# Spring Data JPA with Spring Boot, Hibernate

## Spring Data JPA - Quick Example

### CODE

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>spring-data-jpa-example</artifactId>

<version>0.0.1-SNAPSHOT</version>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.18</version>

<relativePath/>

</parent>

<dependencies>

<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>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**Book.java**

package com.example.demo;

import javax.persistence.\*;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

public Book() {}

public Book(String title) {

this.title = title;}

public String getTitle() {

return title;}

public void setTitle(String title) {

this.title = title;

}}

**BookRepository.java**

package com.example.demo;

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

public interface BookRepository extends JpaRepository<Book, Long> {

Book findByTitle(String title);

}

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.\*;

import org.springframework.boot.autoconfigure.\*;

import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

@Bean

CommandLineRunner run(BookRepository repo) {

return args -> {

repo.save(new Book("Spring Data JPA Guide"));

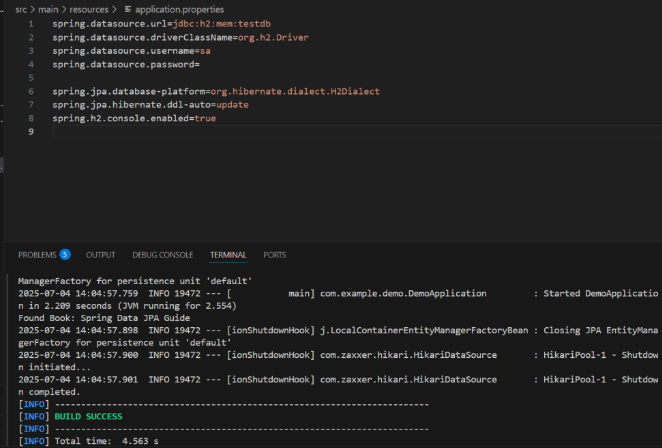
Book book = repo.findByTitle("Spring Data JPA Guide");

System.out.println("Found Book: " + book.getTitle());

};

}}

### OUTPUT



## Difference between JPA, Hibernate and Spring Data JPA

#### **1. JPA (Java Persistence API) – The Specification**

* **What it is**: JPA is a Java specification (interface) for accessing, persisting, and managing data between Java objects and a relational database.
* **Provided by**: Java EE (now Jakarta EE).
* **Role**: It defines a set of rules and annotations such as @Entity, @Id, @OneToMany, etc., but does not contain any implementation.
* **Limitation**: Being a specification, it requires a provider or implementation to function.

Example:

@Entity

public class Student {

@Id

private Long id;

private String name;

}

#### **2. Hibernate – An Implementation of JPA**

* **What it is**: Hibernate is a popular JPA provider. It is a framework that implements the JPA specification and provides the actual persistence logic.
* **Role**: It manages the database operations such as storing, retrieving, and updating Java objects in a relational database.
* **Features**: Hibernate offers additional features beyond JPA, such as caching, lazy loading, and dirty checking.

When you use JPA interfaces like EntityManager, Hibernate is the tool that executes the operations behind the scenes.

#### **3. Spring Data JPA – Abstraction over JPA**

* **What it is**: Spring Data JPA is part of the Spring ecosystem. It provides an abstraction over the JPA standard and typically uses Hibernate as the default JPA provider.
* **Role**: It simplifies JPA-based data access layers by reducing boilerplate code.
* **Features**: It provides ready-to-use interfaces like CrudRepository and JpaRepository, and supports query methods, pagination, and custom queries with minimal code.

Example:

public interface StudentRepository extends JpaRepository<Student, Long> {

List<Student> findByName(String name);

}

This interface is automatically implemented by Spring Data JPA at runtime.

### **Summary Table**

| **Aspect** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (interface) | Implementation (framework) | Abstraction/helper module |
| Provided by | Java EE | Red Hat | Spring Framework |
| Code Implementation | No | Yes | Yes (built on JPA) |
| Extra Features | No | Yes | Yes |
| Ease of Use | Manual | Easier than JPA | Easiest |

JPA defines the rules for persistence, Hibernate provides the implementation of those rules, and Spring Data JPA simplifies the process further by minimizing the amount of boilerplate code developers need to write.