**Spring Data JPA - Quick Example**

1. **LibraryAppApplication.java**

package com.cognizant.libraryapp;

import com.cognizant.libraryapp.model.Book;

import com.cognizant.libraryapp.service.BookService;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import java.util.List;

@SpringBootApplication

public class LibraryAppApplication {

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(LibraryAppApplication.class, args);

BookService bookService = context.getBean(BookService.class);

List<Book> books = bookService.findAll();

books.forEach(System.out::println);

System.out.println("First Book ID: " + books.get(0).getId());

}

}

**Theory:**

* This is the main entry point of the Spring Boot application.
* @SpringBootApplication is a combination of three annotations:
  + @Configuration – marks this as a source of bean definitions.
  + @EnableAutoConfiguration – auto-configures Spring Beans based on dependencies.
  + @ComponentScan – automatically scans and registers components (like services and repositories) from the package.
* SpringApplication.run(...) bootstraps the application and returns the ApplicationContext.
* We use context.getBean(...) to get the BookService bean managed by Spring's IoC container.
* The service method findAll() is called to retrieve all book records from the database.

1. **Book.java (Entity)**

package com.cognizant.libraryapp.model;

import jakarta.persistence.\*;

import lombok.Data;

@Entity

@Table(name = "books")

@Data

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "title")

private String title;

@Column(name = "author")

private String author;

@Override

public String toString() {

return "Book{" +

"id=" + id +

", title='" + title + '\'' +

", author='" + author + '\'' +

'}';

}

}

**Theory:**

* @Entity marks this class as a JPA entity, mapping it to a database table.
* @Table(name = "books") maps the entity to the books table in the database.
* @Id indicates the primary key of the entity.
* @GeneratedValue(strategy = GenerationType.IDENTITY) automatically generates the primary key using the database’s identity column.
* @Column(name = "title") and @Column(name = "author") map the fields to columns in the table.
* @Data is a Lombok annotation that automatically generates getters, setters, toString, equals, and hashCode methods.

1. **BookRepository.java**

package com.cognizant.libraryapp.repository;

import com.cognizant.libraryapp.model.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

// JpaRepository provides CRUD operations

}

**Theory:**

* This is a Spring Data JPA repository interface.
* BookRepository extends JpaRepository<Book, Long> which gives access to all basic CRUD operations like:
  + findAll(), findById(), save(), deleteById(), etc.
* Spring Boot auto-implements this interface at runtime, so we don’t need to write implementation code.
* No need to annotate it with @Repository — Spring auto-detects it during component scanning.

1. **BookService.java**

package com.cognizant.libraryapp.service;

import com.cognizant.libraryapp.model.Book;

import com.cognizant.libraryapp.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 List<Book> findAll() {

return bookRepository.findAll();

}

}

**Theory:**

* This is a Spring Data JPA repository interface.
* BookRepository extends JpaRepository<Book, Long> which gives access to all basic CRUD operations like:
  + findAll(), findById(), save(), deleteById(), etc.
* Spring Boot auto-implements this interface at runtime, so we don’t need to write implementation code.
* No need to annotate it with @Repository — Spring auto-detects it during component scanning.

**Summary**

1. Spring Boot starts and scans for @Entity, @Repository, and @Service classes.
2. The entity Book is mapped to the books table.
3. The repository provides ready-to-use database operations.
4. The service layer calls the repository method.
5. The main application calls the service to fetch and print book data.