**Hands on 4**

**Difference between JPA, Hibernate and Spring Data 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 implementation 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
* Manages transactions

**Refer code snippets below on how the code compares between Hibernate and Spring Data JPA  
Hibernate**

   /\* Method to CREATE an employee in the database \*/

   public Integer addEmployee(Employee employee){

      Session session = factory.openSession();

      Transaction tx = null;

      Integer employeeID = null;

      try {

         tx = session.beginTransaction();

         employeeID = (Integer) session.save(employee);

         tx.commit();

      } catch (HibernateException e) {

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

         e.printStackTrace();

      } finally {

         session.close();

      }

      return employeeID;

   }

**Spring Data JPA**  
EmployeeRespository.java

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

EmployeeService.java

@Autowire

  private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

  employeeRepository.save(employee);

  }

**Solution:**

**JPA (Java Persistence API)**

* **Definition:** A specification (JSR 338) that defines how Java objects are mapped to relational databases.
* **Key Points:**
  + Provides annotations like @Entity, @Id, @OneToMany, etc.
  + Only defines *what* should be done, not *how*.
  + Requires a concrete implementation (e.g., Hibernate, EclipseLink).
* **JPA (Java Persistence API)Example:**

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Column;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name", nullable = false)

private String name;

private String department;

// Constructors

public Employee() {

}

public Employee(String name, String department) {

this.name = name;

this.department = department;

}

// 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 String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

}

**Hibernate**

* **Definition:** A popular **Object-Relational Mapping (ORM)** tool that implements the JPA specification.
* **Key Points:**
  + Provides core ORM functionality (session management, transactions).
  + You write a lot of boilerplate code for session handling, transactions, etc.
  + Uses classes like Session, Transaction, Configuration, etc.

**Hibernate Code Example:**

public Integer addEmployee(Employee employee) {

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

} catch (HibernateException e) {

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

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

**Spring Data JPA**

* **Definition:** A Spring project that builds on top of JPA (and a provider like Hibernate) to **reduce boilerplate code**.
* **Key Points:**
  + Adds another layer of abstraction over JPA.
  + Automatically implements repository patterns via interfaces.
  + No need to write implementation logic for common operations (like save, delete, findAll, etc.).
  + Integrates with Spring’s dependency injection and transaction management.

**Spring Data JPA Code Example:**

**EmployeeRepository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}