Week 3: Spring Data JPA (Spring Boot & Hibernate)

**Exercise 1: Spring Data JPA – Quick Example**

**Scenario:**

Your company needs to develop a database-driven application for managing country information. You need to use Spring Data JPA with Hibernate to handle database operations and retrieve country data from a MySQL database.

**Procedure:**

**Step 1: Set up a Spring Boot Project**

* Create a Spring Boot project using Spring Initializr with group "com.cognizant" and artifact "orm-learn".
* Add Spring Data JPA, MySQL Driver, and Spring Boot DevTools dependencies.

**Step 2: Configure Database Connection**

* Create a MySQL schema named "**ormlearn**" and configure database properties in application.properties.
* Set up logging configuration for Spring Framework, Hibernate SQL, and application logs.

**Step 3: Create Entity and Repository Classes**

* Create a Country entity class with JPA annotations.
* Create a **CountryRepository** interface extending **JpaRepository**.

**Step 4: Implement Service Layer**

* Create a **CountryService** interface and its implementation.
* Use @Service and @Autowired annotations for dependency injection.

**Step 5: Test the Application**

* Modify the main application class to test the database connectivity and data retrieval.

**Implementation:**

**application.properties:**

# Spring Framework and application log  
logging.level.org.springframework=info  
logging.level.com.cognizant=debug  
  
# Hibernate logs for displaying executed SQL, input and output  
logging.level.org.hibernate.SQL=trace  
logging.level.org.hibernate.type.descriptor.sql=trace  
  
# Log pattern  
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n  
  
# Database configuration  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn  
spring.datasource.username=root  
spring.datasource.password=root  
  
# Hibernate configuration  
spring.jpa.hibernate.ddl-auto=validate  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

**Database Schema:**

create table country(co\_code varchar(2) primary key, co\_name varchar(50));  
insert into country values ('IN', 'India');  
insert into country values ('US', 'United States of America');

**Country Entity Class:**

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

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 Interface:**

package com.cognizant.orm\_learn.service;  
import com.cognizant.orm\_learn.model.Country;  
import java.util.List;  
public interface CountryService {  
 List<Country> getAllCountries();  
}

**CountryServiceImpl Class:**

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

**OrmLearnApplication Main Class:**

package com.cognizant.orm\_learn;  
import com.cognizant.orm\_learn.service.CountryService;  
import com.cognizant.orm\_learn.model.Country;  
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);  
 *LOGGER*.info("Inside main");  
 *countryService* = context.getBean(CountryService.class);  
 *testGetAllCountries*();  
 }  
 private static void testGetAllCountries() {  
 *LOGGER*.info("Initializing....");  
 List<Country> countries = *countryService*.getAllCountries();  
 *LOGGER*.debug("countries={}", countries);  
 *LOGGER*.info("Successfully Completed!");  
 }  
}

**Output:**

A screen shot of a computer

AI-generated content may be incorrect.

**Exercise 4: Difference between JPA, Hibernate and Spring Data JPA**

**JPA**

* **Java Persistence API (JPA)** JSR 338 Specification for persisting, reading and managing data from Java objects • Does not contain concrete implementation of the specification • Hibernate is one of the implementations of JPA.

**Hibernate**

* ORM Tool that implements JPA.

**Spring Data JPA**

* Does not have JPA implementation but reduces boiler plate code.
* This is another level of abstraction over JPA implementation provider like Hibernate.
* Manage transactions.

**Scenario:**

Your company needs to develop an employee management system. You need to implement the same functionality using both Spring Data JPA and Hibernate to understand the differences in code complexity and implementation approaches.

**Spring Data JPA Implementation:**

**Employee Entity Class:**

package com.example.springdatajpa;  
import jakarta.persistence.\*;  
@Entity  
public class Employee {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Integer id;  
  
 private String name;  
 private String department;  
  
 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 String getDepartment() {  
 return department;  
 }  
 public void setDepartment(String department) {  
 this.department = department;  
 }  
}

**EmployeeRepository Interface:**

package com.example.springdatajpa;  
import org.springframework.data.jpa.repository.JpaRepository;  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
}

**EmployeeService Class:**

package com.example.springdatajpa;  
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);  
 }  
}

**Main Application Class:**

package com.example.springdatajpa;  
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 SpringDataJpaExampleApplication implements CommandLineRunner {  
 @Autowired  
 private EmployeeService employeeService;  
 public static void main(String[] args) {  
 SpringApplication.run(SpringDataJpaExampleApplication.class, args);  
 }  
 @Override  
 public void run(String... args) throws Exception {  
 Employee emp = new Employee();  
 emp.setName("Sethu Raman");  
 emp.setDepartment("HR");  
 employeeService.addEmployee(emp);  
 System.*out*.println("Employee inserted successfully.");  
 }  
}

**application.properties:**

spring.datasource.url=jdbc:mysql://localhost:3306/springdb  
spring.datasource.username=root  
spring.datasource.password=root  
spring.jpa.hibernate.ddl-auto=update  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
spring.jpa.show-sql=true

**Output:**

A computer screen shot of a computer program

AI-generated content may be incorrect.

**Hibernate Implementation:**

**Employee Entity Class:**

package com.example;  
import javax.persistence.\*;  
@Entity  
@Table(name = "employees")  
public class Employee {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
 @Column(name = "name", nullable = false, length = 100)  
 private String name;  
 @Column(name = "department", nullable = false, length = 100)  
 private String department;  
 public Employee() {}  
 public Employee(String name, String department) {  
 this.name = name;  
 this.department = department;  
 }  
 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 getDepartment() {  
 return department;  
 }  
 public void setDepartment(String department) {  
 this.department = department;  
 }  
 @Override  
 public String toString() {  
 return "Employee{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", department='" + department + '\'' +  
 '}';  
 }  
}

**HibernateUtil Class:**

package com.example;  
import org.hibernate.SessionFactory;  
import org.hibernate.cfg.Configuration;  
public class HibernateUtil {  
 private static final SessionFactory *sessionFactory*;  
 static {  
 try {  
 Configuration configuration = new Configuration();  
 configuration.configure("hibernate.cfg.xml");  
 configuration.addAnnotatedClass(Employee.class);  
 *sessionFactory* = configuration.buildSessionFactory();  
 System.*out*.println("SessionFactory created successfully!");  
 } catch (Throwable ex) {  
 System.*err*.println("Initial SessionFactory creation failed: " + ex);  
 ex.printStackTrace();  
 throw new ExceptionInInitializerError(ex);  
 }  
 }  
 public static SessionFactory getSessionFactory() {  
 return *sessionFactory*;  
 }  
 public static void shutdown() {  
 if (*sessionFactory* != null) {  
 *sessionFactory*.close();  
 }  
 }  
}

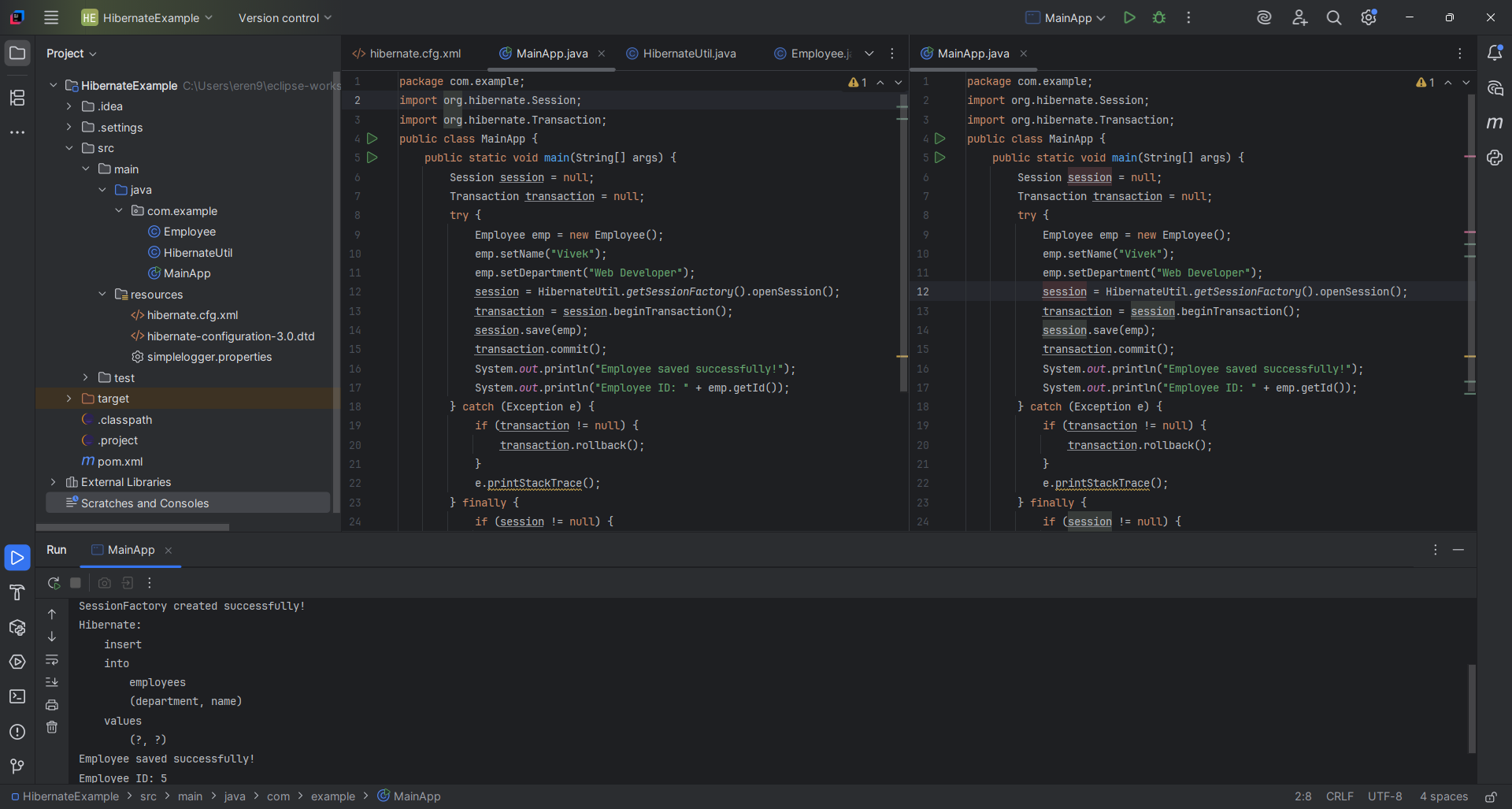
**MainApp Class:**

package com.example;  
import org.hibernate.Session;  
import org.hibernate.Transaction;  
public class MainApp {  
 public static void main(String[] args) {  
 Session session = null;  
 Transaction transaction = null;  
 try {  
 Employee emp = new Employee();  
 emp.setName("Vivek");  
 emp.setDepartment("Web Developer");  
 session = HibernateUtil.*getSessionFactory*().openSession();  
 transaction = session.beginTransaction();  
 session.save(emp);  
 transaction.commit();  
 System.*out*.println("Employee saved successfully!");  
 System.*out*.println("Employee ID: " + emp.getId());  
 } catch (Exception e) {  
 if (transaction != null) {  
 transaction.rollback();  
 }  
 e.printStackTrace();  
 } finally {  
 if (session != null) {  
 session.close();  
 }  
 }  
 HibernateUtil.*shutdown*();  
 }  
}

**hibernate.cfg.xml:**

<?xml version='1.0' encoding='utf-8'?>  
<!DOCTYPE hibernate-configuration SYSTEM "hibernate-configuration-3.0.dtd">  
<hibernate-configuration>  
 <session-factory>  
 <property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>  
 <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/employee\_db?useSSL=false&amp;serverTimezone=UTC&amp;allowPublicKeyRetrieval=true</property>  
 <property name="hibernate.connection.username">root</property>  
 <property name="hibernate.connection.password">root</property>  
 <property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>  
 <property name="hibernate.connection.pool\_size">10</property>  
 <property name="hibernate.hbm2ddl.auto">update</property>  
 <property name="hibernate.show\_sql">true</property>  
 <property name="hibernate.format\_sql">true</property>  
 <mapping class="com.example.Employee"/>  
 </session-factory>  
</hibernate-configuration>

**Output:**



**Conclusion:**

Spring Data JPA significantly reduces code complexity and provides higher-level abstractions, while Hibernate offers more control and flexibility for complex scenarios.