**EXERCISE 1: CONFIGURING A BASIC SPRING APPLICATION**

CONTEXT OF THE EXAMPLE

This exercise simulates a basic library management backend where a BookService retrieves book information from a BookRepository. It demonstrates how Spring Core uses XML-based configuration to manage and inject dependencies between service and repository components.

CODES:

pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.dinesh</groupId>  
 <artifactId>LibraryManagment</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>EX1</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

src/main/resources/applicationContext.xml

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <bean id="bookRepository" class="com.dinesh.EX1.repository.BookRepository"/>  
 <bean id="bookService" class="com.dinesh.EX1.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
  
</beans>

BookService.java

package com.dinesh.EX1.service;  
import com.dinesh.EX1.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void displayBook() {  
 System.*out*.println("Book: " + bookRepository.getBook());  
 }  
}

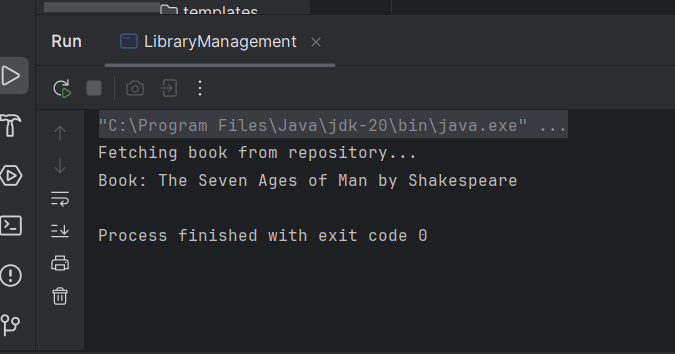
BookRepository.java

package com.dinesh.EX1.repository;  
  
public class BookRepository {  
 public String getBook() {  
 System.*out*.println("Fetching book from repository...");  
 return "The Seven Ages of Man by Shakespeare";  
 }  
}

LibraryManagement.java

package com.dinesh.EX1;  
  
import com.dinesh.EX1.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryManagement {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService service = context.getBean("bookService", BookService.class);  
 service.displayBook();  
 }  
}

OUTPUT



**EXERCISE 2: IMPLEMENTING DEPENDENCY INJECTION**

CONTEXT OF THE EXAMPLE:

This exercise demonstrates Spring’s IoC container injecting the BookRepository dependency into BookService using setter-based XML configuration. It verifies that dependencies can be managed declaratively, without manual object creation.

CODES:

pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.dinesh</groupId>  
 <artifactId>LibraryManagment</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>EX1</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

src/main/resources/applicationContext.xml

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <!-- Bean for Repository -->  
 <bean id="bookRepository" class="com.dinesh.EX1.repository.BookRepository" />  
  
 <!-- Bean for Service with dependency injected -->  
 <bean id="bookService" class="com.dinesh.EX1.service.BookService">  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

BookService.java

package com.dinesh.EX1.service;  
import com.dinesh.EX1.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
 public void displayBook() {  
 System.*out*.println("Please wait while we are Fetching your book from repository...");  
 System.*out*.println("Book: " + bookRepository.getBook());  
 }  
}

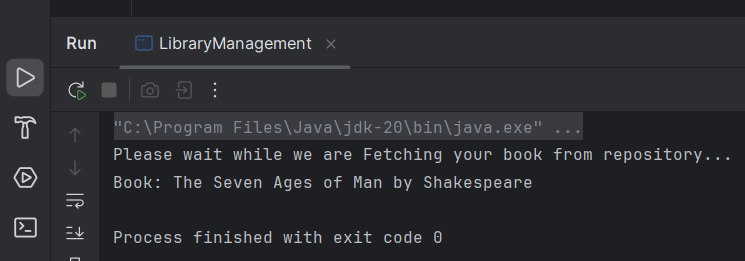
BookRepository.java

package com.dinesh.EX1.repository;  
  
public class BookRepository {  
 public String getBook() {  
 return "The Seven Ages of Man by Shakespeare";  
 }  
}

LibraryManagement.java

package com.dinesh.EX1;  
  
import com.dinesh.EX1.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryManagement {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService service = context.getBean("bookService", BookService.class);  
 service.displayBook();  
 }  
}

OUTPUT:



**EXERCISE 4: CONFIGURING THE SPRING IOC CONTAINER**

CONTEXT OF THE EXAMPLE:

In this exercise, a Maven project is set up in IntelliJ IDE for the Library Management application. Spring Core dependencies such as Spring Context, AOP, and WebMVC were added in the pom.xml, and the Maven Compiler Plugin was configured for Java 1.8 to ensure proper project build and compatibility.

CODES:

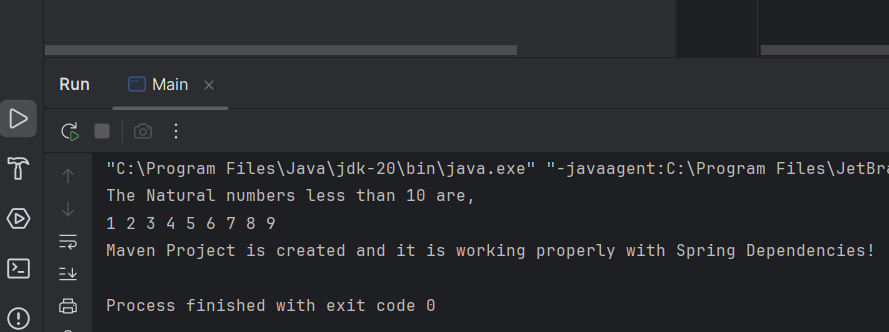
pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  
 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>com.dinesh.library</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <dependencies>  
 <!-- Spring Context -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.32</version>  
 </dependency>  
  
 <!-- Spring AOP -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.32</version>  
 </dependency>  
  
 <!-- Spring WebMVC -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.32</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <!-- Maven Compiler Plugin -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.1</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

Main.java

package com.dinesh;  
  
public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("The Natural numbers less than 10 are,");  
 for(int i=1;i<10;i++){  
 System.*out*.print(i+" ");  
 }  
 System.*out*.println();  
 System.*out*.println("Maven Project is created and it is working properly with Spring Dependencies!");  
 }  
}

OUTPUT:



**EXERCISE 5: CONFIGURING THE SPRING IOC CONTAINER**

CONTEXT OF THE EXAMPLE:

This exercise focuses on configuring Spring's IoC container using an XML file. It centralizes the management of beans (BookService, BookRepository) and handles their wiring through dependency injection using setter methods.

CODES:

pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.dinesh</groupId>  
 <artifactId>LibraryManagment</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>EX1</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

src/main/resources/applicationContext.xml

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <!-- Bean for Repository -->  
 <bean id="bookRepository" class="com.dinesh.EX1.repository.BookRepository" />  
  
 <!-- Bean for Service with Setter Injection -->  
 <bean id="bookService" class="com.dinesh.EX1.service.BookService">  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

BookService.java

package com.dinesh.EX1.service;  
import com.dinesh.EX1.repository.BookRepository;  
public class BookService {  
 private BookRepository bookRepository;  
  
 // Setter Injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void displayBook() {  
 System.*out*.println("Please wait we are retrieving book details...");  
 System.*out*.println("BOOK: " + bookRepository.getBook());  
 }  
}

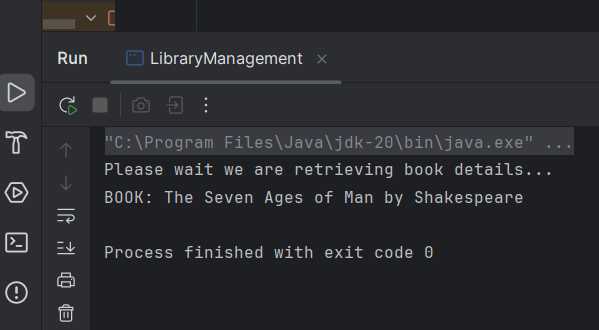
BookRepository.java

package com.dinesh.EX1.repository;  
  
public class BookRepository {  
 public String getBook() {  
 return "The Seven Ages of Man by Shakespeare";  
 }  
}

LibraryManagement.java

package com.dinesh.EX1;  
  
import com.dinesh.EX1.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryManagement {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService service = context.getBean("bookService", BookService.class);  
 service.displayBook();  
 }  
}

OUTPUT:



**EXERCISE 7: IMPLEMENTING CONSTRUCTOR AND SETTER INJECTION**

CONTEXT OF THE EXAMPLE:

This exercise demonstrates how Spring can inject dependencies using both constructor-based and setter-based injection. It gives developers more control over bean initialization.

CODES:

pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.dinesh</groupId>  
 <artifactId>LibraryManagment</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>EX1</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

src/main/resources/applicationContext.xml

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <!-- Bean for Repository -->  
 <bean id="bookRepository" class="com.dinesh.EX1.repository.BookRepository" />  
  
 <!-- Bean for Service using BOTH constructor and setter injection -->  
 <bean id="bookService" class="com.dinesh.EX1.service.BookService">  
 <!-- Constructor injection -->  
 <constructor-arg value="Our College Central Library\*\*!!!" />  
  
 <!-- Setter injection -->  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

BookService.java

package com.dinesh.EX1.service;  
  
import com.dinesh.EX1.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
 private String libraryName; // For constructor injection  
  
 // Constructor for constructor injection  
 public BookService(String libraryName) {  
 this.libraryName = libraryName;  
 }  
  
 // Setter for setter injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void displayBook() {  
 System.*out*.println("!!!\*\*Welcome to " + libraryName);  
 System.*out*.println("You were Searching for->>" + bookRepository.getBook());  
 }  
}

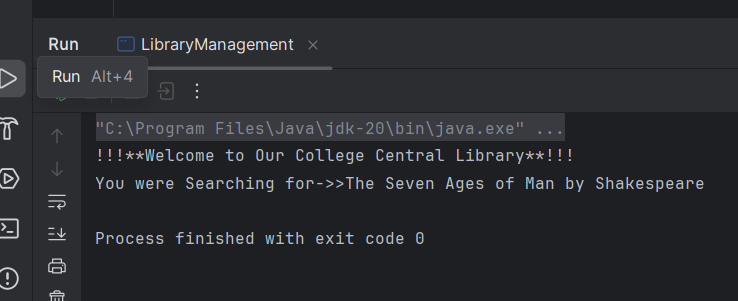
BookRepository.java

package com.dinesh.EX1.repository;  
  
public class BookRepository {  
 public String getBook() {  
 return "The Seven Ages of Man by Shakespeare";  
 }  
}

LibraryManagement.java

package com.dinesh.EX1;  
  
import com.dinesh.EX1.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryManagement {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService service = context.getBean("bookService", BookService.class);  
 service.displayBook();  
 }  
}

OUTPUT:



**EXERCISE 9: CREATING A SPRING BOOT APPLICATION**

CONTEXT OF THE EXAMPLE:

This Spring Boot application is designed to manage a digital library system, allowing users to perform CRUD operations on a collection of books. It demonstrates how to build RESTful APIs using Spring Data JPA with an in-memory H2 database for storing book information like title, author, price, and ID.

CODES:

pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.dinesh</groupId>  
 <artifactId>Library-Management</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>Library-Management</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

application.properties

# H2 Database Configuration  
spring.datasource.url=jdbc:h2:mem:librarydb  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
  
# JPA Settings  
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect  
spring.jpa.hibernate.ddl-auto=update  
spring.h2.console.enabled=true  
spring.jpa.defer-datasource-initialization=true

data.sql

INSERT INTO BOOK (id, title, author, price) VALUES (101, 'The Complete Works of Shakespeare', 'William Shakespeare', 899.00);  
INSERT INTO BOOK (id, title, author, price) VALUES (102, 'Sonnets of Bharathiyar', 'Subramania Bharathiyar', 299.00);  
INSERT INTO BOOK (id, title, author, price) VALUES (103, 'Poems of Bharathidasan', 'Bharathidasan', 250.00);  
INSERT INTO BOOK (id, title, author, price) VALUES (104, 'You Can Win', 'Dr. Sailendra Babu', 499.00);  
INSERT INTO BOOK (id, title, author, price) VALUES (105, 'The Road Not Taken', 'Robert Frost', 199.00);

Book.java

package com.dinesh.Library.Management.model;  
  
import jakarta.persistence.\*;  
  
@Entity  
public class Book {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String title;  
 private String author;  
 private double price;  
  
 public Book() {}  
  
 public Book(String title, String author, double price) {  
 this.title = title;  
 this.author = author;  
 this.price = price;  
 }  
  
 // Getters and Setters  
 public Long getId() {  
 return 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;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
}

BookController.java

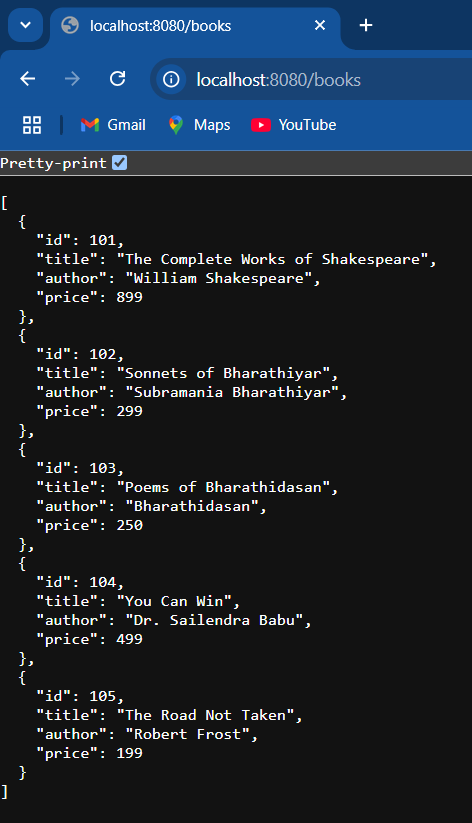
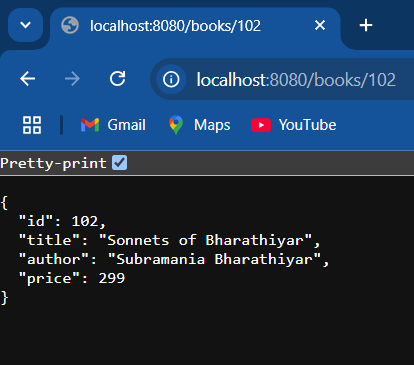
package com.dinesh.Library.Management.controller;  
  
import com.dinesh.Library.Management.model.Book;  
import com.dinesh.Library.Management.repository.BookRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
import java.util.Optional;  
  
@RestController  
@RequestMapping("/books")  
public class BookController {  
  
 @Autowired  
 private BookRepository bookRepository;  
  
 // to GET all books  
 @GetMapping  
 public List<Book> getAllBooks() {  
 return bookRepository.findAll();  
 }  
  
 // to GET book by ID  
 @GetMapping("/{id}")  
 public Optional<Book> getBookById(@PathVariable Long id) {  
 return bookRepository.findById(id);  
 }  
  
 // POST - add new book  
 @PostMapping  
 public Book createBook(@RequestBody Book book) {  
 return bookRepository.save(book);  
 }  
  
 // PUT - update book  
 @PutMapping("/{id}")  
 public Book updateBook(@PathVariable Long id, @RequestBody Book bookDetails) {  
 Book book = bookRepository.findById(id).orElseThrow();  
 book.setTitle(bookDetails.getTitle());  
 book.setAuthor(bookDetails.getAuthor());  
 book.setPrice(bookDetails.getPrice());  
 return bookRepository.save(book);  
 }  
  
 // DELETE - delete book  
 @DeleteMapping("/{id}")  
 public String deleteBook(@PathVariable Long id) {  
 bookRepository.deleteById(id);  
 return "Book deleted with ID: " + id;  
 }  
}

BookRepository.java

package com.dinesh.Library.Management;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class LibraryManagementApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(LibraryManagementApplication.class, args);  
 }  
  
}

LibraryManagementApplication.java

package com.dinesh.Library.Management;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class LibraryManagementApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(LibraryManagementApplication.class, args);  
 }  
  
}

OUTPUT:

**Fetch Data Based On Book-Id**

**Fetch all the provided data**