**Spring Data JPA – Quick Example**

**Scenario:**

Build a LibraryManagement backend application where:

* It should be able to store books in database.
* Users can add and list all the books.

**Structure of the Project**:

LibraryManagement

├── src/main/java/com.library

│ ├── LibraryManagementApplication.java

│ ├── entity

│ │ └── Book.java

│ ├── repository

│ │ └── BookRepository.java

│ ├── service

│ │ └── BookService.java

│ └── controller

│ └── BookController.java

├── src/main/resources

│ ├── application.properties

└── pom.xml

The Spring Data JPA can be built in below steps.

**Step1**: Setup Spring Boot Project

1. Go to Spring initializer
2. Configure – Project: maven

Language: Java

Spring Boot

1. Add dependencies:

Spring Web

Spring Data JPA

MySQL Driver

1. Download and open in eclipse.

The dependencies to be added in pom.xml are:

|  |
| --- |
| <dependencies>  <!-- Spring Web -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <!-- Spring Data JPA -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>  <!-- MySQL Driver -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-j</artifactId>  <scope>runtime</scope>  </dependency>  <!-- Lombok (optional for cleaner code) -->  <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  <optional>true</optional>  </dependency>  </dependencies> |

**Step2**: Configure Database

Create a file i.e. application.properties

|  |
| --- |
| spring.datasource.url=jdbc:mysql://localhost:3306/librarydb  spring.datasource.username=root  spring.datasource.password=\*\*\*\*  spring.jpa.hibernate.ddl-auto=update  spring.jpa.show-sql=true |

Here,

ddl-auto=update will create tables automatically.

**Step3**: Create Repository i.e. BookRepository.java

|  |
| --- |
| package com.library.repository;  import com.library.entity.Book;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  @Repository  public interface BookRepository extends JpaRepository<Book, Long> {  } |

**Step4**: Create Entity i.e. Book.java

|  |
| --- |
| package com.library.entity;  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)  private Long id;  private String title;  private String author;  // Getters and setters  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; }  public String getAuthor() { return author; }  public void setAuthor(String author) { this.author = author; }  } |

**Step5**: Create a Service i.e. BookService.java

|  |
| --- |
| package com.library.service;  import com.library.entity.Book;  import com.library.repository.BookRepository;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import java.util.List;  @Service  public class BookService {  @Autowired  private BookRepository bookRepository;  public Book saveBook (Book book) {  return bookRepository.save(book);  }  public List<Book> getAllBooks () {  return bookRepository.findAll();  }  } |

**Step6**: Rest Controller is to be created i.e. BookController.java

|  |
| --- |
| package com.library.controller;  import com.library.entity.Book;  import com.library.service.BookService;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.\*;  import java.util.List;  @RestController  @RequestMapping("/api/books")  public class BookController {  @Autowired  private BookService bookService;  @PostMapping  public Book addBook (@RequestBody Book book) {  return bookService.saveBook(book);  }  @GetMapping  public List<Book> getAllBooks() {  return bookService.getAllBooks();  }  } |

**Step7**: Run as Java Application.

To run this application the MainApp.java is to be created.

|  |
| --- |
| package com.library;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  public class MainApp {  public static void main (String[] args) {  SpringApplication.run (MainApp.class, args);  }  } |

**Expected Outcome:**

When the MainApp.java file is in running state then,

The output is as follows,

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

=========|\_|==============|\_\_\_/=/\_/\_/\_/

:: Spring Boot :: (v3.1.0)

2025-07-04 12:30:00 INFO 1234 --- [ main] c.library.MainApp : Starting MainApp using Java 17

2025-07-04 12:30:01 INFO 1234 --- [ main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]

2025-07-04 12:30:01 INFO 1234 --- [ main] o.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect

2025-07-04 12:30:01 INFO 1234 --- [ main] org.hibernate.tool.schema.internal.SchemaUpdate : HHH000228: Running schema update

2025-07-04 12:30:01 INFO 1234 --- [ main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService

2025-07-04 12:30:01 INFO 1234 --- [ main] c.library.MainApp : Started MainApp in 2.345 seconds (JVM running for 2.987)