1. **Spring Data JPA - Quick Example**

**Code:**

**pom.xml**

<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>

**Book.java**

package com.library.model;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
@Entity  
public class Book {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY) // Auto-generate ID  
 private Long id;  
 private String title;  
 public Book() {}  
 public Book(String title) {  
 this.title = title;  
 }  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
 public String getTitle() {  
 return title;  
 }  
 public void setTitle(String title) {  
 this.title = title;  
 }  
}

**BookRepository.java**

package com.library.repository;

import com.library.model.Book;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface BookRepository extends CrudRepository<Book, Long> {

// You can define custom query methods here if needed, e.g.:

// List<Book> findByTitle(String title);

}

**LibraryApp.java**

package com.library;

import com.library.model.Book;

import com.library.repository.BookRepository;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LibraryApp implements CommandLineRunner {

@Autowired

private BookRepository bookRepo;

public static void main(String[] args) {

SpringApplication.run(LibraryApp.class, args);

}

@Override

public void run(String... args) {

bookRepo.save(new Book("Spring in Action"));

bookRepo.save(new Book("Effective Java"));

bookRepo.findAll().forEach(book ->

System.out.println(book.getId() + " - " + book.getTitle())

);

}

}

**Application.properties**

spring.datasource.url=jdbc:h2:mem:librarydb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update

**Output:**

1 - Spring in Action

2 - Effective Java

[INFO] BUILD SUCCESS

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

**JPA:**

* **JPA** is a **specification**, not an implementation.
* It defines a standard set of annotations and interfaces for ORM (Object-Relational Mapping).
* You can use it with implementations like **Hibernate**, **EclipseLink**, or **OpenJPA**.
* It abstracts the persistence logic, allowing easier migration between ORM providers.

**Example is as follows:**

import javax.persistence.\*;

public class JPAExample {

public static void main(String[] args) {

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

EntityManager em = emf.createEntityManager();

em.getTransaction().begin();

Book book = new Book(1L, "JPA in Action");

em.persist(book); // Persisting the entity using JPA API

em.getTransaction().commit();

em.close();

emf.close();

}

}

**Hibernate:**

**Hibernate** is an **implementation** of JPA and also provides additional features beyond the JPA standard.

It allows mapping Java objects to database tables.

You work with **SessionFactory**, **Session**, and **Transaction** (Hibernate-specific classes).

**Example is**

import org.hibernate.\*;

import org.hibernate.cfg.Configuration;

public class HibernateExample {

public static void main(String[] args) {

SessionFactory factory = new Configuration().configure().buildSessionFactory();

Session session = factory.openSession();

Transaction tx = session.beginTransaction();

Book book = new Book(2L, "Hibernate in Action");

session.save(book); // Saving using Hibernate API

tx.commit();

session.close();

factory.close();

}

}

**Spring data JPA:**

**Spring Data JPA** is a **Spring Framework module** that simplifies the use of JPA.

It reduces boilerplate code through **repositories** and **method-based queries**.

You just define an interface like BookRepository, and Spring handles the implementation.

**Example is**

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class SpringDataExample implements CommandLineRunner {

@Autowired

private BookRepository bookRepo;

@Override

public void run(String... args) throws Exception {

bookRepo.save(new Book(3L, "Spring Data JPA Magic")); // No need for transaction or session

}

}