Digital Nurture 4.0

Deep Skilling - Java FSE

WEEK-3 HANDS ON

By Kaviya P

**3.SPRING DATA JPA HANSON**

**Hands on 1**

**Introduction to HQL and JPQL** 

 **HQL stands for Hibernate Query Language**  
-> HQL is a powerful, object-oriented query language developed as part of the Hibernate framework. It allows querying against entity objects rather than database tables directly.

 **JPQL stands for Java Persistence Query Language**  
->JPQL is the official query language defined by the Java Persistence API (JPA) specification. Like HQL, it operates on entity objects and their properties.

 **Both HQL and JPQL are object-focused query languages similar to SQL**  
-> While SQL works directly with tables and columns, HQL and JPQL work with Java objects (entities) and their fields, which makes them suitable for ORM (Object Relational Mapping).

 **JPQL is a subset of HQL**  
-> All JPQL features are included in HQL, but HQL has additional capabilities, such as support for certain operations that JPQL does not provide (like native INSERT support).

 **All JPQL queries are valid HQL query, but the reverse is not true**  
-> Since JPQL is a limited subset, anything valid in JPQL will also work in HQL. However, HQL supports more features that JPQL does not, so not all HQL queries will be valid in JPQL.

 **Both HQL and JPQL allow SELECT, UPDATE and DELETE**  
-> These core operations are supported in both languages, allowing for basic data retrieval and manipulation on entities.

 **HQL additionally allows INSERT statement**  
-> Unlike JPQL, HQL supports the INSERT INTO operation, making it more flexible for bulk insertions directly using queries.

**Hands on 2**

**Get all permanent employees using HQL**   
  
Using HQL get all permanent employees. When retrieving the employee details it should also retrieve respective department and skill list as well

**SpringAndMavenApplication.java**

**package** com.week3.SpringAndMaven;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.ConfigurableApplicationContext;

**import** com.week3.SpringAndMaven.model.Employee;

**import** com.week3.SpringAndMaven.model.Skill;

**import** com.week3.SpringAndMaven.service.DepartmentService;

**import** com.week3.SpringAndMaven.service.EmployeeService;

**import** com.week3.SpringAndMaven.service.SkillService;

**import** java.util.List;

**import** java.util.Set;

**import** org.springframework.context.ApplicationContext;

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

**private** **static** EmployeeService *employeeService*;

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

**public** **static** **void** main(String[] args) {

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

*employeeService* = context.getBean(EmployeeService.**class**);

*testGetAllPermanentEmployees*();

}

**private** **static** **void** testGetAllPermanentEmployees() {

***LOGGER***.info("Start");

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

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

employees.forEach(e -> ***LOGGER***.debug("Skills:{}", e.getSkillList()));

***LOGGER***.info("End");

}

}

**EmployeeController.java**

**package** com.week3.SpringAndMaven.controller;

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

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

**import** com.week3.SpringAndMaven.model.Employee;

**import** com.week3.SpringAndMaven.service.EmployeeService;

**import** java.util.List;

@RestController

@RequestMapping("/employees")

**public** **class** EmployeeController {

@Autowired

**private** EmployeeService employeeService;

// Get all permanent employees

@GetMapping("/permanent")

**public** List<Employee> getAllPermanentEmployees() {

**return** employeeService.getAllPermanentEmployees();

}

// Get employee by ID

@GetMapping("/{id}")

**public** Employee getEmployeeById(@PathVariable **int** id) {

**return** employeeService.get(id);

}

}

**Employee.java**

**package** com.week3.SpringAndMaven.model;

**import** jakarta.persistence.\*;

**import** java.util.Date;

**import** java.util.HashSet;

**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.***LAZY***)

@JoinTable(

name = "employee\_skill",

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

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

)

**private** Set<Skill> skillList;

**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;

}

// Getters, Setters, toString()

}

**Department.java**

**package** com.week3.SpringAndMaven.model;

**import** java.util.Set;

**import** jakarta.persistence.\*;

@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.***LAZY***)

**private** Set<Employee> employeeList;

// Getters, Setters, toString()

}

**EmployeeRepository.java**

**package** com.week3.SpringAndMaven.repository;

**import** java.util.List;

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

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

**import** org.springframework.stereotype.Repository;

**import** com.week3.SpringAndMaven.model.Employee;

@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();

}

**DepartmentRepository.java**

**package** com.week3.SpringAndMaven.repository;

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

**import** org.springframework.stereotype.Repository;

**import** com.week3.SpringAndMaven.model.Department;

@Repository

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

}

**SkillRepository.java**

**package** com.week3.SpringAndMaven.repository;

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

**import** org.springframework.stereotype.Repository;

**import** com.week3.SpringAndMaven.model.Skill;

@Repository

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

}

**Skill.java**

**package** com.week3.SpringAndMaven.model;

**import** java.util.Set;

**import** jakarta.persistence.\*;

@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** Set<Employee> employeeList;

// Getters, Setters, toString()

}

**EmployeeService.java**

**package** com.week3.SpringAndMaven.service;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.week3.SpringAndMaven.model.Employee;

**import** com.week3.SpringAndMaven.repository.EmployeeRepository;

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

**import** java.util.List;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

@Service

**public** **class** EmployeeService {

@Autowired

**private** EmployeeRepository employeeRepository;

**public** List<Employee> getAllPermanentEmployees() {

**return** employeeRepository.getAllPermanentEmployees();

}

**public** Employee get(**int** id) {

**return** employeeRepository.findById(id).orElseThrow();

}

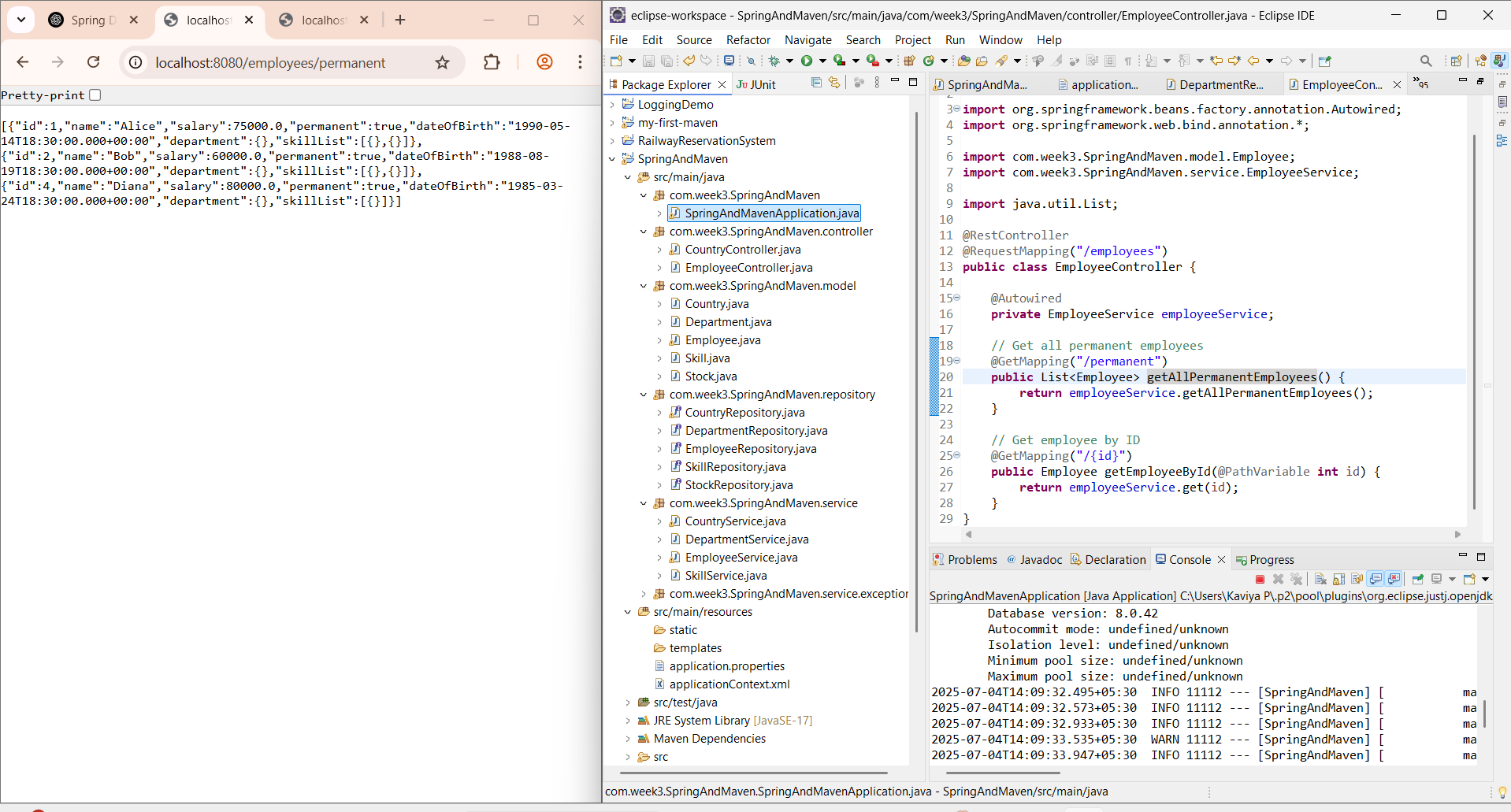
**public** **void** save(Employee employee) {

employeeRepository.save(employee);

}

}

**OUTPUT**

****

**Hands on 3**

**Fetch quiz attempt details using HQL**

**Attempt.java**

**package** com.week3.SpringAndMaven.model1;

**import** java.sql.Date;

**import** java.util.Set;

**import** jakarta.persistence.\*;

@Entity

@Table(name = "attempt")

**public** **class** Attempt {

@Id

**private** **int** id;

@Column(name = "attempted\_date")

**private** Date attemptedDate;

@ManyToOne

@JoinColumn(name = "user\_id")

**private** User user;

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

**private** Set<AttemptQuestion> attemptQuestions;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** Date getAttemptedDate() {

**return** attemptedDate;

}

**public** **void** setAttemptedDate(Date attemptedDate) {

**this**.attemptedDate = attemptedDate;

}

**public** User getUser() {

**return** user;

}

**public** **void** setUser(User user) {

**this**.user = user;

}

**public** Set<AttemptQuestion> getAttemptQuestions() {

**return** attemptQuestions;

}

**public** **void** setAttemptQuestions(Set<AttemptQuestion> attemptQuestions) {

**this**.attemptQuestions = attemptQuestions;

}

// Getters, Setters, toString

}

**AttemptOption.java**

**package** com.week3.SpringAndMaven.model1;

**import** java.sql.Date;

**import** java.util.Set;

**import** jakarta.persistence.\*;

@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, toString

}

**AttemptQuestion.java**

**package** com.week3.SpringAndMaven.model1;

**import** java.sql.Date;

**import** java.util.Set;

**import** jakarta.persistence.\*;

@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")

**private** Set<AttemptOption> attemptOptions;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**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;

}

// Getters, Setters, toString

}

**Question.java**

**package** com.week3.SpringAndMaven.model1;

**import** java.sql.Date;

**import** java.util.Set;

**import** jakarta.persistence.\*;

@Entity

@Table(name = "question")

**public** **class** Question{

@Id

**private** **int** id;

**private** String content;

**private** **double** score;

**private** **boolean** isAnswer;

@ManyToOne

@JoinColumn(name = "question\_id")

**private** Question question;

// Getters, Setters, toString

}

**Option.java**

**package** com.week3.SpringAndMaven.model1;

**import** java.sql.Date;

**import** java.util.Set;

**import** jakarta.persistence.\*;

@Entity

@Table(name = "options")

**public** **class** Option {

@Id

**private** **int** id;

**private** String content;

**private** **double** score;

**private** **boolean** isAnswer;

@ManyToOne

@JoinColumn(name = "question\_id")

**private** Question question;

// Getters, Setters, toString

}

**User.java**

**package** com.week3.SpringAndMaven.model1;

**import** java.sql.Date;

**import** java.util.Set;

**import** jakarta.persistence.\*;

@Entity

@Table(name = "user")

**public** **class** User {

@Id

**private** **int** id;

**private** String name;

@OneToMany(mappedBy = "user")

**private** Set<Attempt> attempts;

// Getters, Setters, toString

}

**AttemptService.java**

**package** com.week3.SpringAndMaven.repository;

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

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

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

**import** org.springframework.stereotype.Repository;

**import** com.week3.SpringAndMaven.model1.Attempt;

@Repository

**public** **interface** AttemptRepository **extends** JpaRepository<Attempt, Integer> {

@Query("SELECT 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 opt "

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

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

}

**package** com.week3.SpringAndMaven.service;

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

**import** org.springframework.stereotype.Service;

**import** com.week3.SpringAndMaven.model1.Attempt;

**import** com.week3.SpringAndMaven.repository.AttemptRepository;

**import** jakarta.transaction.\*;

@Service

@Transactional

**public** **class** AttemptService {

@Autowired

**private** AttemptRepository attemptRepository;

**public** Attempt getAttempt(**int** userId, **int** attemptId) {

**return** attemptRepository.getAttempt(userId, attemptId);

}

}

**SpringAndMavenApplication.java**

**package** com.week3.SpringAndMaven;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.ConfigurableApplicationContext;

**import** java.util.\*;

**import** java.util.Set;

**import** org.springframework.context.ApplicationContext;

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

@Autowired

**private** **static** AttemptService *attemptService*;

**public** **static** **void** main(String[] args) {

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

*attemptService* = context.getBean(AttemptService.**class**);

*testGetAttemptDetails*();

}

**private** **static** **void** testGetAttemptDetails() {

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

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

System.***out***.println(aq.getQuestion().getContent());

**for** (Option option : aq.getQuestion().getOptions()) {

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

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

System.***out***.printf(" %d) %s\t%.1f\t%s\n", option.getId(), option.getContent(), option.getScore(), selected);

}

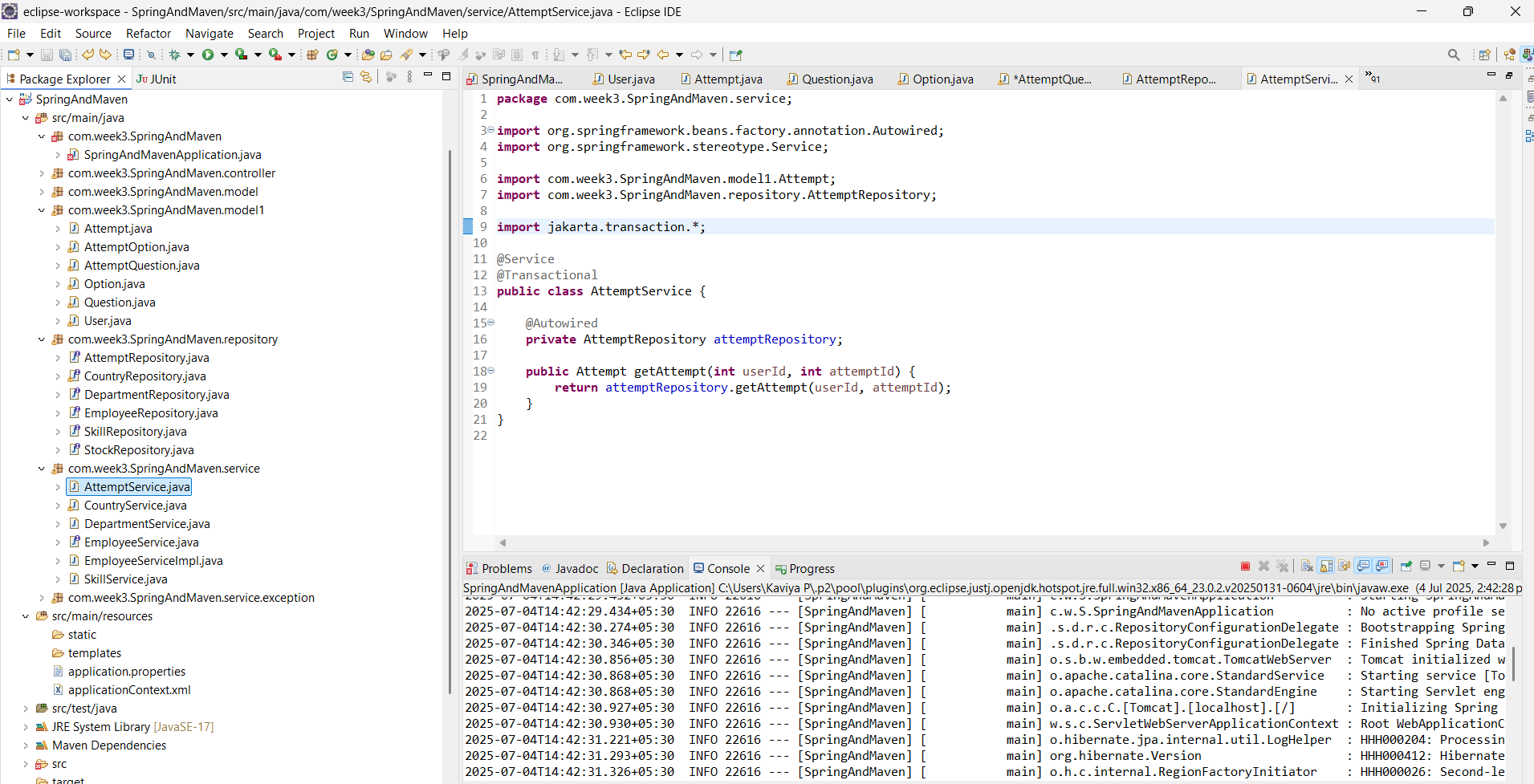
System.***out***.println();

}

}

}

**OUTPUT**

****

**Hands on 4**

**Get average salary using HQL**

**SpringAndMavenApplication.java**  
  
**package** com.week3.SpringAndMaven;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.ConfigurableApplicationContext;

**import** com.week3.SpringAndMaven.model.Employee;

**import** com.week3.SpringAndMaven.model.Skill;

**import** com.week3.SpringAndMaven.service.DepartmentService;

**import** com.week3.SpringAndMaven.service.EmployeeService;

**import** com.week3.SpringAndMaven.service.SkillService;

**import** java.util.List;

**import** java.util.Set;

**import** org.springframework.context.ApplicationContext;

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

**private** **static** EmployeeService *employeeService*;

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

**public** **static** **void** main(String[] args) {

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

*employeeService* = context.getBean(EmployeeService.**class**);

*testGetAverageSalary*();

}

**private** **static** **void** testGetAverageSalary() {

***LOGGER***.info("Start");

**double** avgSalary = *employeeService*.getAverageSalary(1); // department ID = 1

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

***LOGGER***.info("End");

}

}

**EmployeeController.java**

**package** com.week3.SpringAndMaven.controller;

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

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

**import** com.week3.SpringAndMaven.model.Employee;

**import** com.week3.SpringAndMaven.service.EmployeeService;

**import** java.util.List;

@RestController

@RequestMapping("/employees")

**public** **class** EmployeeController {

@Autowired

**private** EmployeeService employeeService;

// Get an employee by ID

@GetMapping("/{id}")

**public** Employee getEmployee(@PathVariable **int** id) {

**return** employeeService.get(id);

}

// Get average salary by department ID

@GetMapping("/avg-salary/{deptId}")

**public** **double** getAverageSalary(@PathVariable("deptId") **int** deptId) {

**return** employeeService.getAverageSalary(deptId);

}

// Add or update an employee

@PostMapping

**public** Employee saveEmployee(@RequestBody Employee employee) {

**return** employeeService.save(employee);

}

}

**Employee.java**

**package** com.week3.SpringAndMaven.model;

**import** jakarta.persistence.\*;

**import** java.util.Date;

**import** java.util.HashSet;

**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.***LAZY***)

@JoinTable(

name = "employee\_skill",

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

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

)

**private** Set<Skill> skillList;

**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;

}

// Getters, Setters, toString()

}

**EmployeeRepository.java**

**package** com.week3.SpringAndMaven.repository;

**import** java.util.List;

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

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

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

**import** org.springframework.stereotype.Repository;

**import** com.week3.SpringAndMaven.model.Employee;

@Repository

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

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

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

}

**EmployeeService.java**

**package** com.week3.SpringAndMaven.service;

**import** com.week3.SpringAndMaven.model.Employee;

**import** java.util.List;

**public** **interface** EmployeeService {

Employee get(**int** id);

**double** getAverageSalary(**int** departmentId);

Employee save(Employee employee);

}

**EmployeeServiceImpl.java**

**package** com.week3.SpringAndMaven.service;

**import** com.week3.SpringAndMaven.model.Employee;

**import** com.week3.SpringAndMaven.repository.EmployeeRepository;

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

**import** org.springframework.stereotype.Service;

**import** java.util.List;

**import** java.util.NoSuchElementException;

@Service

**public** **class** EmployeeServiceImpl **implements** EmployeeService {

@Autowired

**private** EmployeeRepository employeeRepository;

@Override

**public** Employee get(**int** id) {

**return** employeeRepository.findById(id).orElseThrow(() -> **new** NoSuchElementException("Employee not found"));

}

@Override

**public** **double** getAverageSalary(**int** departmentId) {

**return** employeeRepository.getAverageSalary(departmentId);

}

@Override

**public** Employee save(Employee employee) {

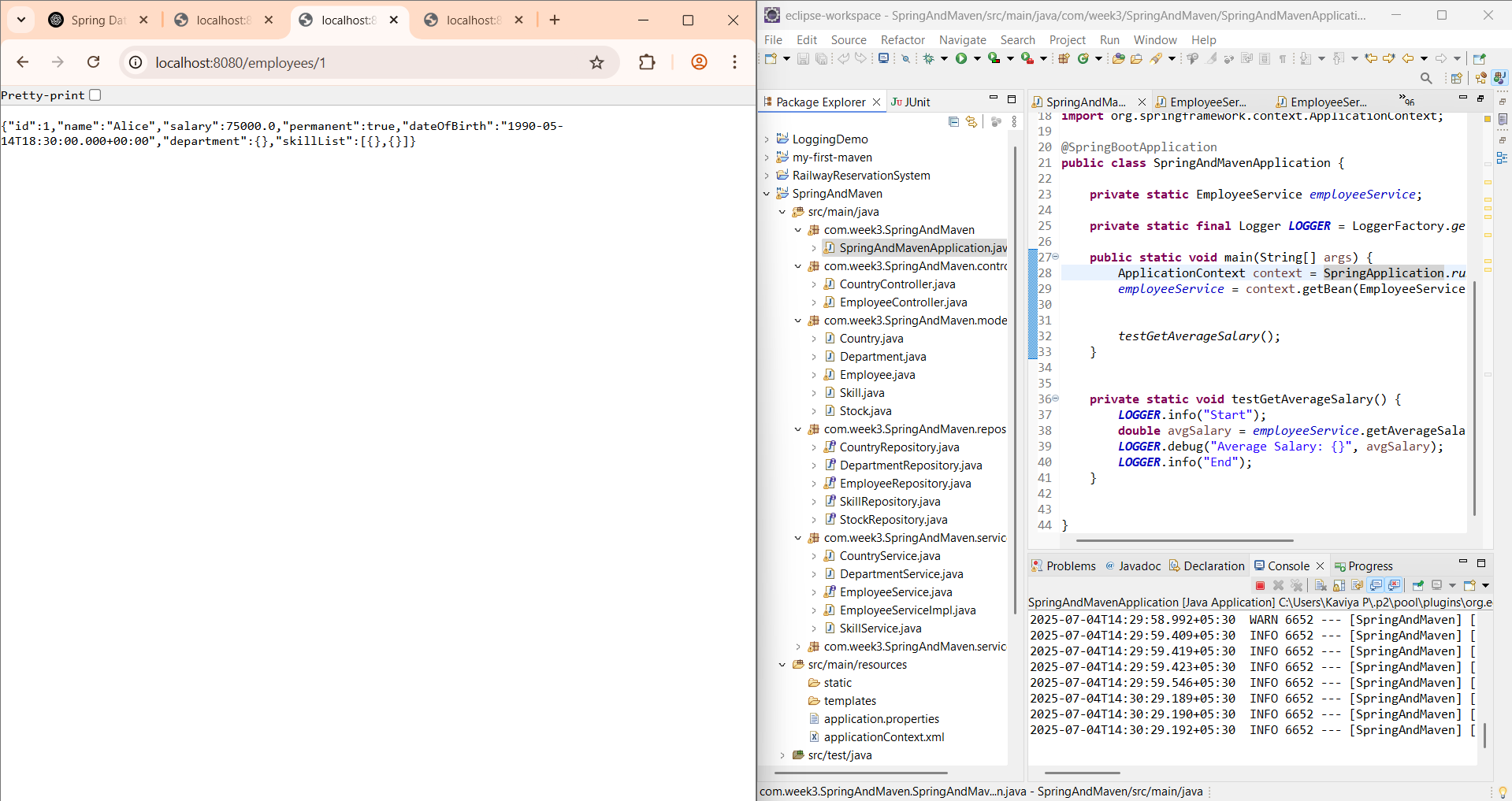
**return** employeeRepository.save(employee);

}

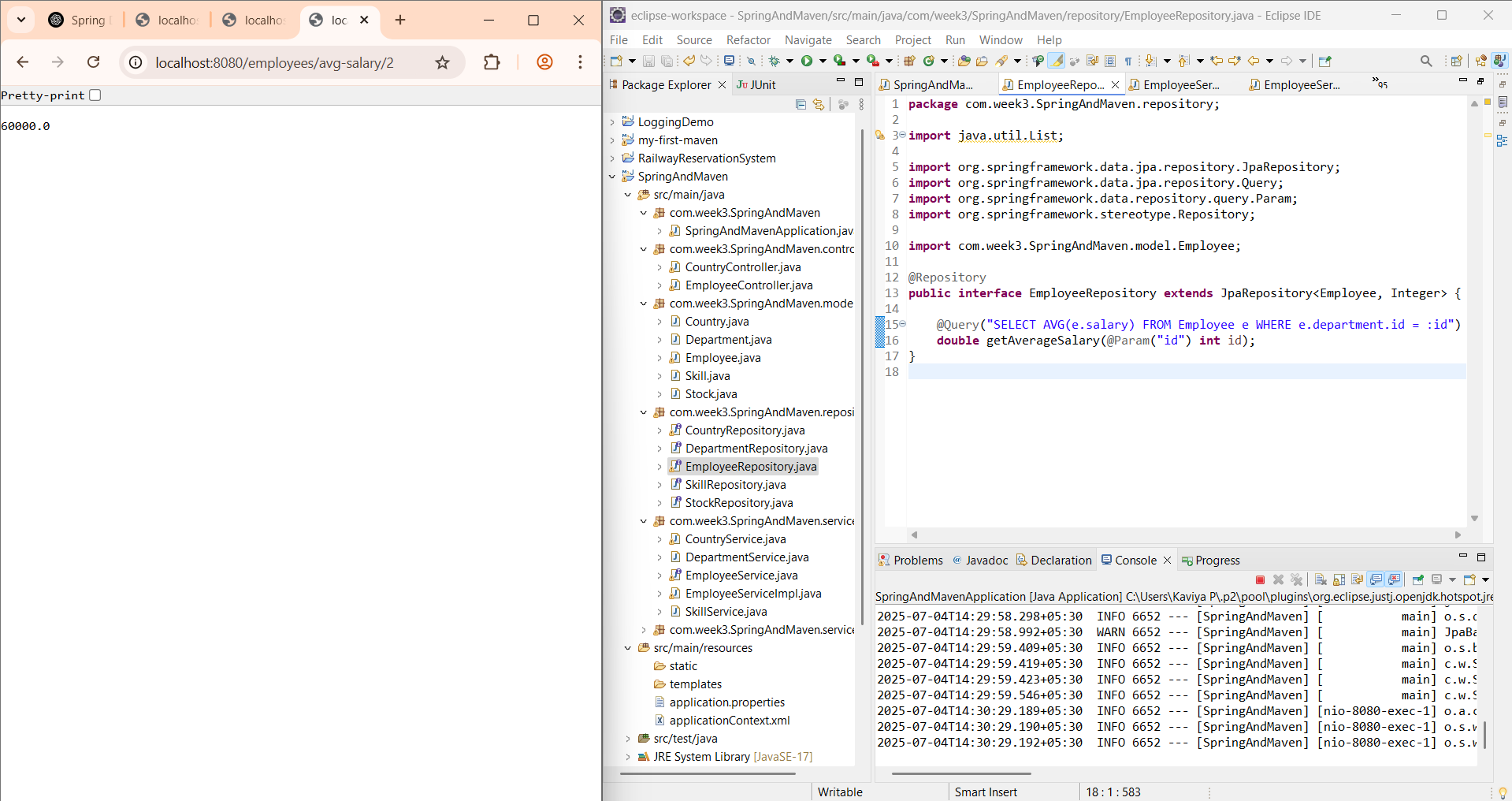
}

**OUTPUT**

**EMPLOYEE 1**

****

**AVERAGE SALARY OF SECOND EMPLOYYE**

****

**Hands on 5**

**Get all employees using Native Query**

**SpringAndMavenApplication.java**

**package** com.week3.SpringAndMaven;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.ConfigurableApplicationContext;

**import** com.week3.SpringAndMaven.model.Employee;

**import** com.week3.SpringAndMaven.model.Skill;

**import** com.week3.SpringAndMaven.service.DepartmentService;

**import** com.week3.SpringAndMaven.service.EmployeeService;

**import** com.week3.SpringAndMaven.service.SkillService;

**import** java.util.List;

**import** java.util.Set;

**import** org.springframework.context.ApplicationContext;

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

**private** **static** EmployeeService *employeeService*;

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

**public** **static** **void** main(String[] args) {

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

*employeeService* = context.getBean(EmployeeService.**class**);

*testGetAverageSalary*();

*testGetAllEmployeesNative*(); // and comment others

}

**private** **static** **void** testGetAllEmployeesNative() {

***LOGGER***.info("Start");

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

***LOGGER***.debug("All Employees (Native): {}", employees);

***LOGGER***.info("End");

}

**private** **static** **void** testGetAverageSalary() {

***LOGGER***.info("Start");

**double** avgSalary = *employeeService*.getAverageSalary(1); // department ID = 1

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

***LOGGER***.info("End");

}

}

**EmployeeController.java**

**package** com.week3.SpringAndMaven.controller;

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

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

**import** com.week3.SpringAndMaven.model.Employee;

**import** com.week3.SpringAndMaven.service.EmployeeService;

**import** java.util.List;

@RestController

@RequestMapping("/employees")

**public** **class** EmployeeController {

@Autowired

**private** EmployeeService employeeService;

// Get an employee by ID

@GetMapping("/{id}")

**public** Employee getEmployee(@PathVariable **int** id) {

**return** employeeService.get(id);

}

@GetMapping("/native")

**public** List<Employee> getAllEmployeesNative() {

**return** employeeService.getAllEmployeesNative();

}

// Get average salary by department ID

@GetMapping("/avg-salary/{deptId}")

**public** **double** getAverageSalary(@PathVariable("deptId") **int** deptId) {

**return** employeeService.getAverageSalary(deptId);

}

// Add or update an employee

@PostMapping

**public** Employee saveEmployee(@RequestBody Employee employee) {

**return** employeeService.save(employee);

}

}

**EmployeeRepository.java**

**package** com.week3.SpringAndMaven.repository;

**import** java.util.List;

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

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

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

**import** org.springframework.stereotype.Repository;

**import** com.week3.SpringAndMaven.model.Employee;

@Repository

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

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

List<Employee> getAllEmployeesNative();

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

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

}

**EmployeeService.java**

**package** com.week3.SpringAndMaven.service;

**import** com.week3.SpringAndMaven.model.Employee;

**import** java.util.List;

**public** **interface** EmployeeService {

Employee get(**int** id);

List<Employee> getAllEmployeesNative();

**double** getAverageSalary(**int** departmentId);

Employee save(Employee employee);

}

**EmployeeServiceImpl.java**

**package** com.week3.SpringAndMaven.service;

**import** com.week3.SpringAndMaven.model.Employee;

**import** com.week3.SpringAndMaven.repository.EmployeeRepository;

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

**import** org.springframework.stereotype.Service;

**import** java.util.List;

**import** java.util.NoSuchElementException;

@Service

**public** **class** EmployeeServiceImpl **implements** EmployeeService {

@Autowired

**private** EmployeeRepository employeeRepository;

@Override

**public** Employee get(**int** id) {

**return** employeeRepository.findById(id).orElseThrow(() -> **new** NoSuchElementException("Employee not found"));

}

@Override

**public** List<Employee> getAllEmployeesNative() {

**return** employeeRepository.getAllEmployeesNative();

}

@Override

**public** **double** getAverageSalary(**int** departmentId) {

**return** employeeRepository.getAverageSalary(departmentId);

}

@Override

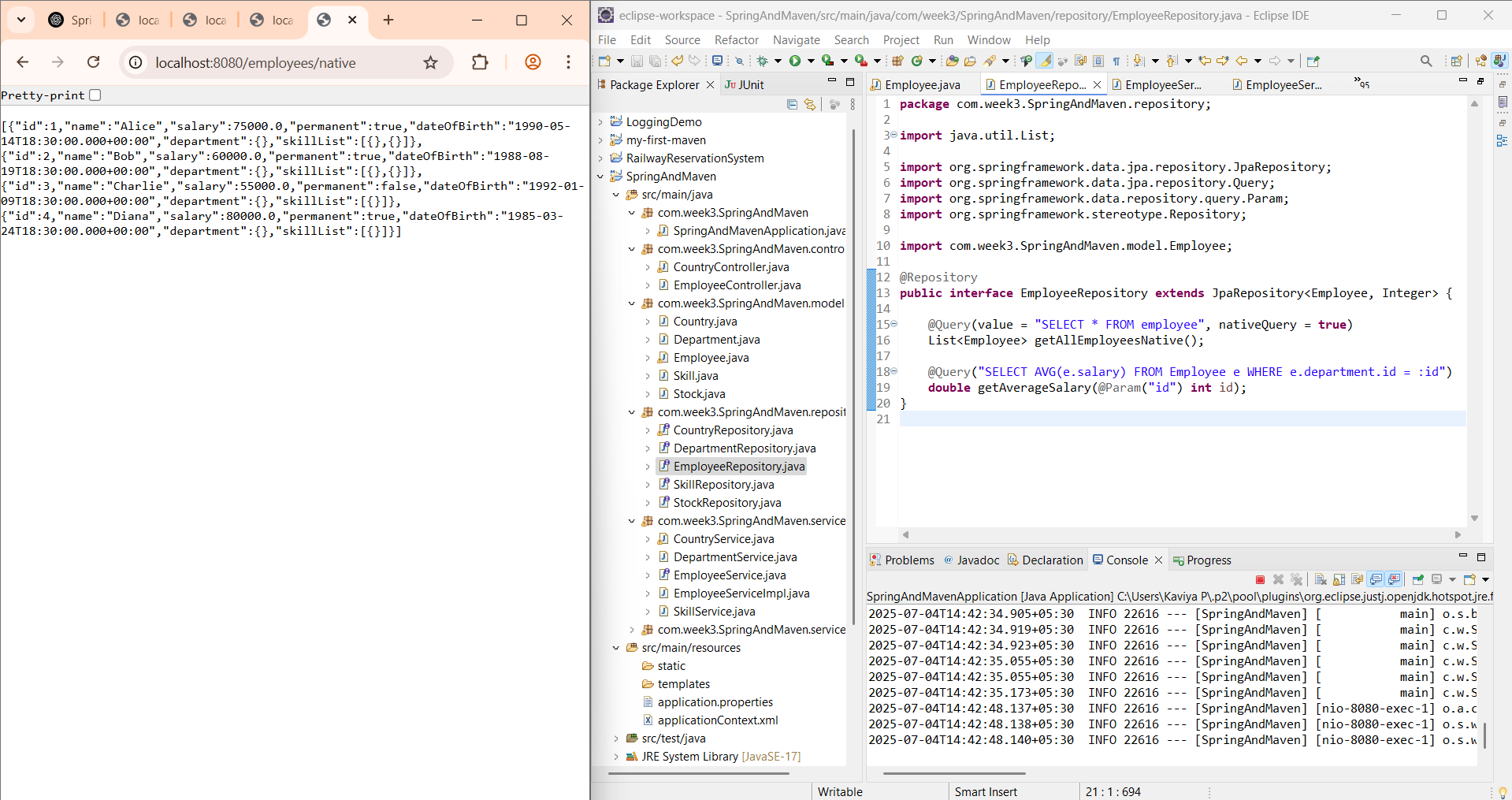
**public** Employee save(Employee employee) {

**return** employeeRepository.save(employee);

}

}

**OUTPUT**

****

**Hands on 6**

**Criteria Query**

In the given scenario of an online retail search (like on Amazon), a user performs a product search using filters such as **Customer Review**, **Hard Disk Size**, **RAM Size**, **CPU Speed**, **Operating System**, **Weight**, and **CPU**. These filters are optional and can be combined in any way, depending on what the user selects.

If we were to write a static HQL query, we'd have to hard-code all possible combinations of these filters, which is inefficient and impractical. This is where **Criteria Query** (also known as the **Criteria API**) in **JPA** or **Hibernate** becomes highly useful.

**Explanation with Scenario**

Let’s say a user searches for "laptop" and applies these filters:

* RAM Size = 16 GB
* Operating System = Windows 11
* CPU = Intel i7

Using **Criteria Query**, we can **programmatically build** the WHERE clause of the query **based on the selected filters only**, instead of writing a fixed HQL string.

**Why Criteria Query is Better for This Scenario**

1. **Dynamic Query Building**:  
   Criteria API lets you add Predicate conditions only for the fields the user has selected filters for.
2. **Type Safety**:  
   Unlike raw HQL, Criteria queries are checked at compile-time, making them less error-prone.
3. **Cleaner and Reusable Code**:  
   Filters can be built using loops, maps, or conditions, avoiding long and unreadable HQL strings.

CriteriaBuilder cb = entityManager.getCriteriaBuilder();

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

Root<Product> product = query.from(Product.class);

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

if (searchKeyword != null) {

    predicates.add(cb.like(product.get("name"), "%" + searchKeyword + "%"));

}

if (ramSize != null) {

    predicates.add(cb.equal(product.get("ramSize"), ramSize));

}

if (operatingSystem != null) {

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

}

if (cpu != null) {

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

}

query.select(product)

     .where(cb.and(predicates.toArray(new Predicate[0])));

List<Product> results = entityManager.createQuery(query).getResultList();