**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.

**Steps:**

1. **Create a New Maven Project:**
   * Create a new Maven project named **LibraryManagement**.
2. **Add Spring Dependencies in pom.xml:**
   * Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. **Configure Maven Plugins:**
   * Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

Solution:

LibraryManagementMavenApplication.java:

package com.library;  
  
import com.library.config.AppConfig;  
import com.library.controller.BookController;  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.annotation.AnnotationConfigApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import java.util.Collections;  
  
public class LibraryManagementMavenApplication {  
  
 public static void main(String[] args) {  
 System.*out*.println("=== Starting Library Management Maven Application ===");  
 System.*out*.println("Exercise 4: Maven Project with Spring Dependencies");  
 System.*out*.println("Java Version: " + System.*getProperty*("java.version"));  
 System.*out*.println("Maven Configuration with Spring Context, AOP, and WebMVC\n");  
  
 *runWithAnnotationConfig*();  
  
 System.*out*.println("\n" + *repeatString*("=", 60) + "\n");  
  
 *runWithXmlConfig*();  
  
 System.*out*.println("\n=== Maven Application Completed Successfully! ===");  
 }  
  
 private static String repeatString(String str, int count) {  
 if (count <= 0) return "";  
 return String.*join*("", Collections.*nCopies*(count, str));  
 }  
  
 private static void runWithAnnotationConfig() {  
 System.*out*.println("=== Running with Annotation-based Configuration ===");  
  
 ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);  
  
 BookController bookController = context.getBean(BookController.class);  
 BookService bookService = context.getBean(BookService.class);  
  
 *testApplication*(bookController, "Annotation Config");  
  
 ((AnnotationConfigApplicationContext) context).close();  
 }  
  
 private static void runWithXmlConfig() {  
 System.*out*.println("=== Running with XML Configuration ===");  
  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookController bookController = context.getBean(BookController.class);  
 BookService bookService = context.getBean(BookService.class);  
  
 *testApplication*(bookController, "XML Config");  
  
 ((ClassPathXmlApplicationContext) context).close();  
 }  
  
 private static void testApplication(BookController controller, String configType) {  
 System.*out*.println("\n=== Testing Application with " + configType + " ===");  
  
 controller.handleGetAllBooks();  
  
 controller.handleGetLibraryStats();  
  
 controller.handleAddBook("Maven in Action", "Brett Porter", "978-1932394887");  
 controller.handleAddBook("Spring Boot in Action", "Craig Walls", "978-1617292545");  
  
 controller.handleSearchBooks("Spring", "title");  
 controller.handleSearchBooks("Craig", "author");  
  
 controller.handleGetBookById(1L);  
 controller.handleGetBookById(999L);  
  
 controller.handleDeleteBook(3L);  
  
 controller.handleGetAllBooks();  
 controller.handleGetLibraryStats();  
 }  
}

BookRepository.java:

package com.library.repository;  
  
import org.springframework.stereotype.Repository;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.concurrent.ConcurrentHashMap;  
import java.util.concurrent.atomic.AtomicLong;  
  
@Repository  
public class BookRepository {  
 private final ConcurrentHashMap<Long, Book> books = new ConcurrentHashMap<>();  
 private final AtomicLong idGenerator = new AtomicLong(1);  
  
 public static class Book {  
 private Long id;  
 private String title;  
 private String author;  
 private String isbn;  
  
 public Book() {}  
  
 public Book(Long id, String title, String author, String isbn) {  
 this.id = id;  
 this.title = title;  
 this.author = author;  
 this.isbn = isbn;  
 }  
  
 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; }  
  
 public String getIsbn() { return isbn; }  
 public void setIsbn(String isbn) { this.isbn = isbn; }  
  
 @Override  
 public String toString() {  
 return "Book{id=" + id + ", title='" + title + "', author='" + author + "', isbn='" + isbn + "'}";  
 }  
 }  
  
 public BookRepository() {  
 // Initialize with sample data  
 initializeSampleBooks();  
 System.*out*.println("BookRepository initialized with " + books.size() + " books");  
 }  
  
 private void initializeSampleBooks() {  
 addBook(new Book(idGenerator.getAndIncrement(), "Spring in Action", "Craig Walls", "978-1617294945"));  
 addBook(new Book(idGenerator.getAndIncrement(), "Java: The Complete Reference", "Herbert Schildt", "978-1260440232"));  
 addBook(new Book(idGenerator.getAndIncrement(), "Clean Code", "Robert C. Martin", "978-0132350884"));  
 addBook(new Book(idGenerator.getAndIncrement(), "Effective Java", "Joshua Bloch", "978-0134685991"));  
 addBook(new Book(idGenerator.getAndIncrement(), "Design Patterns", "Gang of Four", "978-0201633612"));  
 }  
  
 public Book save(Book book) {  
 if (book.getId() == null) {  
 book.setId(idGenerator.getAndIncrement());  
 }  
 books.put(book.getId(), book);  
 System.*out*.println("Book saved: " + book);  
 return book;  
 }  
  
 public Book addBook(Book book) {  
 return save(book);  
 }  
  
 public List<Book> findAll() {  
 return new ArrayList<>(books.values());  
 }  
  
 public Book findById(Long id) {  
 return books.get(id);  
 }  
  
 public void deleteById(Long id) {  
 Book removed = books.remove(id);  
 if (removed != null) {  
 System.*out*.println("Book removed: " + removed);  
 } else {  
 System.*out*.println("Book not found with ID: " + id);  
 }  
 }  
  
 public List<Book> findByTitleContaining(String title) {  
 List<Book> matchingBooks = new ArrayList<>();  
 for (Book book : books.values()) {  
 if (book.getTitle().toLowerCase().contains(title.toLowerCase())) {  
 matchingBooks.add(book);  
 }  
 }  
 return matchingBooks;  
 }  
  
 public List<Book> findByAuthor(String author) {  
 List<Book> matchingBooks = new ArrayList<>();  
 for (Book book : books.values()) {  
 if (book.getAuthor().toLowerCase().contains(author.toLowerCase())) {  
 matchingBooks.add(book);  
 }  
 }  
 return matchingBooks;  
 }  
  
 public long count() {  
 return books.size();  
 }  
}

BookService.java:

package com.library.repository;  
  
import org.springframework.stereotype.Repository;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.concurrent.ConcurrentHashMap;  
import java.util.concurrent.atomic.AtomicLong;  
  
@Repository  
public class BookRepository {  
 private final ConcurrentHashMap<Long, Book> books = new ConcurrentHashMap<>();  
 private final AtomicLong idGenerator = new AtomicLong(1);  
  
 public static class Book {  
 private Long id;  
 private String title;  
 private String author;  
 private String isbn;  
  
 public Book() {}  
  
 public Book(Long id, String title, String author, String isbn) {  
 this.id = id;  
 this.title = title;  
 this.author = author;  
 this.isbn = isbn;  
 }  
  
 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; }  
  
 public String getIsbn() { return isbn; }  
 public void setIsbn(String isbn) { this.isbn = isbn; }  
  
 @Override  
 public String toString() {  
 return "Book{id=" + id + ", title='" + title + "', author='" + author + "', isbn='" + isbn + "'}";  
 }  
 }  
  
 public BookRepository() {  
 // Initialize with sample data  
 initializeSampleBooks();  
 System.*out*.println("BookRepository initialized with " + books.size() + " books");  
 }  
  
 private void initializeSampleBooks() {  
 addBook(new Book(idGenerator.getAndIncrement(), "Spring in Action", "Craig Walls", "978-1617294945"));  
 addBook(new Book(idGenerator.getAndIncrement(), "Java: The Complete Reference", "Herbert Schildt", "978-1260440232"));  
 addBook(new Book(idGenerator.getAndIncrement(), "Clean Code", "Robert C. Martin", "978-0132350884"));  
 addBook(new Book(idGenerator.getAndIncrement(), "Effective Java", "Joshua Bloch", "978-0134685991"));  
 addBook(new Book(idGenerator.getAndIncrement(), "Design Patterns", "Gang of Four", "978-0201633612"));  
 }  
  
 public Book save(Book book) {  
 if (book.getId() == null) {  
 book.setId(idGenerator.getAndIncrement());  
 }  
 books.put(book.getId(), book);  
 System.*out*.println("Book saved: " + book);  
 return book;  
 }  
  
 public Book addBook(Book book) {  
 return save(book);  
 }  
  
 public List<Book> findAll() {  
 return new ArrayList<>(books.values());  
 }  
  
 public Book findById(Long id) {  
 return books.get(id);  
 }  
  
 public void deleteById(Long id) {  
 Book removed = books.remove(id);  
 if (removed != null) {  
 System.*out*.println("Book removed: " + removed);  
 } else {  
 System.*out*.println("Book not found with ID: " + id);  
 }  
 }  
  
 public List<Book> findByTitleContaining(String title) {  
 List<Book> matchingBooks = new ArrayList<>();  
 for (Book book : books.values()) {  
 if (book.getTitle().toLowerCase().contains(title.toLowerCase())) {  
 matchingBooks.add(book);  
 }  
 }  
 return matchingBooks;  
 }  
  
 public List<Book> findByAuthor(String author) {  
 List<Book> matchingBooks = new ArrayList<>();  
 for (Book book : books.values()) {  
 if (book.getAuthor().toLowerCase().contains(author.toLowerCase())) {  
 matchingBooks.add(book);  
 }  
 }  
 return matchingBooks;  
 }  
  
 public long count() {  
 return books.size();  
 }  
}

BookController.java:

package com.library.controller;  
  
import com.library.service.BookService;  
import com.library.repository.BookRepository.Book;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
  
@Controller  
public class BookController {  
  
 @Autowired  
 private BookService bookService;  
  
 public BookController() {  
 System.*out*.println("BookController created");  
 }  
  
 public void handleGetAllBooks() {  
 System.*out*.println("\n=== Controller: Getting All Books ===");  
 bookService.displayAllBooks();  
 }  
  
 public void handleAddBook(String title, String author, String isbn) {  
 System.*out*.println("\n=== Controller: Adding New Book ===");  
 Book book = bookService.addBook(title, author, isbn);  
 System.*out*.println("Controller: Book added successfully - " + book);  
 }  
  
 public void handleDeleteBook(Long id) {  
 System.*out*.println("\n=== Controller: Deleting Book ===");  
 Book book = bookService.getBookById(id);  
 if (book != null) {  
 bookService.deleteBook(id);  
 System.*out*.println("Controller: Book deleted successfully - " + book);  
 } else {  
 System.*out*.println("Controller: Book not found with ID: " + id);  
 }  
 }  
  
 public void handleSearchBooks(String searchTerm, String searchType) {  
 System.*out*.println("\n=== Controller: Searching Books ===");  
 bookService.displaySearchResults(searchTerm, searchType);  
 }  
  
 public void handleGetBookById(Long id) {  
 System.*out*.println("\n=== Controller: Getting Book by ID ===");  
 Book book = bookService.getBookById(id);  
 if (book != null) {  
 System.*out*.println("Controller: Book found - " + book);  
 } else {  
 System.*out*.println("Controller: Book not found with ID: " + id);  
 }  
 }  
  
 public void handleGetLibraryStats() {  
 System.*out*.println("\n=== Controller: Library Statistics ===");  
 long totalBooks = bookService.getTotalBooks();  
 System.*out*.println("Total books in library: " + totalBooks);  
 System.*out*.println("Library status: " + (totalBooks > 0 ? "Active" : "Empty"));  
 }  
}

AppConfig.java:

package com.library.config;  
  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.context.annotation.EnableAspectJAutoProxy;  
  
@Configuration  
@ComponentScan(basePackages = "com.library")  
@EnableAspectJAutoProxy  
public class AppConfig {  
  
 public AppConfig() {  
 System.*out*.println("AppConfig initialized - Component scanning enabled for com.library package");  
 }  
}

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"  
 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">  
  
 <context:component-scan base-package="com.library"/>  
  
 <context:annotation-config/>  
  
 <aop:aspectj-autoproxy/>  
  
 <context:property-placeholder location="classpath:application.properties"/>  
  
</beans>

Pom.xml:

<?xml version="1.0" encoding="UTF-8"?>  
<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>  
  
 <!-- Project Information -->  
 <groupId>com.library</groupId>  
 <artifactId>LibraryManagementMaven</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>LibraryManagementMaven</name>  
 <description>Spring-based Library Management System with Maven Configuration - Exercise 4</description>  
 <url>http://www.example.com</url>  
  
 <!-- Properties -->  
 <properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <spring.version>5.3.21</spring.version>  
 <junit.version>4.13.2</junit.version>  
 <slf4j.version>1.7.32</slf4j.version>  
 </properties>  
  
 <!-- Dependencies -->  
 <dependencies>  
 <!-- Spring Context (includes Core, Beans, AOP, Context) -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
  
 <!-- Spring AOP -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
  
 <!-- Spring WebMVC -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
  
 <!-- Spring Core (explicitly included for clarity) -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
  
 <!-- Spring Beans -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-beans</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
  
 <!-- Spring Test (for testing) -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-test</artifactId>  
 <version>${spring.version}</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- AspectJ for AOP -->  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjrt</artifactId>  
 <version>1.9.7</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 <version>1.9.7</version>  
 </dependency>  
  
 <!-- Servlet API (for WebMVC) -->  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-api</artifactId>  
 <version>4.0.1</version>  
 <scope>provided</scope>  
 </dependency>  
  
 <!-- JSP API (for WebMVC) -->  
 <dependency>  
 <groupId>javax.servlet.jsp</groupId>  
 <artifactId>jsp-api</artifactId>  
 <version>2.2</version>  
 <scope>provided</scope>  
 </dependency>  
  
 <!-- JSTL (for JSP) -->  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>jstl</artifactId>  
 <version>1.2</version>  
 </dependency>  
  
 <!-- Logging -->  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>${slf4j.version}</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-simple</artifactId>  
 <version>${slf4j.version}</version>  
 </dependency>  
  
 <!-- Testing -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>${junit.version}</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- Jackson for JSON processing (useful for WebMVC) -->  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-databind</artifactId>  
 <version>2.13.3</version>  
 </dependency>  
 </dependencies>  
  
 <!-- Build Configuration -->  
 <build>  
 <finalName>LibraryManagementMaven</finalName>  
  
 <plugins>  
 <!-- Maven Compiler 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>  
 <encoding>UTF-8</encoding>  
 </configuration>  
 </plugin>  
  
 <!-- Maven Surefire Plugin (for testing) -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>3.0.0-M7</version>  
 <configuration>  
 <useSystemClassLoader>false</useSystemClassLoader>  
 </configuration>  
 </plugin>  
  
 <!-- Maven Resources Plugin -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-resources-plugin</artifactId>  
 <version>3.2.0</version>  
 <configuration>  
 <encoding>UTF-8</encoding>  
 </configuration>  
 </plugin>  
  
 <!-- Maven Clean Plugin -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-clean-plugin</artifactId>  
 <version>3.1.0</version>  
 </plugin>  
  
 <!-- Maven Install Plugin -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-install-plugin</artifactId>  
 <version>3.0.0-M1</version>  
 </plugin>  
  
 <!-- Maven Deploy Plugin -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-deploy-plugin</artifactId>  
 <version>3.0.0-M1</version>  
 </plugin>  
  
 <!-- Maven Site Plugin -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-site-plugin</artifactId>  
 <version>3.9.1</version>  
 </plugin>  
  
 <!-- Maven JAR Plugin -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-jar-plugin</artifactId>  
 <version>3.2.0</version>  
 <configuration>  
 <archive>  
 <manifest>  
 <addClasspath>true</addClasspath>  
 <mainClass>com.library.LibraryManagementMavenApplication</mainClass>  
 </manifest>  
 </archive>  
 </configuration>  
 </plugin>  
  
 <!-- Exec Maven Plugin (for running the application) -->  
 <plugin>  
 <groupId>org.codehaus.mojo</groupId>  
 <artifactId>exec-maven-plugin</artifactId>  
 <version>3.1.0</version>  
 <configuration>  
 <mainClass>com.library.LibraryManagementMavenApplication</mainClass>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

A screenshot of a computer

AI-generated content may be incorrect.A screen shot of a computer

AI-generated content may be incorrect.Output:

A computer screen with text

AI-generated content may be incorrect.A black screen with white text

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

A black rectangle with white text

AI-generated content may be incorrect.