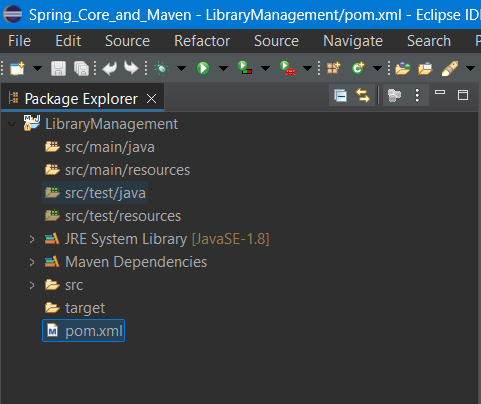
Spring Core and Maven

# Exercise 1: Configuring a Basic Spring Application

## Step: 1: Setting up the maven project

## Creating a maven project name LibraryManagement



## Adding spring dependencies in pom.xml file

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>7.0.0-M6</version>

</dependency>

</dependencies>

## Step: 2: Configure the Application Context

ApplicationConfig.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 id="bookService" class="com.library.service.BookService"/>

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

</beans>

## Step: 3: Define Service and Repository Classes

Service.java

package com.library.service;

public class BookService {

public BookService() {

System.***out***.println("BookService Bean Created");

}

}

Repository.java

package com.library.repository;

public class BookRepository {

public BookRepository() {

System.out.println("BookRepository Bean Created");

}

}

## Step: 4: Run the Application

LibraryManagementApplication.java

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.repository.BookRepository;

import com.library.service.BookService;

public class LibraryManagementApplication {

    public static void main(String args[]) {

        ApplicationContext context =

                new ClassPathXmlApplicationContext("applicationConfig.xml");

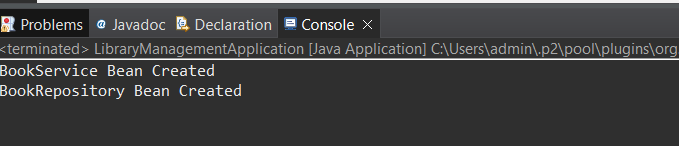
        BookService service = context.getBean("bookService", BookService.class);

        BookRepository repository=context.getBean("bookRepository",BookRepository.class);

    }

}

Output:



# Exercise 2: Implementing Dependency Injection

## Step: 1: Modify the XML configuration

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

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

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

</bean>

</beans>

## Step: 2: Update the BookService Class

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public BookService() {

System.***out***.println("BookService Bean Created");

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository=bookRepository;

System.***out***.println("Setter injection for Book Repository");

}

public void AddBook() {

System.***out***.println("BookAdded Successfully");

bookRepository.save();

}

}

## Step: 3: Verify the Configuration

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.repository.BookRepository;

import com.library.service.BookService;

public class LibraryManagementApplication {

    public static void main(String args[]) {

        ApplicationContext context =

                new ClassPathXmlApplicationContext("applicationConfig.xml");

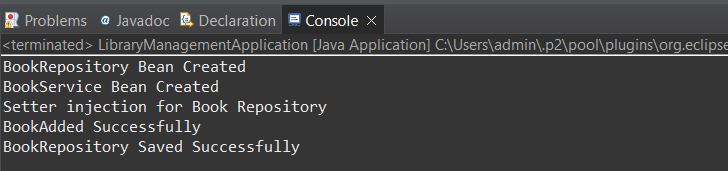
        BookService service = context.getBean("bookService", BookService.class);

        service.AddBook();

    }

}

Output:



# Exercise 3: Implementing Logging with Spring AOP

## Step: 1: Adding Spring AOP Dependency

<!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>7.0.0-M6</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.24</version>

<scope>runtime</scope>

</dependency>

## Step: 2: Create an Aspect for Logging

LoggingAspect.java

package com.library.aspect;

import org.aopalliance.intercept.MethodInterceptor;

import org.aopalliance.intercept.MethodInvocation;

public class LoggingAspect implements MethodInterceptor {

    @Override

    public Object invoke(MethodInvocation invocation) throws Throwable {

        long start = System.currentTimeMillis();

        Object result = invocation.proceed();

        long end = System.currentTimeMillis();

        System.out.println(invocation.getMethod() + " executed in " + (end - start) + "ms");

        return result;

    }

}

## Step: 3: Enable AspectJ Support

<?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:context="http://www.springframework.org/schema/context"

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/context

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

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

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

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

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

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

</bean>

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

<aop:aspectj-autoproxy/>

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

<aop:config>

<aop:advisor advice-ref="loggingAspect"

pointcut="execution(\* com.library.service.\*.\*(..))"/>

</aop:config>

</beans>

## Step: 4: Test the Aspect

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.repository.BookRepository;

import com.library.service.BookService;

public class LibraryManagementApplication {

    public static void main(String args[]) {

        ApplicationContext context =

                new ClassPathXmlApplicationContext("applicationConfig.xml");

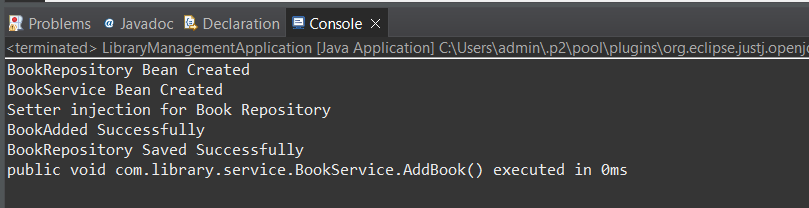
        BookService service = context.getBean("bookService", BookService.class);

        service.AddBook();

    }

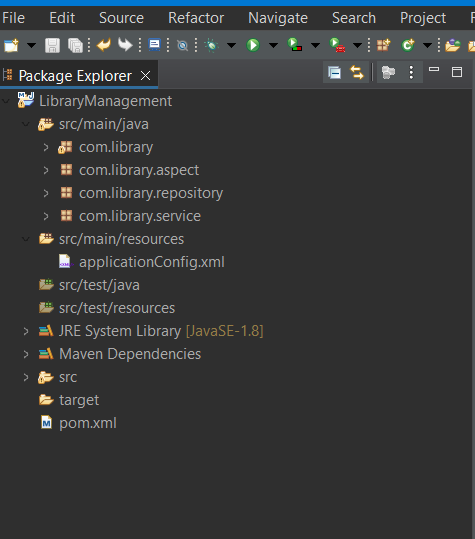
}

Output



# Exercise 4: Creating and Configuring a Maven Project

## Step: 1: Create a new Maven Project



## Step: 2: Adding Spring Dependencies in pom.xml

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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>LibraryManagement</name>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>7.0.0-M6</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>7.0.0-M6</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>7.0.0-M6</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.24</version>

<scope>runtime</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>7.0.0-M6</version>

</dependency>

</dependencies>

</project>

## Step: 3: Configure Maven Plugin

Add to pom.xml

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

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

<version>3.10.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

# Exercise 5: Configuring the Spring IoC Container

## Step: 1: Create Spring Configuration file

<?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">

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

</bean>

</beans>

## Step: 2: Update the Book Service class

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public BookService() {

System.***out***.println("BookService Bean Created");

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository=bookRepository;

System.***out***.println("Setter injection for Book Repository");

}

public void AddBook() {

System.***out***.println("BookAdded Successfully");

bookRepository.save();

}

}

## Step: 3: Run the Application

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.repository.BookRepository;

import com.library.service.BookService;

public class LibraryManagementApplication {

    public static void main(String args[]) {

        ApplicationContext context =

                new ClassPathXmlApplicationContext("applicationConfig.xml");

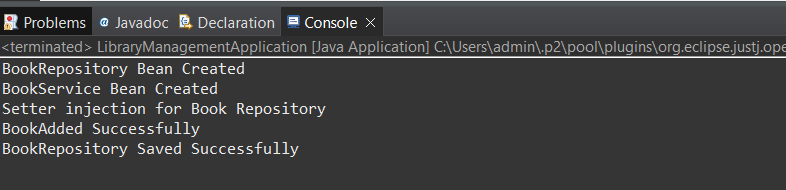
        BookService service = context.getBean("bookService", BookService.class);

        service.AddBook();

    }

}

**Output**



# Exercise 6: Configuring Beans with Annotations

## Step: 1: Create Spring Configuration File

<?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:context="http://www.springframework.org/schema/context"

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"/>

<context:annotation-config/>

</beans>

## Step: 2: Annotate Classes

BookService.java

package com.library.repository;

import org.springframework.stereotype.Repository;

*@Repository*

public class BookRepository {

public BookRepository() {

System.***out***.println("BookRepository Bean Created");

}

public void save() {

System.***out***.println("BookRepository Saved Successfully");

}

}

BookRepository.java

package com.library.repository;

import org.springframework.stereotype.Repository;

*@Repository*

public class BookRepository {

public BookRepository() {

System.***out***.println("BookRepository Bean Created");

}

public void save() {

System.***out***.println("BookRepository Saved Successfully");

}

}

## Step: 3: Test the Configuration

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.repository.BookRepository;

import com.library.service.BookService;

public class LibraryManagementApplication {

    public static void main(String args[]) {

        ApplicationContext context =

                new ClassPathXmlApplicationContext("applicationContext.xml");

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

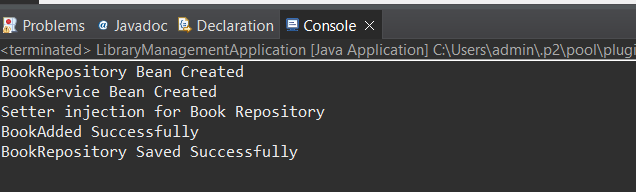
        service.AddBook();

        ((ClassPathXmlApplicationContext) context).close();

    }

}

Output:



# Exercise 7: Implementing Constructor and Setter Injection

## Step: 1: Configure Constructor 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"

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

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">

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

<bean id="notificationService" class="com.library.service.NotificationService"/>

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

<constructor-arg ref="bookRepository"></constructor-arg>

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

</bean>

</beans>

## Step: 2: Configure Setter Injection

NotificationService.java

package com.library.service;

public class NotificationService {

public NotificationService() {

System.***out***.println("Notification Service Bean Created");

}

public void NotifyUser() {

System.***out***.println("User Notified");

}

}

BookService.java

package com.library.service;

import org.springframework.stereotype.Service;

import com.library.repository.BookRepository;

*@Service*

public class BookService {

private BookRepository bookRepository;// via constructor

private NotificationService notificationService;//via setter

public BookService(BookRepository bookRepository) {

this.bookRepository=bookRepository;

System.***out***.println("BookService Bean Created via Constructor Injection");

}

public void setNotificationService(NotificationService notificationService) {

this.notificationService=notificationService;

System.***out***.println("NotificationService Bean created via Setter Injection");

}

public void addBook() {

System.***out***.println("BookAdded Successfully");

bookRepository.save();

if (notificationService != null) {

notificationService.NotifyUser();

}

}

}

## Step: 3: Test the Configuration

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

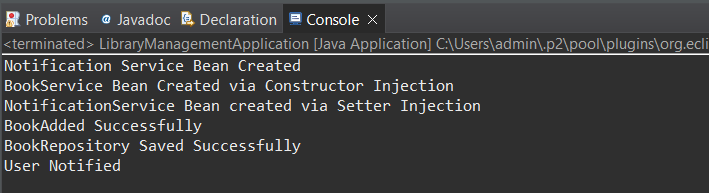
service.addBook();

((ClassPathXmlApplicationContext) context).close();

}

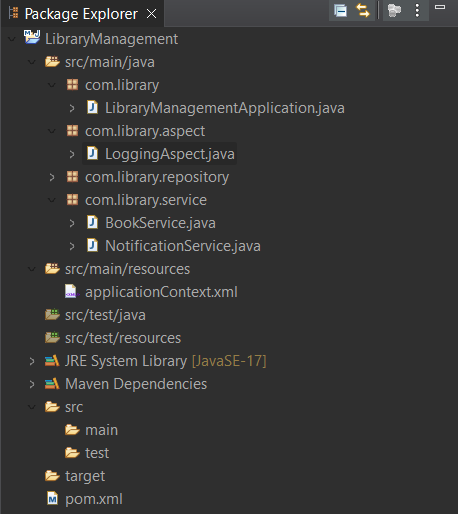
}

Output:



# Exercise 8: Implementing Basic AOP with Spring

## Step: 1: Define an Aspect



## Step: 2: Create Advise Methods

package com.library.aspect;

public class LoggingAspect {

public void beforeMethod() {

System.***out***.println("[LOG] Before executing BookService method");

}

public void afterMethod() {

System.***out***.println("[LOG] After executing BookService method");

}

}

## Step: 3: Configure the Aspect

<?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:context="http://www.springframework.org/schema/context"

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

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

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

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

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

<aop:config>

<aop:aspect ref="loggingAspect">

<aop:pointcut id="bookServiceMethods"

expression="execution(\* com.library.service.BookService.\*(..))"/>

<aop:before method="beforeMethod" pointcut-ref="bookServiceMethods"/>

<aop:after method="afterMethod" pointcut-ref="bookServiceMethods"/>

</aop:aspect>

</aop:config>

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

</beans>

## Step: 4: Test the Aspect

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

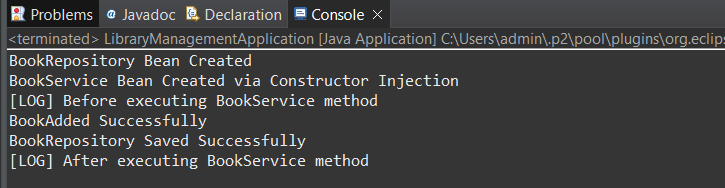
        service.addBook();

        ((ClassPathXmlApplicationContext) context).close();

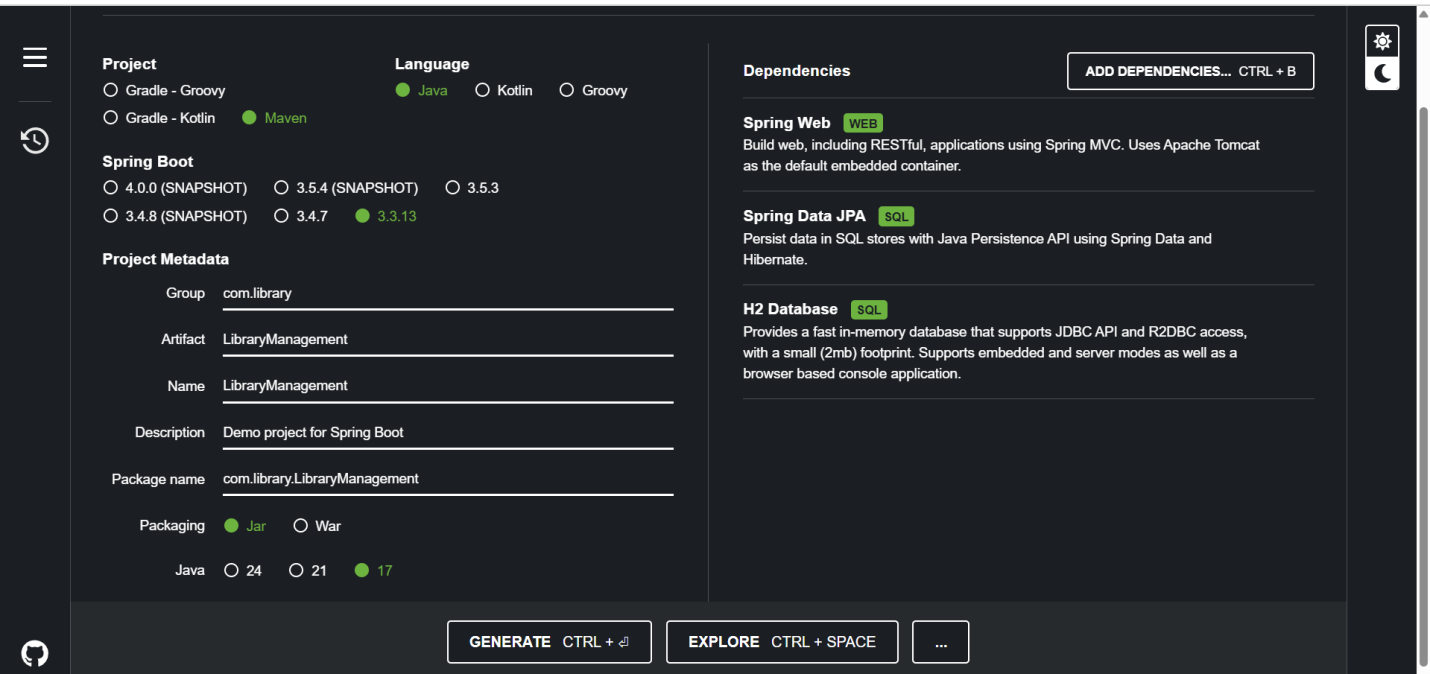
    }

}

Output:



# Exercise 9: Creating a Spring Boot Application



Application.properties

spring.application.name=LibraryManagement

# H2 Database config

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

# JPA config

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

spring.jpa.hibernate.ddl-auto=update

# Enable H2 console

spring.h2.console.enabled=true

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

Book.java

package com.library.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; }

}

BookController.java

package com.library.controller;

import com.library.entity.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) {

            book.setTitle(bookDetails.getTitle());

            book.setAuthor(bookDetails.getAuthor());

            return bookRepository.save(book);

        }

        return null;

    }

    @DeleteMapping("/{id}")

    public void deleteBook(@PathVariable Long id) {

        bookRepository.deleteById(id);

    }

}

BookRepository.java

package com.library.repository;

import com.library.entity.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

LibraryManagement.java

package com.library.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);

    }

}

Output:

