**WEEK 3 SPRING DATA JPA**

**SUPERSET ID : 6386707**

**Handson 1**

**HQL**

 HQL = Hibernate Query Language

 A powerful, object-oriented query language developed for Hibernate ORM

 Works with entity objects and their property names, *not*

table or column names

 HQL supports SELECT, UPDATE, DELETE, and also

INSERT (which JPQL doesn't)

**JPQL**

 JPQL = Java Persistence Query Language

 Defined in the JPA (Java Persistence API) speciflcation

 JPQL is a subset of HQL and works similarly

It supports SELECT, UPDATE, DELETE (but not INSERT)



**Sample HQL/JPQL Syntax**

// SELECT

Query query = session.createQuery("FROM Employee WHERE salary > :salary");

query.setParameter("salary", 50000); List<Employee> list = query.list();

// UPDATE

Query updateQuery = session.createQuery("UPDATE Employee SET salary = :newSal WHERE id = :empId");

updateQuery.setParameter("newSal", 60000);

updateQuery.setParameter("empId", 1); int result = updateQuery.executeUpdate();

// DELETE

Query deleteQuery = session.createQuery("DELETE FROM Employee WHERE id = :empId");

deleteQuery.setParameter("empId", 1); deleteQuery.executeUpdate();

**Handson 2**

**Employee.java**

package com.cognizant.ormlearn.model;

import javax.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 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(fetch = FetchType.EAGER)

@JoinTable(name = "employee\_skill",

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

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

private Set<Skill> skillList;

// Getters and 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 Set<Skill> getSkillList() { return skillList; }

public void setSkillList(Set<Skill> skillList) { this.skillList = skillList; }

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", salary=" + salary +

", permanent=" + permanent + ", dateOfBirth=" + dateOfBirth + "]";

}

}

**Department.java**

package com.cognizant.ormlearn.model;

import javax.persistence.\*;

import java.util.Set;

@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", fetch = FetchType.EAGER)

private Set<Employee> employeeList;

// Getters and 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 Set<Employee> getEmployeeList() { return employeeList; }

public void setEmployeeList(Set<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.Set;

@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", fetch = FetchType.EAGER)

private Set<Employee> employeeList;

// Getters and 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 Set<Employee> getEmployeeList() { return employeeList; }

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

@Override

public String toString() {

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

}

}

**EmployeeRepository.java**

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

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

List<Employee> getAllPermanentEmployees();

}

**EmployeeService.java**

@Service

public class EmployeeService {

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

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public List<Employee> getAllPermanentEmployees() {

LOGGER.info("Fetching all permanent employees");

return employeeRepository.getAllPermanentEmployees();

}

}

**OrmLearnApplication.java**

private static void testGetAllPermanentEmployees() {

LOGGER.info("Start");

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

LOGGER.debug("Permanent Employees: {}", employees);

for (Employee e : employees) {

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

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

LOGGER.debug("Skills: {}", e.getSkillList());

}

LOGGER.info("End");

}

**Handson 3**

**User.java**

@Entity

@Table(name = "user")

public class User {

@Id

private int id;

private String name;

@OneToMany(mappedBy = "user", fetch = FetchType.LAZY)

private List<Attempt> attempts;

// Getters, setters, toString()

}

**Attempt.java**

@Entity

@Table(name = "attempt")

public class Attempt {

@Id

private int id;

@ManyToOne

@JoinColumn(name = "user\_id")

private User user;

private Date date;

@OneToMany(mappedBy = "attempt", fetch = FetchType.LAZY)

private List<AttemptQuestion> attemptQuestions;

// Getters, setters, toString()

}

**Question.java**

@Entity

@Table(name = "question")

public class Question {

@Id

private int id;

private String text;

private double score;

@OneToMany(mappedBy = "question", fetch = FetchType.LAZY)

private List<Option> options;

// Getters, setters

}

**Option.java**

@Entity

@Table(name = "options")

public class Option {

@Id

private int id;

private String text;

private boolean isCorrect;

@ManyToOne

@JoinColumn(name = "question\_id")

private Question question;

// Getters, setters

}

**AttemptQuestion.java**

@Entity

@Table(name = "attempt\_question")

public class AttemptQuestion {

@Id

private int id;

@ManyToOne

@JoinColumn(name = "attempt\_id")

private Attempt attempt;

@ManyToOne

@JoinColumn(name = "question\_id")

private Question question;

@OneToMany(mappedBy = "attemptQuestion", fetch = FetchType.LAZY)

private List<AttemptOption> attemptOptions;

// Getters, setters

}

**AttemptOption.java**

@Entity

@Table(name = "attempt\_option")

public class AttemptOption {

@Id

private int id;

@ManyToOne

@JoinColumn(name = "attempt\_question\_id")

private AttemptQuestion attemptQuestion;

@ManyToOne

@JoinColumn(name = "option\_id")

private Option option;

private boolean selected;

// Getters, setters

}

**AttemptRepository.java**

@Repository

public interface AttemptRepository extends JpaRepository<Attempt, Integer> {

@Query("SELECT a FROM Attempt a " +

"LEFT JOIN FETCH a.user u " +

"LEFT JOIN FETCH a.attemptQuestions aq " +

"LEFT JOIN FETCH aq.question q " +

"LEFT JOIN FETCH q.options o " +

"LEFT JOIN FETCH aq.attemptOptions ao " +

"LEFT JOIN FETCH ao.option opt " +

"WHERE u.id = :userId AND a.id = :attemptId")

Attempt getAttempt(@Param("userId") int userId, @Param("attemptId") int attemptId);

}

**AttemptService.java**

@Service

public class AttemptService {

@Autowired

private AttemptRepository attemptRepository;

@Transactional

public Attempt getAttempt(int userId, int attemptId) {

return attemptRepository.getAttempt(userId, attemptId);

}

}

**OrmLearnApplication.java**

attemptService = context.getBean(AttemptService.class);

testGetAttemptDetails();

**testGetAttemptDetails()**

private static void testGetAttemptDetails() {

LOGGER.info("Start");

Attempt attempt = attemptService.getAttempt(1, 1);

LOGGER.debug("User: {}", attempt.getUser().getName());

LOGGER.debug("Attempt Date: {}", attempt.getDate());

for (AttemptQuestion aq : attempt.getAttemptQuestions()) {

Question q = aq.getQuestion();

System.out.println(q.getText());

for (Option option : q.getOptions()) {

boolean selected = aq.getAttemptOptions().stream()

.anyMatch(ao -> ao.getOption().getId() == option.getId() && ao.isSelected());

System.out.printf(" %d) %-10s %-4.1f %s\n",

option.getId(), option.getText(), q.getScore(), selected);

}

System.out.println();

}

LOGGER.info("End");

}

**Handson 4**

**Modify EmployeeRepository**

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

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

double getAverageSalary();

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

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

}

**Update EmployeeService**

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public double getAverageSalary() {

return employeeRepository.getAverageSalary();

}

@Transactional

public double getAverageSalary(int departmentId) {

return employeeRepository.getAverageSalary(departmentId);

}

}

**Modify OrmLearnApplication.java**

employeeService = context.getBean(EmployeeService.class);

testGetAverageSalary();

testGetAverageSalaryForDepartment();

**Add Test methods**

private static void testGetAverageSalary() {

LOGGER.info("Start");

double avgSalary = employeeService.getAverageSalary();

LOGGER.debug("Average Salary (All Employees): {}", avgSalary);

LOGGER.info("End");

}

private static void testGetAverageSalaryForDepartment() {

LOGGER.info("Start");

int departmentId = 2;

double avgSalary = employeeService.getAverageSalary(departmentId);

LOGGER.debug("Average Salary (Dept ID {}): {}", departmentId, avgSalary);

LOGGER.info("End");

}

**Handson 5**

**Update EmployeeRepository**

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

List<Employee> getAllEmployeesNative();

**Update EmployeeService**

@Transactional

public List<Employee> getAllEmployeesNative() {

return employeeRepository.getAllEmployeesNative();

}

**Update OrmLearnApplication.java**

employeeService = context.getBean(EmployeeService.class);

testGetAllEmployeesNative();

**Add Test method**

private static void testGetAllEmployeesNative() {

LOGGER.info("Start");

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

for (Employee e : employees) {

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

}

LOGGER.info("End");

}

**Handson 6**

**Product.java**

@Entity

public class Product {

@Id

private int id;

private String name;

private int ramSize;

private double cpuSpeed;

private String hardDiskSize;

private double weight;

private String cpu;

private double customerReview;

private String operatingSystem;

// Getters and Setters

}

**ProductRepositoryCustom.java**

public interface ProductRepositoryCustom {

List<Product> searchProducts(String name, Integer ramSize, String cpu, String os, Double review, Double cpuSpeed);

}

**ProductRepositoryImpl.java**

@Repository

public class ProductRepositoryImpl implements ProductRepositoryCustom {

@PersistenceContext

private EntityManager entityManager;

@Override

public List<Product> searchProducts(String name, Integer ramSize, String cpu, String os, Double review, Double cpuSpeed) {

CriteriaBuilder cb = entityManager.getCriteriaBuilder();

CriteriaQuery<Product> query = cb.createQuery(Product.class); Root<Product> root = query.from(Product.class);

List<Predicate> predicates = new ArrayList<>();

if (name != null && !name.isEmpty()) {

predicates.add(cb.like(cb.lower(root.get("name")), "%" + name.toLowerCase() + "%"));

}

if (ramSize != null) {

predicates.add(cb.ge(root.get("ramSize"), ramSize));

}

if (cpu != null && !cpu.isEmpty()) {

predicates.add(cb.equal(root.get("cpu"), cpu));

}

if (os != null && !os.isEmpty()) {

predicates.add(cb.equal(root.get("operatingSystem"), os));

}

if (review != null) {

predicates.add(cb.ge(root.get("customerReview"), review));

}

if (cpuSpeed != null) {

predicates.add(cb.ge(root.get("cpuSpeed"), cpuSpeed));

}

query.select(root).where(cb.and(predicates.toArray(new Predicate[0]))); return entityManager.createQuery(query).getResultList();

}

}

**ProductRepository.java**

public interface ProductRepository extends JpaRepository<Product, Integer>, ProductRepositoryCustom {

}

**ProductService.java**

public interface ProductService {

List<Product> dynamicSearch(String name, Integer ramSize, String cpu, String os, Double review, Double cpuSpeed);

}

**ProductServiceImpl.java**

@Service

public class ProductServiceImpl implements ProductService {

@Autowired

private ProductRepository productRepository;

@Override

public List<Product> dynamicSearch(String name, Integer ramSize, String cpu, String os, Double review, Double cpuSpeed) {

return productRepository.searchProducts(name, ramSize, cpu, os, review, cpuSpeed);

}

}

**OrmLearnApplication.java**

@Autowired

private ProductService productService;

private void testDynamicSearch() {

List<Product> results = productService.dynamicSearch("laptop", 8, "Intel i5", "Windows", 4.0, null); for (Product product : results) {

System.out.println(product.getName() + " - " + product.getRamSize() + "GB RAM - " + product.getCpu());

}

}