**Superset Id:** **6376594**

**Spring Core And Maven(Hands-On)**

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

Project Name:LibraryManagement

applicationContent.xml Code:

<?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 -->

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

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

</bean>

</beans>

BookRepository.java Code:

**package** com.library.repository;

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

**public** **void** save(String bookName) {

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

}

}

BookService.java Code:

**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(String bookName) {

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

bookRepository.save(bookName);

}

}

App.java Code:

**package** com.library.LibraryManagement;

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

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

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

**public** **class** App {

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

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

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

bookService.addBook("The Alchemist");

}

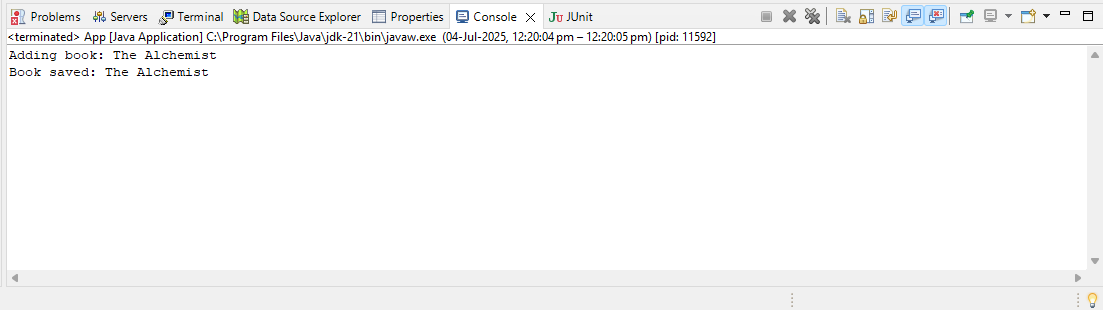
}

Excepted Output:

Adding book: The Alchemist

Book saved: The Alchemist

Output:



**Exercise 2: Implementing Dependency Injection**

Project Name:LibraryManagement

applicationContent.xml Code:

<?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"*>

<!-- Book Repository Bean -->

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

<!-- Book Service Bean with DI -->

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

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

</bean>

</beans>

BookRepository.java Code:

**package** com.library.repository;

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

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

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

}

}

BookService.java Code:

**package** com.library.service;

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

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

**private** BookRepository bookRepository;

// ✅ Setter for DI

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

**this**.bookRepository = bookRepository;

}

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

bookRepository.saveBook(book);

}

}

App.java Code:

**package** com.library.LibraryManagement;

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

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

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

**public** **class** App {

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

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

// ✅ Bean ID is "bookService" (lowercase)

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

service.addBook("The Alchemist");

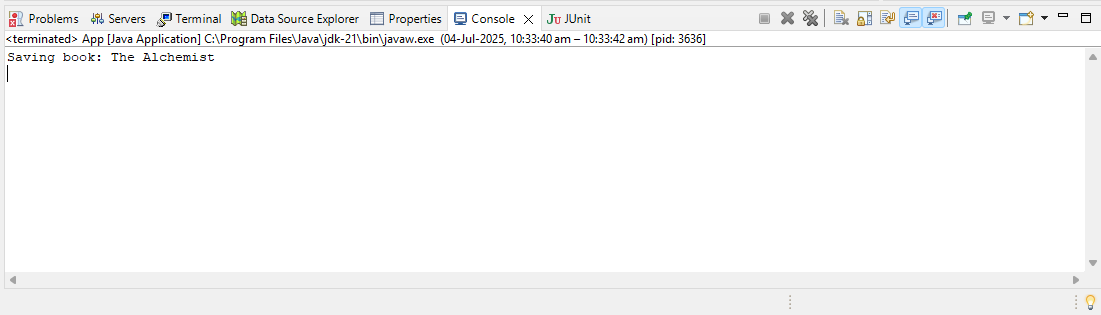
}

}

Expected Output:

Saving book: The Alchemist

Output:



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

Project Name:LibraryManagement

applicationContent.xml Code:

<?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"*>

<!-- Book Repository Bean -->

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

<!-- Book Service Bean -->

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

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

</bean>

</beans>

BookRepository.java Code:

**package** com.library.repository;

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

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

System.***out***.println("Saving book: " + bookName);

}

}

BookService.java Code:

**package** com.library.service;

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

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

**private** BookRepository bookRepository;

// Setter for DI

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

**this**.bookRepository = bookRepository;

}

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

bookRepository.saveBook(bookName);

}

}

App.java Code:

**package** com.library.LibraryManagement;

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

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

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

**public** **class** App {

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

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

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

bookService.addBook("The Alchemist");

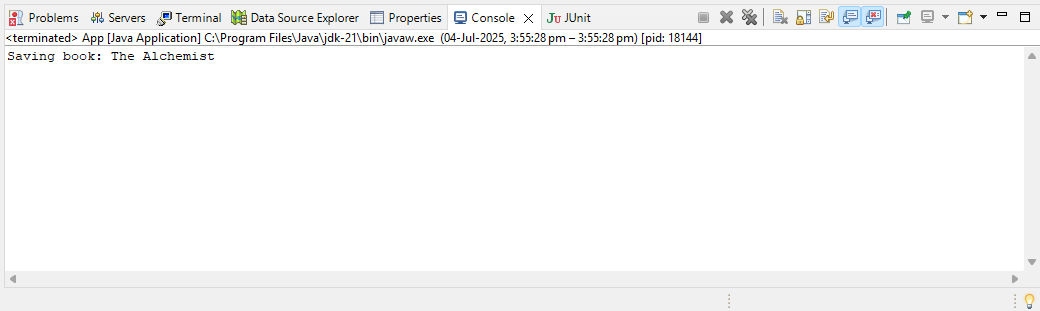
}

}

Expected Output:

Saving book: The Alchemist

Output:



**Additional Hands-On:**

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

Project Name:LibraryManagement

applicationContent.xml Code:

<?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"*>

<!-- Book Repository Bean -->

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

<!-- Book Service Bean with DI -->

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

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

</bean>

</beans>

BookRepository.java Code:

**package** com.library.repository;

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

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

System.***out***.println("Saving book: " + bookName);

}

}

BookService.java Code:

**package** com.library.service;

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

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

**private** BookRepository bookRepository;

// ✅ Setter method for DI

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

**this**.bookRepository = bookRepository;

}

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

bookRepository.saveBook(bookName);

}

}

App.java Code:

**package** com.library.LibraryManagement;

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

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

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

**public** **class** App {

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

// ✅ Load Spring IoC container

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

// ✅ Get bookService bean

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

// ✅ Test it

bookService.addBook("The Alchemist");

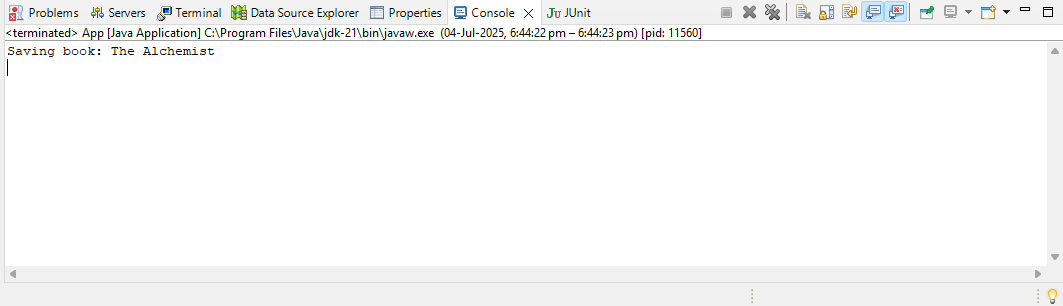
}

}

Excepted Output:

Saving book: The Alchemist

Output:



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

Project Name:LibraryManagement

applicationContent.xml Code:

<?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"*>

<!-- LoggerService Bean (for constructor injection) -->

<bean id=*"loggerService"* class=*"com.library.service.LoggerService"* />

<!-- BookRepository Bean (for setter injection) -->

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

<!-- BookService Bean using constructor and setter injection -->

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

<!-- Constructor injection -->

<constructor-arg ref=*"loggerService"* />

<!-- Setter injection -->

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

</bean>

</beans>

BookRepository.java Code:

**package** com.library.repository;

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

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

System.***out***.println("Saving book: " + bookName);

}

}

BookService.java Code:

**package** com.library.service;

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

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

**private** BookRepository bookRepository; // Setter injection

**private** LoggerService loggerService; // Constructor injection

// ✅ Constructor for LoggerService

**public** BookService(LoggerService loggerService) {

**this**.loggerService = loggerService;

}

// ✅ Setter for BookRepository

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

**this**.bookRepository = bookRepository;

}

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

loggerService.log("Adding a book...");

bookRepository.saveBook(bookName);

}

}

LoggerService.java Code:

**package** com.library.service;

**public** **class** LoggerService {

**public** **void** log(String message) {

System.***out***.println("LOG: " + message);

}

}

App.java Code:

**package** com.library.LibraryManagement;

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

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

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

**public** **class** App {

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

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

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

service.addBook("The Power of Habit");

}

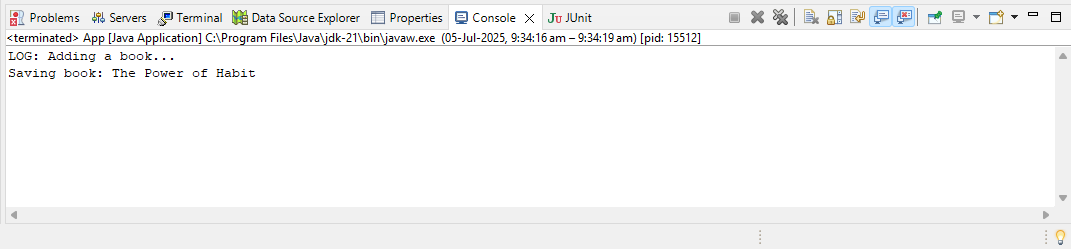
}

Expected Output:

LOG: Adding a book...

Saving book: The Power of Habit

Output:



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

Project Name:LibraryManagementBoot

LibraryManagementBootApplication.java Code:

**package** com.example.LibraryManagementBoot;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.boot.CommandLineRunner;

**import** org.springframework.context.annotation.Bean;

**import** com.example.LibraryManagementBoot.entity.Book;

**import** com.example.LibraryManagementBoot.repository.BookRepository;

@SpringBootApplication

**public** **class** LibraryManagementBootApplication {

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

SpringApplication.*run*(LibraryManagementBootApplication.**class**, args);

}

@Bean

**public** CommandLineRunner addSampleBooks(BookRepository bookRepository) {

**return** args -> {

bookRepository.save(**new** Book("Wings of Fire", "A. P. J. Abdul Kalam"));

bookRepository.save(**new** Book("The Alchemist", "Paulo Coelho"));

bookRepository.save(**new** Book("Ignited Minds", "A. P. J. Abdul Kalam"));

};

}

}

BookController.java Code:

**package** com.example.LibraryManagementBoot.controller;

**import** com.example.LibraryManagementBoot.entity.Book;

**import** com.example.LibraryManagementBoot.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;

@GetMapping

**public** List<Book> getAllBooks() {

**return** bookRepository.findAll();

}

@PostMapping

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

**return** bookRepository.save(book);

}

}

Book.java Code:

**package** com.example.LibraryManagementBoot.entity;

**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** 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; }

}

BookRepository.java Code:

**package** com.example.LibraryManagementBoot.repository;

**import** com.example.LibraryManagementBoot.entity.Book;

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

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

}

application.propeties Code:

# === Database Configuration ===

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

# === Hibernate / JPA Settings ===

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

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

# === H2 Console ===

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

# === Custom Port (to avoid port 8080 conflict) ===

server.port=8081

Output:

