# **Spring Data JPA – Quick Example**

Spring Data JPA is a part of the Spring Data family that simplifies data persistence using JPA (Java Persistence API). It builds on top of standard JPA and allows developers to create repositories by simply defining interfaces.

### **✅ Objective:**

Simplify the interaction with databases by eliminating boilerplate code (like writing queries for basic CRUD operations).

## **🔧 Example using Spring Boot + Spring Data JPA + H2 (In-Memory Database)**

### **1. Maven Dependencies (pom.xml):**

xml

CopyEdit

<dependencies>  
 <!-- Spring Boot Starter JPA -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
  
 <!-- H2 Database -->  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
</dependencies>

### **2. Entity Class (Book.java):**

java

CopyEdit

@Entity  
public class Book {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
 private String title;  
  
 // Constructors, Getters, Setters  
}

### **3. Repository Interface (BookRepository.java):**

java

CopyEdit

public interface BookRepository extends JpaRepository<Book, Long> {  
 List<Book> findByTitle(String title); // Auto-implemented by Spring Data JPA  
}

### **4. Service or Runner Class Example:**

java

CopyEdit

@SpringBootApplication  
public class LibraryApp implements CommandLineRunner {  
 @Autowired  
 private BookRepository bookRepository;  
  
 public static void main(String[] args) {  
 SpringApplication.run(LibraryApp.class, args);  
 }  
  
 @Override  
 public void run(String... args) {  
 bookRepository.save(new Book("Spring in Action"));  
 List<Book> books = bookRepository.findByTitle("Spring in Action");  
 books.forEach(book -> System.out.println(book.getTitle()));  
 }  
}

### **✅ Benefits:**

* No need to implement BookRepository.
* Auto-generated SQL queries for basic methods like findByTitle, save, findAll.
* Works with in-memory (H2) and external (MySQL, PostgreSQL) databases.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| Type | Specification (JSR 338) | JPA implementation & ORM framework | Abstraction over JPA implementations |
| Boilerplate Code | Some manual writing | Reduced from JDBC | Almost none for common operations |
| Implementation Required | Yes (e.g., Hibernate, EclipseLink) | N/A (It is the implementation) | Built on top of Hibernate or JPA |
| Query Language | JPQL | HQL (Hibernate-specific) | Method name-based or custom JPQL |
| Ease of Use | Moderate | Easier than JDBC | Easiest and most productive |
| Repository Support | No | No | Yes (JpaRepository, CrudRepository, etc.) |
| Use Case | Standard data API | Advanced ORM capabilities | Rapid development in Spring applications |

### **✅ Summary:**

* **JPA**: A **standard interface** for data persistence in Java. It needs an implementation.
* **Hibernate**: A **popular implementation of JPA** and a powerful ORM framework.
* **Spring Data JPA**: A **layer on top of JPA and Hibernate** that removes boilerplate code and makes database access extremely easy.

**Mohammed Ayaan**

**Supersetid – 6409789**