**Hands On 3: Fetch quiz attempt details using HQL**

**Attempt.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.time.LocalDate;

import java.util.List;

@Entity

public class Attempt {

@Id

@Column(name = "at\_id")

private int atId;

@Column(name = "at\_date")

private LocalDate atDate;

@Column(name = "at\_score")

private double atScore;

@ManyToOne

@JoinColumn(name = "at\_us\_id")

private User user;

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

private List<AttemptQuestion> attemptQuestions;

public int getAtId() {

return atId;

}

public void setAtId(int atId) {

this.atId = atId;

}

public LocalDate getAtDate() {

return atDate;

}

public void setAtDate(LocalDate atDate) {

this.atDate = atDate;

}

public double getAtScore() {

return atScore;

}

public void setAtScore(double atScore) {

this.atScore = atScore;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public List<AttemptQuestion> getAttemptQuestions() {

return attemptQuestions;

}

public void setAttemptQuestions(List<AttemptQuestion> attemptQuestions) {

this.attemptQuestions = attemptQuestions;

}

@Override

public String toString() {

return "Attempt{" +

"id=" + atId +

", date=" + atDate +

", score=" + atScore +

", user=" + (user != null ? user.getName() : null) +

'}';

}

}

**AttemptOption.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "attempt\_option")

public class AttemptOption {

@Id

@Column(name = "ao\_id")

private int aoId;

@ManyToOne

@JoinColumn(name = "ao\_aq\_id")

private AttemptQuestion attemptQuestion;

@ManyToOne

@JoinColumn(name = "ao\_op\_id")

private Options option;

@Column(name = "ao\_selected")

private boolean aoSelected;

public int getAoId() {

return aoId;

}

public void setAoId(int aoId) {

this.aoId = aoId;

}

public AttemptQuestion getAttemptQuestion() {

return attemptQuestion;

}

public void setAttemptQuestion(AttemptQuestion attemptQuestion) {

this.attemptQuestion = attemptQuestion;

}

public Options getOption() {

return option;

}

public void setOption(Options option) {

this.option = option;

}

public boolean isAoSelected() {

return aoSelected;

}

public void setAoSelected(boolean aoSelected) {

this.aoSelected = aoSelected;

}

@Override

public String toString() {

return "AttemptOption{" +

"aoId=" + aoId +

", option=" + (option != null ? option.getText() : null) +

", score=" + (option != null ? option.getScore() : 0) +

", selected=" + aoSelected +

'}';

}

}

**AttemptQuestion.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.util.Set;

@Entity

@Table(name = "attempt\_question")

public class AttemptQuestion {

@Id

@Column(name = "aq\_id")

private int aqId;

@ManyToOne

@JoinColumn(name = "aq\_at\_id")

private Attempt attempt;

@ManyToOne

@JoinColumn(name = "aq\_qt\_id")

private Question question;

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

private Set<AttemptOption> attemptOptions;

public int getAqId() {

return aqId;

}

public void setAqId(int aqId) {

this.aqId = aqId;

}

public Attempt getAttempt() {

return attempt;

}

public void setAttempt(Attempt attempt) {

this.attempt = attempt;

}

public Question getQuestion() {

return question;

}

public void setQuestion(Question question) {

this.question = question;

}

public Set<AttemptOption> getAttemptOptions() {

return attemptOptions;

}

public void setAttemptOptions(Set<AttemptOption> attemptOptions) {

this.attemptOptions = attemptOptions;

}

@Override

public String toString() {

return "AttemptQuestion{" +

"aqId=" + aqId +

", question=" + (question != null ? question.getText() : null) +

'}';

}

}

**Options.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "options")

public class Options {

@Id

@Column(name = "op\_id")

private int id;

@Column(name = "op\_score")

private double score;

@Column(name = "op\_text")

private String text;

@ManyToOne

@JoinColumn(name = "op\_qt\_id")

private Question question;

// ✅ Getters and Setters

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public double getScore() {

return score;

}

public void setScore(double score) {

this.score = score;

}

public String getText() {

return text;

}

public void setText(String text) {

this.text = text;

}

public Question getQuestion() {

return question;

}

public void setQuestion(Question question) {

this.question = question;

}

@Override

public String toString() {

return "Options{" +

"id=" + id +

", score=" + score +

", text='" + text + '\'' +

'}';

}

}

**Question.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.util.Set;

@Entity

@Table(name = "question")

public class Question {

@Id

@Column(name = "qt\_id")

private int id;

@Column(name = "qt\_text")

private String text;

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

private Set<Options> options;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getText() {

return text;

}

public void setText(String text) {

this.text = text;

}

public Set<Options> getOptions() {

return options;

}

public void setOptions(Set<Options> options) {

this.options = options;

}

@Override

public String toString() {

return "Question{" +

"id=" + id +

", text='" + text + '\'' +

'}';

}

}

**User.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.util.List;

@Entity

@Table(name = "quiz\_user") // ✅ matches schema table name

public class User {

@Id

@Column(name = "us\_id")

private int id;

@Column(name = "us\_name")

private String name;

@Column(name = "us\_email")

private String email;

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

private List<Attempt> attempts;

// ✅ 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 String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public List<Attempt> getAttempts() {

return attempts;

}

public void setAttempts(List<Attempt> attempts) {

this.attempts = attempts;

}

@Override

public String toString() {

return "User{" +

"id=" + id +

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

", email='" + email + '\'' +

'}';

}

}

**AttemptRepository**

package com.cognizant.orm\_learn.repository;

import com.cognizant.orm\_learn.model.Attempt;

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

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

import org.springframework.data.repository.CrudRepository;

public interface AttemptRepository extends CrudRepository<Attempt, Integer> {

@Query("""

SELECT DISTINCT a FROM Attempt a

JOIN FETCH a.user u

JOIN FETCH a.attemptQuestions aq

JOIN FETCH aq.question q

JOIN FETCH q.options o

JOIN FETCH aq.attemptOptions ao

JOIN FETCH ao.option op

WHERE u.id = :userId AND a.atId = :attemptId

""")

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

}

**UserRepository**

package com.cognizant.orm\_learn.repository;

import org.springframework.stereotype.Repository;

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

import com.cognizant.orm\_learn.model.User;

@Repository

public interface UserRepository extends JpaRepository<User, Integer> {}

**Attempt.service**

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Attempt;

public interface AttemptService {

Attempt getAttempt(int userId, int attemptId);

}

**AttemptServiceImpl**

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Attempt;

import com.cognizant.orm\_learn.repository.AttemptRepository;

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

import org.springframework.stereotype.Service;

@Service

public class AttemptServiceImpl implements AttemptService {

@Autowired

private AttemptRepository attemptRepository;

@Override

public Attempt getAttempt(int userId, int attemptId) {

return attemptRepository.getAttemptDetails(userId, attemptId);

}

}

**Datasql**

INSERT INTO department (dp\_id, dp\_name) VALUES

(1, 'HR'),

(2, 'IT');

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

(1, 'Alice', 50000, true, 1, '1990-01-01'),

(2, 'Bob', 60000, true, 1, '1985-06-15');

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

(3, 'Charlie', 80000, true, 2, '1992-11-20');

INSERT INTO question (qt\_id, qt\_text) VALUES

(1, 'What is the extension of the hyper text markup language file?'),

(2, 'What is the maximum level of heading tag can be used in a HTML page?'),

(3, 'The HTML document itself begins with <html> and ends </html>. State True or False'),

(4, 'Choose the right option to store text value in a variable');

-- Q1

INSERT INTO options (op\_id, op\_text, op\_score, op\_qt\_id) VALUES

(1, '.xhtm', 0.0, 1),

(2, '.ht', 0.0, 1),

(3, '.html', 1.0, 1),

(4, '.htmx', 0.0, 1);

-- Q2

INSERT INTO options (op\_id, op\_text, op\_score, op\_qt\_id) VALUES

(5, '5', 0.0, 2),

(6, '3', 0.0, 2),

(7, '4', 0.0, 2),

(8, '6', 1.0, 2);

-- Q3

INSERT INTO options (op\_id, op\_text, op\_score, op\_qt\_id) VALUES

(9, 'false', 0.0, 3),

(10, 'true', 1.0, 3);

-- Q4

INSERT INTO options (op\_id, op\_text, op\_score, op\_qt\_id) VALUES

(11, '''John''', 0.5, 4),

(12, 'John', 0.0, 4),

(13, '"John"', 0.5, 4),

(14, '/John/', 0.0, 4);

**OrmLearnApplication**

package com.cognizant.orm\_learn;

import java.time.LocalDate;

import java.util.List;

import java.util.Set;

import java.util.HashSet;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

import com.cognizant.orm\_learn.model.\*;

import com.cognizant.orm\_learn.repository.CountryRepository;

import com.cognizant.orm\_learn.repository.UserRepository;

import com.cognizant.orm\_learn.repository.AttemptRepository;

import com.cognizant.orm\_learn.service.CountryService;

import com.cognizant.orm\_learn.service.EmployeeService;

import com.cognizant.orm\_learn.service.AttemptService;

@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");

}

@Bean

CommandLineRunner run(CountryRepository repo,

CountryService countryService,

EmployeeService employeeService,

AttemptService attemptService,

UserRepository userRepository,

AttemptRepository attemptRepository) {

return args -> {

LOGGER.info("Start inserting quiz attempt data");

User user1 = new User();

user1.setId(1);

user1.setName("Test User");

user1.setEmail("test@example.com");

userRepository.save(user1); // user ID 1

Question q1 = new Question();

q1.setId(1);

q1.setText("What is the extension of the hyper text markup language file?");

Options q1o1 = new Options();

q1o1.setId(1);

q1o1.setText(".xhtm");

q1o1.setScore(0.0);

q1o1.setQuestion(q1);

Options q1o2 = new Options();

q1o2.setId(2);

q1o2.setText(".ht");

q1o2.setScore(0.0);

q1o2.setQuestion(q1);

Options q1o3 = new Options();

q1o3.setId(3);

q1o3.setText(".html");

q1o3.setScore(1.0);

q1o3.setQuestion(q1);

Options q1o4 = new Options();

q1o4.setId(4);

q1o4.setText(".htmx");

q1o4.setScore(0.0);

q1o4.setQuestion(q1);

q1.setOptions(Set.of(q1o1, q1o2, q1o3, q1o4));

Question q2 = new Question();

q2.setId(2);

q2.setText("What is the maximum level of heading tag can be used in a HTML page?");

Options q2o1 = new Options();

q2o1.setId(5);

q2o1.setText("5");

q2o1.setScore(0.0);

q2o1.setQuestion(q2);

Options q2o2 = new Options();

q2o2.setId(6);

q2o2.setText("3");

q2o2.setScore(0.0);

q2o2.setQuestion(q2);

Options q2o3 = new Options();

q2o3.setId(7);

q2o3.setText("4");

q2o3.setScore(0.0);

q2o3.setQuestion(q2);

Options q2o4 = new Options();

q2o4.setId(8);

q2o4.setText("6");

q2o4.setScore(1.0);

q2o4.setQuestion(q2);

q2.setOptions(Set.of(q2o1, q2o2, q2o3, q2o4));

Question q3 = new Question();

q3.setId(3);

q3.setText("The HTML document itself begins with <html> and ends </html>. State True or False");

Options q3o1 = new Options();

q3o1.setId(9);

q3o1.setText("False");

q3o1.setScore(0.0);

q3o1.setQuestion(q3);

Options q3o2 = new Options();

q3o2.setId(10);

q3o2.setText("Tru");

q3o2.setScore(1.0);

q3o2.setQuestion(q3);

q3.setOptions(Set.of(q3o1, q3o2));

Question q4 = new Question();

q4.setId(4);

q4.setText("Choose the right option to store text value in a variable");

Options q4o1 = new Options();

q4o1.setId(11);

q4o1.setText("John");

q4o1.setScore(0.5);

q4o1.setQuestion(q4);

Options q4o2 = new Options();

q4o2.setId(12);

q4o2.setText("John");

q4o2.setScore(0.0);

q4o2.setQuestion(q4);

Options q4o3 = new Options();

q4o3.setId(13);

q4o3.setText("\"John\"");

q4o3.setScore(0.5);

q4o3.setQuestion(q4);

Options q4o4 = new Options();

q4o4.setId(14);

q4o4.setText("/John/");

q4o4.setScore(0.0);

q4o4.setQuestion(q4);

q4.setOptions(Set.of(q4o1, q4o2, q4o3, q4o4));

Attempt attempt = new Attempt();

attempt.setAtId(101);

attempt.setAtDate(LocalDate.now());

attempt.setAtScore(1);

attempt.setUser(user1);

AttemptQuestion aq1 = new AttemptQuestion();

aq1.setAqId(1001);

aq1.setAttempt(attempt);

aq1.setQuestion(q1);

AttemptOption ao1 = new AttemptOption();

ao1.setAoId(501);

ao1.setAoSelected(true);

ao1.setAttemptQuestion(aq1);

ao1.setOption(q1o3); // .html

aq1.setAttemptOptions(Set.of(ao1));

AttemptQuestion aq2 = new AttemptQuestion();

aq2.setAqId(1002);

aq2.setAttempt(attempt);

aq2.setQuestion(q2);

AttemptOption ao2 = new AttemptOption();

ao2.setAoId(502);

ao2.setAoSelected(true);

ao2.setAttemptQuestion(aq2);

ao2.setOption(q2o4); // 6

aq2.setAttemptOptions(Set.of(ao2));

AttemptQuestion aq3 = new AttemptQuestion();

aq3.setAqId(1003);

aq3.setAttempt(attempt);

aq3.setQuestion(q3);

AttemptOption ao3 = new AttemptOption();

ao3.setAoId(503);

ao3.setAoSelected(true);

ao3.setAttemptQuestion(aq3);

ao3.setOption(q3o2); // true

aq3.setAttemptOptions(Set.of(ao3));

AttemptQuestion aq4 = new AttemptQuestion();

aq4.setAqId(1004);

aq4.setAttempt(attempt);

aq4.setQuestion(q4);

AttemptOption ao4 = new AttemptOption();

ao4.setAoId(504);

ao4.setAoSelected(true);

ao4.setAttemptQuestion(aq4);

ao4.setOption(q4o1); // 'John'

aq4.setAttemptOptions(Set.of(ao4));

attempt.setAttemptQuestions(List.of(aq1, aq2, aq3, aq4));

attemptRepository.save(attempt);

LOGGER.info("Start fetching quiz attempt details");

int userId = 1;

int attemptId = 101;

var fetchedAttempt = attemptService.getAttempt(userId, attemptId);

if (fetchedAttempt != null) {

LOGGER.info("Username: {}", fetchedAttempt.getUser().getName());

LOGGER.info("Attempted Date: {}", fetchedAttempt.getAtDate());

Set<Integer> printedQuestionIds = new HashSet<>();

fetchedAttempt.getAttemptQuestions().forEach(aqFetched -> {

Question questionFetched = aqFetched.getQuestion();

if (!printedQuestionIds.contains(questionFetched.getId())) {

LOGGER.info("❓ {}", questionFetched.getText());

Set<Options> allOptions = questionFetched.getOptions();

allOptions.forEach(opt -> {

boolean isSelected = aqFetched.getAttemptOptions().stream()

.anyMatch(selectedAo -> selectedAo.getOption().getId() == opt.getId());

LOGGER.info(" ➤ {}. {} | Score: {} | Selected: {}",

opt.getId(), opt.getText(), opt.getScore(), isSelected);

});

printedQuestionIds.add(questionFetched.getId());

}

});

LOGGER.info("✅ End of quiz attempt details");

} else {

LOGGER.warn("❌ Attempt not found for userId={} and attemptId={}", userId, attemptId);

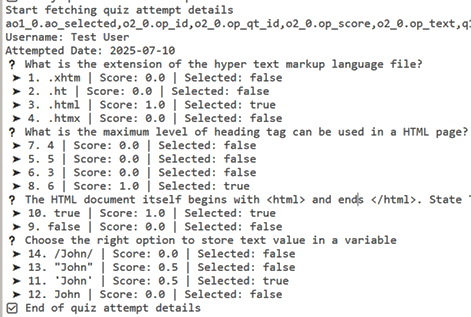
}

};

}

}

**Output**

****