Exercise 1,2,5,7 : **Configuring a Basic Spring Application & Dependency Injection ,bean allocation.. spring IOC**

**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.library</groupId>  
 <artifactId>SPRING1</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 </properties>  
  
 <repositories>  
 <repository>  
 <id>central</id>  
 <url>https://repo1.maven.org/maven2</url>  
 </repository>  
 </repositories>  
  
 <dependencies>  
 <!-- Spring Core Dependencies -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-beans</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jcl</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-expression</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
 </dependencies>  
  
</project>

MainAPP.JAVA

package com.library;  
  
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 = (BookService) context.getBean("bookService");  
 bookService.addBook("The Harry Potter");  
 }  
}

BookService

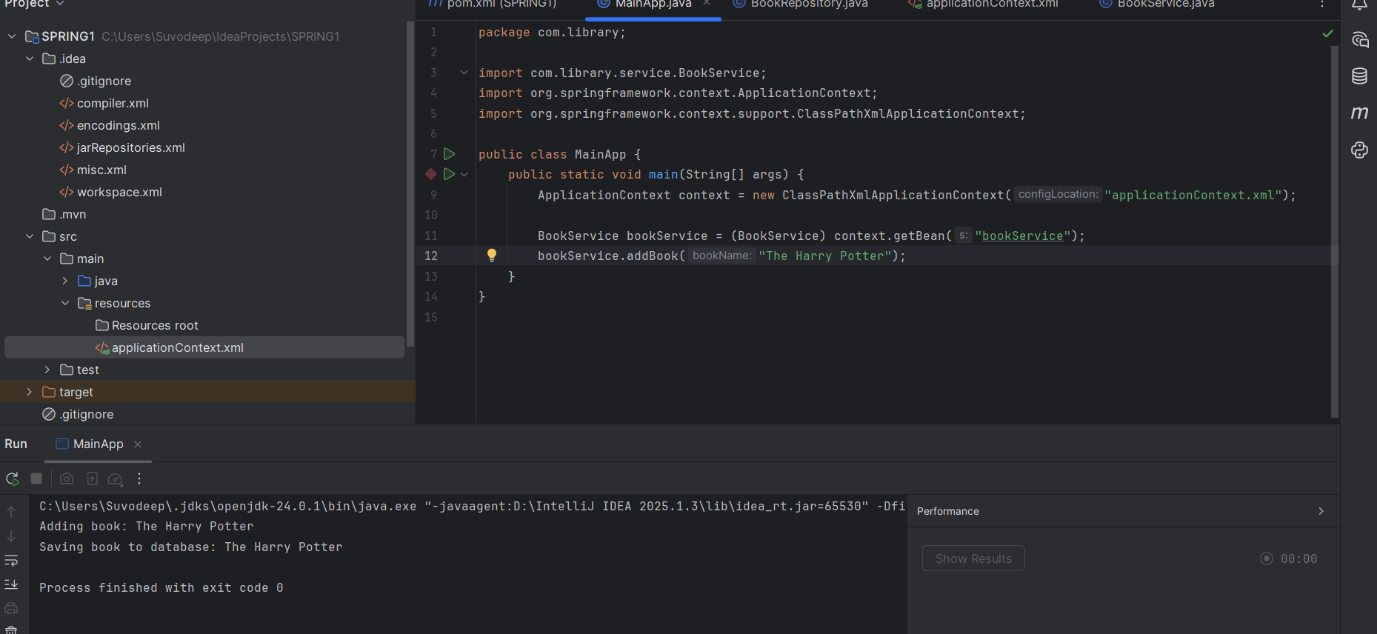
package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 // Setter injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook(String bookName) {  
 System.*out*.println("Adding book: " + bookName);  
 bookRepository.saveBook(bookName);  
 }  
}

Book Repo

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook(String bookName) {  
 System.*out*.println("Saving book to database: " + bookName);  
 }  
}

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.library.repository.BookRepository" />  
  
   
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
  
</beans>



Exercise 4

<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.library</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <java.version>1.8</java.version>  
 </properties>  
  
 <repositories>  
 <repository>  
 <id>maven-central</id>  
 <name>Maven Central</name>  
 <url>https://repo1.maven.org/maven2</url>  
 </repository>  
 </repositories>  
  
  
 <dependencies>  
Spring Context (Core DI features) -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
  
Spring AOP -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
  
Spring WebMVC -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
  
Spring Expression -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-expression</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
  
Spring JCL (logging) -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jcl</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
 </dependencies>  
  
Maven Compiler Plugin -->  
 <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>

EXERCISE 9

LibraryManagement

package com.library;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class LibraryManagement2Application {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(LibraryManagement2Application.class, args);  
 }  
  
}

application props

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

Entity book

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;  
  
 public Book() {}  
  
 public Book(String title, String author) {  
 this.title = title;  
 this.author = author;  
 }  
  
 // 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; }  
}

book repo

package com.library.repository;  
  
import com.library.entity.Book;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface BookRepository extends JpaRepository<Book, Long> {  
}

book controller

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("/books")  
public class BookController {  
  
 @Autowired  
 private BookRepository bookRepository;  
  
 @PostMapping  
 public Book addBook(@RequestBody Book book) {  
 return bookRepository.save(book);  
 }  
  
 @GetMapping  
 public List<Book> getAllBooks() {  
 return bookRepository.findAll();  
 }  
  
 @GetMapping("/{id}")  
 public Book getBookById(@PathVariable Long id) {  
 return bookRepository.findById(id).orElse(null);  
 }  
  
 @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());  
 return bookRepository.save(existing);  
 }  
 return null;  
 }  
  
 @DeleteMapping("/{id}")  
 public void deleteBook(@PathVariable Long id) {  
 bookRepository.deleteById(id);  
 }

}  
