**Exercise 1: Configuring a Basic Spring Application**

**Pom.xml**:

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version> <!-- Or latest stable version -->

</dependency>

</dependencies>

**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>">

<!-- Repository Bean -->

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

<!-- Service Bean with Dependency -->

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

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

</bean>

</beans>

**BookRepository.java:**

**package** com.library.repository;

**public** **class** BookRepository {

**public** **void** saveBook(String title) {

System.***out***.println("Book saved: " + title);

}

}

**BookService.java:**

**package** com.library.service;

**import** com.library.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Setter method for Spring

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String title) {

System.***out***.println("Adding book: " + title);

bookRepository.saveBook(title);

}

}

**MainApp.java:**

**package** com.library.test;

**import** com.library.service.BookService;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** MainApp {

**public** **static** **void** main(String[] args) {

// Load Spring context from XML

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

// Get BookService bean

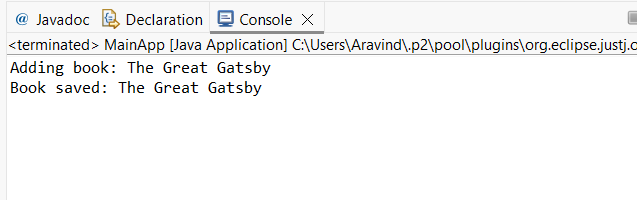
BookService bookService = context.getBean("bookService", BookService.**class**);

bookService.addBook("The Great Gatsby");

}

}

**OUTPUT:**

****

**Exercise 2: Implementing Dependency Injection**

Modify the XML Configuration

**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

https://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- Bean for BookRepository -->

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

<!-- Bean for BookService with DI -->

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

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

</bean>

</beans>

**BookService.java:**

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 displayBookTitle() {

String title = bookRepository.getBookTitle();

System.out.println("Book Title: " + title);

}

}

**MainApp.java**:

package com.library.main;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

// Load Spring context from XML

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

// Get the BookService bean

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

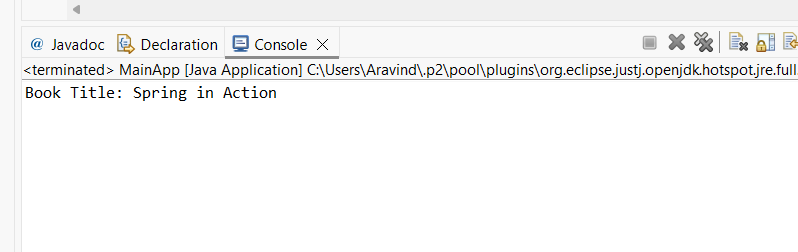
// Use the service method

bookService.displayBookTitle();

}

}

**OUTPUT:**

****

**Exercise 4: Creating and Configuring a Maven Project**

Step 1: Create a New Maven Project

Set the following:

* **Group ID:** com.library
* **Artifact ID:** LibraryManagement

Step 2: Add Spring Dependencies to pom.xml

**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](http://maven.apache.org/POM/4.0.0%20http://maven.apache.org/xsd/maven-4.0.0.xsd)

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Spring Context for IoC/DI -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring WebMVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Servlet API (required for WebMVC) -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

</dependencies>

Step 3: Configure Maven Compiler Plugin

Inside the pom.xml, add the Maven Compiler Plugin to ensure Java 1.8 compatibility.

<build>

<plugins>

<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>

The setup the maven configuration is successfully completed.

Additional Hands-on:

**Exercise 5: Configuring the Spring IoC Container**

**Pom.xml**:

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version> <!-- Or latest stable version -->

</dependency>

</dependencies>

**BookRepository.java**

package com.library.repository;

public class BookRepository {

public String getBookTitle() {

return "Clean Code by Robert C. Martin";

}

}

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for Spring DI

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayBookTitle() {

System.out.println("Book retrieved from repository: " + bookRepository.getBookTitle());

}

}

**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

https://www.springframework.org/schema/beans/spring-beans.xsd">

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

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

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

</bean>

</beans>

**MainApp.java**

package com.library.main;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

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

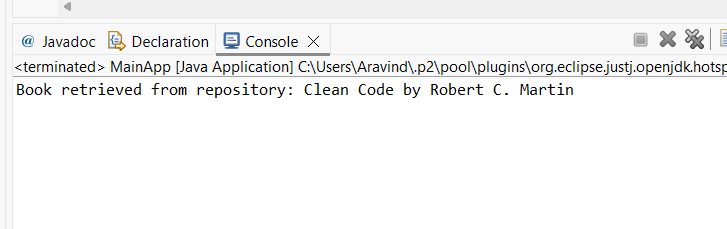
BookService bookService = context.getBean("bookService", BookService.class);

bookService.displayBookTitle();

}

}

**OUTPUT:**



**Exercise 7: Implementing Constructor and Setter Injection**

**Pom.xml**:

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version> <!-- Or latest stable version -->

</dependency>

</dependencies>

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

private String infoMessage; // Injected via constructor

// Constructor for Constructor Injection

public BookService(String infoMessage) {

this.infoMessage = infoMessage;

}

// Setter for Setter Injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayBookInfo() {

System.out.println("Message: " + infoMessage);

System.out.println("Book retrieved from repository: " + bookRepository.getBookTitle());

}

}

**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/beanshttps://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- Bean for BookRepository -->

<beanid="bookRepositoryclass="com.library.repository.BookRepository" />

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

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

<!-- Constructor injection -->

<constructor-arg value="Welcome to the Library Management System!" />

<!-- Setter injection -->

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

</bean> </beans>

**BookRepository.java:**

package com.library.repository;

public class BookRepository {

public String getBookTitle() {

return "Effective Java by Joshua Bloch";

}

}

**MainApp.java:**

package com.library.main;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

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

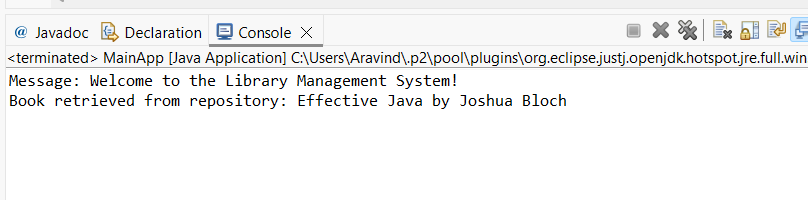
BookService bookService = context.getBean("bookService", BookService.class);

bookService.displayBookInfo();

}

}

**OUTPUT:**



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

pom.xml:

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

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

<dependency>

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

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

</dependency>

<!-- H2 Database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

**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=

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

# H2 Console (enabled)

spring.h2.console.enabled=true

# Hibernate auto schema update

spring.jpa.hibernate.ddl-auto=update

**Book.java:**

package com.library.entity;

import jakarta.persistence.\*;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

private double price;

// Getters and 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; }

public double getPrice() { return price; }

public void setPrice(double price) { this.price = price; }

}

**BookRepository.java**

package com.library.repository;

import com.library.entity.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**BookController.java**

package com.library.controller;

import com.library.entity.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("/api/books")

public class BookController {

@Autowired

private BookRepository bookRepository;

// Get all books

@GetMapping

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

// Get book by ID

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

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

}

// Add a new book

@PostMapping

public Book addBook(@RequestBody Book book) {

return bookRepository.save(book);

}

// Update existing book

@PutMapping("/{id}")

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

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

if (existing != null) {

existing.setTitle(book.getTitle());

existing.setAuthor(book.getAuthor());

existing.setPrice(book.getPrice());

return bookRepository.save(existing);

}

return null;

}

// Delete a book

@DeleteMapping("/{id}")

public void deleteBook(@PathVariable Long id) {

bookRepository.deleteById(id);

}

}

**LibraryManagementApplication.java**

package com.library;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LibraryManagementApplication {

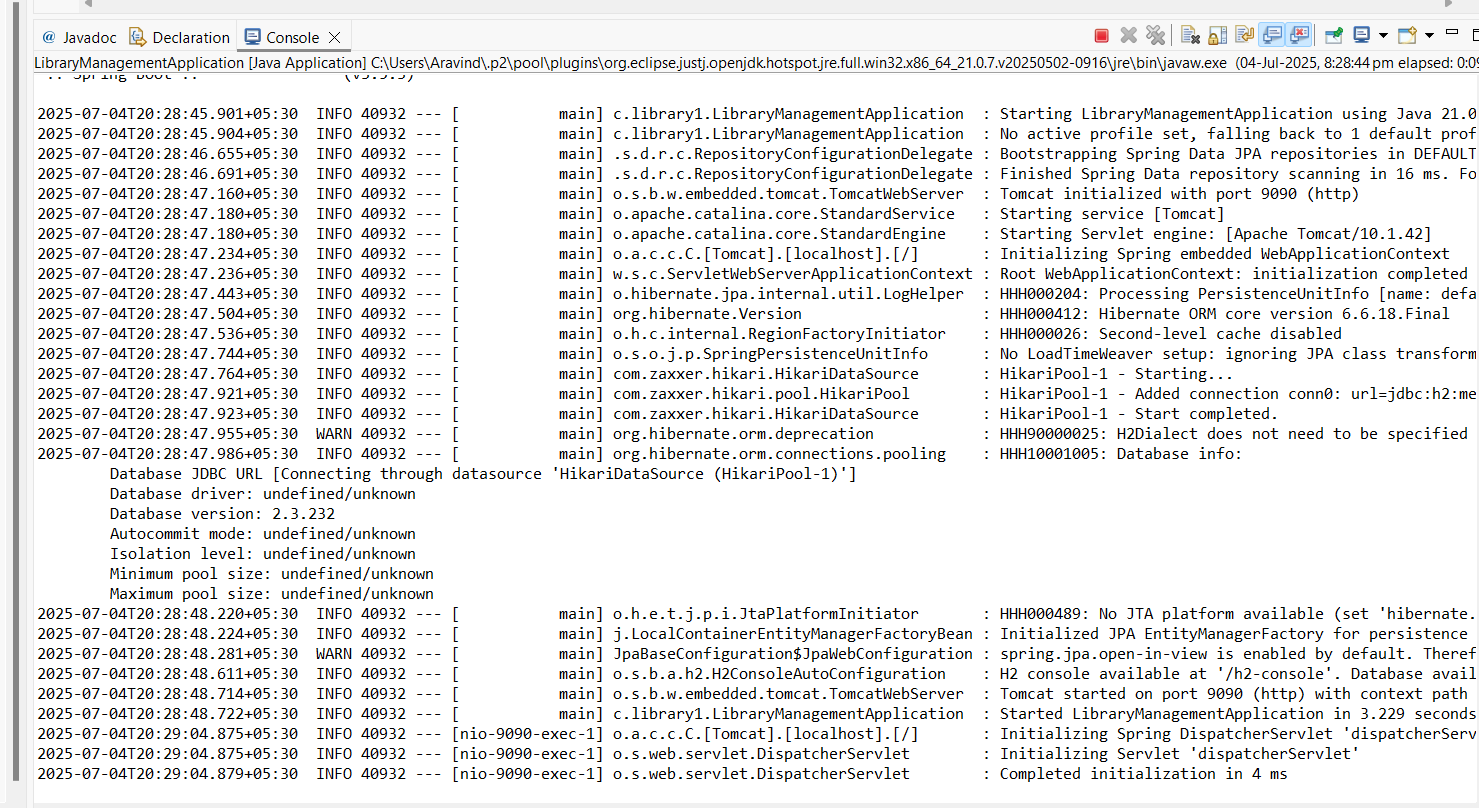
public static void main(String[] args) {

SpringApplication.run(LibraryManagementApplication.class, args);

}

}

**OUTPUT:**

****

****