# SPRING\_CORE\_MAVEN

Exercise 1: Conflguring a Basic Spring Application

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.printBookDetails();

}

}

BookService.java

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for dependency injection

public void setBookRepository(BookRepository bookRepository) { this.bookRepository = bookRepository;

}

public void printBookDetails() {

System.out.println("BookService: " + bookRepository.getBookInfo());

}

}

#### BookRepository.java

package com.library.repository;

public class BookRepository { public String getBookInfo() {

return "Book info from repository";

}

}

## output

#### Exercise 2: Implementing Dependency Injection

BookRepository.java

package com.library;

public class BookRepository { public void save() {

System.out.println("BookRepository: Saving book to database...");

}

}

##### BookService.java

package com.library; public class BookService {

private BookRepository bookRepository;

// Setter for Dependency Injection

public void setBookRepository(BookRepository bookRepository) { this.bookRepository = bookRepository;

}

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

}

}

##### LibraryManagementApplication.java

package com.library;

import org.springframework.context.ApplicationContext; import

org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication { public static void main(String[] args) {

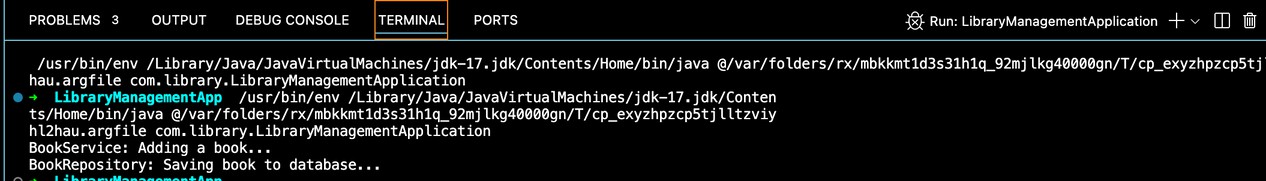
ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

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

bookService.addBook();

}

}

Output

#### Exercise 3: Implementing Logging with Spring AOP

BookRepository.java

package com.library.repository; public class BookRepository {

public void save() { System.out.println("BookRepository: Saving book...");

}

}

### 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();

}

}

LoggingAspect.java

package com.library.aspect;

import org.aspectj.lang.ProceedingJoinPoint; import org.aspectj.lang.annotation.Around; import org.aspectj.lang.annotation.Aspect;

@Aspect

public class LoggingAspect {

@Around("execution(\* com.library.service.\*.\*(..))")

public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {

long start = System.currentTimeMillis();

Object result = joinPoint.proceed(); // execute the method long end = System.currentTimeMillis();

System.out.println("Execution time of " + joinPoint.getSignature().getName() + ": " + (end - start) + " ms");

return result;

}

}

LibraryManagementApplication.java

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext; import

org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication { public static void main(String[] args) {

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

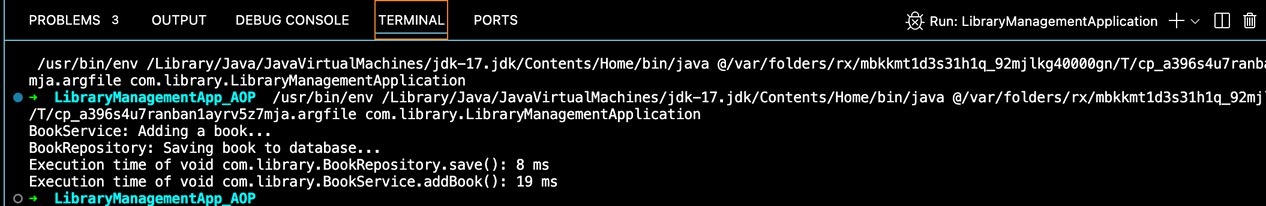
BookService bookService = context.getBean("bookService", BookService.class);

bookService.addBook();

}

}

### output



#### Exercise 4: Creating and Conflguring a Maven Project

BookRepository.java

package com.library.repository; public class BookRepository {

public void save() { System.out.println("BookRepository: Saving book...");

}

}

### 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();

}

}

LibraryManagementApplication.java

package com.library;

import org.springframework.context.ApplicationContext; import

org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication { public static void main(String[] args) {

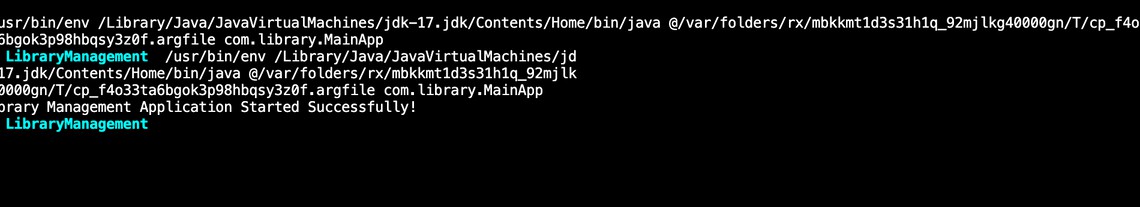
ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

System.out.println("Library Management Application Started Successfully!");

}

}

## output



##### Exercise 5: Conflguring the Spring IoC Container

BookRepository.java

package com.library.repository; public class BookRepository {

public void save() {

System.out.println("BookRepository: Saving book data...");

}

}

BookService.java

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() {

System.out.println("BookService: Adding a book..."); bookRepository.save();

}

}

LibraryManagementApplication.java

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext; import

org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication { public static void main(String[] args) {

ApplicationContext context = new

ClassPathXmlApplicationContext("applicationContext.xml");

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

bookService.addBook();

}

}

output

##### Exercise 6: Conflguring Beans with Annotations

BookService.java

package com.library;

import org.springframework.stereotype.Service; @Service

public class BookService {

private BookRepository bookRepository;

// Use constructor injection (recommended) or setter injection with @Autowired

@org.springframework.beans.factory.annotation.Autowired public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook() {

System.out.println("Adding a new book..."); bookRepository.save();

}

}

BookRepository.java

package com.library;

import org.springframework.stereotype.Repository;

@Repository

public class BookRepository { public void save() {

System.out.println("Saving book to repository...");

}

}

LibraryManagementApplication.java

package com.library;

import org.springframework.context.ApplicationContext; import

org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication { public static void main(String[] args) {

ApplicationContext context =

new ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = context.getBean(BookService.class); bookService.addBook();

}

}

# output

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

BookService.java package com.library; public class BookService {

private BookRepository bookRepository;

// Constructor injection

public BookService(BookRepository bookRepository) { this.bookRepository = bookRepository;

}

// Setter injection

public void setBookRepository(BookRepository bookRepository) { this.bookRepository = bookRepository;

}

public void addBook() {

System.out.println("Adding a new book..."); bookRepository.save();

}

}

LibraryManagementApplication.java

package com.library;

import org.springframework.context.ApplicationContext; import

org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication { public static void main(String[] args) {

ApplicationContext context =

new ClassPathXmlApplicationContext("applicationContext.xml");

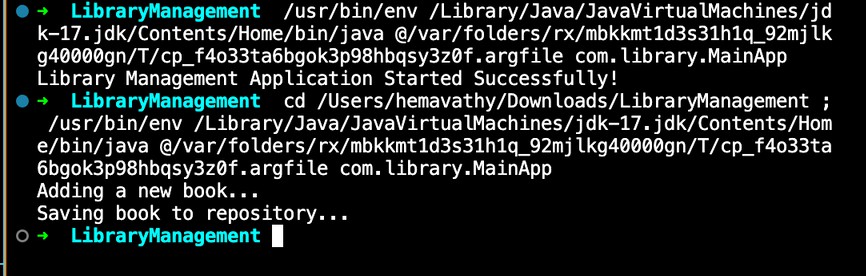
BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook();

}

}

output



Exercise 8: Implementing Basic AOP with Spring

##### LoggingAspect.java

package com.library.aspect;

import org.aspectj.lang.JoinPoint; import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.Aspect; import org.aspectj.lang.annotation.Before;

@Aspect

public class LoggingAspect {

@Before("execution(\* com.library.BookService.\*(..))") public void logBefore(JoinPoint joinPoint) {

System.out.println("[LOG] Before method: " +

joinPoint.getSignature().getName());

}

@After("execution(\* com.library.BookService.\*(..))") public void logAfter(JoinPoint joinPoint) {

System.out.println("[LOG] After method: " +

joinPoint.getSignature().getName());

}

}

LibraryManagementApplication.java

package com.library;

import org.springframework.context.ApplicationContext; import

org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication { public static void main(String[] args) {

ApplicationContext context =

new ClassPathXmlApplicationContext("applicationContext.xml");

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

bookService.addBook();

}

}

BookService.java

package com.library;

public class BookService {

private BookRepository bookRepository;

// Constructor injection

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

// Setter injection

public void setBookRepository(BookRepository bookRepository) { this.bookRepository = bookRepository;

}

public void addBook() {

System.out.println("Adding a new book..."); bookRepository.save();

}

}

# output



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

##### LibraryManagementApplication.java

package com.example.librarymanagement;

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);

}

}

Book.java

package com.example.librarymanagement.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;

// 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

package com.example.librarymanagement.repository;

import com.example.librarymanagement.entity.Book;

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

public interface BookRepository extends JpaRepository<Book, Long>

{

}

BookController.java

package com.example.librarymanagement.controller;

import com.example.librarymanagement.entity.Book;

import com.example.librarymanagement.repository.BookRepository; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.web.bind.annotation.\*;

import java.util.List; import java.util.Optional;

@RestController @RequestMapping("/books") public class BookController {

@Autowired

private BookRepository bookRepository;

@GetMapping

public List<Book> getAllBooks() { return bookRepository.findAll();

}

@GetMapping("/{id}")

public Optional<Book> getBookById(@PathVariable Long id) { return bookRepository.findById(id);

}

@PostMapping

public Book createBook(@RequestBody Book book) { return bookRepository.save(book);

}

@PutMapping("/{id}")

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

Book book = bookRepository.findById(id).orElseThrow();

book.setTitle(bookDetails.getTitle()); book.setAuthor(bookDetails.getAuthor()); return bookRepository.save(book);

}

@DeleteMapping("/{id}")

public void deleteBook(@PathVariable Long id) { bookRepository.deleteById(id);

}

}

output

