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

Solution:

JpaHibernateDemoApplication.java:

package com.example.jpahibernatedemo;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.transaction.annotation.EnableTransactionManagement;  
  
@SpringBootApplication  
@EnableTransactionManagement  
public class JpaHibernateDemoApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(JpaHibernateDemoApplication.class, args);  
 }  
}

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>3.2.0</version>  
 <relativePath/>  
 </parent>  
 <groupId>com.example</groupId>  
 <artifactId>jpa-hibernate-demo</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>jpa-hibernate-demo</name>  
 <description>Demo project for JPA, Hibernate, and Spring Data JPA</description>  
 <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-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>8.0.33</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </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>

Employee.java:

package com.example.jpahibernatedemo.entity;  
  
import jakarta.persistence.\*;  
import java.time.LocalDate;  
  
@Entity  
@Table(name = "employees")  
public class Employee {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Integer id;  
  
 @Column(name = "first\_name", nullable = false, length = 50)  
 private String firstName;  
  
 @Column(name = "last\_name", nullable = false, length = 50)  
 private String lastName;  
  
 @Column(name = "email", unique = true, nullable = false)  
 private String email;  
  
 @Column(name = "department")  
 private String department;  
  
 @Column(name = "salary")  
 private Double salary;  
  
 @Column(name = "hire\_date")  
 private LocalDate hireDate;  
  
 // Default constructor  
 public Employee() {}  
  
 // Constructor with parameters  
 public Employee(String firstName, String lastName, String email, String department, Double salary, LocalDate hireDate) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.email = email;  
 this.department = department;  
 this.salary = salary;  
 this.hireDate = hireDate;  
 }  
  
 // Getters and Setters  
 public Integer getId() { return id; }  
 public void setId(Integer id) { this.id = id; }  
  
 public String getFirstName() { return firstName; }  
 public void setFirstName(String firstName) { this.firstName = firstName; }  
  
 public String getLastName() { return lastName; }  
 public void setLastName(String lastName) { this.lastName = lastName; }  
  
 public String getEmail() { return email; }  
 public void setEmail(String email) { this.email = email; }  
  
 public String getDepartment() { return department; }  
 public void setDepartment(String department) { this.department = department; }  
  
 public Double getSalary() { return salary; }  
 public void setSalary(Double salary) { this.salary = salary; }  
  
 public LocalDate getHireDate() { return hireDate; }  
 public void setHireDate(LocalDate hireDate) { this.hireDate = hireDate; }  
  
 @Override  
 public String toString() {  
 return "Employee{" +  
 "id=" + id +  
 ", firstName='" + firstName + '\'' +  
 ", lastName='" + lastName + '\'' +  
 ", email='" + email + '\'' +  
 ", department='" + department + '\'' +  
 ", salary=" + salary +  
 ", hireDate=" + hireDate +  
 '}';  
 }  
}

EmployeeRepository.java:

package com.example.jpahibernatedemo.repository;  
  
import com.example.jpahibernatedemo.entity.Employee;  
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 java.util.List;  
  
@Repository  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
  
 // Custom query methods using method naming convention  
 List<Employee> findByDepartment(String department);  
 List<Employee> findByFirstNameAndLastName(String firstName, String lastName);  
 List<Employee> findBySalaryGreaterThan(Double salary);  
  
 // Custom JPQL query  
 @Query("SELECT e FROM Employee e WHERE e.email = :email")  
 Employee findByEmail(@Param("email") String email);  
  
 // Native SQL query  
 @Query(value = "SELECT \* FROM employees WHERE department = ?1 ORDER BY salary DESC", nativeQuery = true)  
 List<Employee> findByDepartmentOrderBySalaryDesc(String department);  
}

EmployeeService.java:

package com.example.jpahibernatedemo.service;  
  
import com.example.jpahibernatedemo.entity.Employee;  
import com.example.jpahibernatedemo.repository.EmployeeRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
@Transactional  
public class EmployeeService {  
  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 // CREATE  
 public Employee addEmployee(Employee employee) {  
 return employeeRepository.save(employee);  
 }  
  
 // READ  
 public List<Employee> getAllEmployees() {  
 return employeeRepository.findAll();  
 }  
  
 public Optional<Employee> getEmployeeById(Integer id) {  
 return employeeRepository.findById(id);  
 }  
  
 public List<Employee> getEmployeesByDepartment(String department) {  
 return employeeRepository.findByDepartment(department);  
 }  
  
 public Employee getEmployeeByEmail(String email) {  
 return employeeRepository.findByEmail(email);  
 }  
  
 // UPDATE  
 public Employee updateEmployee(Employee employee) {  
 return employeeRepository.save(employee);  
 }  
  
 // DELETE  
 public void deleteEmployee(Integer id) {  
 employeeRepository.deleteById(id);  
 }  
  
 public void deleteAllEmployees() {  
 employeeRepository.deleteAll();  
 }  
  
 // Custom business logic  
 public List<Employee> getHighEarners(Double minSalary) {  
 return employeeRepository.findBySalaryGreaterThan(minSalary);  
 }  
  
 public long getEmployeeCount() {  
 return employeeRepository.count();  
 }  
}

EmployeeHibernateService.java:

package com.example.jpahibernatedemo.service;  
  
import com.example.jpahibernatedemo.entity.Employee;  
import org.hibernate.Session;  
import org.hibernate.SessionFactory;  
import org.hibernate.Transaction;  
import org.hibernate.query.Query;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class EmployeeHibernateService {  
  
 @Autowired  
 private SessionFactory sessionFactory;  
  
 // CREATE - Traditional Hibernate approach  
 public Integer addEmployee(Employee employee) {  
 Session session = sessionFactory.openSession();  
 Transaction tx = null;  
 Integer employeeID = null;  
  
 try {  
 tx = session.beginTransaction();  
 employeeID = (Integer) session.save(employee);  
 tx.commit();  
 System.*out*.println("Employee created with ID: " + employeeID);  
 } catch (Exception e) {  
 if (tx != null) tx.rollback();  
 e.printStackTrace();  
 } finally {  
 session.close();  
 }  
 return employeeID;  
 }  
  
 // READ - Get employee by ID  
 public Employee getEmployeeById(Integer id) {  
 Session session = sessionFactory.openSession();  
 Employee employee = null;  
  
 try {  
 employee = session.get(Employee.class, id);  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 session.close();  
 }  
 return employee;  
 }  
  
 // READ - Get all employees  
 public List<Employee> getAllEmployees() {  
 Session session = sessionFactory.openSession();  
 List<Employee> employees = null;  
  
 try {  
 Query<Employee> query = session.createQuery("FROM Employee", Employee.class);  
 employees = query.getResultList();  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 session.close();  
 }  
 return employees;  
 }  
  
 // UPDATE  
 public void updateEmployee(Employee employee) {  
 Session session = sessionFactory.openSession();  
 Transaction tx = null;  
  
 try {  
 tx = session.beginTransaction();  
 session.update(employee);  
 tx.commit();  
 System.*out*.println("Employee updated successfully");  
 } catch (Exception e) {  
 if (tx != null) tx.rollback();  
 e.printStackTrace();  
 } finally {  
 session.close();  
 }  
 }  
  
 // DELETE  
 public void deleteEmployee(Integer id) {  
 Session session = sessionFactory.openSession();  
 Transaction tx = null;  
  
 try {  
 tx = session.beginTransaction();  
 Employee employee = session.get(Employee.class, id);  
 if (employee != null) {  
 session.delete(employee);  
 System.*out*.println("Employee deleted successfully");  
 }  
 tx.commit();  
 } catch (Exception e) {  
 if (tx != null) tx.rollback();  
 e.printStackTrace();  
 } finally {  
 session.close();  
 }  
 }  
  
 // Custom HQL query  
 public List<Employee> getEmployeesByDepartment(String department) {  
 Session session = sessionFactory.openSession();  
 List<Employee> employees = null;  
  
 try {  
 Query<Employee> query = session.createQuery(  
 "FROM Employee e WHERE e.department = :dept", Employee.class);  
 query.setParameter("dept", department);  
 employees = query.getResultList();  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 session.close();  
 }  
 return employees;  
 }  
}

EmployeeController:

package com.example.jpahibernatedemo.controller;  
  
import com.example.jpahibernatedemo.entity.Employee;  
import com.example.jpahibernatedemo.service.EmployeeService;  
import com.example.jpahibernatedemo.service.EmployeeHibernateService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.time.LocalDate;  
import java.util.List;  
import java.util.Optional;  
  
@RestController  
@RequestMapping("/api/employees")  
public class EmployeeController {  
  
 @Autowired  
 private EmployeeService employeeService;  
  
 @Autowired  
 private EmployeeHibernateService employeeHibernateService;  
  
 // Spring Data JPA Endpoints  
 @PostMapping("/jpa")  
 public ResponseEntity<Employee> createEmployeeJPA(@RequestBody Employee employee) {  
 Employee savedEmployee = employeeService.addEmployee(employee);  
 return ResponseEntity.*ok*(savedEmployee);  
 }  
  
 @GetMapping("/jpa")  
 public ResponseEntity<List<Employee>> getAllEmployeesJPA() {  
 List<Employee> employees = employeeService.getAllEmployees();  
 return ResponseEntity.*ok*(employees);  
 }  
  
 @GetMapping("/jpa/{id}")  
 public ResponseEntity<Employee> getEmployeeByIdJPA(@PathVariable Integer id) {  
 Optional<Employee> employee = employeeService.getEmployeeById(id);  
 return employee.map(ResponseEntity::*ok*)  
 .orElse(ResponseEntity.*notFound*().build());  
 }  
  
 @PutMapping("/jpa/{id}")  
 public ResponseEntity<Employee> updateEmployeeJPA(@PathVariable Integer id, @RequestBody Employee employee) {  
 Optional<Employee> existingEmployee = employeeService.getEmployeeById(id);  
 if (existingEmployee.isPresent()) {  
 employee.setId(id);  
 Employee updatedEmployee = employeeService.updateEmployee(employee);  
 return ResponseEntity.*ok*(updatedEmployee);  
 }  
 return ResponseEntity.*notFound*().build();  
 }  
  
 @DeleteMapping("/jpa/{id}")  
 public ResponseEntity<Void> deleteEmployeeJPA(@PathVariable Integer id) {  
 Optional<Employee> employee = employeeService.getEmployeeById(id);  
 if (employee.isPresent()) {  
 employeeService.deleteEmployee(id);  
 return ResponseEntity.*noContent*().build();  
 }  
 return ResponseEntity.*notFound*().build();  
 }  
  
 // Hibernate Endpoints  
 @PostMapping("/hibernate")  
 public ResponseEntity<String> createEmployeeHibernate(@RequestBody Employee employee) {  
 Integer employeeId = employeeHibernateService.addEmployee(employee);  
 return ResponseEntity.*ok*("Employee created with ID: " + employeeId);  
 }  
  
 @GetMapping("/hibernate")  
 public ResponseEntity<List<Employee>> getAllEmployeesHibernate() {  
 List<Employee> employees = employeeHibernateService.getAllEmployees();  
 return ResponseEntity.*ok*(employees);  
 }  
  
 @GetMapping("/hibernate/{id}")  
 public ResponseEntity<Employee> getEmployeeByIdHibernate(@PathVariable Integer id) {  
 Employee employee = employeeHibernateService.getEmployeeById(id);  
 if (employee != null) {  
 return ResponseEntity.*ok*(employee);  
 }  
 return ResponseEntity.*notFound*().build();  
 }  
  
 @PutMapping("/hibernate/{id}")  
 public ResponseEntity<String> updateEmployeeHibernate(@PathVariable Integer id, @RequestBody Employee employee) {  
 employee.setId(id);  
 employeeHibernateService.updateEmployee(employee);  
 return ResponseEntity.*ok*("Employee updated successfully");  
 }  
  
 @DeleteMapping("/hibernate/{id}")  
 public ResponseEntity<String> deleteEmployeeHibernate(@PathVariable Integer id) {  
 employeeHibernateService.deleteEmployee(id);  
 return ResponseEntity.*ok*("Employee deleted successfully");  
 }  
  
 // Test endpoint to create sample data  
 @PostMapping("/sample-data")  
 public ResponseEntity<String> createSampleData() {  
 Employee emp1 = new Employee("John", "Doe", "john.doe@example.com", "IT", 75000.0, LocalDate.*of*(2020, 1, 15));  
 Employee emp2 = new Employee("Jane", "Smith", "jane.smith@example.com", "HR", 65000.0, LocalDate.*of*(2019, 3, 20));  
 Employee emp3 = new Employee("Bob", "Johnson", "bob.johnson@example.com", "IT", 80000.0, LocalDate.*of*(2021, 6, 10));  
  
 employeeService.addEmployee(emp1);  
 employeeService.addEmployee(emp2);  
 employeeService.addEmployee(emp3);  
  
 return ResponseEntity.*ok*("Sample data created successfully");  
 }  
}

HibernateConfig.java:

package com.example.jpahibernatedemo.config;  
  
import org.hibernate.SessionFactory;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;  
  
import jakarta.persistence.EntityManagerFactory;  
  
@Configuration  
public class HibernateConfig {  
  
 @Bean  
 public SessionFactory sessionFactory(EntityManagerFactory entityManagerFactory) {  
 return entityManagerFactory.unwrap(SessionFactory.class);  
 }  
}

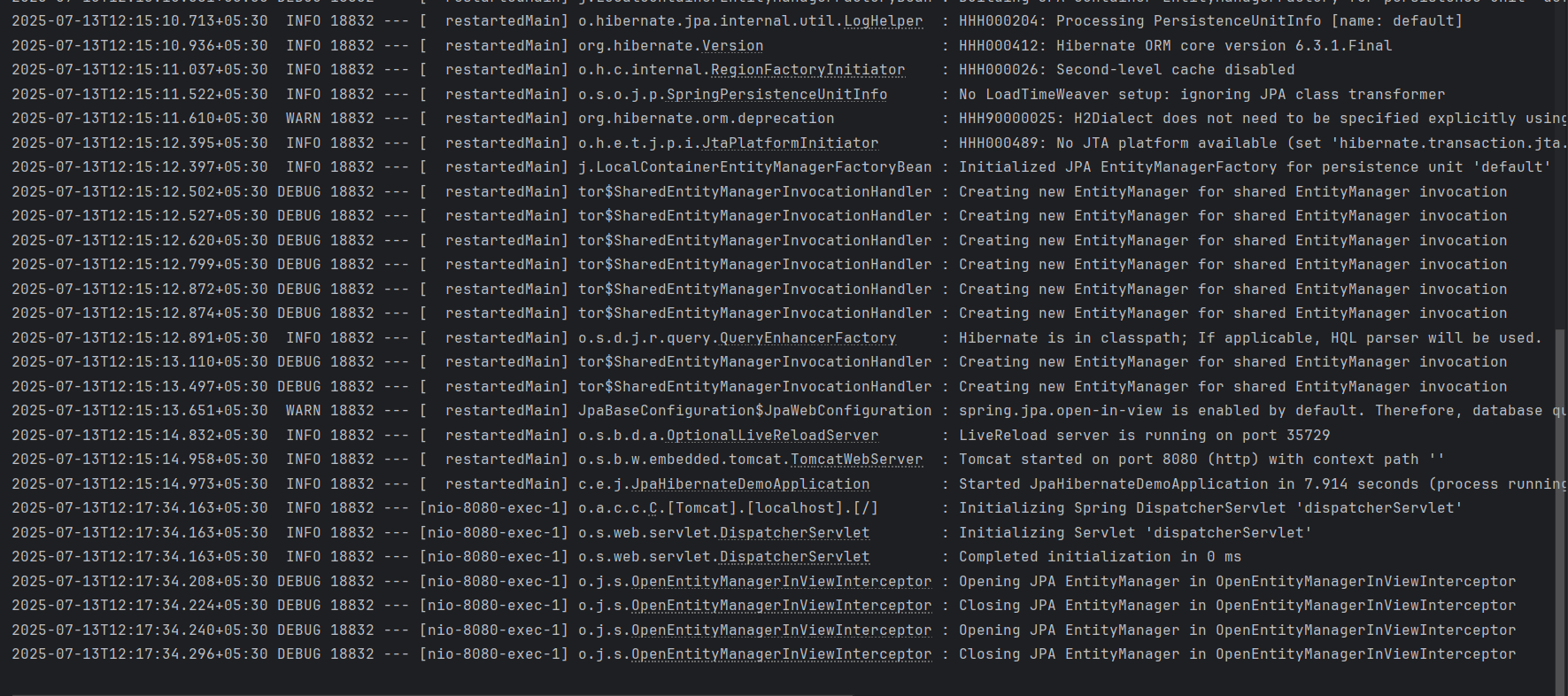
JPAConfig:

package com.example.jpahibernatedemo.config;  
  
import org.springframework.boot.autoconfigure.domain.EntityScan;  
import org.springframework.boot.orm.jpa.EntityManagerFactoryBuilder;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.context.annotation.Primary;  
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;  
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;  
  
import javax.sql.DataSource;  
  
@Configuration  
@EnableJpaRepositories(basePackages = "com.example.jpahibernatedemo.repository")  
@EntityScan(basePackages = "com.example.jpahibernatedemo.entity")  
public class JPAConfig {  
  
 @Bean  
 @Primary  
 public LocalContainerEntityManagerFactoryBean entityManagerFactory(  
 EntityManagerFactoryBuilder builder, DataSource dataSource) {  
 return builder  
 .dataSource(dataSource)  
 .packages("com.example.jpahibernatedemo.entity")  
 .persistenceUnit("default")  
 .build();  
 }  
}

application.properties:

# H2 Database Configuration (for testing)  
spring.datasource.url=jdbc:h2:mem:testdb  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=password  
spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console  
  
# JPA/Hibernate Configuration  
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect  
spring.jpa.hibernate.ddl-auto=create-drop  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.format\_sql=true  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect  
  
# EntityManagerFactory Configuration  
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl  
spring.jpa.hibernate.naming.implicit-strategy=org.hibernate.boot.model.naming.ImplicitNamingStrategyLegacyJpaImpl  
  
# Server Configuration  
server.port=8080  
  
# Logging  
logging.level.org.hibernate.SQL=DEBUG  
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE  
logging.level.org.springframework.orm.jpa=DEBUG  
logging.level.org.springframework.transaction=DEBUG

Output:

A black screen with many small squares

AI-generated content may be incorrect.