**WEEK-3**

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

**1.** **Spring Data JPA - Quick Example**

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/>   
 </parent>  
 <groupId>com.cognizant</groupId>  
 <artifactId>orm-learn</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>orm-learn</name>  
 <description>Demo project for Spring Data JPA and Hibernate</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>8.0.33</version>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**Country.java**

package com.cognizant.orm\_learn.model;  
  
import jakarta.persistence.Column;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
  
@Entity  
@Table(name = "country")  
public class Country {  
  
 @Id  
 @Column(name = "code")  
 private String code;  
  
 @Column(name = "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.orm\_learn.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
import com.cognizant.orm\_learn.model.Country;  
  
@Repository  
public interface CountryRepository extends JpaRepository<Country, String> {  
}

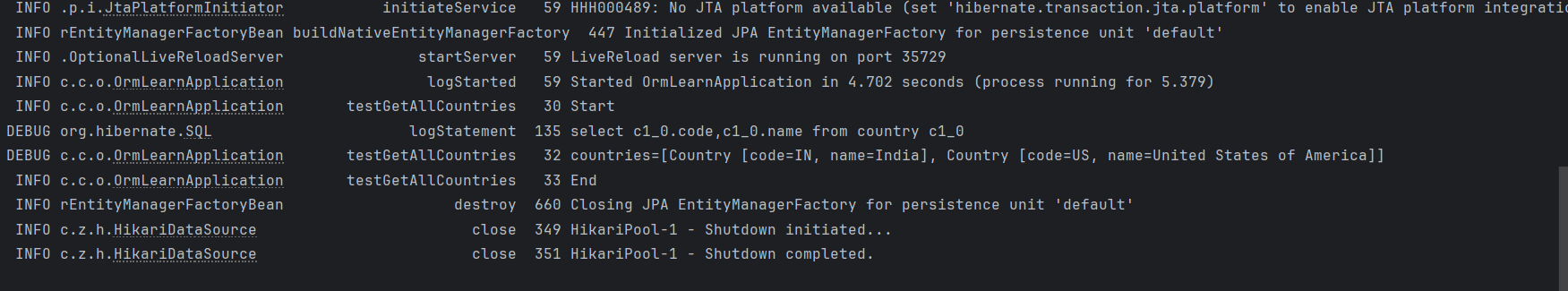
**CountryService.java**

package com.cognizant.orm\_learn.service;  
  
import java.util.List;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.repository.CountryRepository;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
import java.util.List;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.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;  
  
@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);  
 *testGetAllCountries*();  
 }  
  
 private static void testGetAllCountries() {  
 *LOGGER*.info("Start");  
 List<Country> countries = *countryService*.getAllCountries();  
 *LOGGER*.debug("countries={}", countries);  
 *LOGGER*.info("End");  
 }  
}

**Output:**



**2. Difference between JPA, Hibernate and Spring Data JPA**

**Employee.java**

package com.cognizant.orm\_learn.model;  
  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "employee")  
public class Employee {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Integer id;  
  
 @Column(name = "name")  
 private String name;

@Column(name = "salary")  
 private Double salary;  
  
 // Getters and Setters  
 public Integer getId() {  
 return id;  
 }  
 public void setId(Integer 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;  
 }  
  
 @Override  
 public String toString() {  
 return "Employee [id=" + id + ", name=" + name + ", salary=" + salary + "]";  
 }  
}

**EmployeeRepository.java**

package com.cognizant.orm\_learn.repository;  
  
import com.cognizant.orm\_learn.model.Employee;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
}

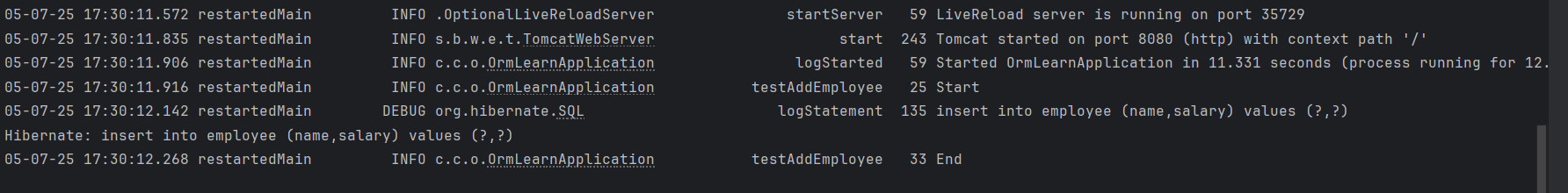
**EmployeeService.java**

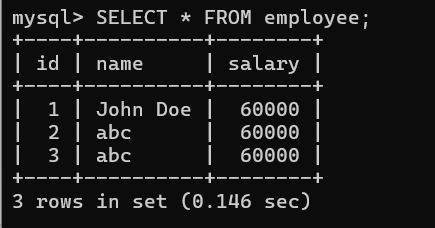
package com.cognizant.orm\_learn.service;  
  
import com.cognizant.orm\_learn.model.Employee;  
import com.cognizant.orm\_learn.repository.EmployeeRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
@Service  
public class EmployeeService {  
  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 @Transactional  
 public void addEmployee(Employee employee) {  
 employeeRepository.save(employee);  
 }  
}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.model.Employee;  
import com.cognizant.orm\_learn.service.EmployeeService;  
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 EmployeeService *employeeService*;  
  
 public static void main(String[] args) {  
 ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.class, args);  
 *employeeService* = context.getBean(EmployeeService.class);  
  
 *testAddEmployee*();  
 }  
  
 private static void testAddEmployee() {  
 *LOGGER*.info("Start");  
  
 Employee emp = new Employee();  
 emp.setName("abc");  
 emp.setSalary(60000);  
  
 *employeeService*.addEmployee(emp);  
  
 *LOGGER*.info("End");  
 }  
}

**Output:**

****



**3. Implement services for managing Country**

insert into country (code, name) values ("AF", "Afghanistan");

insert into country (code, name) values ("AL", "Albania");

insert into country (code, name) values ("DZ", "Algeria");

insert into country (code, name) values ("AS", "American Samoa");

insert into country (code, name) values ("AD", "Andorra");

insert into country (code, name) values ("AO", "Angola");

insert into country (code, name) values ("AI", "Anguilla");

insert into country (code, name) values ("AQ", "Antarctica");

insert into country (code, name) values ("AG", "Antigua and Barbuda");

insert into country (code, name) values ("AR", "Argentina");

insert into country (code, name) values ("AM", "Armenia");

insert into country (code, name) values ("AW", "Aruba");

insert into country (code, name) values ("AU", "Australia");

insert into country (code, name) values ("AT", "Austria");

insert into country (code, name) values ("AZ", "Azerbaijan");

insert into country (code, name) values ("BS", "Bahamas");

insert into country (code, name) values ("BH", "Bahrain");

insert into country (code, name) values ("BD", "Bangladesh");

insert into country (code, name) values ("BB", "Barbados");

insert into country (code, name) values ("BY", "Belarus");

insert into country (code, name) values ("BE", "Belgium");

insert into country (code, name) values ("BZ", "Belize");

insert into country (code, name) values ("BJ", "Benin");

insert into country (code, name) values ("BM", "Bermuda");

insert into country (code, name) values ("BT", "Bhutan");

insert into country (code, name) values ("BO", "Bolivia, Plurinational State of");

insert into country (code, name) values ("BQ", "Bonaire, Sint Eustatius and Saba");

insert into country (code, name) values ("BA", "Bosnia and Herzegovina");

insert into country (code, name) values ("BW", "Botswana");

insert into country (code, name) values ("BV", "Bouvet Island");

insert into country (code, name) values ("BR", "Brazil");

insert into country (code, name) values ("IO", "British Indian Ocean Territory");

insert into country (code, name) values ("BN", "Brunei Darussalam");

insert into country (code, name) values ("BG", "Bulgaria");

insert into country (code, name) values ("BF", "Burkina Faso");

insert into country (code, name) values ("BI", "Burundi");

insert into country (code, name) values ("KH", "Cambodia");

insert into country (code, name) values ("CM", "Cameroon");

insert into country (code, name) values ("CA", "Canada");

insert into country (code, name) values ("CV", "Cape Verde");

insert into country (code, name) values ("KY", "Cayman Islands");

insert into country (code, name) values ("CF", "Central African Republic");

insert into country (code, name) values ("TD", "Chad");

insert into country (code, name) values ("CL", "Chile");

insert into country (code, name) values ("CN", "China");

insert into country (code, name) values ("CX", "Christmas Island");

insert into country (code, name) values ("CC", "Cocos (Keeling) Islands");

insert into country (code, name) values ("CO", "Colombia");

insert into country (code, name) values ("KM", "Comoros");

insert into country (code, name) values ("CG", "Congo");

insert into country (code, name) values ("CD", "Congo, the Democratic Republic of the");

insert into country (code, name) values ("CK", "Cook Islands");

insert into country (code, name) values ("CR", "Costa Rica");

insert into country (code, name) values ("HR", "Croatia");

insert into country (code, name) values ("CU", "Cuba");

insert into country (code, name) values ("CW", "Curaçao");

insert into country (code, name) values ("CY", "Cyprus");

insert into country (code, name) values ("CZ", "Czech Republic");

insert into country (code, name) values ("CI", "Côte d'Ivoire");

insert into country (code, name) values ("DK", "Denmark");

insert into country (code, name) values ("DJ", "Djibouti");

insert into country (code, name) values ("DM", "Dominica");

insert into country (code, name) values ("DO", "Dominican Republic");

insert into country (code, name) values ("EC", "Ecuador");

insert into country (code, name) values ("EG", "Egypt");

insert into country (code, name) values ("SV", "El Salvador");

insert into country (code, name) values ("GQ", "Equatorial Guinea");

insert into country (code, name) values ("ER", "Eritrea");

insert into country (code, name) values ("EE", "Estonia");

insert into country (code, name) values ("ET", "Ethiopia");

insert into country (code, name) values ("FK", "Falkland Islands (Malvinas)");

insert into country (code, name) values ("FO", "Faroe Islands");

insert into country (code, name) values ("FJ", "Fiji");

insert into country (code, name) values ("FI", "Finland");

insert into country (code, name) values ("FR", "France");

insert into country (code, name) values ("GF", "French Guiana");

insert into country (code, name) values ("PF", "French Polynesia");

insert into country (code, name) values ("TF", "French Southern Territories");

insert into country (code, name) values ("GA", "Gabon");

insert into country (code, name) values ("GM", "Gambia");

insert into country (code, name) values ("GE", "Georgia");

insert into country (code, name) values ("DE", "Germany");

insert into country (code, name) values ("GH", "Ghana");

insert into country (code, name) values ("GI", "Gibraltar");

insert into country (code, name) values ("GR", "Greece");

insert into country (code, name) values ("GL", "Greenland");

insert into country (code, name) values ("GD", "Grenada");

insert into country (code, name) values ("GP", "Guadeloupe");

insert into country (code, name) values ("GU", "Guam");

insert into country (code, name) values ("GT", "Guatemala");

insert into country (code, name) values ("GG", "Guernsey");

insert into country (code, name) values ("GN", "Guinea");

insert into country (code, name) values ("GW", "Guinea-Bissau");

insert into country (code, name) values ("GY", "Guyana");

insert into country (code, name) values ("HT", "Haiti");

insert into country (code, name) values ("HM", "Heard Island and McDonald Islands");

insert into country (code, name) values ("VA", "Holy See (Vatican City State)");

insert into country (code, name) values ("HN", "Honduras");

insert into country (code, name) values ("HK", "Hong Kong");

insert into country (code, name) values ("HU", "Hungary");

insert into country (code, name) values ("IS", "Iceland");

insert into country (code, name) values ("IN", "India");

insert into country (code, name) values ("ID", "Indonesia");

insert into country (code, name) values ("IR", "Iran, Islamic Republic of");

insert into country (code, name) values ("IQ", "Iraq");

insert into country (code, name) values ("IE", "Ireland");

insert into country (code, name) values ("IM", "Isle of Man");

insert into country (code, name) values ("IL", "Israel");

insert into country (code, name) values ("IT", "Italy");

insert into country (code, name) values ("JM", "Jamaica");

insert into country (code, name) values ("JP", "Japan");

insert into country (code, name) values ("JE", "Jersey");

insert into country (code, name) values ("JO", "Jordan");

insert into country (code, name) values ("KZ", "Kazakhstan");

insert into country (code, name) values ("KE", "Kenya");

insert into country (code, name) values ("KI", "Kiribati");

insert into country (code, name) values ("KP", "Democratic People's Republic of Korea");

insert into country (code, name) values ("KR", "Republic of Korea");

insert into country (code, name) values ("KW", "Kuwait");

insert into country (code, name) values ("KG", "Kyrgyzstan");

insert into country (code, name) values ("LA", "Lao People's Democratic Republic");

insert into country (code, name) values ("LV", "Latvia");

insert into country (code, name) values ("LB", "Lebanon");

insert into country (code, name) values ("LS", "Lesotho");

insert into country (code, name) values ("LR", "Liberia");

insert into country (code, name) values ("LY", "Libya");

insert into country (code, name) values ("LI", "Liechtenstein");

insert into country (code, name) values ("LT", "Lithuania");

insert into country (code, name) values ("LU", "Luxembourg");

insert into country (code, name) values ("MO", "Macao");

insert into country (code, name) values ("MK", "Macedonia, the Former Yugoslav Republic of");

insert into country (code, name) values ("MG", "Madagascar");

insert into country (code, name) values ("MW", "Malawi");

insert into country (code, name) values ("MY", "Malaysia");

insert into country (code, name) values ("MV", "Maldives");

insert into country (code, name) values ("ML", "Mali");

insert into country (code, name) values ("MT", "Malta");

insert into country (code, name) values ("MH", "Marshall Islands");

insert into country (code, name) values ("MQ", "Martinique");

insert into country (code, name) values ("MR", "Mauritania");

insert into country (code, name) values ("MU", "Mauritius");

insert into country (code, name) values ("YT", "Mayotte");

insert into country (code, name) values ("MX", "Mexico");

insert into country (code, name) values ("FM", "Micronesia, Federated States of");

insert into country (code, name) values ("MD", "Moldova, Republic of");

insert into country (code, name) values ("MC", "Monaco");

insert into country (code, name) values ("MN", "Mongolia");

insert into country (code, name) values ("ME", "Montenegro");

insert into country (code, name) values ("MS", "Montserrat");

insert into country (code, name) values ("MA", "Morocco");

insert into country (code, name) values ("MZ", "Mozambique");

insert into country (code, name) values ("MM", "Myanmar");

insert into country (code, name) values ("NA", "Namibia");

insert into country (code, name) values ("NR", "Nauru");

insert into country (code, name) values ("NP", "Nepal");

insert into country (code, name) values ("NL", "Netherlands");

insert into country (code, name) values ("NC", "New Caledonia");

insert into country (code, name) values ("NZ", "New Zealand");

insert into country (code, name) values ("NI", "Nicaragua");

insert into country (code, name) values ("NE", "Niger");

insert into country (code, name) values ("NG", "Nigeria");

insert into country (code, name) values ("NU", "Niue");

insert into country (code, name) values ("NF", "Norfolk Island");

insert into country (code, name) values ("MP", "Northern Mariana Islands");

insert into country (code, name) values ("NO", "Norway");

insert into country (code, name) values ("OM", "Oman");

insert into country (code, name) values ("PK", "Pakistan");

insert into country (code, name) values ("PW", "Palau");

insert into country (code, name) values ("PS", "Palestine, State of");

insert into country (code, name) values ("PA", "Panama");

insert into country (code, name) values ("PG", "Papua New Guinea");

insert into country (code, name) values ("PY", "Paraguay");

insert into country (code, name) values ("PE", "Peru");

insert into country (code, name) values ("PH", "Philippines");

insert into country (code, name) values ("PN", "Pitcairn");

insert into country (code, name) values ("PL", "Poland");

insert into country (code, name) values ("PT", "Portugal");

insert into country (code, name) values ("PR", "Puerto Rico");

insert into country (code, name) values ("QA", "Qatar");

insert into country (code, name) values ("RO", "Romania");

insert into country (code, name) values ("RU", "Russian Federation");

insert into country (code, name) values ("RW", "Rwanda");

insert into country (code, name) values ("RE", "Réunion");

insert into country (code, name) values ("BL", "Saint Barthélemy");

insert into country (code, name) values ("SH", "Saint Helena, Ascension and Tristan da Cunha");

insert into country (code, name) values ("KN", "Saint Kitts and Nevis");

insert into country (code, name) values ("LC", "Saint Lucia");

insert into country (code, name) values ("MF", "Saint Martin (French part)");

insert into country (code, name) values ("PM", "Saint Pierre and Miquelon");

insert into country (code, name) values ("VC", "Saint Vincent and the Grenadines");

insert into country (code, name) values ("WS", "Samoa");

insert into country (code, name) values ("SM", "San Marino");

insert into country (code, name) values ("ST", "Sao Tome and Principe");

insert into country (code, name) values ("SA", "Saudi Arabia");

insert into country (code, name) values ("SN", "Senegal");

insert into country (code, name) values ("RS", "Serbia");

insert into country (code, name) values ("SC", "Seychelles");

insert into country (code, name) values ("SL", "Sierra Leone");

insert into country (code, name) values ("SG", "Singapore");

insert into country (code, name) values ("SX", "Sint Maarten (Dutch part)");

insert into country (code, name) values ("SK", "Slovakia");

insert into country (code, name) values ("SI", "Slovenia");

insert into country (code, name) values ("SB", "Solomon Islands");

insert into country (code, name) values ("SO", "Somalia");

insert into country (code, name) values ("ZA", "South Africa");

insert into country (code, name) values ("GS", "South Georgia and the South Sandwich Islands");

insert into country (code, name) values ("SS", "South Sudan");

insert into country (code, name) values ("ES", "Spain");

insert into country (code, name) values ("LK", "Sri Lanka");

insert into country (code, name) values ("SD", "Sudan");

insert into country (code, name) values ("SR", "Suriname");

insert into country (code, name) values ("SJ", "Svalbard and Jan Mayen");

insert into country (code, name) values ("SZ", "Swaziland");

insert into country (code, name) values ("SE", "Sweden");

insert into country (code, name) values ("CH", "Switzerland");

insert into country (code, name) values ("SY", "Syrian Arab Republic");

insert into country (code, name) values ("TW", "Taiwan, Province of China");

insert into country (code, name) values ("TJ", "Tajikistan");

insert into country (code, name) values ("TZ", "Tanzania, United Republic of");

insert into country (code, name) values ("TH", "Thailand");

insert into country (code, name) values ("TL", "Timor-Leste");

insert into country (code, name) values ("TG", "Togo");

insert into country (code, name) values ("TK", "Tokelau");

insert into country (code, name) values ("TO", "Tonga");

insert into country (code, name) values ("TT", "Trinidad and Tobago");

insert into country (code, name) values ("TN", "Tunisia");

insert into country (code, name) values ("TR", "Turkey");

insert into country (code, name) values ("TM", "Turkmenistan");

insert into country (code, name) values ("TC", "Turks and Caicos Islands");

insert into country (code, name) values ("TV", "Tuvalu");

insert into country (code, name) values ("UG", "Uganda");

insert into country (code, name) values ("UA", "Ukraine");

insert into country (code, name) values ("AE", "United Arab Emirates");

insert into country (code, name) values ("GB", "United Kingdom");

insert into country (code, name) values ("US", "United States");

insert into country (code, name) values ("UM", "United States Minor Outlying Islands");

insert into country (code, name) values ("UY", "Uruguay");

insert into country (code, name) values ("UZ", "Uzbekistan");

insert into country (code, name) values ("VU", "Vanuatu");

insert into country (code, name) values ("VE", "Venezuela, Bolivarian Republic of");

insert into country (code, name) values ("VN", "Viet Nam");

insert into country (code, name) values ("VG", "Virgin Islands, British");

insert into country (code, name) values ("VI", "Virgin Islands, U.S.");

insert into country (code, name) values ("WF", "Wallis and Futuna");

insert into country (code, name) values ("EH", "Western Sahara");

insert into country (code, name) values ("YE", "Yemen");

insert into country (code, name) values ("ZM", "Zambia");

insert into country (code, name) values ("ZW", "Zimbabwe");

insert into country (code, name) values ("AX", "Åland Islands");

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.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);  
  
 *testFindCountryByCode*();  
 *testAddCountry*();  
 *testUpdateCountry*();  
 *testDeleteCountry*();  
 *testFindCountriesByPartialName*();  
 }  
  
 private static void testFindCountryByCode() {  
 *LOGGER*.info("Start: testFindCountryByCode");  
 Country country = *countryService*.findCountryByCode("IN");  
 *LOGGER*.debug("Country: {}", country);  
 *LOGGER*.info("End");  
 }  
  
 private static void testAddCountry() {  
 *LOGGER*.info("Start: testAddCountry");  
 Country country = new Country();  
 country.setCode("ZZ");  
 country.setName("Zootopia");  
 *countryService*.addCountry(country);  
 *LOGGER*.debug("Added: {}", *countryService*.findCountryByCode("ZZ"));  
 *LOGGER*.info("End");  
 }  
  
 private static void testUpdateCountry() {  
 *LOGGER*.info("Start: testUpdateCountry");  
 *countryService*.updateCountry("ZZ", "Zootopia Updated");  
 *LOGGER*.debug("Updated: {}", *countryService*.findCountryByCode("ZZ"));  
 *LOGGER*.info("End");  
 }  
  
 private static void testDeleteCountry() {  
 *LOGGER*.info("Start: testDeleteCountry");  
 *countryService*.deleteCountry("ZZ");  
 *LOGGER*.debug("Deleted: {}", *countryService*.findCountryByCode("ZZ"));  
 *LOGGER*.info("End");  
 }  
  
 private static void testFindCountriesByPartialName() {  
 *LOGGER*.info("Start: testFindCountriesByPartialName");  
 List<Country> countries = *countryService*.findCountriesByPartialName("an");  
 countries.forEach(c -> *LOGGER*.debug("Country: {}", c));  
 *LOGGER*.info("End");  
 }  
}

**Country.java**

package com.cognizant.orm\_learn.model;  
  
import jakarta.persistence.Column;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
  
@Entity  
@Table(name = "country")  
public class Country {  
  
 @Id  
 @Column(name = "code")  
 private String code;  
  
 @Column(name = "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**

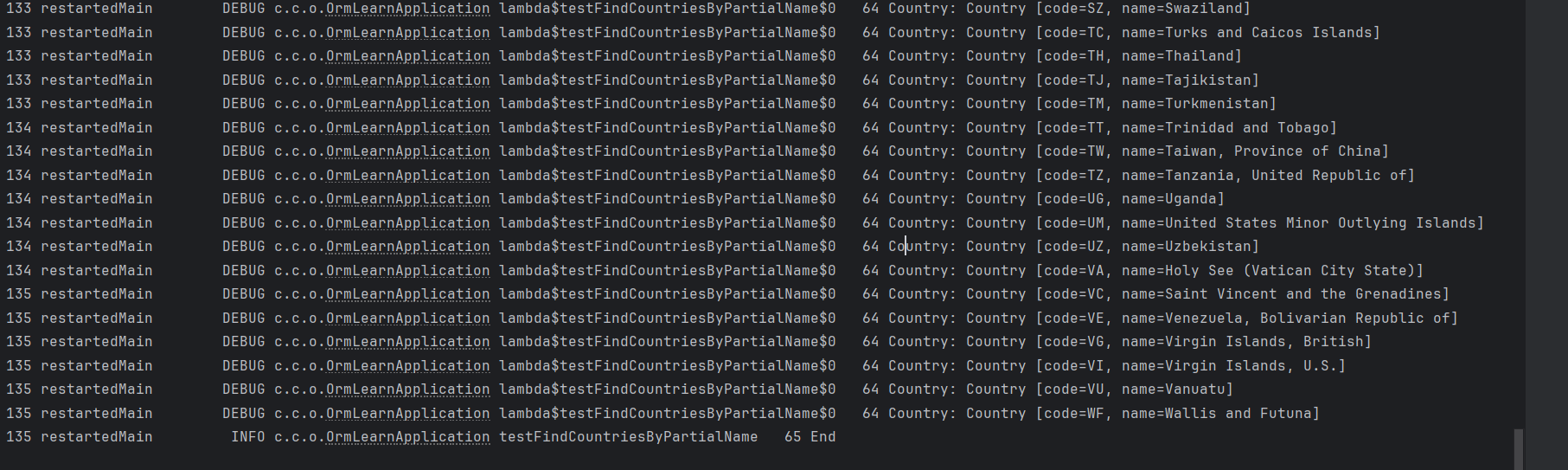
package com.cognizant.orm\_learn.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
import com.cognizant.orm\_learn.model.Country;  
import java.util.\*;  
  
@Repository  
public interface CountryRepository extends JpaRepository<Country, String> {  
 List<Country> findByNameContainingIgnoreCase(String name);  
}

**CountryService.java**

package com.cognizant.orm\_learn.service;  
  
import java.util.List;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.repository.CountryRepository;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional  
 public Country findCountryByCode(String code) {  
 return countryRepository.findById(code).orElse(null);  
 }  
  
 @Transactional  
 public void addCountry(Country country) {  
 countryRepository.save(country);  
 }  
  
 @Transactional  
 public void updateCountry(String code, String name) {  
 Country c = findCountryByCode(code);  
 if (c != null) {  
 c.setName(name);  
 countryRepository.save(c);  
 }  
 }  
  
 @Transactional  
 public void deleteCountry(String code) {  
 countryRepository.deleteById(code);  
 }  
  
 @Transactional  
 public List<Country> findCountriesByPartialName(String name) {  
 return countryRepository.findByNameContainingIgnoreCase(name);  
 }}

**Output:**

****



**4. Find a country based on country code**

**5. Add a new country**

**6. Update a country based on code**

**7. Delete a country based on code**

**CountryService.java**

package com.cognizant.orm\_learn.service;  
  
import java.util.List;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.repository.CountryRepository;  
import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;  
import java.util.Optional;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional  
 public Country findCountryByCode(String code) throws CountryNotFoundException {  
 Optional<Country> result = countryRepository.findById(code);  
 if (!result.isPresent()) {  
 throw new CountryNotFoundException("Country not found for code: " + code);  
 }  
 return result.get();  
 }  
  
 @Transactional  
 public void addCountry(Country country) {  
 countryRepository.save(country);  
 }  
  
 @Transactional  
 public void updateCountry(String code, String name) throws CountryNotFoundException {  
 Country country = findCountryByCode(code);  
 country.setName(name);  
 countryRepository.save(country);  
 }  
  
 @Transactional  
 public void deleteCountry(String code) throws CountryNotFoundException {  
 Country country = findCountryByCode(code);  
 countryRepository.delete(country);  
 }  
  
 @Transactional  
 public List<Country> findCountriesByPartialName(String name) {  
 return countryRepository.findByNameContainingIgnoreCase(name);  
 }  
}

**CountryNotFoundException.java**

package com.cognizant.orm\_learn.service.exception;  
  
public class CountryNotFoundException extends Exception {  
 public CountryNotFoundException(String message) {  
 super(message);  
 }  
}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.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 com.cognizant.orm\_learn.service.exception.CountryNotFoundException;  
  
  
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) throws CountryNotFoundException {  
 ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.class, args);  
 *countryService* = context.getBean(CountryService.class);  
  
 *testFindCountryByCode*();  
 *testAddCountry*();  
 *testUpdateCountry*();  
 *testDeleteCountry*();  
 *testFindCountriesByPartialName*();  
 }  
  
 private static void testFindCountryByCode() throws CountryNotFoundException {  
 *LOGGER*.info("Start");  
 Country country = *countryService*.findCountryByCode("IN");  
 *LOGGER*.debug("Country: {}", country);  
 *LOGGER*.info("End");  
 }  
  
 private static void testAddCountry() throws CountryNotFoundException {  
 *LOGGER*.info("Start");  
 Country country = new Country();  
 country.setCode("ZZ");  
 country.setName("Zootopia");  
 *countryService*.addCountry(country);  
 *LOGGER*.debug("Added: {}", *countryService*.findCountryByCode("ZZ"));  
 *LOGGER*.info("End");  
 }  
  
 private static void testUpdateCountry() throws CountryNotFoundException {  
 *LOGGER*.info("Start");  
 *countryService*.updateCountry("ZZ", "Zootopia Updated");  
 *LOGGER*.debug("Updated: {}", *countryService*.findCountryByCode("ZZ"));  
 *LOGGER*.info("End");  
 }  
  
 private static void testDeleteCountry() {  
 *LOGGER*.info("Start");  
 try {  
 *countryService*.deleteCountry("ZZ");  
 *LOGGER*.debug("Deleted country with code ZZ");  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Error deleting country: {}", e.getMessage());  
 }  
 *LOGGER*.info("End");  
 }

private static void testFindCountriesByPartialName() {  
 *LOGGER*.info("Start");  
 List<Country> countries = *countryService*.findCountriesByPartialName("an");  
 countries.forEach(c -> *LOGGER*.debug("Country: {}", c));  
 *LOGGER*.info("End");  
 }  
}

**Output:**

****

**8. Write queries on country table using Query Methods**

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.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 com.cognizant.orm\_learn.service.exception.CountryNotFoundException;  
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) throws CountryNotFoundException {  
 ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.class, args);  
 *countryService* = context.getBean(CountryService.class);  
  
 *testFindCountriesByPartialMatch*();  
 *testFindCountriesByPartialMatchSorted*();  
 *testFindCountriesStartingWith*();  
 }  
 private static void testFindCountriesByPartialMatch() {  
 *LOGGER*.info("Start");  
 List<Country> countries = *countryService*.findCountriesByPartialName("ou");  
 countries.forEach(c -> *LOGGER*.debug("Country: {}", c));  
 *LOGGER*.info("End");  
 }  
 private static void testFindCountriesByPartialMatchSorted() {  
 *LOGGER*.info("Start");  
 List<Country> countries = *countryService*.findCountriesByPartialNameSorted("ou");  
 countries.forEach(c -> *LOGGER*.debug("Country: {}", c));  
 *LOGGER*.info("End");  
 }  
 private static void testFindCountriesStartingWith() {  
 *LOGGER*.info("Start");  
 List<Country> countries = *countryService*.findCountriesStartingWith("Z");  
 countries.forEach(c -> *LOGGER*.debug("Country: {}", c));  
 *LOGGER*.info("End");  
 }  
}

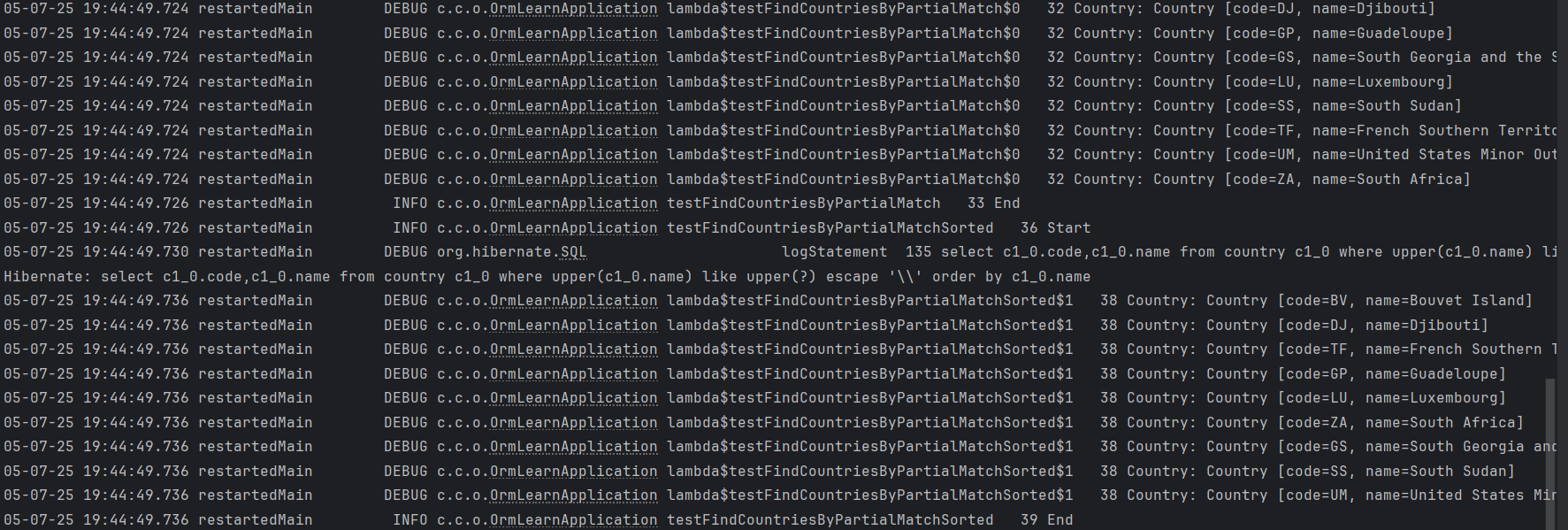
**CountryService.java**

package com.cognizant.orm\_learn.service;  
  
import java.util.List;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.repository.CountryRepository;  
import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;  
import java.util.Optional;  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional  
 public List<Country> findCountriesByPartialName(String name) {  
 return countryRepository.findByNameContainingIgnoreCase(name);  
 }  
  
 @Transactional  
 public List<Country> findCountriesByPartialNameSorted(String name) {  
 return countryRepository.findByNameContainingIgnoreCaseOrderByNameAsc(name);  
 }  
  
 @Transactional  
 public List<Country> findCountriesStartingWith(String letter) {  
 return countryRepository.findByNameStartingWithIgnoreCase(letter);  
 }  
}

**CountryRepository.java**

package com.cognizant.orm\_learn.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
import com.cognizant.orm\_learn.model.Country;  
import java.util.\*;  
  
@Repository  
public interface CountryRepository extends JpaRepository<Country, String> {  
 List<Country> findByNameContainingIgnoreCase(String substring);  
 List<Country> findByNameContainingIgnoreCaseOrderByNameAsc(String substring);  
 List<Country> findByNameStartingWithIgnoreCase(String letter);  
}

**Output:**



**9. Write queries on stock table using Query Methods**

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.model.Employee;  
import com.cognizant.orm\_learn.repository.EmployeeRepository;  
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.ApplicationContext;  
import org.springframework.transaction.annotation.EnableTransactionManagement;  
import org.springframework.transaction.annotation.Transactional;  
  
  
import java.util.Optional;  
@SpringBootApplication  
@EnableTransactionManagement  
public class OrmLearnApplication {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(OrmLearnApplication.class);  
  
 private static EmployeeRepository *employeeRepository*;  
  
 public static void main(String[] args) {  
 ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.class, args);  
 *employeeRepository* = context.getBean(EmployeeRepository.class);  
  
 *testGetEmployee*();   
 }  
  
 @Transactional  
 private static void testGetEmployee() {  
 *LOGGER*.info("Start");  
  
 Optional<Employee> optional = *employeeRepository*.findById(1);  
 if (optional.isPresent()) {  
 Employee employee = optional.get();  
 *LOGGER*.debug("Employee: {}", employee.getName());   
 *LOGGER*.debug("Department: {}", employee.getDepartment().getName());  
  
 employee.getSkillList().forEach(skill ->  
 *LOGGER*.debug("Skill: {}", skill.getName())  
 );  
 }  
  
 *LOGGER*.info("End");  
 }  
  
}

**Employee.java**   
package com.cognizant.orm\_learn.model;  
  
import jakarta.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(fetch = FetchType.*EAGER*)  
 @JoinTable(  
 name = "employee\_skill",  
 joinColumns = @JoinColumn(name = "es\_em\_id"),  
 inverseJoinColumns = @JoinColumn(name = "es\_sk\_id")  
 )  
 private List<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 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 +  
 ", department=" + department +  
 ", skillList=" + skillList +  
 '}';  
 }  
}

**Department.java**

package com.cognizant.orm\_learn.model;  
  
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;  
  
 // 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;  
 }  
  
 @Override  
 public String toString() {  
 return "Department{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 '}';  
 }  
}

**Skill.java**

package com.cognizant.orm\_learn.model;  
  
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;  
  
 // 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;  
 }  
  
 @Override  
 public String toString() {  
 return "Skill{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 '}';  
 }  
}

**EmployeeRepository.java**

package com.cognizant.orm\_learn.repository;  
  
import com.cognizant.orm\_learn.model.Employee;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

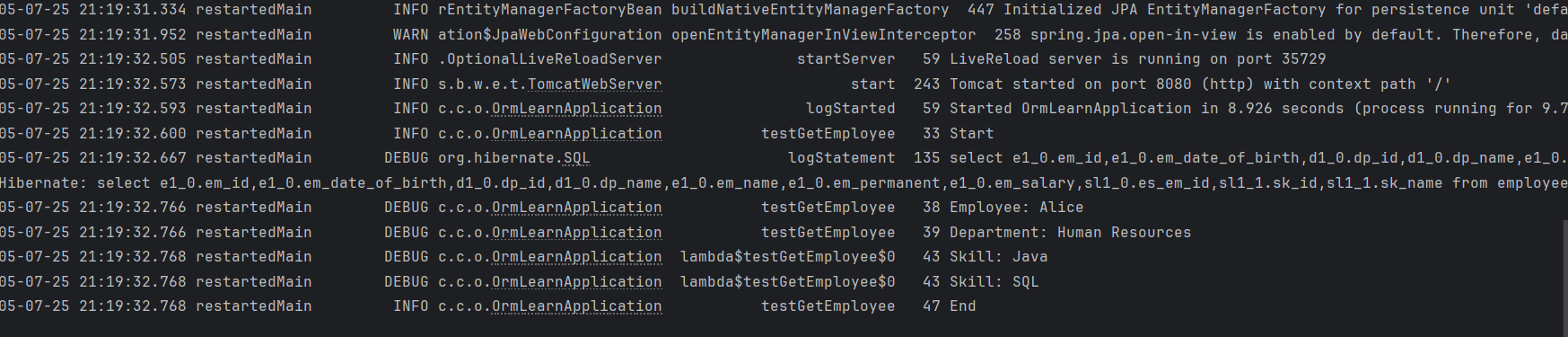
**DepartmentRepository.java**

package com.cognizant.orm\_learn.repository;  
  
import com.cognizant.orm\_learn.model.Department;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface DepartmentRepository extends JpaRepository<Department, Integer> {}

**SkillRepository.java**

package com.cognizant.orm\_learn.repository;  
  
import com.cognizant.orm\_learn.model.Skill;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface SkillRepository extends JpaRepository<Skill, Integer> {}

**Output:**

****

**10. Get all permanent employees using HQL   
EmployeeRepository.java**

package com.cognizant.orm\_learn.repository;  
  
import com.cognizant.orm\_learn.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> {  
  
 @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**

package com.cognizant.orm\_learn.service;  
  
import java.util.List;  
import com.cognizant.orm\_learn.model.Employee;  
  
public interface EmployeeService {  
 List<Employee> getAllPermanentEmployees();  
}

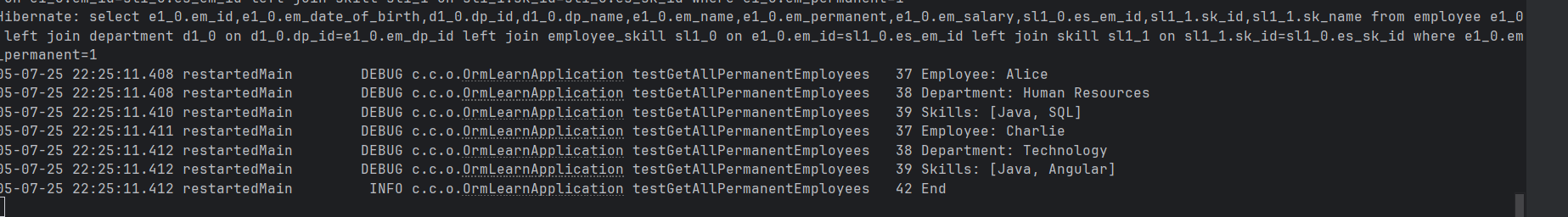
**EmployeeServiceImpl.java**

package com.cognizant.orm\_learn.service;  
  
import java.util.List;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import com.cognizant.orm\_learn.model.Employee;  
import com.cognizant.orm\_learn.repository.EmployeeRepository;  
  
@Service  
public class EmployeeServiceImpl implements EmployeeService {  
  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 @Override  
 public List<Employee> getAllPermanentEmployees() {  
 return employeeRepository.getAllPermanentEmployees();  
 }  
}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
import java.util.List;  
  
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.transaction.annotation.EnableTransactionManagement;  
import org.springframework.transaction.annotation.Transactional;  
  
import com.cognizant.orm\_learn.model.Employee;  
import com.cognizant.orm\_learn.service.EmployeeService;  
  
@SpringBootApplication  
@EnableTransactionManagement  
public class OrmLearnApplication {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(OrmLearnApplication.class);  
  
 @Autowired  
 private EmployeeService employeeService;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(OrmLearnApplication.class, args).getBean(OrmLearnApplication.class).testGetAllPermanentEmployees();  
 }  
  
 @Transactional  
 public void testGetAllPermanentEmployees() {  
 *LOGGER*.info("Start");  
  
 List<Employee> employees = employeeService.getAllPermanentEmployees();  
 for (Employee e : employees) {  
 *LOGGER*.debug("Employee: {}", e.getName());  
 *LOGGER*.debug("Department: {}", e.getDepartment().getName());  
 *LOGGER*.debug("Skills: {}", e.getSkillList().stream().map(s -> s.getName()).toList());  
 }  
  
 *LOGGER*.info("End");  
 }  
}

**Output:**



**11. Fetch quiz attempt details using HQL**

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.model.\*;  
import com.cognizant.orm\_learn.service.AttemptService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import java.util.List;  
  
@SpringBootApplication  
public class OrmLearnApplication implements CommandLineRunner {  
  
 @Autowired  
 private AttemptService attemptService;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(OrmLearnApplication.class, args);  
 }  
  
 @Override  
 public void run(String... args) {  
 System.*out*.println("Fetching all attempts:");  
 List<Attempt> allAttempts = attemptService.getAllAttempts();  
 for (Attempt a : allAttempts) {  
 System.*out*.println("Attempt ID: " + a.getId() + ", User: " + a.getUser().getUsername());  
 }  
  
 System.*out*.println("\nFetching specific attempt with userId=1, attemptId=1:");  
 Attempt attempt = attemptService.getAttemptDetails(1, 1);  
  
 if (attempt != null) {  
 System.*out*.println("Username: " + attempt.getUser().getUsername());  
 System.*out*.println("Attempt Date: " + attempt.getAttemptDate());  
  
 for (AttemptQuestion aq : attempt.getAttemptQuestions()) {  
 Question question = aq.getQuestion();  
 System.*out*.println("\n" + question.getQuestionText());  
  
 for (AttemptOption ao : aq.getAttemptOptions()) {  
 Option option = ao.getOption();  
 String isCorrect = option.isCorrect() ? "1.0" : "0.0";  
 System.*out*.println(option.getOptionText() + "\t" + isCorrect + "\t" + ao.isSelected());  
 }  
 }  
 } else {  
 System.*out*.println("No attempt found for userId=1 and attemptId=1.");  
 }  
 }  
}

**User.java**

package com.cognizant.orm\_learn.model;  
  
import jakarta.persistence.\*;  
import java.util.List;  
  
@Entity  
@Table(name = "user")  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "us\_id")  
 private int id;  
  
 @Column(name = "us\_username")  
 private String username;  
  
 @OneToMany(mappedBy = "user")  
 private List<Attempt> attempts;  
  
 public int getId() { return id; }  
 public void setId(int id) { this.id = id; }  
 public String getUsername() { return username; }  
 public void setUsername(String username) { this.username = username; }  
 public List<Attempt> getAttempts() { return attempts; }  
 public void setAttempts(List<Attempt> attempts) { this.attempts = attempts; }  
}

**Attempt.java**

package com.cognizant.orm\_learn.model;  
  
import jakarta.persistence.\*;  
import java.util.Date;  
import java.util.List;  
  
@Entity  
@Table(name = "attempt")  
public class Attempt {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "at\_id")  
 private int id;  
  
 @Column(name = "at\_attempt\_date")  
 private Date attemptDate;  
  
 @ManyToOne  
 @JoinColumn(name = "at\_us\_id")  
 private User user;  
  
 @OneToMany(mappedBy = "attempt", cascade = CascadeType.*ALL*, fetch = FetchType.*LAZY*)  
 private List<AttemptQuestion> attemptQuestions;  
  
 public int getId() { return id; }  
 public void setId(int id) { this.id = id; }  
 public Date getAttemptDate() { return attemptDate; }  
 public void setAttemptDate(Date attemptDate) { this.attemptDate = attemptDate; }  
 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; }  
}

**Question.java**

package com.cognizant.orm\_learn.model;  
  
import jakarta.persistence.\*;  
import java.util.List;  
  
@Entity  
@Table(name = "question")  
public class Question {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "qu\_id")  
 private int id;  
  
 @Column(name = "qu\_text")  
 private String questionText;  
  
 @Column(name = "qu\_score")  
 private double score;  
  
 @OneToMany(mappedBy = "question")  
 private List<Option> options;  
  
 public int getId() { return id; }  
 public void setId(int id) { this.id = id; }  
 public String getQuestionText() { return questionText; }  
 public void setQuestionText(String questionText) { this.questionText = questionText; }  
 public double getScore() { return score; }  
 public void setScore(double score) { this.score = score; }  
 public List<Option> getOptions() { return options; }  
 public void setOptions(List<Option> options) { this.options = options; }  
}

**Option.java**

package com.cognizant.orm\_learn.model;  
  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "option")  
public class Option {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "op\_id")  
 private int id;  
  
 @Column(name = "op\_text")  
 private String optionText;  
  
 @Column(name = "op\_is\_correct")  
 private boolean correct;  
  
 @ManyToOne  
 @JoinColumn(name = "op\_qu\_id")  
 private Question question;  
  
 // Getters and Setters  
 public int getId() { return id; }  
 public void setId(int id) { this.id = id; }  
  
 public String getOptionText() { return optionText; }  
 public void setOptionText(String optionText) { this.optionText = optionText; }  
  
 public boolean isCorrect() { return correct; }  
 public void setCorrect(boolean correct) { this.correct = correct; }  
  
 public Question getQuestion() { return question; }  
 public void setQuestion(Question question) { this.question = question; }  
  
 @Override  
 public String toString() {  
 return "Option{" +  
 "id=" + id +  
 ", optionText='" + optionText + '\'' +  
 ", correct=" + correct +  
 '}';  
 }  
}

**AttemptQuestion.java**

package com.cognizant.orm\_learn.model;  
  
import jakarta.persistence.\*;  
import java.util.List;  
  
@Entity  
@Table(name = "attempt\_question")  
public class AttemptQuestion {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "aq\_id")  
 private int id;  
  
 @ManyToOne  
 @JoinColumn(name = "aq\_qu\_id")  
 private Question question;  
  
 @ManyToOne  
 @JoinColumn(name = "aq\_at\_id")  
 private Attempt attempt;  
  
 @OneToMany(mappedBy = "attemptQuestion")  
 private List<AttemptOption> attemptOptions;  
  
 public int getId() { return id; }  
 public void setId(int id) { this.id = id; }  
 public Question getQuestion() { return question; }  
 public void setQuestion(Question question) { this.question = question; }  
 public Attempt getAttempt() { return attempt; }  
 public void setAttempt(Attempt attempt) { this.attempt = attempt; }  
 public List<AttemptOption> getAttemptOptions() { return attemptOptions; }  
 public void setAttemptOptions(List<AttemptOption> attemptOptions) { this.attemptOptions = attemptOptions; }  
}

**AttemptOption.java**

package com.cognizant.orm\_learn.model;  
  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "attempt\_option")  
public class AttemptOption {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "ao\_id")  
 private int id;  
  
 @ManyToOne  
 @JoinColumn(name = "ao\_op\_id")  
 private Option option;  
  
 @ManyToOne  
 @JoinColumn(name = "ao\_aq\_id")  
 private AttemptQuestion attemptQuestion;  
  
 @Column(name = "ao\_selected")  
 private boolean selected;   
  
 // Getters and Setters  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public Option getOption() {  
 return option;  
 }  
  
 public void setOption(Option option) {  
 this.option = option;  
 }  
  
 public AttemptQuestion getAttemptQuestion() {  
 return attemptQuestion;  
 }  
  
 public void setAttemptQuestion(AttemptQuestion attemptQuestion) {  
 this.attemptQuestion = attemptQuestion;  
 }  
  
 public boolean isSelected() {  
 return selected;  
 }  
  
 public void setSelected(boolean selected) {  
 this.selected = selected;  
 }  
  
 @Override  
 public String toString() {  
 return "AttemptOption{" +  
 "id=" + id +  
 ", option=" + option +  
 ", attemptQuestion=" + attemptQuestion +  
 ", selected=" + selected +  
 '}';  
 }  
}

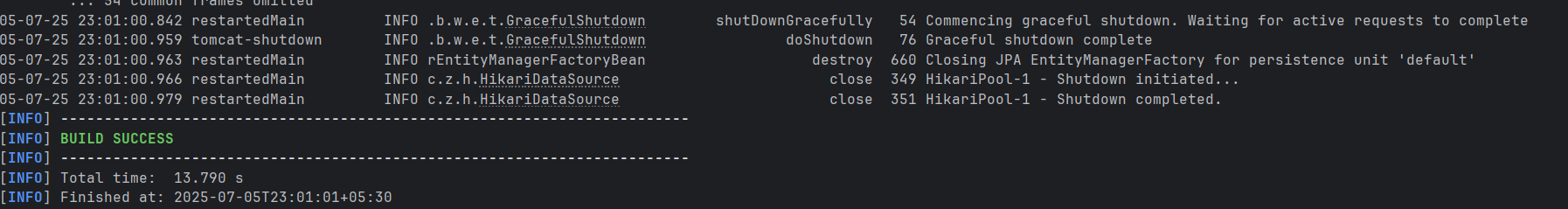
**AttemptRepository.java**

package com.cognizant.orm\_learn.repository;  
  
import com.cognizant.orm\_learn.model.Attempt;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
  
import java.util.List;  
import java.util.Optional;  
  
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 aq.attemptOptions ao " +  
 "JOIN FETCH ao.option o " +  
 "WHERE u.id = :userId AND a.id = :attemptId")  
 Optional<Attempt> getAttemptDetails(@Param("userId") int userId, @Param("attemptId") int attemptId);  
  
 @Query("SELECT DISTINCT a FROM Attempt a JOIN FETCH a.user u")  
 List<Attempt> findAllWithUser();  
}

**AttemptService.java**

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;  
  
import java.util.List;  
  
@Service  
public class AttemptService {  
  
 @Autowired  
 private AttemptRepository attemptRepository;  
  
 public Attempt getAttemptDetails(int userId, int attemptId) {  
 return attemptRepository.getAttemptDetails(userId, attemptId).orElse(null);  
 }  
  
 public List<Attempt> getAllAttempts() {  
 return attemptRepository.findAllWithUser();  
 }  
}

**Output:**



**12. Get average salary using HQL**

**EmployeeRepository.java**

package com.cognizant.orm\_learn.repository;  
  
import com.cognizant.orm\_learn.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> {   
 @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 departmentId);  
}

**EmployeeService.java**

package com.cognizant.orm\_learn.service;  
  
public interface EmployeeService {  
 double getAverageSalary();  
 double getAverageSalaryByDepartment(int departmentId);  
}

**EmployeeServiceImpl.java**

package com.cognizant.orm\_learn.service;  
  
import com.cognizant.orm\_learn.repository.EmployeeRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
@Service  
public class EmployeeServiceImpl implements EmployeeService {  
  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 @Override  
 public double getAverageSalary() {  
 return employeeRepository.getAverageSalary();  
 }  
  
 @Override  
 public double getAverageSalaryByDepartment(int departmentId) {  
 return employeeRepository.getAverageSalary(departmentId);  
 }  
}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.service.EmployeeService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class OrmLearnApplication implements CommandLineRunner {  
  
 @Autowired  
 private EmployeeService employeeService;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(OrmLearnApplication.class, args);  
 }  
  
 @Override  
 public void run(String... args) {  
  
 // 1. Get average salary of all employees  
 double avgSalaryAll = employeeService.getAverageSalary();  
 System.*out*.println("Average salary of all employees: " + avgSalaryAll);  
  
 // 2. Get average salary of a department  
 int testDepartmentId = 1;

double avgSalaryByDept = employeeService.getAverageSalaryByDepartment(testDepartmentId);  
 System.*out*.println("Average salary for department ID " + testDepartmentId + ": " + avgSalaryByDept);  
 }  
}

**Output:**

