**Digital Nurture 4.0 – Week 3**

**Spring Data JPA with Spring Boot, Hibernate**

**Mandatory Hands-On**

1. **spring-data-jpa-handson**

**Spring Data JPA - Quick Example**

**Objective:**

Create a Spring Boot application using **Spring Data JPA** to fetch data from a MySQL database with a Book entity.

**STEP 1: Spring Initializr - Project Creation**

Fill the details as:

Group: com.example

Artifact: ormexample

Name: ormexample

Package name: com.example.ormexample

Select:

* Language: Java
* Java Version: 17 (or your version)
* Packaging: Jar

Click **Add Dependencies** and choose:

* Spring Web
* Spring Boot DevTools
* Spring Data JPA
* MySQL Driver

Click **Generate**, unzip it and open in **IntelliJ**

**STEP 2: Create MySQL Schema & Table**

1. Open **MySQL Workbench**

CREATE DATABASE ormexample;

USE ormexample;

CREATE TABLE book (

id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(100),

author VARCHAR(100)

);

INSERT INTO book (title, author) VALUES

('Spring Boot Basics', 'John Doe'),

('JPA Mastery', 'Jane Smith');

**STEP 3: application.properties Setup**

# Logging

logging.level.org.springframework=info

logging.level.com.example.ormexample=debug

logging.level.org.hibernate.SQL=trace

# MySQL Configuration

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

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

spring.datasource.username=root

spring.datasource.password=root@123

# Hibernate

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

**STEP 4: Create Folder Structure**

**Inside src/main/java/com/example/ormexample, create:**

* **model**
* **repository**
* **service**

**STEP 5: Book Entity (Model)**

src/main/java/com/example/ormexample/model/Book.java

package com.example.ormexample.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Column;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column

private String title;

@Column

private String author;

// Getters & Setters

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getTitle() { return title; }

public void setTitle(String title) { this.title = title; }

public String getAuthor() { return author; }

public void setAuthor(String author) { this.author = author; }

@Override

public String toString() {

return "Book [id=" + id + ", title=" + title + ", author=" + author + "]";

}

}

**STEP 6: Repository Interface**

src/main/java/com/example/ormexample/repository/BookRepository.java

package com.example.ormexample.repository;

import com.example.ormexample.model.Book;

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

import org.springframework.stereotype.Repository;

@Repository

public interface BookRepository extends JpaRepository<Book, Integer> {

}

**STEP 7: Service Class**

src/main/java/com/example/ormexample/service/BookService.java

package com.example.ormexample.service;

import com.example.ormexample.model.Book;

import com.example.ormexample.repository.BookRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class BookService {

@Autowired

private BookRepository bookRepository;

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

}

**STEP 8: Main Class Update**

src/main/java/com/example/ormexample/OrmexampleApplication.java

package com.example.ormexample;

import com.example.ormexample.model.Book;

import com.example.ormexample.service.BookService;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.List;

@SpringBootApplication

public class OrmexampleApplication implements CommandLineRunner {

@Autowired

private BookService bookService;

public static void main(String[] args) {

SpringApplication.run(OrmexampleApplication.class, args);

}

@Override

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

System.out.println("---- Book List ----");

List<Book> books = bookService.getAllBooks();

books.forEach(System.out::println);

}

}

**STEP 9: Run the Project**

1. Right-click OrmexampleApplication.java → Run
2. Output will display books from your MySQL DB:

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

1. **spring-data-jpa-handson**

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

**Hibernate implementation:**

**Step 1:**

Create a project named **hibernate-example.**Then configure pom.xml.

**Pom.xml:**

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

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>hibernate-example</artifactId>

<version>1.0</version>

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>6.4.4.Final</version>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<version>8.3.0</version>

</dependency>

<dependency>

<groupId>jakarta.persistence</groupId>

<artifactId>jakarta.persistence-api</artifactId>

<version>3.1.0</version>

</dependency>

</dependencies>

</project>

**Step 2:**

Create **hibernate.cfg.xml** in the **src/main/resources** directory.

**hibernate.cfg.xml**:

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/test1</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root@123</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="hibernate.hbm2ddl.auto">update</property>

<property name="show\_sql">true</property>

<!-- Mapping your model class -->

<mapping class="com.example.hibernate.Employee"/>

</session-factory>

</hibernate-configuration>

**Step 3:**

In **src/main**/**java/com/example/hibernate,**Create the classes named **Employee.java,EmployeeDAO.java,App.java.**

**Employee.java:**

package com.example.hibernate;

import jakarta.persistence.\*;

@Entity

@Table(name = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

private String name;

private String dept;

public Employee() {}

public Employee(String name, String dept) {

this.name = name;

this.dept = dept;

}

public Integer getId() { return id; }

public void setId(Integer id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getDept() { return dept; }

public void setDept(String dept) { this.dept = dept; }

}

**EmployeeDAO.java:**

package com.example.hibernate;

import org.hibernate.\*;

import org.hibernate.cfg.Configuration;

public class EmployeeDAO {

private static final SessionFactory sessionFactory;

static {

sessionFactory = new Configuration()

.configure()

.addAnnotatedClass(Employee.class)

.buildSessionFactory();

}

public Integer addEmployee(Employee empObj) {

Transaction tx = null;

Integer empId = null;

try (Session session = sessionFactory.openSession()) {

tx = session.beginTransaction();

empId = (Integer) session.save(empObj);

tx.commit();

} catch (Exception err) {

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

err.printStackTrace();

}

return empId;

}

public void shutdown() {

sessionFactory.close();

}

}

**App.java**:

package com.example.hibernate;

public class App {

public static void main(String[] args) {

EmployeeDAO dao = new EmployeeDAO();

Employee emp = new Employee("John", "Operations");

Integer id = dao.addEmployee(emp);

System.out.println("New Employee created with ID: " + id);

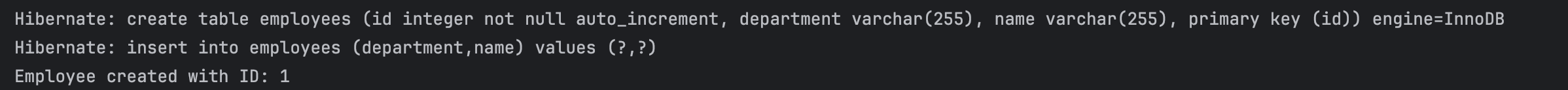
dao.shutdown();

}

}

**Step 4:**

Create a schema called test1 on your MySQL Database.And ensure your username and password on **hibernate.cfg.xml**.



**Spring JPA implentation:**

**Step 1:**

Configure spring Initializer by using below table details.

|  |  |
| --- | --- |
| **Field** | **Value** |
| Project | Maven |
| Language | Java |
| Spring Boot | 3.5.3 |
| Group | com.example |
| Artifact | employeeapp |
| Name | employeeapp |
| Package Name | com.example.employeeapp |
| Packaging | Jar |
| Java | 21 |

Dependencies:

1.Spring Data JPA

2.Spring Web

3.MySQL Driver

**Step 2:**

Now open it on eclipse idle.In **com/example/employeeapp** create the classes named Employee.java,EmployeeRepository.java,EmployeeService.java, and EmployeeController.java.

**Employee.java:**

package com.example.employeeapp;

import jakarta.persistence.\*;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

private String name;

private String dept;

public Employee() {}

public Employee(String name, String dept) {

this.name = name;

this.dept = dept;

}

public Integer getId() { return id; }

public void setId(Integer id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getDept() { return dept; }

public void setDept(String dept) { this.dept = dept; }

}

**EmployeeRepository.java**:**(Interface)**

package com.example.employeeapp;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**:

package com.example.employeeapp;

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

import org.springframework.stereotype.Service;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository repo;

public Employee register(Employee e) {

return repo.save(e);

}

}

**EmployeeController.java**:

package com.example.employeeapp;

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

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

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeService service;

@PostMapping

public Employee saveEmployee(@RequestBody Employee e) {

return service.register(e);

}

}

**Step 3:**

In **src/main/resources**,update **application.properties**.

**application.properties**:

spring.datasource.url=jdbc:mysql://localhost:3306/test2

spring.datasource.username=root

spring.datasource.password=root@123

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

Create scheme called test2 on your MySQL Database.And ensure the username and password

**Step 4:**

Run the **EmployeeappApplication.java**  as a java application.And run the following command on the terminal.

A black screen with white text

AI-generated content may be incorrect.

**(b) Additional important hands-on**

**Filename :** 1. spring-data-jpa-handson

**(i)Implement services for managing Country,**

**(ii)Find a country based on country code,**

**(iii)Add new Country:**

**Step 1:**

Configure spring Initializer by using below table details.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Field** | **Value** | | Project | Maven | | Language | Java | | Spring Boot | 3.5.3 | | Group | com.example | | Artifact | countryservice | | Name | countryservice | | Package Name | com.example.countryservice | | Packaging | Jar | | Java | 21 | |

Dependencies:

1.Spring Data JPA

2.Spring Web

3.MySQL Driver

**Step 2:**

Modify **application.properties** as your username,password and schema\_name.

**application.properties**:

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

spring.datasource.username=root

spring.datasource.password=290319

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.show-sql=true

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

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

**Step 3:**

Create a table(country) on MySQL workbench by using following query,

CREATE TABLE country (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

co\_code VARCHAR(10) UNIQUE NOT NULL,

co\_name VARCHAR(255) NOT NULL

);

Then Insert all values which is given in the hands on workbook.

**Step 4:**

**Country.java (model package)**

package com.example.countryservice.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "co\_code", nullable = false, unique = true)

private String countryCode;

@Column(name = "co\_name", nullable = false)

private String countryName;

public Country() {}

public Country(String countryCode, String countryName) {

this.countryCode = countryCode;

this.countryName = countryName;

}

public Long getId() { return id; }

public String getCountryCode() { return countryCode; }

public void setCountryCode(String countryCode) { this.countryCode = countryCode; }

public String getCountryName() { return countryName; }

public void setCountryName(String countryName) { this.countryName = countryName; }

}

**CountryRepository.java (repository package)**

package com.example.countryservice.repository;

import com.example.countryservice.model.Country;

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

import java.util.Optional;

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, Long> {

Optional<Country> findByCountryCode(String countryCode);

List<Country> findByCountryNameContainingIgnoreCase(String keyword);

}

**CountryService.java (service package)**

package com.example.countryservice.service;

import com.example.countryservice.model.Country;

import com.example.countryservice.repository.CountryRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

public class CountryService {

@Autowired

private CountryRepository repo;

public Optional<Country> getByCode(String code) {

return repo.findByCountryCode(code);

}

public Country create(Country country) {

return repo.save(country);

}

public Optional<Country> modify(String code, Country input) {

return repo.findByCountryCode(code).map(c -> {

c.setCountryName(input.getCountryName());

c.setCountryCode(input.getCountryCode());

return repo.save(c);

});

}

public void remove(String code) {

repo.findByCountryCode(code).ifPresent(repo::delete);

}

public List<Country> searchByName(String name) {

return repo.findByCountryNameContainingIgnoreCase(name);

}

public List<Country> listAll() {

return repo.findAll();

}

}

**CountryController.java (controller package)**

package com.example.countryservice.controller;

import com.example.countryservice.model.Country;

import com.example.countryservice.service.CountryService;

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

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/countries")

public class CountryController {

@Autowired

private CountryService service;

@GetMapping("/{code}")

public ResponseEntity<Country> fetchByCode(@PathVariable String code) {

return service.getByCode(code)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@PostMapping

public ResponseEntity<Country> insert(@RequestBody Country country) {

return ResponseEntity.ok(service.create(country));

}

@PutMapping("/{code}")

public ResponseEntity<Country> revise(@PathVariable String code, @RequestBody Country country) {

return service.modify(code, country)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@DeleteMapping("/{code}")

public ResponseEntity<Void> erase(@PathVariable String code) {

service.remove(code);

return ResponseEntity.noContent().build();

}

@GetMapping("/search")

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

return ResponseEntity.ok(service.searchByName(name));

}

@GetMapping

public ResponseEntity<List<Country>> fetchAll() {

return ResponseEntity.ok(service.listAll());

}

}

**CountryServiceApplication.java**:

package com.example.countryservice;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CountryServiceApplication {

public static void main(String[] args) {

SpringApplication.run(CountryServiceApplication.class, args);

}

}

**Step 5:**

Run **CountryServiceApplication.java** as a java application then use below terminal commands to access the output.

* It provides get country by code
* Add new country
* Update existing country details
* Delete country by code

A computer screen with white text

AI-generated content may be incorrect.

A computer screen with white text

AI-generated content may be incorrect.

A computer screen with white text

AI-generated content may be incorrect.

2. spring-data-jpa-handson

**(i)Demonstrate implementation of Query Methods feature of Spring Data JPA:**

**Step 1:**

Configure spring Initializr by using below table details.

|  |  |
| --- | --- |
| **Field** | **Value** |
| Project | Maven |
| Language | Java |
| Spring Boot | 3.5.3 |
| Group | com.cognizant |
| Artifact | ormlearn |
| Name | ormlearn |
| Package Name | com.cognizant.ormlearn |
| Packaging | Jar |
| Java | 21 |

Dependencies:

1.Spring Data JPA

2.Spring Web

3.MySQL Driver

**Step 2:**

Update **application.properties** by username,password and schema name.

**application.properties**:

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn1

spring.datasource.username=root

spring.datasource.password=290319

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

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

**Step 3:**

Create the classes named **Country.java** in model package

**Country.java**:

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String countryCode;

@Column(name = "co\_name")

private String countryName;

public Country() {}

public Country(String countryCode, String countryName) {

this.countryCode = countryCode;

this.countryName = countryName;

}

public String getCountryCode() {

return countryCode;

}

public void setCountryCode(String countryCode) {

this.countryCode = countryCode;

}

public String getCountryName() {

return countryName;

}

public void setCountryName(String countryName) {

this.countryName = countryName;

}

@Override

public String toString() {

return "Country [countryCode=" + countryCode + ", countryName=" + countryName + "]";

}

}

**Stock.java**:

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

import java.math.BigDecimal;

import java.util.Date;

@Entity

@Table(name = "stock")

public class Stock {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "st\_id")

private int stockId;

@Column(name = "st\_code")

private String companyCode;

@Temporal(TemporalType.DATE)

@Column(name = "st\_date")

private Date stockDate;

@Column(name = "st\_open")

private BigDecimal openingPrice;

@Column(name = "st\_close")

private BigDecimal closingPrice;

@Column(name = "st\_volume")

private long stockVolume;

public int getStockId() {

return stockId;

}

public void setStockId(int stockId) {

this.stockId = stockId;

}

public String getCompanyCode() {

return companyCode;

}

public void setCompanyCode(String companyCode) {

this.companyCode = companyCode;

}

public Date getStockDate() {

return stockDate;

}

public void setStockDate(Date stockDate) {

this.stockDate = stockDate;

}

public BigDecimal getOpeningPrice() {

return openingPrice;

}

public void setOpeningPrice(BigDecimal openingPrice) {

this.openingPrice = openingPrice;

}

public BigDecimal getClosingPrice() {

return closingPrice;

}

public void setClosingPrice(BigDecimal closingPrice) {

this.closingPrice = closingPrice;

}

public long getStockVolume() {

return stockVolume;

}

public void setStockVolume(long stockVolume) {

this.stockVolume = stockVolume;

}

@Override

public String toString() {

return "Stock{" +

"stockId=" + stockId +

", companyCode='" + companyCode + '\'' +

", stockDate=" + stockDate +

", openingPrice=" + openingPrice +

", closingPrice=" + closingPrice +

", stockVolume=" + stockVolume +

'}';

}

}

**CountryRepository.java:**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Country;

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

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByCountryNameContaining(String value);

List<Country> findByCountryNameContainingOrderByCountryNameAsc(String value);

List<Country> findByCountryNameStartingWith(String start);

}

**StockRepository.java** :

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Stock;

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

import java.util.Date;

import java.util.List;

public interface StockRepository extends JpaRepository<Stock, Integer> {

List<Stock> findByCompanyCodeAndStockDateBetween(String code, Date from, Date to);

List<Stock> findByCompanyCodeAndClosingPriceGreaterThan(String code, double threshold);

List<Stock> findTop3ByOrderByStockVolumeDesc();

List<Stock> findTop3ByCompanyCodeOrderByClosingPriceAsc(String code);

}

**OrmLearnApplication.java**:

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.model.Stock;

import com.cognizant.ormlearn.repository.CountryRepository;

import com.cognizant.ormlearn.repository.StockRepository;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private CountryRepository countryRepo;

@Autowired

private StockRepository stockRepo;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

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

showCountriesWithText();

showCountriesSorted();

showCountriesStartingWithLetter();

fetchFacebookStocks();

fetchHighPriceGoogle();

fetchTopVolumeStocks();

fetchLowNetflixStocks();

}

private void showCountriesWithText() {

LOGGER.info("Countries with 'ou' in name:");

countryRepo.findByCountryNameContaining("ou")

.forEach(country -> LOGGER.info(country.toString()));

}

private void showCountriesSorted() {

LOGGER.info("Countries with 'ou' sorted by name:");

countryRepo.findByCountryNameContainingOrderByCountryNameAsc("ou")

.forEach(country -> LOGGER.info(country.toString()));

}

private void showCountriesStartingWithLetter() {

LOGGER.info("Countries starting with 'Z':");

countryRepo.findByCountryNameStartingWith("Z")

.forEach(country -> LOGGER.info(country.toString()));

}

private void fetchFacebookStocks() throws Exception {

LOGGER.info("Facebook stocks in September 2019:");

Date from = new SimpleDateFormat("yyyy-MM-dd").parse("2019-09-01");

Date to = new SimpleDateFormat("yyyy-MM-dd").parse("2019-09-30");

stockRepo.findByCompanyCodeAndStockDateBetween("FB", from, to)

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

}

private void fetchHighPriceGoogle() {

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

stockRepo.findByCompanyCodeAndClosingPriceGreaterThan("GOOGL", 1250.0)

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

}

private void fetchTopVolumeStocks() {

LOGGER.info("Top 3 stocks by volume:");

stockRepo.findTop3ByOrderByStockVolumeDesc()

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

}

private void fetchLowNetflixStocks() {

LOGGER.info("3 lowest Netflix stocks:");

stockRepo.findTop3ByCompanyCodeOrderByClosingPriceAsc("NFLX")

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

}

}

**Step 4:**

Run **OrmLearnApplication.java** as a java application to see the output on console.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**(ii)Demonstrate implementation of O/R Mapping:**

**Step 1:**

Alter the program files from the previous spring.io file.

**Step 2:**

Update **application.properties** by your username,password and schema name.

**application.properties**:

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn1

spring.datasource.username=root

spring.datasource.password=290319

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

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

**Step 3:**

Create the classes names and write the codes.

**Department.java:**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

import java.util.Set;

@Entity

@Table(name = "department")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "dp\_id")

private int deptId;

@Column(name = "dp\_name")

private String deptName;

@OneToMany(mappedBy = "department", fetch = FetchType.EAGER)

private Set<Employee> employees;

public int getDeptId() {

return deptId;

}

public void setDeptId(int deptId) {

this.deptId = deptId;

}

public String getDeptName() {

return deptName;

}

public void setDeptName(String deptName) {

this.deptName = deptName;

}

public Set<Employee> getEmployees() {

return employees;

}

public void setEmployees(Set<Employee> employees) {

this.employees = employees;

}

@Override

public String toString() {

return "Department [deptId=" + deptId + ", deptName=" + deptName + "]";

}

}

**Skill.java:**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

import java.util.Set;

@Entity

@Table(name = "skill")

public class Skill {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "sk\_id")

private int skillId;

@Column(name = "sk\_name")

private String skillName;

@ManyToMany(mappedBy = "skills", fetch = FetchType.EAGER)

private Set<Employee> employees;

public int getSkillId() {

return skillId;

}

public void setSkillId(int skillId) {

this.skillId = skillId;

}

public String getSkillName() {

return skillName;

}

public void setSkillName(String skillName) {

this.skillName = skillName;

}

public Set<Employee> getEmployees() {

return employees;

}

public void setEmployees(Set<Employee> employees) {

this.employees = employees;

}

@Override

public String toString() {

return "Skill [skillId=" + skillId + ", skillName=" + skillName + "]";

}

}

**Employee.java:**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

import java.util.Date;

import java.util.Set;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "em\_id")

private int empId;

@Column(name = "em\_name")

private String empName;

@Column(name = "em\_salary")

private double empSalary;

@Column(name = "em\_permanent")

private boolean isPermanent;

@Temporal(TemporalType.DATE)

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

private Date dob;

@ManyToOne

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

private Department department;

@ManyToMany(fetch = FetchType.EAGER)

@JoinTable(name = "employee\_skill",

joinColumns = @JoinColumn(name = "es\_em\_id"),

inverseJoinColumns = @JoinColumn(name = "es\_sk\_id"))

private Set<Skill> skills;

public int getEmpId() { return empId; }

public void setEmpId(int empId) { this.empId = empId; }

public String getEmpName() { return empName; }

public void setEmpName(String empName) { this.empName = empName; }

public double getEmpSalary() { return empSalary; }

public void setEmpSalary(double empSalary) { this.empSalary = empSalary; }

public boolean isPermanent() { return isPermanent; }

public void setPermanent(boolean permanent) { isPermanent = permanent; }

public Date getDob() { return dob; }

public void setDob(Date dob) { this.dob = dob; }

public Department getDepartment() { return department; }

public void setDepartment(Department department) { this.department =department;

}

public Set<Skill> getSkills() { return skills; }

public void setSkills(Set<Skill> skills) { this.skills = skills; }

@Override

public String toString() {

return "Employee [empId=" + empId + ", empName=" + empName +

", salary=" + empSalary + ", permanent=" + isPermanent +

", dob=" + dob + "]";

}

}

**repository package**

**DepartmentRepository.java:**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Department;

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

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

**EmployeeRepository.java:**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Employee;

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

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

**SkillRepository.java:**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Skill;

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

public interface SkillRepository extends JpaRepository<Skill, Integer> {}

**service package**

**EmployeeService.java:**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.repository.EmployeeRepository;

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 repo;

@Transactional

public Employee fetchById(int id) {

return repo.findById(id).orElse(null);

}

@Transactional

public void save(Employee emp) {

repo.save(emp);

}

}

**DepartmentService.java:**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Department;

import com.cognizant.ormlearn.repository.DepartmentRepository;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

public class DepartmentService {

@Autowired

private DepartmentRepository repo;

@Transactional

public Department fetchById(int id) {

return repo.findById(id).orElse(null);

}

@Transactional

public void save(Department dept) {

repo.save(dept);

}

}

**SkillService.java:**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Skill;

import com.cognizant.ormlearn.repository.SkillRepository;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

public class SkillService {

@Autowired

private SkillRepository repo;

@Transactional

public Skill fetchById(int id) {

return repo.findById(id).orElse(null);

}

@Transactional

public void save(Skill skill) {

repo.save(skill);

}

}

**OrmLearnApplication.java:**

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.\*;

import com.cognizant.ormlearn.service.\*;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.text.SimpleDateFormat;

import java.util.HashSet;

import java.util.Set;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private EmployeeService empService;

@Autowired

private DepartmentService deptService;

@Autowired

private SkillService skillService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

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

insertEmployee();

fetchEmployeeDetails();

changeEmployeeDepartment();

printDepartmentDetails();

assignSkillToEmployee();

}

private void insertEmployee() throws Exception {

LOG.info("Creating new employee...");

Employee emp = new Employee();

emp.setEmpName("Giri Kumar");

emp.setEmpSalary(80000);

emp.setPermanent(true);

emp.setDob(new SimpleDateFormat("yyyy-MM-dd").parse("1996-05-01"));

emp.setDepartment(deptService.fetchById(1));

empService.save(emp);

LOG.info("Employee added: {}", emp);

}

private void fetchEmployeeDetails() {

LOG.info("Fetching employee by ID...");

Employee emp = empService.fetchById(1);

LOG.info("Employee: {}", emp);

LOG.info("Department: {}", emp.getDepartment());

LOG.info("Skills: {}", emp.getSkills());

}

private void changeEmployeeDepartment() {

LOG.info(" Updating employee department...");

Employee emp = empService.fetchById(1);

emp.setDepartment(deptService.fetchById(2));

empService.save(emp);

LOG.info("Updated employee department: {}", emp);

}

private void printDepartmentDetails() {

LOG.info("Fetching department info...");

Department dept = deptService.fetchById(1);

LOG.info("Department: {}", dept);

LOG.info("Employees: {}", dept.getEmployees());

}

private void assignSkillToEmployee() {

LOG.info("Assigning skill to employee...");

Employee emp = empService.fetchById(1);

Skill skill = skillService.fetchById(1);

Set<Skill> skillSet = emp.getSkills();

if (skillSet == null) skillSet = new HashSet<>();

skillSet.add(skill);

emp.setSkills(skillSet);

empService.save(emp);

LOG.info("Skill assigned to employee: {}", emp);

}

}

**Step 4:**

Run the query below on the specific schema which was mentioned in application.properties.

**Query.sql:**

CREATE TABLE department (

dp\_id INT AUTO\_INCREMENT PRIMARY KEY,

dp\_name VARCHAR(255)

);

CREATE TABLE skill (

sk\_id INT AUTO\_INCREMENT PRIMARY KEY,

sk\_name VARCHAR(255)

);

CREATE TABLE employee (

em\_id INT AUTO\_INCREMENT PRIMARY KEY,

em\_name VARCHAR(255),

em\_salary DOUBLE,

em\_permanent BOOLEAN,

em\_date\_of\_birth DATE,

em\_dp\_id INT,

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

);

CREATE TABLE employee\_skill (

es\_em\_id INT,

es\_sk\_id INT,

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

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

);

INSERT INTO department (dp\_name) VALUES ('HR'), ('IT');

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

**Step 5:**

Run OrmLearnApplication.java, you will see logs like:

A screenshot of a computer

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

A black rectangle with white dots

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

A black rectangle with white dots

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

3. spring-data-jpa-handson

**(i)Demonstrate writing Hibernate Query Language and Native Query**

**Step 1:**

Configure spring Initializr(ormlearn) by using below table details.

|  |  |  |
| --- | --- | --- |
| **Step** | **Setting** | **Value** |
| 1 | Project | Maven |
| 2 | Language | Java |
| 3 | Spring Boot | 3.5.3 |
| 4 | Group id & Artifact Name | com.demo & ormlearn |
| 5 | Packaging | Jar |
| 6 | Java Version | 21 |
| 7 | Dependencies | Spring Web,Spring Data JPA,MySQL Driver |

**Step 2: MySQL Schema Setup**

Create a schema named `ormlearn3` in MySQL Workbench. Run the following script:

**Query.sql:**

USE ormlearn3;

DROP TABLE IF EXISTS employee\_skill;

DROP TABLE IF EXISTS employee;

DROP TABLE IF EXISTS skill;

DROP TABLE IF EXISTS department;

CREATE TABLE department (

id INT PRIMARY KEY,

name VARCHAR(50)

);

CREATE TABLE skill (

id INT PRIMARY KEY,

name VARCHAR(50)

);

CREATE TABLE employee (

em\_id INT PRIMARY KEY,

em\_name VARCHAR(50),

em\_salary DOUBLE,

em\_permanent BOOLEAN,

em\_date\_of\_birth VARCHAR(20),

em\_dp\_id INT,

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

);

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(id)

);

INSERT INTO department VALUES (1, 'HR'), (2, 'IT');

INSERT INTO skill VALUES (1, 'Java'), (2, 'Spring'), (3, 'Communication');

INSERT INTO employee VALUES

(1, 'John Doe', 50000, true, '1990-01-01', 2),

(2, 'Emma Smith', 60000, true, '1992-05-05', 1),

(3, 'Alex Brown', 45000, false, '1988-07-12', 2);

INSERT INTO employee\_skill VALUES (1, 1), (1, 2), (2, 3);

**Step 3:**

## In **application.properties**,ensure username and password

**application.properties**:

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn3

spring.datasource.username=root

spring.datasource.password=290319

spring.jpa.hibernate.ddl-auto=none

spring.jpa.show-sql=true

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

**Step 4:**

## **pom.xml:**

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.demo</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<parent>

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

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

<version>3.5.3</version>

</parent>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

</project>

**Step 5:**

Model, Repository, Service and Application Code

Classes:

**Department.java:**

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "department")

public class Department {

@Id

private int id;

private String name;

public Department() {

}

public Department(int id, String name) {

this.id = id;

this.name = name;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Department{" +

"id=" + id +

", name='" + name + '\'' +

'}';

}

}

**Skill.java:**

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "skill")

public class Skill {

@Id

private int id;

private String name;

public Skill() {

}

public Skill(int id, String name) {

this.id = id;

this.name = name;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Skill{" +

"id=" + id +

", name='" + name + '\'' +

'}';

}

}

**Employee.java:**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

import java.util.List;

import java.util.stream.Collectors;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@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 String dateOfBirth;

@ManyToOne

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

private Department department;

@ManyToMany

@JoinTable(

name = "employee\_skill",

joinColumns = @JoinColumn(name = "es\_em\_id"),

inverseJoinColumns = @JoinColumn(name = "es\_sk\_id")

)

private List<Skill> skillList;

public Employee() {

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public boolean isPermanent() {

return permanent;

}

public void setPermanent(boolean permanent) {

this.permanent = permanent;

}

public String getDateOfBirth() {

return dateOfBirth;

}

public void setDateOfBirth(String dateOfBirth) {

this.dateOfBirth = dateOfBirth;

}

public Department getDepartment() {

return department;

}

public void setDepartment(Department department) {

this.department = department;

}

public List<Skill> getSkillList() {

return skillList;

}

public void setSkillList(List<Skill> skillList) {

this.skillList = skillList;

}

@Override

public String toString() {

String deptName = (department != null) ? department.getName() : "null";

String skills = (skillList != null) ? skillList.stream().map(Skill::getName).collect(Collectors.joining(", ")) : "null";

return "Employee{" +

"id=" + id +

", name='" + name + '\'' +

", salary=" + salary +

", permanent=" + permanent +

", dateOfBirth='" + dateOfBirth + '\'' +

", department=" + deptName +

", skills=" + skills +

'}';

}

}

**EmployeeRepository.java:**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Employee;

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

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

import org.springframework.data.repository.query.Param;

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

@Query("SELECT e FROM Employee e WHERE e.permanent = true")

List<Employee> getAllPermanentEmployees();

@Query("SELECT e FROM Employee e LEFT JOIN FETCH e.department d LEFT JOIN FETCH e.skillList WHERE e.permanent = true")

List<Employee> getAllPermanentEmployeesWithFetch();

@Query("SELECT AVG(e.salary) FROM Employee e WHERE e.department.id = :id")

double getAverageSalary(@Param("id") int id);

@Query(value = "SELECT \* FROM employee", nativeQuery = true)

List<Employee> getAllEmployeesNative();

}

**EmployeeService.java:**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Employee;

import java.util.List;

public interface EmployeeService {

List<Employee> getAllPermanentEmployees();

List<Employee> getAllPermanentEmployeesWithFetch();

double getAverageSalary(int deptId);

List<Employee> getAllEmployeesNative();

}

**EmployeeServiceImpl.java:**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class EmployeeServiceImpl implements EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Override

public List<Employee> getAllPermanentEmployees() {

return employeeRepository.getAllPermanentEmployees();

}

@Override

public List<Employee> getAllPermanentEmployeesWithFetch() {

return employeeRepository.getAllPermanentEmployeesWithFetch();

}

@Override

public double getAverageSalary(int deptId) {

return employeeRepository.getAverageSalary(deptId);

}

@Override

public List<Employee> getAllEmployeesNative() {

return employeeRepository.getAllEmployeesNative();

}

}

**OrmLearnApplication.java:**

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.service.EmployeeService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) {

LOGGER.info("START");

testGetAllPermanentEmployeesWithFetch();

testAverageSalary();

testNativeQuery();

LOGGER.info("END");

}

private void testGetAllPermanentEmployeesWithFetch() {

LOGGER.info("START - testGetAllPermanentEmployeesWithFetch");

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

employees.forEach(e -> LOGGER.info("Employee (Fetched): {}", e));

LOGGER.info("END - testGetAllPermanentEmployeesWithFetch");

}

private void testAverageSalary() {

LOGGER.info("START - testAverageSalary");

double avgSalary = employeeService.getAverageSalary(2);

LOGGER.info("Average Salary for Department 2: {}", avgSalary);

LOGGER.info("END - testAverageSalary");

}

private void testNativeQuery() {

LOGGER.info("START - testNativeQuery");

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

employees.forEach(e -> LOGGER.info("Employee (Native): id={}, name={}, salary={}, permanent={}",

e.getId(), e.getName(), e.getSalary(), e.isPermanent()));

LOGGER.info("END - testNativeQuery");

}

}

**Step 6:**

Run Application

Run `OrmLearnApplication.java` and check the console for:

- Permanent employees (with fetch join)

- Department-wise average salary

- List of all employees (via native query)

A screenshot of a computer

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.