**Module 6 - Spring Data JPA with Spring Boot, Hibernate**

**Spring Data JPA - Quick Example**

**Solution:**

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

<dependencies>

<!-- Spring Boot Starter Data JPA -->

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- H2 Database for testing -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

**Step 2:** **. Application Configuration (application.properties)**

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

**Step 3: Entity Class (Book.java)**

package com.example.demo.model;

import jakarta.persistence.\*;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

// Constructors

public Book() {}

public Book(String title, String author) {

this.title = title;

this.author = author;

}

// Getters & 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; }

}

**Step 4: Repository Interface (BookRepository.java)**

package com.example.demo.repository;

import com.example.demo.model.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

// No need to write code, JPA provides built-in methods like save(), findAll(), findById(), deleteById(), etc.

}

**Step 5: Controller (BookController.java)**

package com.example.demo.controller;

import com.example.demo.model.Book;

import com.example.demo.service.BookService;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

private final BookService service;

public BookController(BookService service) {

this.service = service;

}

@GetMapping

public List<Book> listBooks() {

return service.getAllBooks();

}

@PostMapping

public Book createBook(@RequestBody Book book) {

return service.saveBook(book);

}

@DeleteMapping("/{id}")

public void deleteBook(@PathVariable Long id) {

service.deleteBook(id);

}

}

**Step 6: Main Application Class (DemoApplication.java)**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

**CONCLUSION:**

Sample API Calls using Postman or curl:

* GET /books → List all books
* POST /books → Add a new book (provide JSON like: {"title": "Spring", "author": "Rod Johnson"})
* DELETE /books/{id} → Delete a book by ID