**Exercise 9: Creating a Spring Boot Application**

**Step 1: Create a Spring Boot Project**

Use Spring Initializr to create a new Spring Boot project named LibraryManagement.

**Step 2: Add Dependencies**

Include dependencies for Spring Web, Spring Data JPA, and H2 Database in the pom.xml file:

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

</dependency>

</dependencies>

**Step 3: Create Application Properties**

Configure database connection properties in application.properties:

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

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

**Step 4: Define Entities and Repositories**

Create Book entity and BookRepository interface:

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

// Getters and setters

}

public interface BookRepository extends JpaRepository<Book, Long> {

}

**Step 5: Create a REST Controller**

Create a BookController class to handle CRUD operations:

@RestController

@RequestMapping("/api/books")

public class BookController {

@Autowired

private BookRepository bookRepository;

@GetMapping

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

@PostMapping

public Book createBook(@RequestBody Book book) {

return bookRepository.save(book);

}

@GetMapping("/{id}")

public Book getBook(@PathVariable Long id) {

return bookRepository.findById(id).orElseThrow();

}

@PutMapping("/{id}")

public Book updateBook(@PathVariable Long id, @RequestBody Book book) {

Book existingBook = bookRepository.findById(id).orElseThrow();

existingBook.setTitle(book.getTitle());

existingBook.setAuthor(book.getAuthor());

return bookRepository.save(existingBook);

}

@DeleteMapping("/{id}")

public void deleteBook(@PathVariable Long id) {

bookRepository.deleteById(id);

}

}

**Step 6: Run the Application**

Run the Spring Boot application and test the REST endpoints using Postman or cURL.

**Output:**

GET /api/books

- HTTP Status: 200 OK

- Response Body:

[

{

"id": 1,

"title": "Book Title 1",

"author": "Author 1"

},

{

"id": 2,

"title": "Book Title 2",

"author": "Author 2"

}

]

POST /api/books

- Request Body:

{

"title": "New Book Title",

"author": "New Author"

}

- HTTP Status: 201 Created

- Response Body:

{

"id": 3,

"title": "New Book Title",

"author": "New Author"

}

GET /api/books/1

- HTTP Status: 200 OK

- Response Body:

{

"id": 1,

"title": "Book Title 1",

"author": "Author 1"

}

PUT /api/books/1

- Request Body:

{

"title": "Updated Book Title",

"author": "Updated Author"

}

- HTTP Status: 200 OK

- Response Body:

{

"id": 1,

"title": "Updated Book Title",

"author": "Updated Author"

}

DELETE /api/books/1

- HTTP Status: 204 No Content