**Week -3**

1. spring-data-jpa-handson

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

| **Feature** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Type** | Specification / API | Framework / ORM tool | Spring abstraction layer |
| **Purpose** | Defines standard API for ORM in Java | Concrete implementation of JPA & extra features | Simplifies data access using repositories |
| **Implementation** | Only defines interfaces (like EntityManager) | Provides implementation for JPA APIs and more | Uses existing JPA implementations like Hibernate under the hood |
| **Boilerplate Code** | Required (EntityManager, queries) | Less than plain JDBC but still manual | Greatly reduced — CRUD, pagination, sorting with no boilerplate |
| **Transaction Management** | Manual | Managed via Hibernate’s API | Managed by Spring Framework |
| **Query Support** | JPQL (Java Persistence Query Language) | JPQL + Hibernate Query Language (HQL) + Criteria API | JPQL + Derived Query Methods + @Query support |
| **Integration** | Can be used with any Java application | Used in many frameworks like Spring, Struts | Deep integration with Spring Boot and Spring ecosystem |
| **Caching** | No caching API defined | Supports 1st and 2nd level caching | Uses Hibernate caching when applicable |
| **Learning Curve** | Medium | Medium to High | Low, especially for Spring developers |
| **Flexibility** | Flexible and standard | More powerful and feature-rich | Focused on productivity with convention-over-configuration |

**The code compares between JPA, Hibernate, and Spring Data JPA:-**

Below is a side-by-side code comparison of the **same functionality** — saving an employee record — using:

1. **JPA (pure Java Persistence API)**
2. **Hibernate (direct usage)**
3. **Spring Data JPA (with Spring Boot)**

**1.Using JPA :**

// Employee.java

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

private String name;

private double salary;

// getters & setters

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// JPAExample.java

public class JPAExample {

public static void main(String[] args) {

EntityManagerFactory emf = Persistence.createEntityManagerFactory("employeePU");

EntityManager em = emf.createEntityManager();

Employee emp = new Employee();

emp.setName("John Doe");

emp.setSalary(50000);

em.getTransaction().begin();

em.persist(emp);

em.getTransaction().commit();

em.close();

emf.close();

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

<!-- persistence.xml (in META-INF folder) -->

<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence" version="2.2">

<persistence-unit name="employeePU">

<class>com.example.Employee</class>

<properties>

<property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/testdb"/>

<property name="javax.persistence.jdbc.user" value="root"/>

<property name="javax.persistence.jdbc.password" value="root"/>

<property name="javax.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver"/>

<property name="hibernate.hbm2ddl.auto" value="update"/>

</properties>

</persistence-unit>

</persistence>

**2.Using Hibernate :**

// Employee.java – same as above

// HibernateExample.java

public class HibernateExample {

public static void main(String[] args) {

SessionFactory factory = new Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(Employee.class)

.buildSessionFactory();

Session session = factory.openSession();

Employee emp = new Employee();

emp.setName("Jane Smith");

emp.setSalary(60000);

Transaction tx = null;

try {

tx = session.beginTransaction();

session.save(emp);

tx.commit();

} catch (Exception e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

factory.close();

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

<!-- hibernate.cfg.xml -->

<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/testdb</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>

<property name="hibernate.hbm2ddl.auto">update</property>

</session-factory>

</hibernate-configuration>

**3.Using Spring Data JPA :**

// Employee.java – same entity

// EmployeeRepository.java

@Repository

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

// EmployeeService.java

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public void saveEmployee() {

Employee emp = new Employee();

emp.setName("Alex Johnson");

emp.setSalary(70000);

employeeRepository.save(emp);

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// Application.java

@SpringBootApplication

public class Application implements CommandLineRunner {

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

@Override

public void run(String... args) {

employeeService.saveEmployee();

}

}

# application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/testdb

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true