**SPRING CORE MAVEN HANDS-ON**

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

**Scenario:**

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

**Code:**

**BookRepository.java  
public** **class** BookRepository {

**public** **void** fetchBook() {

        System.out.println("Fetching book from repository...");

    }

}

**BookService.java**

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

**private** **BookRepository** bookRepository;

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

        this.bookRepository **=** bookRepository;

    }

**public** **void** displayBook() {

        bookRepository.fetchBook();

    }

}

**LibraryManagementApplication.java  
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");

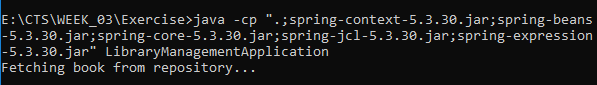
**BookService** bookService **=** context.getBean(BookService.class);

        bookService.displayBook();

    }

}

**Output:**

****

**Exercise 2: Implementing Dependency Injection**

**Scenario:**

In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

**Code:**

**BookRepository.java  
public** **class** BookRepository {

**public** **void** fetchBook() {

        System.out.println("Fetching book from repository...");

    }

}

**BookService.java**

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

**private** **BookRepository** bookRepository;

**public** BookService(**BookRepository** bookRepository) {

        this.bookRepository **=** bookRepository;

    }

**public** **void** displayBook() {

        bookRepository.fetchBook();

    }

}

**LibraryManagementApplication**

**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");

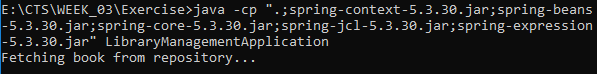
**BookService** bookService **=** context.getBean(BookService.class);

        bookService.displayBook();

    }

}

**Output:**

****

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

**Scenario:**

The library management application requires logging capabilities to track method execution times.

**Code:**

**Appconfig.java**

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

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

@**Configuration**

**public** **class** AppConfig {

    @**Bean**

**public** **BookRepository** bookRepository() {

**return** **new** BookRepository();

    }

    @**Bean**

**public** **BookService** bookService() {

**return** **new** BookService(bookRepository());

    }

}

**BookRepository.java**

**import** **org.springframework.stereotype.Component**;

@**Component**

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

**public** **void** fetchBook() {

        System.out.println("Fetching book from repository (via annotations)...");

    }

}

**BookService.java  
import** **org.springframework.beans.factory.annotation.Autowired**;

**import** **org.springframework.stereotype.Component**;

@**Component**

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

**private** **final** **BookRepository** bookRepository;

    @**Autowired**

**public** BookService(**BookRepository** bookRepository) {

        this.bookRepository **=** bookRepository;

    }

**public** **void** displayBook() {

        bookRepository.fetchBook();

    }

}

**LibraryManagementApplication.java**

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

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

**public** **class** LibraryManagementApplication {

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

**ApplicationContext** context **=** **new** AnnotationConfigApplicationContext(AppConfig.class);

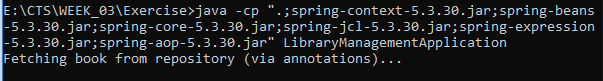
**BookService** bookService **=** context.getBean(BookService.class);

        bookService.displayBook();

    }

}

**Output:**

****

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

**Scenario:**

You need to set up a new Maven project for the library management application and add Spring dependencies.

**Code:**

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

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

@**Configuration**

**public** **class** AppConfig {

    @**Bean**

**public** **BookService** bookService() {

**return** **new** BookService(); *// You can also add scope here*

    }

}

**BookRepository.java**

**import** **org.springframework.stereotype.Component**;

@**Component**

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

**public** **void** fetchBook() {

        System.out.println("Fetching book from repository (via annotations)...");

    }

}

**BookService.java**

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

**public** BookService() {

        System.out.println("BookService instance created: " **+** this);

    }

**public** **void** displayBook() {

        System.out.println("Displaying book from instance: " **+** this);

    }

}

**LibraryManagementApplication.java**

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

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

**public** **class** LibraryManagementApplication {

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

**ApplicationContext** context **=** **new** AnnotationConfigApplicationContext(AppConfig.class);

**BookService** service1 **=** context.getBean(BookService.class);

        service1.displayBook();

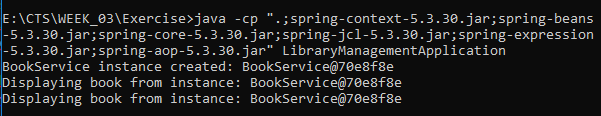
**BookService** service2 **=** context.getBean(BookService.class);

        service2.displayBook();

    }

}

**Output:**

****

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

**Scenario:**

The library management application requires a central configuration for beans and dependencies.

**Code:**

**BookService.java  
public** **class** BookService {

**public** BookService() {

        System.out.println("BookService constructor called");

    }

**public** **void** init() {

        System.out.println("BookService init() method called");

    }

**public** **void** destroy() {

        System.out.println("BookService destroy() method called");

    }

**public** **void** displayBook() {

        System.out.println("Displaying book from BookService");

    }

}

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

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

@**Configuration**

**public** **class** AppConfig {

    @**Bean**(initMethod **=** "init", destroyMethod **=** "destroy")

**public** **BookService** bookService() {

**return** **new** BookService();

    }

}

**LibraryManagementApplication.java**

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

**public** **class** LibraryManagementApplication {

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

**AnnotationConfigApplicationContext** context **=** **new** AnnotationConfigApplicationContext(AppConfig.class);

**BookService** service **=** context.getBean(BookService.class);

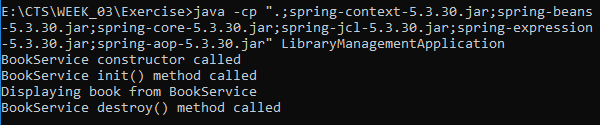
        service.displayBook();

        context.close(); *// 🔁 Required to trigger destroy()*

    }

}

**Output:**

****

**Exercise 6: Configuring Beans with Annotations**

**Scenario:**

You need to simplify the configuration of beans in the library management application using annotations.

**Code:**

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

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

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

@**Configuration**

@**EnableAspectJAutoProxy**

**public** **class** AppConfig {

    @**Bean**

**public** **BookService** bookService() {

**return** **new** BookService();

    }

    @**Bean**

**public** **LoggingAspect** loggingAspect() {

**return** **new** LoggingAspect();

    }

}

**LibraryManagementApplication.java**

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

**public** **class** LibraryManagementApplication {

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

**AnnotationConfigApplicationContext** context **=** **new** AnnotationConfigApplicationContext(AppConfig.class);

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

        bookService.displayBook();

        context.close();

    }

}

**LoggingAspects.java**

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

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

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

@**Aspect**

**public** **class** LoggingAspect {

    @**Before**("execution(\* BookService.displayBook(..))")

**public** **void** logBefore() {

        System.out.println("LOG: Before executing displayBook()");

    }

    @**After**("execution(\* BookService.displayBook(..))")

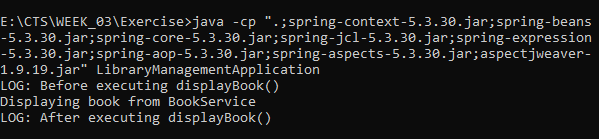
**public** **void** logAfter() {

        System.out.println("LOG: After executing displayBook()");

    }

}

**Output:**

****

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

**Scenario:**

The library management application requires both constructor and setter injection for better control over bean initialization.

**Code:**

**Book.java  
public** **interface** Book {

**void** displayInfo();

}

**BookCollections.java  
import** **java.util.List**;

**import** **org.springframework.beans.factory.annotation.Autowired**;

**public** **class** BookCollectionService {

    @**Autowired**

**private** **List**<**Book**> books;

**public** **void** displayAllBooks() {

        System.out.println("Displaying all books:");

**for** (**Book** book **:** books) {

            book.displayInfo();

        }

    }

}

**BookService.java  
public** **class** BookService {

**public** **void** displayBook() {

        System.out.println("Displaying book from BookService");

    }

}

**FictionBook.java  
public** **class** FictionBook **implements** Book {

**public** **void** displayInfo() {

        System.out.println("Fiction Book: The Great Gatsby");

    }

}

ScienceBook.java  
**public** **class** ScienceBook **implements** Book {

**public** **void** displayInfo() {

        System.out.println("Science Book: A Brief History of Time");

    }

}

**LibraryManagementApplication.java  
import** **org.springframework.context.annotation.AnnotationConfigApplicationContext**;

**public** **class** LibraryManagementApplication {

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

**AnnotationConfigApplicationContext** context **=** **new** AnnotationConfigApplicationContext(AppConfig.class);

**BookCollectionService** service **=** context.getBean(BookCollectionService.class);

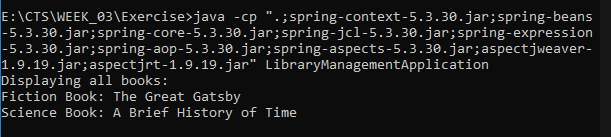
        service.displayAllBooks();

        context.close();

    }

}

**Output:**

****

**Exercise 8: Implementing Basic AOP with Spring**

**Scenario:**

The library management application requires basic AOP functionality to separate cross-cutting concerns like logging and transaction management.

**Code:**

**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.example</groupId>

    <artifactId>library</artifactId>

    <version>1.0.0</version>

    <packaging>jar</packaging>

    <name>LibraryManagement</name>

    <parent>

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

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

        <version>2.7.18</version>

        <relativePath/> *<!-- lookup parent from repository -->*

    </parent>

    <properties>

        <java.version>21</java.version>

    </properties>

    <dependencies>

        <dependency>

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

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

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

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

                <artifactId>spring-boot-maven-plugin</artifactId>

            </plugin>

        </plugins>

    </build>

</project>

**Book.java  
package** **com.example**;

**public** **class** Book {

**private** **String** title;

**private** **String** author;

*// Getters and Setters*

**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 controller.java  
package** **com.example**;

**import** **org.springframework.web.bind.annotation.\***;

@**RestController**

@**RequestMapping**("/books")

**public** **class** BookController {

    @**PostMapping**

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

**return** "Book added: " **+** book.getTitle();

    }

    @**GetMapping**

**public** **String** getBooks() {

**return** "Returning list of books";

    }

}

**LibraryApplication.java  
package** **com.example**;

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

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

@**SpringBootApplication**

**public** **class** LibraryApplication {

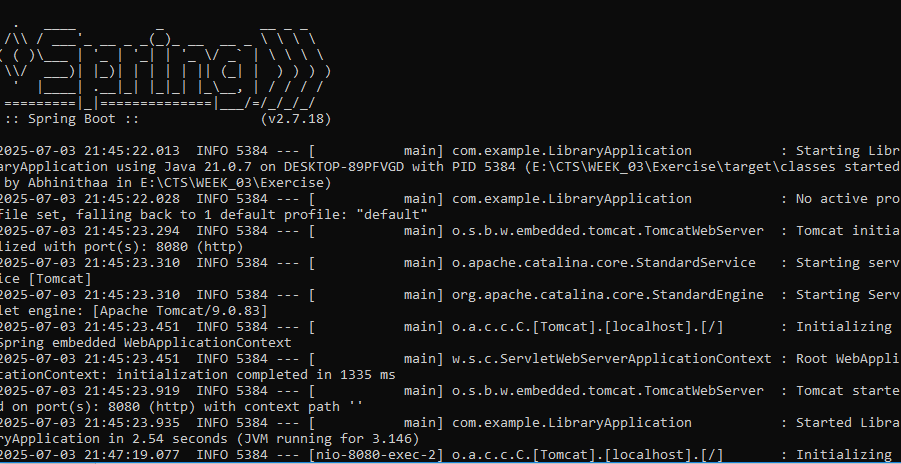
**public** **static** **void** main(**String**[] args) {

        SpringApplication.run(LibraryApplication.class, args);

    }

}

**Output:**

****

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

**Scenario:**

You need to create a Spring Boot application for the library management system to simplify configuration and deployment.

**Code:**

**BookController.java  
package** **com.example**;

**import** **java.util.ArrayList**;

**import** **java.util.List**;

**import** **org.springframework.web.bind.annotation.GetMapping**;

**import** **org.springframework.web.bind.annotation.PostMapping**;

**import** **org.springframework.web.bind.annotation.RequestBody**;

**import** **org.springframework.web.bind.annotation.RequestMapping**;

**import** **org.springframework.web.bind.annotation.RestController**;

@**RestController**

@**RequestMapping**("/books")

**public** **class** BookController {

**private** **List**<**Book**> books **=** **new** **ArrayList**<>();

    @**PostMapping**

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

        books.add(book);

        System.out.println("Book added: " **+** book.getTitle());

**return** "Book added: " **+** book.getTitle();

    }

    @**GetMapping**

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

        System.out.println("Returning list of books");

**return** books;

    }

}

**Book.java**

**package** **com.example**;

**public** **class** Book {

**private** **String** title;

**private** **String** author;

*// Constructors*

**public** Book() {}

**public** Book(**String** title, **String** author) {

        this.title **=** title;

        this.author **=** author;

    }

*// Getters and setters*

**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;

    }

}

**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.example</groupId>

    <artifactId>library</artifactId>

    <version>1.0.0</version>

    <packaging>jar</packaging>

    <name>LibraryManagement</name>

    <parent>

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

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

        <version>2.7.18</version>

        <relativePath/> *<!-- lookup parent from repository -->*

    </parent>

    <properties>

        <java.version>21</java.version>

    </properties>

    <dependencies>

        <dependency>

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

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

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

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

                <artifactId>spring-boot-maven-plugin</artifactId>

            </plugin>

        </plugins>

    </build>

</project>

**Output:**

