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

**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/maven-v4\_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>5.3.30</version>

    </dependency>

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>4.13.2</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.LibraryManagementApplication</mainClass>

        </configuration>

      </plugin>

    </plugins>

  </build>

</project>

**ApplicationContext.xml**

<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">

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

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

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

    </bean>

</beans>

**BookService and BookRepository Class**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Setter injection

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook() {

        bookRepository.saveBook();

        System.out.println("Book added successfully!");

    }

}

package com.library.repository;

public class BookRepository {

    public void saveBook() {

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

    }

}

**Main Class**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryManagementApplication {

    public static void main(String[] args) {

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

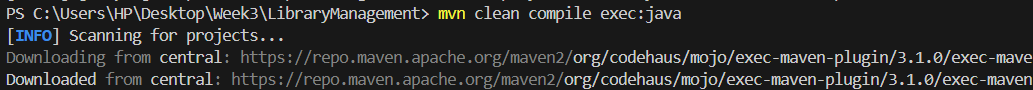
        BookService bookService = context.getBean("bookService", BookService.class);

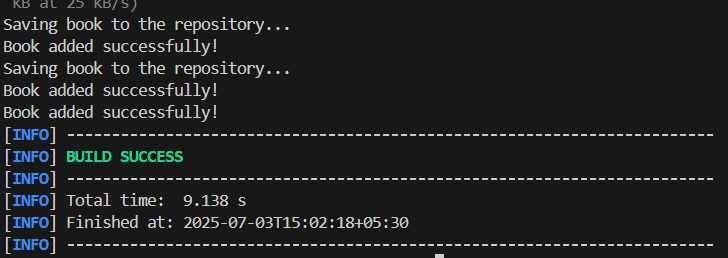
        bookService.addBook();

    }

}

**Output**

****

****

**Exercise 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/maven-v4\_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>5.3.30</version>

    </dependency>

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>4.13.2</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.LibraryManagementApplication</mainClass>

        </configuration>

      </plugin>

    </plugins>

  </build>

</project>

**ApplicationContext.xml**

<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">

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

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

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

    </bean>

</beans>

**BookService and BookRepository Class**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Setter injection

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook() {

        bookRepository.saveBook();

        System.out.println("Book added successfully!");

    }

}

package com.library.repository;

public class BookRepository {

    public void saveBook() {

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

    }

}

**Main Class**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryManagementApplication {

    public static void main(String[] args) {

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

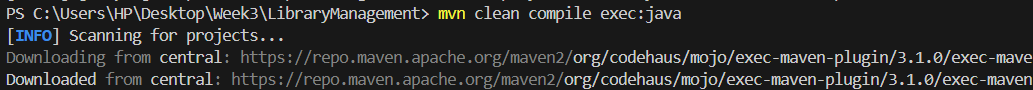
        BookService bookService = context.getBean("bookService", BookService.class);

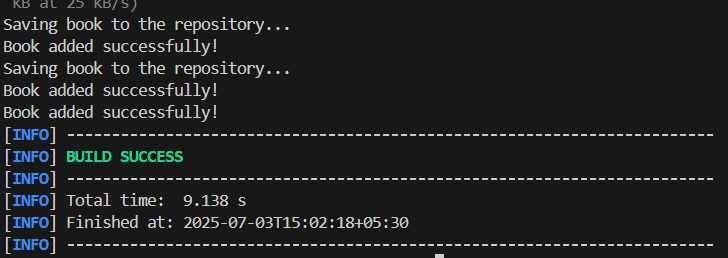
        bookService.addBook();

    }

}

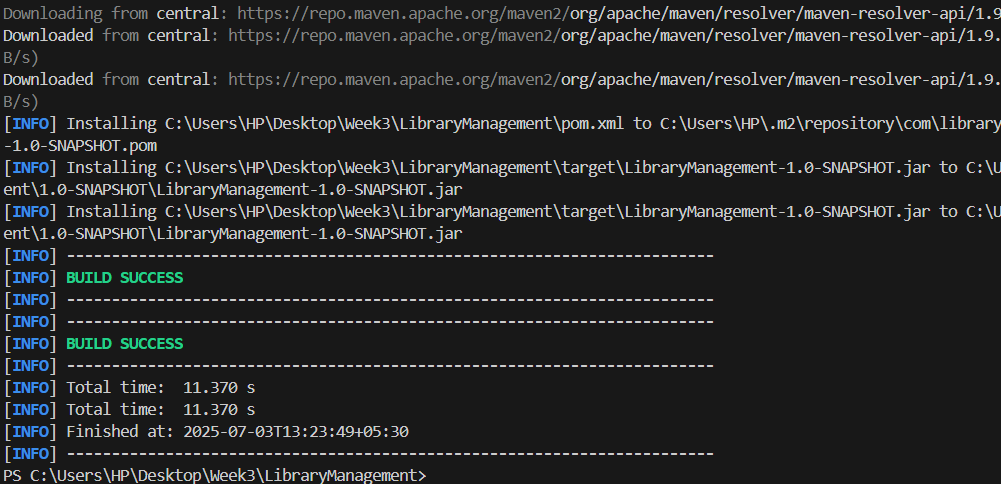
**Output**

****

****

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

**Build a new 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/maven-v4\_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>5.3.30</version>

    </dependency>

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>4.13.2</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.LibraryManagementApplication</mainClass>

      </configuration>

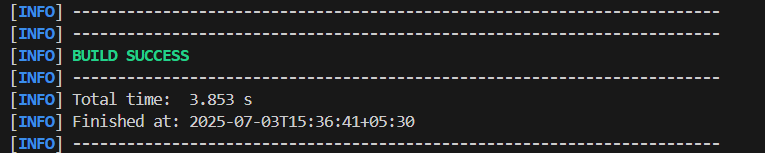
      </plugin>

    </plugins>

  </build>

</project>

**OUTPUT**

****

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

**Create Spring Configuration File:**

<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>

**Update the BookService Class:**

package com.library.service;

import java.util.ArrayList;

import java.util.List;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Internal list to hold book names

    private List<String> books = new ArrayList<>();

    // Setter injection

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    // Add a book

    public void addBook(String bookName) {

        bookRepository.saveBook(bookName);

        books.add(bookName);

        System.out.println("Book added successfully!");

    }

    // Getter for books list

    public List<String> getBooks() {

        return books;

    }

}

**Run the Application:**

package com.library;

import com.library.repository.BookRepository;

import com.library.service.BookService;

public class LibraryManagementApplication {

    public static void main(String[] args) {

        BookRepository bookRepository = new BookRepository();

        BookService bookService = new BookService();

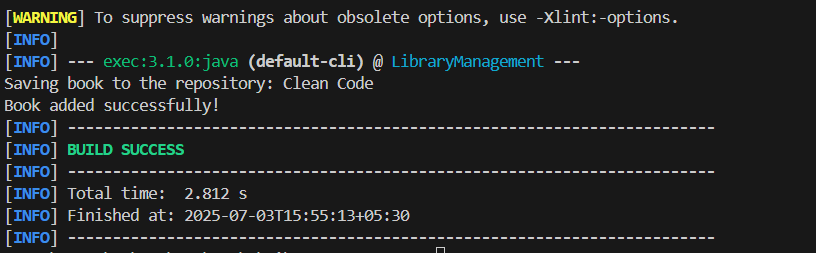
        bookService.setBookRepository(bookRepository);

        bookService.addBook("Clean Code");

    }

}

**Output**

****

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

**Configure Constructor Injection:**

<?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">

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

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

        <!-- Constructor injection -->

        <constructor-arg value="My Library Book Service"/>

        <!-- Setter injection -->

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

    </bean>

</beans>

**Configure Setter Injection:**

package com.library.service;

import com.library.repository.BookRepository;

import java.util.ArrayList;

import java.util.List;

public class BookService {

    private String serviceName;           // Constructor injection

    private BookRepository bookRepository; // Setter injection

    private List<String> books = new ArrayList<>();

    public BookService(String serviceName) {

        this.serviceName = serviceName;

    }

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook(String bookName) {

        System.out.println("Service Name: " + serviceName);

        bookRepository.saveBook();

        books.add(bookName);

        System.out.println("Book added successfully!");

    }

    public List<String> getBooks() {

        return books;

    }

}

package com.library.repository;

public class BookRepository {

    public void saveBook() {

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

    }

}

**Test the Injection:**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryManagementApplication {

    public static void main(String[] args) {

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

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

        bookService.addBook("Clean Code");

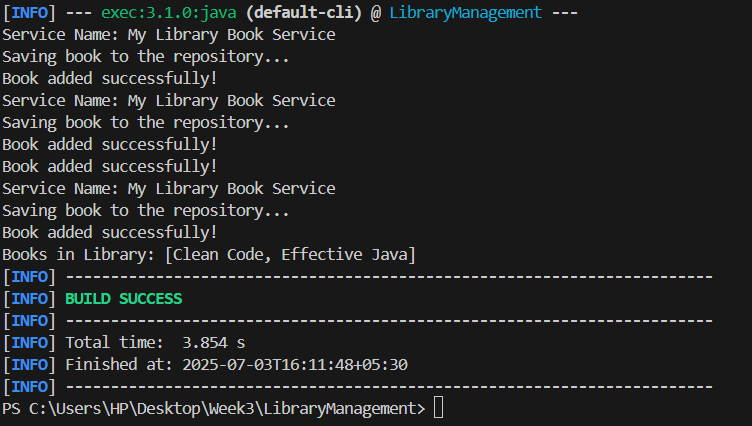
        bookService.addBook("Effective Java");

        System.out.println("Books in Library: " + bookService.getBooks());

    }

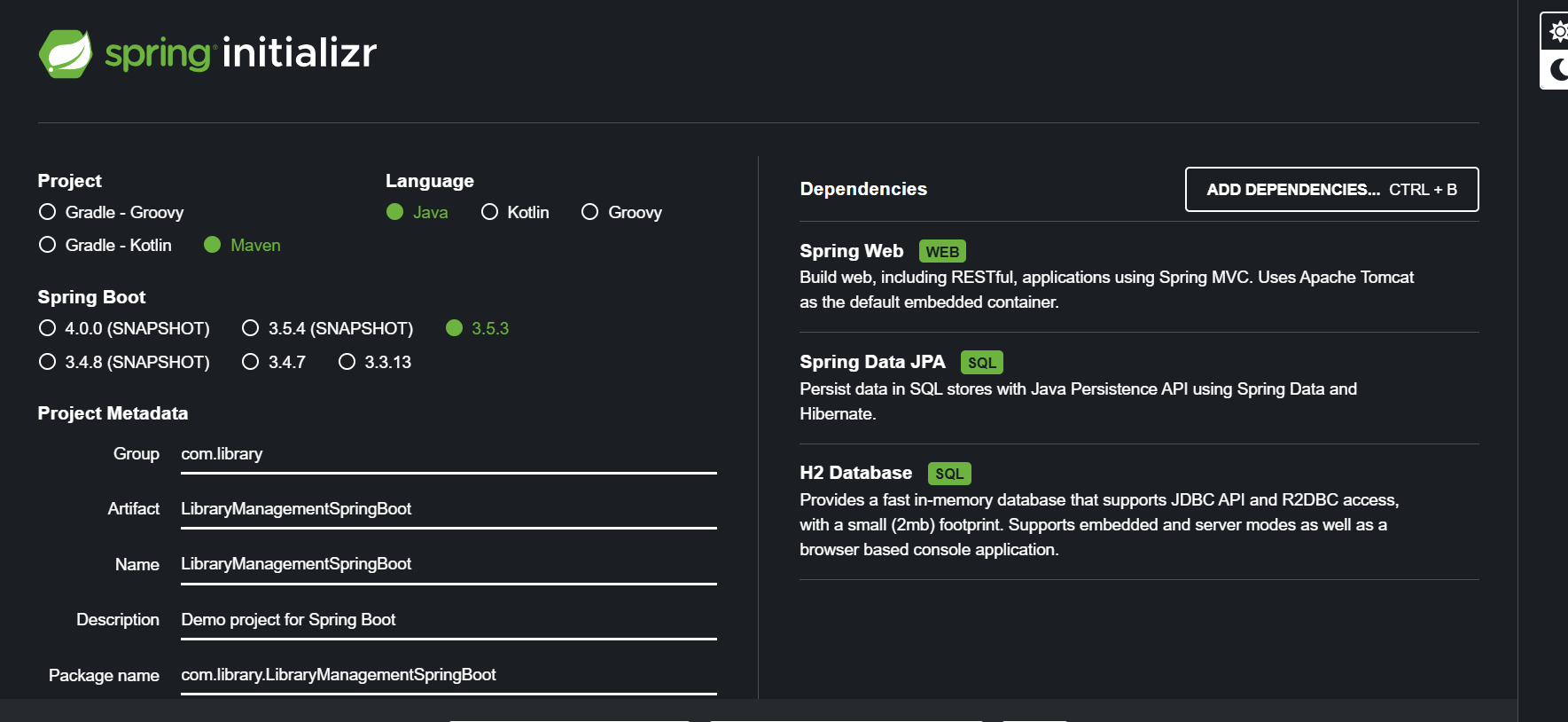
}

**Output**



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

**Create a Spring Boot Project:**



**Create Application Properties:**

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.h2.console.enabled=true

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

**Define Entities and Repositories:**

package com.library.LibraryManagementSpringBoot.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;

    //Constructors

    public Book() {}

    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; }

}

package com.library.LibraryManagementSpringBoot.repository;

import com.library.LibraryManagementSpringBoot.model.Book;

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

import org.springframework.stereotype.Repository;

@Repository

public interface BookRepository extends JpaRepository<Book, Long> {

}

**Create a REST Controller:**

package com.library.LibraryManagementSpringBoot.controller;

import com.library.LibraryManagementSpringBoot.model.Book;

import com.library.LibraryManagementSpringBoot.repository.BookRepository;

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

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

import java.util.List;

@RestController

@RequestMapping("/api/books")

public class BookController {

    @Autowired

    private BookRepository bookRepository;

    @GetMapping

    public List<Book> getAllBooks() {

        return bookRepository.findAll();

    }

    @PostMapping

    public Book createBook(@RequestBody Book book) {

        return bookRepository.save(book);

    }

    @GetMapping("/{id}")

    public Book getBookById(@PathVariable Long id) {

        return bookRepository.findById(id).orElseThrow(() -> new RuntimeException("Book not found"));

    }

    @PutMapping("/{id}")

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

        Book book = bookRepository.findById(id)

                .orElseThrow(() -> new RuntimeException("Book not found"));

        book.setTitle(updatedBook.getTitle());

        book.setAuthor(updatedBook.getAuthor());

        return bookRepository.save(book);

    }

    @DeleteMapping("/{id}")

    public void deleteBook(@PathVariable Long id) {

        bookRepository.deleteById(id);

    }

}

**Run the Application:**

