**Question-1:**

**Problem statement1-Configuring a Basic Spring Application**

**Solution:**

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>

<dependencies> <dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

</dependencies>

</project>

Java code:

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

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

}

}

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("Add book: " + bookName);

bookRepository.saveBook(bookName);

}

}

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

bookService.addBook("Effective Java");

}

}

Output:

Add book: Effective Java

Save book: Effective Java

**Problem statement-2: Configuring the Spring IoC Container**

**Solution:**

<?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.BookRepo" />

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

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

</bean>

</beans>

Code:

package com.library.service;

import com.library.repository.BookRepo;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepo bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String title) {

System.out.println("BookService: Add book \"" + title + "\"");

bookRepository.saveBook(title);

}

}

package com.library.repository;

public class BookRepo {

public void saveBook(String title) {

System.out.println("BookRepository: Save book \"" + title + "\"");

}

}

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", BookService.class);

bookService.addBook("Clean Code");

}

}

Output:

BookService: Add book "Clean Code"

BookRepository: Save book "Clean Code"

**Question-2:**

**Problem statement-1: Implementing Dependency Injection**

**Solution:**

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>

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 title) {

System.out.println("BookService: Adding book \"" + title + "\"");

bookRepository.saveBook(title);

}

}

package com.library.repository;

public class BookRepository {

public void saveBook(String title) {

System.out.println("BookRepository: Saving book \"" + title + "\"");

}

}

package com.library;

import com.library.service.BookService;

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", BookService.class);

bookService.addBook("Design Patterns in Java");

}

}

**Output:**

BookService: Adding book "Design Patterns in Java"

BookRepository: Saving book "Design Patterns in Java"

**Problem statement-2: Implementing Constructor and Setter Injection**

**Solution:**

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

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

<constructor-arg ref="notificationService" />

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

</bean>

</beans>

package com.library.service;

public class NotificationService {

public void notifyUser(String message) {

System.out.println("NotificationService: " + message);

}

}

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

private NotificationService notificationService;

public BookService(NotificationService notificationService) {

this.notificationService = notificationService;

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String title) {

System.out.println("BookService: Adding book \"" + title + "\"");

bookRepository.saveBook(title);

notificationService.notifyUser("Book \"" + title + "\" has been added.");

}

}

package com.library;

import com.library.service.BookService;

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", BookService.class);

bookService.addBook("Spring in Action");

}

}

Output:

BookService: Adding book "Spring in Action"

BookRepository: Saving book "Spring in Action"

NotificationService: Book "Spring in Action" has been added.

**Question -3:**

**Problem statement-1: Creating and Configuring a Maven Project**

**Solution:**

<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>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.30</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.30</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

</dependencies>

<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></project>

**Problem statement-2: Creating a Spring Boot Application**

**Solution:**

package com.library.model;

import javax.persistence.\*;

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

public Book() {}

public Book(String title, String author) {

this.title = title;

this.author = author;

}

public Long getId() { return 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.repository;

import com.library.model.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {}

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;

@RequestMapping("/api/books")

public class BookController {

private BookRepository bookRepository;

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

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

}

@PostMapping

public Book createBook(@RequestBody Book book) {

return bookRepository.save(book);

}

@PutMapping("/{id}")

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

return bookRepository.findById(id) .map(book -> {

book.setTitle(updatedBook.getTitle());

book.setAuthor(updatedBook.getAuthor());

return bookRepository.save(book);

}).orElse(null);

}

@DeleteMapping("/{id}")

public void deleteBook(@PathVariable Long id) {

bookRepository.deleteById(id);

}

}

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);

}

}

**Question-4:**

**Poblem statement**: **Explain the difference between Java Persistence API, Hibernate and Spring Data JPA**

**Solution:**

**JPA:**

JPA is a specification, part of the official Java EE/ Jakarta EE standard.

It defines a set of interfaces and annotations for object-relational mapping (ORM), but does not implement them.

**Purpose:**

Provides a standard way to map Java objects to database tables.

Supports queries using JPQL (Java Persistence Query Language).

**Key features:**

Annotations like @Entity, @Id, @OneToMany, etc.

Query language (JPQL)

Entity Manager (EntityManager interface)

**Hibernate:**

Hibernate is a popular ORM framework that implements the JPA specification (and adds extra features).

It can be used with or without JPA.

**Purpose:**

Maps Java objects to relational tables and vice versa.

Provides a powerful query language (HQL), caching, and performance optimizations.

**Spring data JPA:**

A Spring Framework project that builds on top of JPA and Hibernate (or any JPA provider).

Provides automatic implementation of repositories, reducing boilerplate code

**Purpose:**

Makes working with JPA/Hibernate easier and faster in Spring applications.

Uses CrudRepository, JpaRepository, and custom queries with @Query.

**Key features:**

No need to write DAO or boilerplate queries

Integrates easily with Spring Boot

Supports derived query methods