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

**Exercise 1: Spring Data JPA - Quick Example Software Pre-requisites**:

//application.properties

# App & Spring Logs

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

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

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

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=validate

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

//OrmLearnApplication.java

package com.cognizant.ormlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class OrmLearnApplication {

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

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

LOGGER.info("Inside main");

}

}

//Country.java

package com.cognizant.ormlearn.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_name")

private String name;

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

//CountryRepository.java

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Country;

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

import org.springframework.stereotype.Repository;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

//CountryService.java

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

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

import org.springframework.stereotype.Service;

import javax.transaction.Transactional;

import java.util.List;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**Exercise 2: Difference between JPA, Hibernate and Spring Data JPA Java Persistence API (JPA):**

/\* Method to CREATE an employee in the database \*/

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

//EmployeeService.java

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

**Exercise 3: Implement services for managing Country An application requires for features to be implemented with regards to country. These features needs to be supported by implementing them as service using Spring Data JPA.**

//application.properties

spring.jpa.hibernate.ddl-auto=validate # or update

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

spring.datasource.username=root

spring.datasource.password=root

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

//CountryRepository.java

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Country;

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

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

// Find countries by partial name

List<Country> findByNameContaining(String name);

}

//CountryService.java

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

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

import org.springframework.stereotype.Service;

import javax.transaction.Transactional;

import java.util.List;

import java.util.Optional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public Country findCountryByCode(String code) {

Optional<Country> countryOpt = countryRepository.findById(code);

if (countryOpt.isPresent()) {

return countryOpt.get();

} else {

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

}

}

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

@Transactional

public void updateCountry(Country country) {

if (countryRepository.existsById(country.getCode())) {

countryRepository.save(country);

} else {

throw new RuntimeException("Country code not found: " + country.getCode());

}

}

@Transactional

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

public List<Country> findCountriesByNameContaining(String partialName) {

return countryRepository.findByNameContaining(partialName);

}

}

//OrmLearnApplication.java

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import java.util.List;

@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);

countryService = context.getBean(CountryService.class);

LOGGER.info("Inside main");

testCountryFeatures();

}

private static void testCountryFeatures() {

LOGGER.info("Start");

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

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

Country newCountry = new Country();

newCountry.setCode("XY");

newCountry.setName("Xyland");

countryService.addCountry(newCountry);

newCountry.setName("New Xyland");

countryService.updateCountry(newCountry);

List<Country> countries = countryService.findCountriesByNameContaining("Uni");

LOGGER.debug("Countries containing 'Uni': {}", countries);

countryService.deleteCountry("XY");

LOGGER.info("End");

}

}

**Exercise 4: Find a country based on country code:**

//CountryNotFoundException.java

package com.cognizant.springlearn.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

//CountryService.java

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

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

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

import org.springframework.stereotype.Service;

import javax.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();

}

// Other methods remain same...

}

//OrmLearnApplication.java

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.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);

private static CountryService countryService;

public static void main(String[] args) throws CountryNotFoundException {

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

countryService = context.getBean(CountryService.class);

LOGGER.info("Inside main");

getCountryByCodeTest();

}

private static void getCountryByCodeTest() throws CountryNotFoundException {

LOGGER.info("Start");

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

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

LOGGER.info("End");

}

}

**Exercise 5: Add a new country:**

//addCountry.java

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

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

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

import org.springframework.stereotype.Service;

import javax.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();

}

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

// other methods...

}

**Exercise 6: Write queries on stock table using Query Methods With one year stock data of Facebook, Google and Netflix, we need to implement Spring Data JPA Query Methods for the following scenarios: Sample Data Sample data for implementing this hands on is provided to you in the platform**

//Stock.java

package com.cognizant.ormlearn.model;

import javax.persistence.\*;

import java.util.Date;

@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 Date date;

@Column(name = "st\_open")

private Double open;

@Column(name = "st\_close")

private Double close;

@Column(name = "st\_volume")

private Double volume;

// Getters & setters

public int getId() { return id; }

public String getCode() { return code; }

public void setCode(String code) { this.code = code; }

public Date getDate() { return date; }

public void setDate(Date date) { this.date = date; }

public Double getOpen() { return open; }

public void setOpen(Double open) { this.open = open; }

public Double getClose() { return close; }

public void setClose(Double close) { this.close = close; }

public Double getVolume() { return volume; }

public void setVolume(Double volume) { this.volume = volume; }

@Override

public String toString() {

return "Stock{" +

"id=" + id +

", code='" + code + '\'' +

", date=" + date +

", open=" + open +

", close=" + close +

", volume=" + volume +

'}';

}

}

//OrmLearnApplication.java

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Stock;

import com.cognizant.ormlearn.repository.StockRepository;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication {

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

private static StockRepository stockRepository;

public static void main(String[] args) throws ParseException {

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

stockRepository = context.getBean(StockRepository.class);

getFacebookStockSeptember2019();

getGoogleStockPriceGreaterThan1250();

getTop3HighestVolume();

getLowest3NetflixStocks();

}

private static void getFacebookStockSeptember2019() throws ParseException {

LOGGER.info("Start");

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

Date start = sdf.parse("2019-09-01");

Date end = sdf.parse("2019-09-30");

List<Stock> stocks = stockRepository.findByCodeAndDateBetween("FB", start, end);

LOGGER.debug("FB Sept 2019 Stocks: {}", stocks);

LOGGER.info("End");

}

private static void getGoogleStockPriceGreaterThan1250() {

LOGGER.info("Start");

List<Stock> stocks = stockRepository.findByCodeAndCloseGreaterThan("GOOGL", 1250.0);

LOGGER.debug("Google Stocks > 1250: {}", stocks);

LOGGER.info("End");

}

private static void getTop3HighestVolume() {

LOGGER.info("Start");

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

LOGGER.debug("Top 3 by Volume: {}", stocks);

LOGGER.info("End");

}

private static void getLowest3NetflixStocks() {

LOGGER.info("Start");

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

LOGGER.debug("Lowest 3 Netflix Stocks: {}", stocks);

LOGGER.info("End");

}

}

**Exercise 7: Create payroll tables and bean mapping To demonstrate one to many, many to one and many to many relationships in Hibernate, a schema with entities employee, department and skill will be used. In this hands on we will setup the tables and data, which forms the basis for learning the mappings in Hibernate. Schema Structure Follow steps below to create necessary tables:**

//Department.java

package com.cognizant.ormlearn.model;

import javax.persistence.\*;

import java.util.List;

@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;

@OneToMany(mappedBy = "department")

private List<Employee> employeeList;

// Getters & Setters

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 List<Employee> getEmployeeList() { return employeeList; }

public void setEmployeeList(List<Employee> employeeList) { this.employeeList = employeeList; }

@Override

public String toString() {

return "Department [id=" + id + ", name=" + name + "]";

}

}

//skill.java

package com.cognizant.ormlearn.model;

import javax.persistence.\*;

import java.util.List;

@Entity

@Table(name = "skill")

public class Skill {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "sk\_id")

private int id;

@Column(name = "sk\_name")

private String name;

@ManyToMany(mappedBy = "skillList")

private List<Employee> employeeList;

// Getters & Setters

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 List<Employee> getEmployeeList() { return employeeList; }

public void setEmployeeList(List<Employee> employeeList) { this.employeeList = employeeList; }

@Override

public String toString() {

return "Skill [id=" + id + ", name=" + name + "]";

}

}

//Employee.java

package com.cognizant.ormlearn.model;

import javax.persistence.\*;

import java.util.Date;

import java.util.List;

@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;

@ManyToMany

@JoinTable(name = "employee\_skill",

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

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

private List<Skill> skillList;

// Getters & Setters

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 Date getDateOfBirth() { return dateOfBirth; }

public void setDateOfBirth(Date 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() {

return "Employee [id=" + id +

", name=" + name +

", salary=" + salary +

", permanent=" + permanent +

", dateOfBirth=" + dateOfBirth +

"]";

}

}

**Exercise 8: Get average salary using HQL Compute the average salary of a department using HQL. Refer steps below to implement:**

//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;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

// Get average salary for ALL employees

@Query(value = "SELECT AVG(e.salary) FROM Employee e")

double getAverageSalary();

// Get average salary filtered by department ID

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

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

}

//EmployeeService.java

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public double getAverageSalary() {

return employeeRepository.getAverageSalary();

}

public double getAverageSalary(int departmentId) {

return employeeRepository.getAverageSalary(departmentId);

}

}

//OrmLearnApplication.java

private static void testGetAverageSalary() {

LOGGER.info("Start");

double avgSalary = employeeService.getAverageSalary();

LOGGER.debug("Average Salary for all employees: {}", avgSalary);

LOGGER.info("End");

}

private static void testGetAverageSalaryForDepartment() {

LOGGER.info("Start");

int deptId = 1; // Example department ID

double avgSalary = employeeService.getAverageSalary(deptId);

LOGGER.debug("Average Salary for Department ID {}: {}", deptId, avgSalary);

LOGGER.info("End");

}

//Main.java

testGetAverageSalary();

testGetAverageSalaryForDepartment();

**Exercise 9: Get all employees using Native Query About Native Queries:**

//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 java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

// Native SQL to get all employees

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

List<Employee> getAllEmployeesNative();

// (Existing methods: HQL average salary, etc.)

}

//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 java.util.List;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public List<Employee> getAllEmployeesNative() {

return employeeRepository.getAllEmployeesNative();

}

// (Existing methods: getAverageSalary(), etc.)

}

//OrmLearnApplication.java

private static void testGetAllEmployeesNative() {

LOGGER.info("Start");

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

LOGGER.debug("Employees (Native Query): {}", employeeList);

LOGGER.info("End");

}

//Main.java

testGetAllEmployeesNative();