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

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

Vaishnavi

4/7/2025

In Java applications, interacting with a relational database involves object-relational mapping (ORM), where Java classes are mapped to database tables. To simplify this process, frameworks like **JPA**, **Hibernate**, and **Spring Data JPA** are used.

**JPA (Java Persistence API)** is a **specification** that defines a standard way to map Java objects to relational databases.

**Hibernate** is a popular **implementation** of JPA and provides additional features such as caching, lazy loading, and custom HQL queries.

**Spring Data JPA** is a **Spring-based abstraction** built on top of JPA and Hibernate, which **eliminates boilerplate code** and simplifies database access using repository interfaces.

**Objective:**

Understand the purpose and functionality of JPA, Hibernate, and Spring Data JPA.

Learn how they differ in terms of configuration, complexity, and coding.

Demonstrate practical code examples for persisting an Employee entity using Hibernate and Spring Data JPA.

Highlight the benefits of using Spring Data JPA for faster development and cleaner architecture.

**Implementation:**

### Setup a Project Initialization

### ****Open Spring Initializer****

Group: com.cognizant

Artifact: orm-learn

Dependencies:

Spring Boot DevTools

Spring Data JPA

MySQL Driver

### ****Open a Existing Maven Project****:

File > Import > Maven > Existing Maven Projects

## Add Application Properties

# Logging

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

# Console Log Pattern

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

# DB 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=my\_passoword

# Hibernate

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

**Hibernate Example**

/\* 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); // Manual session handling

tx.commit();

} catch (HibernateException e) {

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

e.printStackTrace();

} finally {

session.close(); // Manual resource management

}

return employeeID;

}

**Model: Employee.java**

package com.example.employeeapp.model;

import jakarta.persistence.\*;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

private String name;

private String department;

// Getters and Setters

}

**Repository: EmployeeRepository.java**

package com.example.employeeapp.repository;

import com.example.employeeapp.model.Employee;

import org.springframework.data.jpa.repository.JpaRepository;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**Service: EmployeeService.java**

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee); // Simple one-liner for save

}

}

**Controller: EmployeeController.java**

package com.example.employeeapp.controller;

import com.example.employeeapp.model.Employee;

import com.example.employeeapp.service.EmployeeService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/employee")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PostMapping("/add")

public String addEmployee(@RequestBody Employee employee) {

employeeService.addEmployee(employee);

return "Employee added!";

}

}

**Run this application and open postman**

POST--------http://localhost:8080/employee/add

Content-Type: application/json

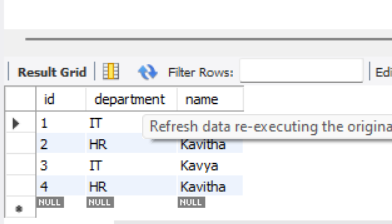
{

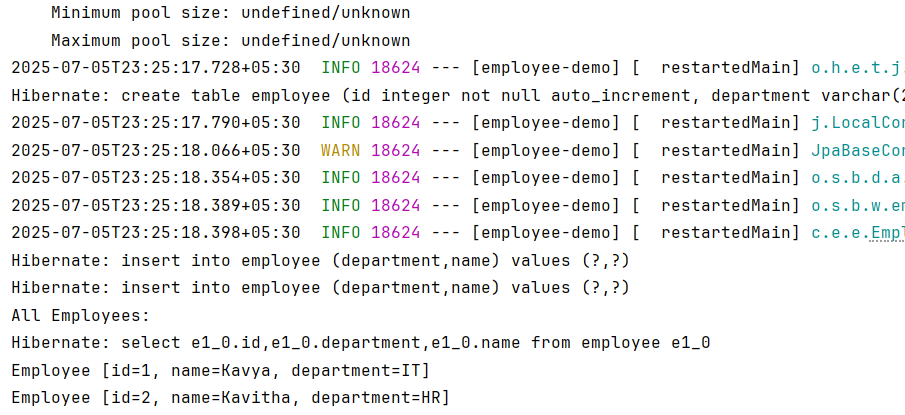
"name": "Vaishnavi",

"department": "Engineering"

}

**Output:**

****

****

**Difference:**

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (API) | Implementation (ORM provider) | Framework/Abstraction |
| Role | Defines standard annotations & interfaces for persistence | Implements JPA & adds extra features (e.g., caching, lazy loading) | Simplifies JPA-based development with built-in CRUD, queries |
| Exampel in Code | @Entity, @Id, EntityManager | hibernate.dialect, @GeneratedValue(strategy=...) | extends JpaRepository<Employee, Integer> |
| Who provides it? | Oracle (Java EE spec) | Red Hat | Spring Framework |