**EXERCISE : 1 SPRING-DATA-JPA**

SQL:

mysql -u root -p

create schema ormlearn;

create table country (

co\_code varchar(2) primary key,

co\_name varchar(50)

);

JAVA:

OrmLearnApplication.java

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

public static void main(String[] args) {

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

LOGGER.info("Inside main");

countryService = context.getBean(CountryService.class);

testGetAllCountries();

}

Country.java

@Entity

@Table(name="country")

public class Country {

@Id

@Column(name="code")

private String code;

@Column(name="name")

private String name;

// Getters, Setters, toString()

}

CountryRepository.java

@Repository

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

CountryService.java

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

@Transactional

public Country findCountryByCode(String code) throws CountryNotFoundException {

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

if (!result.isPresent()) throw new CountryNotFoundException();

return result.get();

}

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

@Transactional

public void updateCountry(String code, String newName) {

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

if (result.isPresent()) {

Country country = result.get();

country.setName(newName);

countryRepository.save(country);

}

}

@Transactional

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

}

MAIN:

private static void testGetAllCountries() {

LOGGER.info("Start");

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

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

LOGGER.info("End");

}

private static void getCountryTest() {

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

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

}

private static void testAddCountry() {

Country country = new Country();

country.setCode("ZZ");

country.setName("Zootopia");

countryService.addCountry(country);

getCountryTest();

}

private static void testUpdateCountry() {

countryService.updateCountry("ZZ", "New Zootopia");

}

private static void testDeleteCountry() {

countryService.deleteCountry("ZZ");

}

**EXERCISE : 2 SPRING-DATA-JPA**

//Searching countries by containing texts

List<Country> findByNameContaining(String text);

//Sort by name in Ascending order

List<Country> findByNameContainingOrderByNameAsc(String text);

//Countries starting with a specific letter

List<Country> findByNameStartingWith(String prefix);

//Stock entity mapping

@Entity

@Table(name = "stock")

public class Stock {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

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

}

//Facebook stock 2019

List<Stock> findByCodeAndDateBetween(String code, LocalDate start, LocalDate end);

//Google stocks > 1250

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

//Top 3 Highest volume

List<Stock> findTop3ByOrderByVolumeDesc();

//Netflix lowest closing price

List<Stock> findTop3ByCodeOrderByCloseAsc(String code);

Employee.java:

@ManyToOne

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

private Department department;

Department.java:

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

private Set<Employee> employeeList;

Employee.java:

@ManyToMany(fetch = FetchType.EAGER)

@JoinTable(name = "employee\_skill",

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

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

private Set<Skill> skillList;

Skill.java:

@ManyToMany(mappedBy = "skillList")

private Set<Employee> employeeList;

OrmLearnApplication.java

Employee employee = employeeService.get(1);

System.out.println(employee.getDepartment());

System.out.println(employee.getSkillList());

Employee emp = new Employee();

emp.setName("John Doe");

emp.setSalary(50000);

emp.setPermanent(true);

emp.setDateOfBirth(LocalDate.of(1990, 1, 1));

Department dept = departmentService.get(1);

emp.setDepartment(dept);

employeeService.save(emp);

Employee emp = employeeService.get(1);

Department newDept = departmentService.get(2);

emp.setDepartment(newDept);

employeeService.save(emp);

Employee emp = employeeService.get(1);

Skill skill = skillService.get(3);

emp.getSkillList().add(skill);

employeeService.save(emp);

**EXERCISE : 3 SPRING-DATA-JPA**

//Employee class

@Entity

@Table(name = "EMPLOYEE")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private int id;

@Column(name = "first\_name")

private String firstName;

@Column(name = "last\_name")

private String lastName;

@Column(name = "salary")

private int salary;

// Getters and Setters

}

### @Entity

Declares the class as a persistent Java object (POJO) that should be mapped to a table.

Hibernate will recognize this class during session factory setup.

Must be present to allow the class to be managed by Hibernate.

### @Table(name = "EMPLOYEE")

### Specifies the table in the database this entity is mapped to.

### Optional—if not used, Hibernate will assume the table name is the same as the class name (Employee).

### @Id

### Marks a field as the primary key.

### This is required for every entity to uniquely identify rows in the table.

@GeneratedValue

Indicates that the primary key value should be automatically generated.

Strategies include:

1. AUTO (Default)
2. IDENTITY (used for auto-increment in MySQL)
3. SEQUENCE
4. TABLE

@Column

Maps the field to a specific column in the table.

Attributes like nullable, length, unique can also be used for further configuration.

//hibernate.cfg.xml

<hibernate-configuration>

<session-factory>

<!-- Database connection settings -->

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

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

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

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

<!-- SQL dialect -->

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

<!-- Echo all executed SQL to stdout -->

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

<!-- Automatically create/update DB schema -->

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

<!-- Register entity class -->

<mapping class="Employee"/>

</session-factory>

</hibernate-configuration>

**EXERCISE: 4 SPRING-DATA-JPA**

Java Persistence API (JPA)

· JPA is a Java specification (JSR 338) that standardizes ORM (Object-Relational Mapping) in Java.

It defines interfaces and annotations for managing relational data in Java applications.

JPA itself is not an implementation, but rather a set of rules/contracts.

//hibernate.cfg.xml

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;

}

Spring data jpa code:

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

**EXERCISE: 5 SPRING-DATA-JPA**

//application.propeties:

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

spring.datasource.username=root

spring.datasource.password=your\_password

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

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

//Country Entity:

import jakarta.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code", length = 2)

private String code;

@Column(name = "co\_name")

private String name;

// Constructors

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

// Getters and Setters

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; }

}

//Country Repository:

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

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String partialName);

}

//Country Service

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

import java.util.List;

import java.util.Optional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public Country findByCode(String code) {

return countryRepository.findById(code).orElse(null);

}

@Transactional

public Country addCountry(Country country) {

return countryRepository.save(country);

}

@Transactional

public Country updateCountry(String code, Country updatedCountry) {

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

if (existing.isPresent()) {

Country country = existing.get();

country.setName(updatedCountry.getName());

return countryRepository.save(country);

}

return null;

}

@Transactional

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

public List<Country> searchCountries(String partialName) {

return countryRepository.findByNameContainingIgnoreCase(partialName);

}

}

//REST Controller:

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

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

import java.util.List;

@RestController

@RequestMapping("/countries")

public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping("/{code}")

public Country getByCode(@PathVariable String code) {

return countryService.findByCode(code);

}

@PostMapping

public Country add(@RequestBody Country country) {

return countryService.addCountry(country);

}

@PutMapping("/{code}")

public Country update(@PathVariable String code, @RequestBody Country country) {

return countryService.updateCountry(code, country);

}

@DeleteMapping("/{code}")

public void delete(@PathVariable String code) {

countryService.deleteCountry(code);

}

@GetMapping("/search")

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

return countryService.searchCountries(name);

}

}