**Spring Core and Maven**

**Configuring a basic Spring Application**

LibraryApp.java

package com.library.repository;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryApp {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

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

bookService.listBooks();

}

}

BookRepository.java

package com.library.repository;

public class BookRepository {

public void displayBooks() {

System.***out***.println("Fetching books from repository...");

}

}

BookService.java

ackage com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void listBooks() {

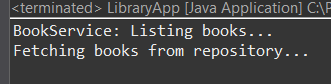
System.***out***.println("BookService: Listing books...");

bookRepository.displayBooks();

}

}

Output



**Implementing Dependency Injection**

Modify the XML Configuration:

<?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">

<!-- Repository Bean -->

<bean id="bookRepository" class="com.library.repository.BookRepository" />

<!-- Service Bean with Dependency Injection -->

<bean id="bookService" class="com.library.service.BookService">

<property name="bookRepository" ref="bookRepository" />

</bean>

</beans>

Update the BookService Class: Ensure that BookService class has a setter method for BookRepository.

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void listBooks() {

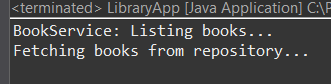
System.***out***.println("BookService: Listing books...");

bookRepository.displayBooks();

}

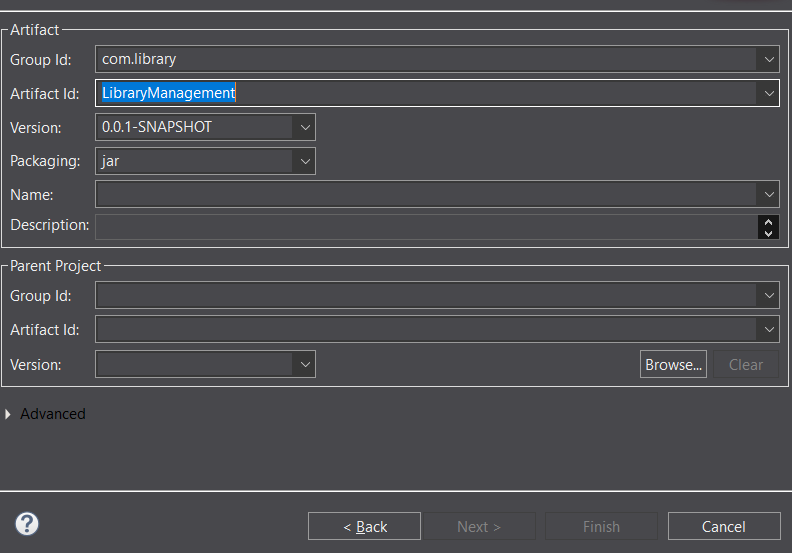
}

Run the LibraryManagementApplication main class to verify the dependency injection.



**Creating and Configuring a Maven Project**

**Create a New Maven Project:**



**Add Spring Dependencies in pom.xml:**

<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>

Configure Maven Compiler Plugin for Java 1.8

<build>

<plugins>

<!-- Maven Compiler Plugin -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

**Configuring the Spring IoC Container**

**Create Spring Configuration File:**

<?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">

<!-- Repository Bean -->

<bean id="bookRepository" class="com.library.repository.BookRepository" />

<!-- Service Bean with Dependency Injection -->

<bean id="bookService" class="com.library.service.BookService">

<property name="bookRepository" ref="bookRepository" />

</bean>

</beans>

**Update the BookService Class:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void listBooks() {

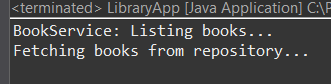
System.***out***.println("BookService: Listing books...");

bookRepository.displayBooks();

}

}

**Run the Application:**



**Implementing Constructor and Setter Injection**

**Configure Constructor Injection:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

private String libraryName;

public BookService(String libraryName) {

this.libraryName = libraryName;

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void listBooks() {

System.***out***.println("Welcome to " + libraryName + " Library");

System.***out***.println("BookService: Listing books...");

bookRepository.displayBooks();

}

}

**Configure Setter Injection:**

<?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">

<!-- BookRepository Bean -->

<bean id="bookRepository" class="com.library.repository.BookRepository" />

<!-- BookService Bean with Constructor and Setter Injection -->

<bean id="bookService" class="com.library.service.BookService">

<!-- Constructor injection -->

<constructor-arg value="City Central" />

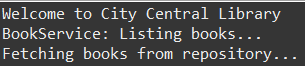
<!-- Setter injection -->

<property name="bookRepository" ref="bookRepository" />

</bean>

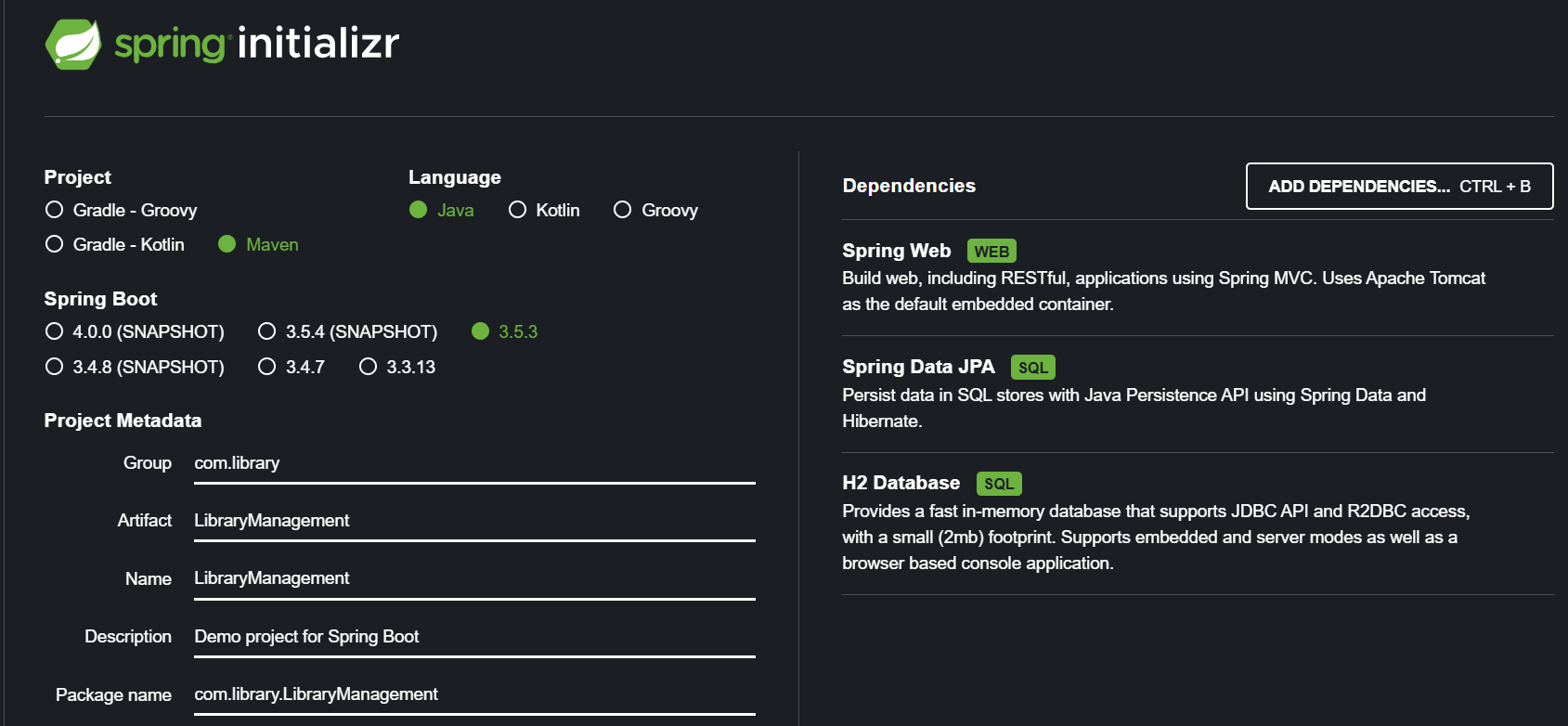
</beans>

**Test the Injection:**



**Creating a Spring Boot Application**

**Create a Spring Boot Project:**



**Create Application Properties:**

spring.application.name=LibraryManagement

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

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

spring.jpa.hibernate.ddl-auto=update

**Define Entities and Repositories:**

**Book.java**

package com.library.model;

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;

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; }

}

**BookRepository.java**

package com.library.repository;

import com.library.model.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**Create a REST Controller**

package com.library.controller;

import com.library.model.Book;

import com.library.repository.BookRepository;

import org.springframework.beans.factory.annotation.Autowired;

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

import java.util.List;

*@RestController*

*@RequestMapping*("/books")

public class BookController {

*@Autowired*

private BookRepository bookRepository;

*@PostMapping*

public Book createBook(*@RequestBody* Book book) {

return bookRepository.save(book);

}

*@GetMapping*

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

// Read one

*@GetMapping*("/{id}")

public Book getBookById(*@PathVariable* Long id) {

return bookRepository.findById(id).orElse(null);

}

*@PutMapping*("/{id}")

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

Book book = bookRepository.findById(id).orElse(null);

if (book != null) {

book.setTitle(bookDetails.getTitle());

book.setAuthor(bookDetails.getAuthor());

return bookRepository.save(book);

}

return null;

}

*@DeleteMapping*("/{id}")

public void deleteBook(*@PathVariable* Long id) {

bookRepository.deleteById(id);

}

}

**Run the Application:**

