**WEEK-3**

**Spring Core and Maven**

**1.** **Configuring a Basic Spring Application**

**2. Implementing 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

                             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>

    <packaging>jar</packaging>

    <name>LibraryManagement</name>

    <url>http://maven.apache.org</url>

    <dependencies>

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-context</artifactId>

            <version>5.3.34</version>

        </dependency>

        <dependency>

            <groupId>junit</groupId>

            <artifactId>junit</artifactId>

            <version>3.8.1</version>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.codehaus.mojo</groupId>

                <artifactId>exec-maven-plugin</artifactId>

                <version>3.1.0</version>

                <configuration>

                    <mainClass>com.library.MainApp</mainClass>

                </configuration>

            </plugin>

            <plugin>

                <artifactId>maven-compiler-plugin</artifactId>

                <version>3.8.1</version>

                <configuration>

                    <source>1.8</source>

                    <target>1.8</target>

                </configuration>

            </plugin>

        </plugins>

    </build>

</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 Definitions -->

    <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 saveBook(String book) {

        System.out.println("Saving book: " + 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(String bookName){

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

        bookRepository.saveBook(bookName);

    }

}

**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) {

  // Load Spring container

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

// Get bean from Spring container

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

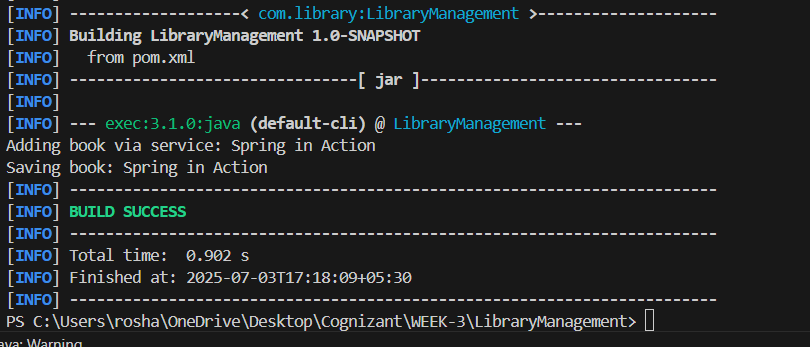
// Use the bean

        bookService.addBook("Spring in Action");

    }

}

**Output:**

****

**3. Implementing Logging with Spring AOP**

**pom.xml**

<dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-aspects</artifactId>

            <version>5.3.34</version>

  </dependency>

<dependency>

            <groupId>org.aspectj</groupId>

            <artifactId>aspectjweaver</artifactId>

            <version>1.9.21</version>

</dependency>

**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"

       xmlns:aop="http://www.springframework.org/schema/aop"

       xsi:schemaLocation=" http://www.springframework.org/schema/beans

           http://www.springframework.org/schema/beans/spring-beans.xsd

           http://www.springframework.org/schema/aop

           http://www.springframework.org/schema/aop/spring-aop.xsd">

    <aop:aspectj-autoproxy/>

    <!-- Bean Definitions -->

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

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

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

    </bean>

    <bean id="loggingAspect" class="com.library.aspect.LoggingAspect"/>

</beans>

**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 res=joinPoint.proceed();

        long end=System.currentTimeMillis();

        System.out.println("[LOG] " + joinPoint.getSignature() + " executed in " + (end - start) + " ms");

return res;

    }

}

**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) {

        // Load Spring container

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

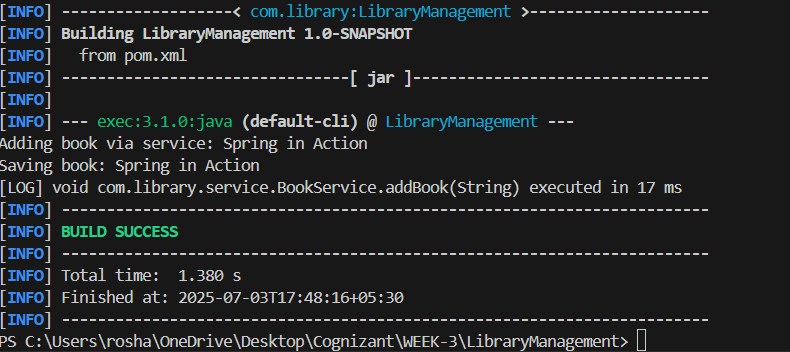
// Get bean from Spring container

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

// Use the bean

        bookService.addBook("Spring in Action");}}

**Output:**

****

**4. Creating and Configuring a Maven Project**

**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>LibraryManagement</artifactId>

    <version>1.0-SNAPSHOT</version>

    <packaging>jar</packaging>

    <name>LibraryManagement</name>

    <url>http://maven.apache.org</url>

    <dependencies>

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-context</artifactId>

            <version>5.3.34</version>

        </dependency>

        <dependency>

            <groupId>junit</groupId>

            <artifactId>junit</artifactId>

            <version>3.8.1</version>

            <scope>test</scope>

        </dependency>

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-aspects</artifactId>

            <version>5.3.34</version>

        </dependency>

         <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-webmvc</artifactId>

            <version>5.3.34</version>

        </dependency>

        <dependency>

            <groupId>org.aspectj</groupId>

            <artifactId>aspectjweaver</artifactId>

            <version>1.9.21</version>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.codehaus.mojo</groupId>

                <artifactId>exec-maven-plugin</artifactId>

                <version>3.1.0</version>

                <configuration>

                    <mainClass>com.library.MainApp</mainClass>

                </configuration>

            </plugin>

            <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>

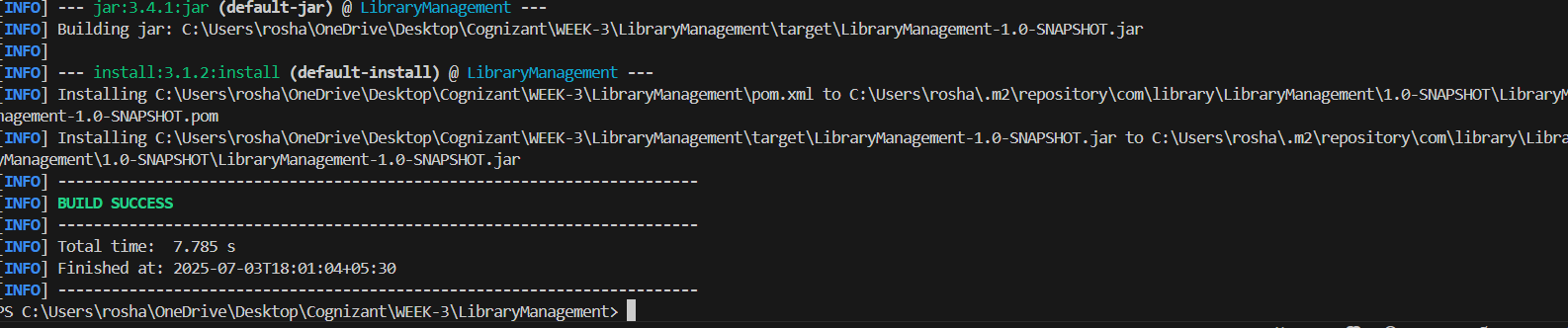
            </plugin>

        </plugins>

    </build>

</project>

**Output:**



**5. Configuring the Spring IoC Container**

**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(String bookName) {

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

        bookRepository.saveBook(bookName);

    }

}

**BookRepository.java**

package com.library.repository;

public class BookRepository {

    public void saveBook(String bookName) {

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

    }

}

**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) {

        // Load Spring container

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

        // Get bean from Spring container

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

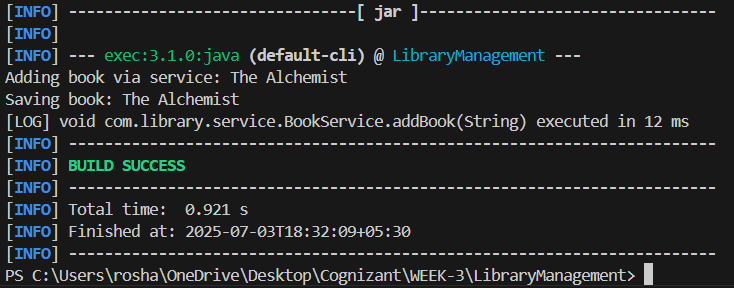
        // Use the bean

        bookService.addBook("The Alchemist");

    }

}

**Output:**

****

**6. Configuring Beans with Annotations**

**applicationContext.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

       xmlns:context="http://www.springframework.org/schema/context"

       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

       xsi:schemaLocation="

           http://www.springframework.org/schema/beans

           https://www.springframework.org/schema/beans/spring-beans.xsd

           http://www.springframework.org/schema/context

           https://www.springframework.org/schema/context/spring-context.xsd">

    <context:component-scan base-package="com.library" />

</beans>

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

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

import org.springframework.stereotype.Service;

@Service

public class BookService {

   private BookRepository bookRepository;

@Autowired

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

public void addBook(String bookName) {

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

        bookRepository.saveBook(bookName);

    }

}

**BookRepository.java**

package com.library.repository;

import org.springframework.stereotype.Repository;

@Repository

public class BookRepository {

    public void saveBook(String bookName) {

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

    }

}

**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");

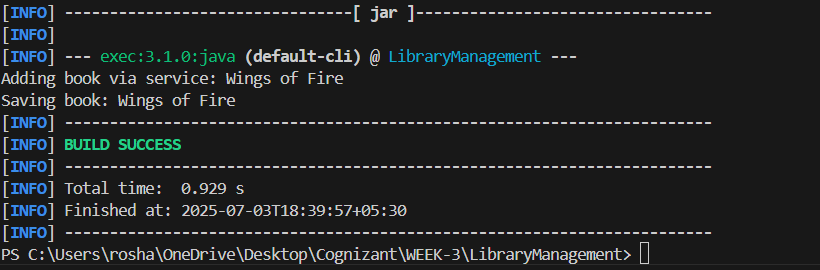
        BookService bookService = context.getBean(BookService.class);

        bookService.addBook("Wings of Fire");

    }

}

**Output:**



**7. Implementing Constructor and Setter Injection**

**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

           https://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- BookRepository Bean -->

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

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

        <constructor-arg value="LIB123" />

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

    </bean>

</beans>

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private String serviceId;

    private BookRepository bookRepository;

    public BookService(String serviceId) {

        this.serviceId = serviceId;

    }

public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

public void addBook(String bookName) {

        System.out.println("Service ID: " + serviceId);

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

        bookRepository.saveBook(bookName);

    }

}

**BookRepository.java**

package com.library.repository;

public class BookRepository {

    public void saveBook(String bookName) {

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

    }

}

**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");

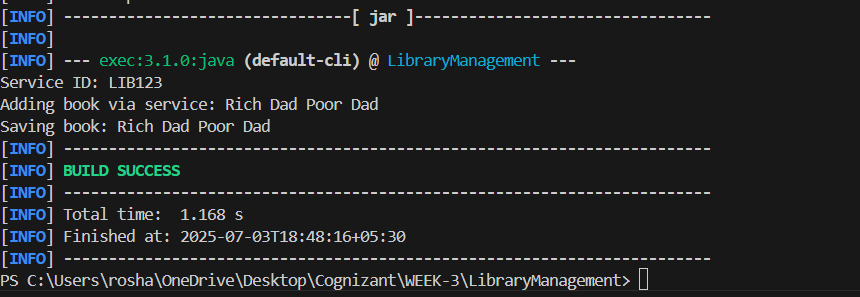
        BookService bookService = context.getBean(BookService.class);

        bookService.addBook("Rich Dad Poor Dad");

    }

}

**Output:**

****

**9. Creating a Spring Boot Application**

**application.properties**

spring.application.name=LibManagement

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

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

spring.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true

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

**Book.java**

package com.library.model;

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() {} // Default constructor

public Book(String title, String author) {

        this.title = title;

        this.author = author;

    }

    // Getters & 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.library.repository;

import com.library.model.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**BookController.java**

package com.library.controller;

import com.library.model.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;

    @GetMapping

    public List<Book> getAllBooks() {

        return bookRepository.findAll();

    }

@PostMapping

    public Book addBook(@RequestBody Book book) {

        return bookRepository.save(book);

    }

@GetMapping("/{id}")

    public Book getBook(@PathVariable Long id) {

        return bookRepository.findById(id).orElse(null);

    }

@PutMapping("/{id}")

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

        Book book = bookRepository.findById(id).orElse(null);

        if (book == null) return null;

        book.setTitle(bookDetails.getTitle());

        book.setAuthor(bookDetails.getAuthor());

        return bookRepository.save(book);

    }

@DeleteMapping("/{id}")

    public void deleteBook(@PathVariable Long id) {

        bookRepository.deleteById(id);

    }

}

**LibraryManagementApplication.java**

package com.library;

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);

    }

}

**Output:**

