Name: Vidya C S Track: JAVA FSE

Skill: Spring Core and Maven

Exercise 1: Configuring a Basic Spring Application

Scenario: Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

Code:

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>
  <dependencies>
    <!-- Spring Core Context Dependency -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.34</version>
    </dependency>
  </dependencies>
</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
                    https://www.springframework.org/schema/beans/spring-beans.xsd">
```

```
    <!-- Repository Bean -->
    <bean id="bookRepository" class="com.library.repository.BookRepository" />
    <!-- Service Bean -->
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository" />
    </bean>
</beans>
```

BookRepository.java

Path: com.library.repository.BookRepository

```
package com.library.repository;
public class BookRepository {
    public void saveBook(String bookName) {
        System.out.println("Book saved: " + bookName);
    }
}
```

BookService.java

Path: com.library.service.BookService

```
package com.library.service;
import com.library.repository.BookRepository;
public class BookService {
    private BookRepository bookRepository;
    // Setter for Dependency Injection
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }
    public void addBook(String bookName) {
        System.out.println("Adding book in service...");
        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) {

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

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

        bookService.addBook("The Alchemist");

    }

}
```

Summary:

- Created Maven project `LibraryManagement`
- Added Spring Core dependency in pom.xml
- Defined beans in applicationContext.xml
- Created BookService and BookRepository classes
- Verified functionality using a main method and got expected output

OUTPUT:

```
9         ApplicationContext context = new ClassPathXmlApplicationCon
10        BookService bookService = (BookService) context.getBean("bo
11        bookService.addBook("The Alchemist");
12    }
```

```
Console
<terminated> MainApp [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.h
Adding book in service...
Book saved: The Alchemist
```

Exercise 2: Implementing Dependency Injection

Scenario: In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

Code:

LibraryManagementApplication.java

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

        bookService.addBook("The Alchemist");

    }

}
```

BookService.java

```
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Setter for Dependency Injection

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook(String bookName) {

        System.out.println("Adding book in service...");

        bookRepository.saveBook(bookName);

    }

}
```

BookRepository.java

```
package com.library.repository;

public class BookRepository {

    public void saveBook(String bookName) {

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

    }

}
```

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">

    <!-- Repository Bean -->

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

    <!-- Service Bean with Setter Injection -->

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

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

    </bean>

</beans>
```

Summary:

This exercise demonstrates setter-based dependency injection using Spring Framework.

- BookService has a setter method for BookRepository.

- applicationContext.xml wires the dependency using <property>.

- MainApp tests the setup by calling addBook("The Alchemist").

Output:

```
9          ApplicationContext context = new ClassPathXmlApplicationCon
10         BookService bookService = (BookService) context.getBean("bo
11         bookService.addBook("The Alchemist");
12     }
<
Console
<terminated> MainApp [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.h
Adding book in service...
Book saved: The Alchemist
```

Exercise 3: Implementing Logging with Spring AOP

Scenario: The library management application requires logging capabilities to track method execution times.

Code:

LibraryManagement.java

```
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.class);

        bookService.addBook("Spring in Action");

        bookService.listBooks();

    }

}
```

BookService.java

```
package com.library.service;

import org.springframework.stereotype.Component;

import java.util.ArrayList;

import java.util.List;

@Component

public class BookService {

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

    public void addBook(String book) {

        books.add(book);

        System.out.println("Book added: " + book);

    }

    public void listBooks() {

        System.out.println("Books: " + books);

    }

}
```

Summary:

- Implemented AOP using @Aspect and @Around advice.

- LoggingAspect logs method execution time for methods in com.library.service.

- Spring configuration includes context scan and AspectJ support.

Output:

```
    public void addBook(String book) {
        books.add(book);
        System.out.println("Book added: " + book);
    }

    public void listBooks() {
        System.out.println("Books: " + books);
    }
```

```
Console
<terminated> LibraryManagementApplication [Java Application] C:\Users\DELL\.p
Book added: Spring in Action
[LOG] BookService.addBook(..) executed in 16 ms
Books: [Spring in Action]
[LOG] BookService.listBooks() executed in 1 ms
```

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.

Code:

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>LibraryManagementAOP</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
    <!-- Spring Context -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.34</version>
    </dependency>
    <!-- Spring AOP -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-aspects</artifactId>
      <version>5.3.34</version>
    </dependency>
    <!-- Spring WebMVC -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>5.3.34</version>
    </dependency>
  </dependencies>
  <build>
```

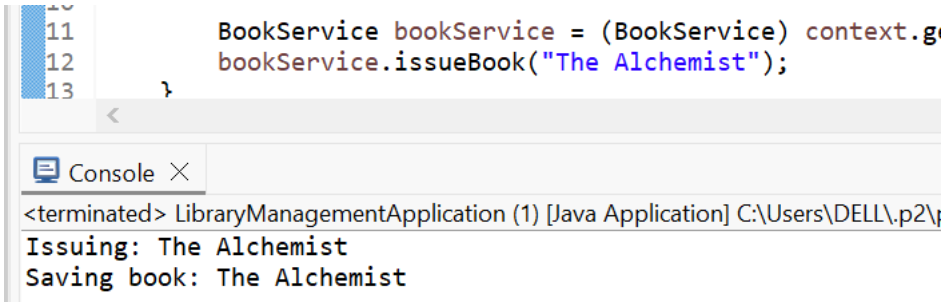
```
  <plugins>
    <!-- Maven Compiler Plugin -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
 </build>
</project>
```

Summary:

- Created a Maven project named LibraryManagementAOP using maven-archetype-quickstart.

- Added dependencies: Spring Context, Spring AOP, and Spring WebMVC (version 5.3.34).

- Configured Maven Compiler Plugin to use Java version 1.8.



Exercise 5: Configuring the Spring IoC Container

Scenario:

The library management application requires a central configuration for beans and dependencies.

Code:

LibraryManagementApplication.java

```
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 = (BookService) context.getBean("bookService");
        bookService.issueBook("The Alchemist");
    }
}
```

BookService.java

```
package com.library.service;
import com.library.repository.BookRepository;
public class BookService {
    private BookRepository bookRepository;
    // Setter method for dependency injection
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }
    // Example method to test
    public void issueBook(String bookName) {
        System.out.println("Issuing: " + bookName);
        bookRepository.save(bookName);
    }
}
```

BookRepository.java

```
package com.library.repository;

public class BookRepository {

    public void save(String bookName) {

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

    }

}
```

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">


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

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

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

    </bean>

</beans>
```

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>LibraryManagementAOP</artifactId>

  <version>0.0.1-SNAPSHOT</version>

  <dependencies>

    <!-- Spring Context -->

    <dependency>
```

```
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.34</version>
  </dependency>
  <!-- Spring AOP -->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aspects</artifactId>
    <version>5.3.34</version>
  </dependency>
  <!-- Spring WebMVC -->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.3.34</version>
  </dependency>
</dependencies>
<build>
  <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>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

Summary:

- Created applicationContext.xml to define Spring beans.

- BookRepository and BookService beans configured using XML.

- Used <property> tag to inject BookRepository into BookService.

- LibraryManagementApplication.java loads Spring context and tests the DI setup.

- Output verified: issuing and saving a book message is printed.

Output:

```
<dependencies>
```

Overview | Dependencies | Dependency Hierarchy | Effective POM

Console

```
<terminated> LibraryManagementApplication (1) [Java Applicatior
Issuing: The Alchemist
Saving book: The Alchemist
```

Exercise 6: Configuring Beans with Annotations

Scenario:

You need to simplify the configuration of beans in the library management application using annotations.

Code:

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
      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">
 <!-- Enable annotation-based configuration -->
 <context:component-scan base-package="com.library" />
```

```
  <!-- Enable AspectJ support -->
  <aop:aspectj-autoproxy />
  <!-- Register Logging Aspect -->
  <bean id="loggingAspect" class="com.library.aspect.LoggingAspect" />
</beans>
```

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 in repository: " + bookName);
    }
}
```

BookService.java

```
package com.library.service;
import org.springframework.stereotype.Service;
import java.util.ArrayList;
import java.util.List;
@Service
public class BookService {
    private List<String> books = new ArrayList<>();
    public void addBook(String book) {
        books.add(book);
        System.out.println("Book added: " + book);
    }
    public void listBooks() {
        System.out.println("Books: " + books);
    }
}
```

LibraryManagementApplication.java

```
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.class);

        bookService.addBook("The Alchemist");

        bookService.addBook("1984");

        bookService.listBooks();

    }

}
```

Summary:

- Used @Service and @Repository to define beans
- Enabled component scanning via XML
- No need for explicit <bean> tags
- Application worked with AOP logging and proper bean wiring

Output:

```
12          https://www.springframework.org/schema/
13
14  <!-- Enable annotation-based configuration -->
<

Console
<terminated> LibraryManagementApplication [Java Application] C:\Users\DEL
Book added: The Alchemist
[LOG] BookService.addBook(..) executed in 12 ms
Book added: 1984
[LOG] BookService.addBook(..) executed in 0 ms
Books: [The Alchemist, 1984]
[LOG] BookService.listBooks() executed in 0 ms
```

Exercise 8: Implementing Basic AOP with Spring

Scenario: The library management application requires basic AOP functionality to separate cross-cutting concerns like logging and transaction management.

Code:

LoggingAspect.java

```
package com.library.aspect;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.aspectj.lang.annotation.After;

@Aspect

public class LoggingAspect {

    @Before("execution(* com.library.service.*.*(..))")

    public void logBeforeMethod() {

        System.out.println("📘 [LOG] Method execution started...");

    }

    @After("execution(* com.library.service.*.*(..))")

    public void logAfterMethod() {

        System.out.println("📙 [LOG] Method execution finished...");

    }

}
```

BookService.java

```
package com.library.service;

public class BookService {

    public void issueBook() {

        System.out.println("📚 Book issued.");

    }

    public void returnBook() {

        System.out.println("📥 Book returned.");

    }

}
```

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:aop="http://www.springframework.org/schema/aop"
      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
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop.xsd">
   <!-- Enable annotation-based AOP -->
   <aop:aspectj-autoproxy/>
   <!-- Register service bean -->
   <bean id="bookService" class="com.library.service.BookService"/>
   <!-- Register aspect bean -->
   <bean class="com.library.aspect.LoggingAspect"/>
</beans>
```

LibraryManagementApplication.java

```
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 = (BookService) context.getBean("bookService");
bookService.issueBook();
bookService.returnBook();
}
}
```

Output:

```
18 }
19
```

Console

```
<terminated> LibraryManagementApplication (3) [Java Application] C:\Users\DELL\.p2\poo
[AROUND] Starting method: issueBook
[LOG] Method execution started...
Book issued.
[LOG] Method execution finished...
[AROUND] Finished method: issueBook in 29 ms
[AROUND] Starting method: returnBook
[LOG] Method execution started...
Book returned.
[LOG] Method execution finished...
[AROUND] Finished method: returnBook in 2 ms
```