Mandatory\_handsOn\_Spring\_Maven

# Exercise 1 & 2: Configuring a Basic Spring Application with Dependency Injection

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 https://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>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
 </dependencies>  
</project>

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>

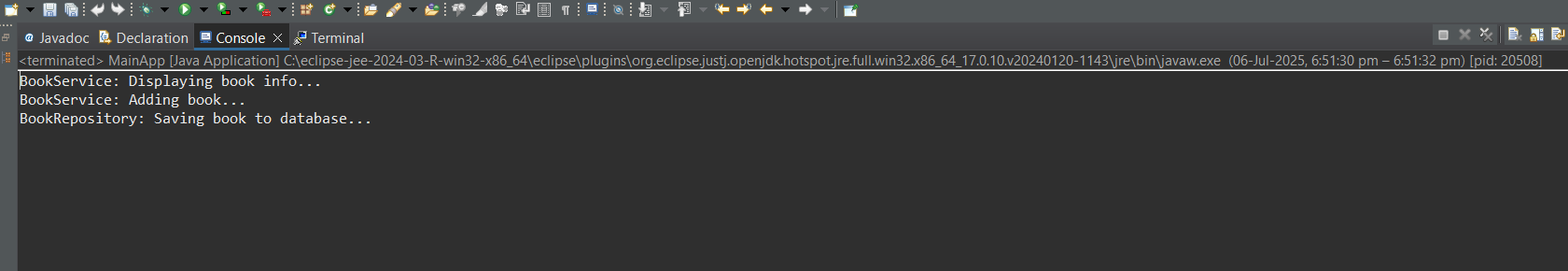
BookRepository.java

package com.library.repository;  
  
public class BookRepository {  
 public void save() {  
 System.out.println("BookRepository: Saving book to database...");  
 }  
}

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 addBook() {  
 System.out.println("BookService: Adding book...");  
 bookRepository.save();  
 }  
}

MainApp.java

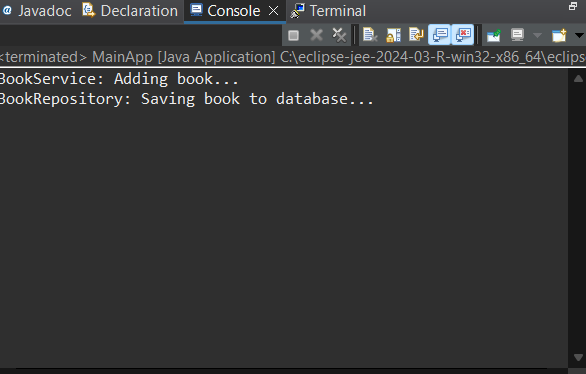
package com.library;  
  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import com.library.service.BookService;  
  
public class MainApp {  
 public static void main(String[] args) {  
 ApplicationContext context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

# Exercise 4: Creating and Configuring a Maven Project

Already included above with pom.xml setup and Java 1.8 compiler plugin.

# Exercise 5: Configuring the Spring IoC Container

No code changed. I Used the same applicationContext.xml and setter injection setup from Exercises 1 and 2.



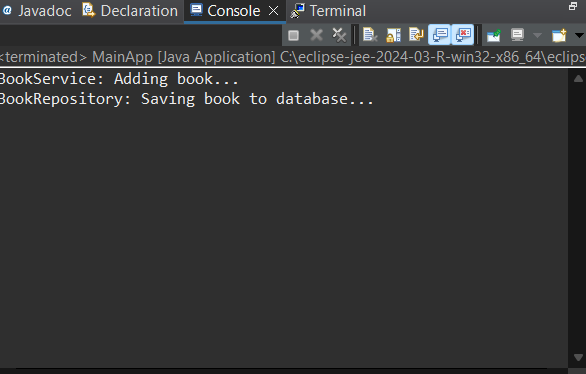
# Exercise 7: Implementing Constructor and Setter Injection

Modified BookService.java

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

Updated applicationContext.xml

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



# Exercise 9: Creating a Spring Boot Application

Main Application

@SpringBootApplication  
public class LibraryManagementApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(LibraryManagementApplication.class, args);  
 }  
}

Book.java

@Entity  
public class Book {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
 private String title;  
 private String author;  
 // getters and setters  
}

BookRepository.java

public interface BookRepository extends JpaRepository<Book, Long> {}

BookController.java

@RestController  
@RequestMapping("/books")  
public class BookController {  
 @Autowired  
 private BookRepository bookRepository;  
  
 @GetMapping  
 public List<Book> getAllBooks() {  
 return bookRepository.findAll();  
 }  
  
 @PostMapping  
 public Book addBook(@RequestBody Book book) {  
 return bookRepository.save(book);  
 }  
}

application.properties

spring.datasource.url=jdbc:h2:mem:librarydb  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
spring.h2.console.enabled=true  
spring.jpa.show-sql=true  
spring.jpa.hibernate.ddl-auto=update