dateOfBirth=1992-07-21, department=Department{id=3, name='IT'}}

INFO - Updated salary for Employee ID: 1 to 90000.0

INFO - Deleted employee with name: John Doe

INFO - End