# Week3: Spring Data JPA with Spring Boot, Hibernate

# Hands-on 4: Difference between JPA, Hibernate, and Spring Data JPA

## 1. Definitions

**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 boilerplate code

• Provides an abstraction layer over JPA implementations like Hibernate

• Manages transactions automatically

## 2. Code Comparison

**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);   
 tx.commit();  
 } catch (HibernateException e) {  
 if (tx != null) tx.rollback();  
 e.printStackTrace();   
 } finally {  
 session.close();   
 }  
 return employeeID;  
}

**Spring Data JPA Example:**

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

import org.springframework.stereotype.Repository;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**FILE NAME: EmployeeRepository**

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

import org.springframework.stereotype.Repository;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**FILE NAME: EmployeeService**

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);

}

}

**FILE NAME: main**

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class EmployeeApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeApplication.class, args);

}

@Bean

CommandLineRunner run(EmployeeService employeeService) {

return args -> {

Employee employee = new Employee();

employee.setName("John Doe");

employee.setDepartment("Engineering");

employee.setSalary(35000);

Integer id = employeeService.addEmployee(employee);

System.out.println("Employee added with ID: " + id);

};

}

}

3. Feature Comparison Table

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA (Specification) | Hibernate (Implementation) | Spring Data JPA (Abstraction) |
| Provides Implementation | No | Yes | No |
| Session/Transaction Control | No (API only) | Manual (Session, TX) | Automatic (via @Transactional) |
| Boilerplate Code | High | Medium | Low |
| Integration with Spring | N/ | Works standalone | Deep Spring integration |