LAPORAN HASIL PRAKTIKUM 4

PEMROGRAMAN BERORIENTASI OBJEK LANJUT

“Spring-3”

Dosen Pengampu :

Sri Hartati Wijono, M.Kom.

Oleh

Nama : Maria Gresia Plena Br Purba

NIM : 235314094

Kelas : DP

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS SANATA DHARMA

YOGYAKARTA

2024

A. TUJUAN

1) Memahami dan menerapkan **Spring AOP** untuk memisahkan logika lintas aplikasi (cross-cutting concerns) seperti logging, keamanan, dan validasi.
2) Menerapkan **Spring Transaction Management** untuk mengelola transaksi di aplikasi.

B. PRAKTIKUM

1. Capture code class App

```java
package com.example.library.managment;

import com.example.library.managment.controller.LibraryController;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext(configLocation: "applicationContext.xml");
        LibraryController libraryController = context.getBean(type:LibraryController.class);

        try {
            libraryController.addNewBook(isbn:"978-1234567897", title:"Spring Framework", author:"Craig Walls");
            libraryController.addNewBook(isbn:"978-9876543210", title:null, author:"James Gosling");
        } catch (IllegalArgumentException e) {
            System.out.println(x:e.getMessage());
        }
        System.out.println(x:"All books in library:");
        libraryController.displayAllBooks();
    }
}
```

Penjelasan : class ini mengimplementasikan penggunaan dari ApplicationContext yang digunakan untuk mengkonfigurasi “ApplicationContext.xml” dengan menggunakan ClassPathXmlApplicationContext. Lalu, menggunakan getBean() untuk mengambil bean LibraryController.class. Menggunakan try and catch untuk menangani pengecualian agar program tidak langsung berhenti jika terjadi kesalahan. Kemudian, memanggil method addNewBook() untuk menambahkan buku baru dan menampilkan daftar buku dengan memanggil method displayAllBooks().

2. Capture code class LoggingAspect

```java
package com.example.library.managment.aspect;

import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class LoggingAspect {

    @Before("execution(* com.example.library.managment.service.BookService.*(..))")
    public void logBeforeServiceMethods() {
        System.out.println(x:"LoggingAspect: A method in BookService is being called");
    }
}
```

Penjelasan : @Aspect untuk mendefinisikan bahwa class ini merupakan Spring Bean yang dikelola oleh Spring IoC Container. **@Component** berarti class akan didaftarkan oleh Spring sebagai bean yang dikelola oleh IoC Container. **@Before** berarti menjalankan advice sebelum eksekusi method BookService. Method logBeforeServiceMethods() berfungsi untuk menampilkan pesan jika suatu method di dalam BookService telah dipanggil.

3. Capture code class ValidationAspect

```java
package com.example.library.managment.aspect;

import com.example.library.managment.model.Book;
import org.springframework.stereotype.Component;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;

@Aspect
@Component
public class ValidationAspect {

    @Before("execution(* com.example.library.managment.service.BookService.addBook(..)) && args(book)")
    public void validateBook(Book book) {
        if (book.getTitle() == null || book.getTitle().trim().isEmpty()) {
            throw new IllegalArgumentException(s: "Book title cannot be null or empty!");
        }
        System.out.println(x: "ValidationAspect: Book title is valid");
    }
}
```

Penjelasan : @Aspect untuk mendefinisikan bahwa class ini merupakan Spring Bean yang dikelola oleh Spring IoC Container. **@Component** berarti class akan didaftarkan oleh Spring sebagai bean yang dikelola oleh IoC Container. **@Before** berarti menjalankan advice sebelum eksekusi method addBook. Mehod validateBook berfungsi untuk mengetahui apakah judul buku sudah valid atau tidak. Jika tidak valid, maka akan menampilkan pesan "ValidationAspect: Book title is valid".

4. Capture code class LibraryController

```java
package com.example.library.managment.controller;

import com.example.library.managment.model.Book;
import com.example.library.managment.service.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;

@Controller
public class LibraryController {

    @Autowired
    private BookService bookService;

    public void addNewBook(String isbn, String title, String author) {
        Book book = new Book(isbn, title, author);
        bookService.addBook(book);
    }

    public void displayAllBooks() {
        for (Book book : bookService.getAllBooks()) {
            System.out.println(x: book);
        }
    }
}
```

Penjelasan : @Controller berfungsi untuk membuat class ini sebagai Spring MVC Controller. **@Autowired** berfungsi untuk melakukan Dependency Injection atau tidak. Method addNewBook() berfungsi untuk membuat objek pada class Book dengan nama book yang memiliki parameter berisi isbn, title, author. Setelah objek Book dibuat, method ini akan memanggil method addBook() dengan parameter book yang akan disimpan ke dalam objek bookService untuk menambahkan buku baru. Method displayBooks() berfungsi untuk mengembalikan dann menampilkan daftar buku. Pada method ini, terdapat pemanggilan bookService.getAllBooks() yang berfungsi untuk mendapatkan buku-buku yang ada. Lalu, akan melakukan perulangan untuk mencetak informasi dari buku.

5. Capture code class Book

```java
package com.example.library.managment.model;

public class Book {

    private String isbn;
    private String title;
    private String author;

    public Book(String isbn, String title, String author) {
        this.isbn = isbn;
        this.title = title;
        this.author = author;
    }

    public String getIsbn() {
        return isbn;
    }

    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }

    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;
    }

    @Override
    public String toString() {
        return "Book [ISBN=" + isbn + ", Title=" + title + ", Author=" + author + "]";
    }
}
```

Penjelasan : terdapat beberapa method dengan setter dan getter. Getter berfungsi untuk mengembalikan nilai dari variabel-variabel, sedangkan setter untuk mengubah nilai dari variabel. Kemudian, terdapat method toString() yang akan mengembalikan informasi dari buku berupa isbn, title, dan author yang telah disimpan ke dalam variabel yang ada.

6. Capture code interface BookRepository

```java
package com.example.library.managment.repository;

import com.example.library.managment.model.Book;
import java.util.List;

public interface BookRepository {
    void save(Book book);
    List<Book> findAll();
}
```

Penjelasan : interface ini memiliki method save() dengan isian parameter book yang bertipe Book. Method ini berfungsi untuk menyimpan ataupun menambahkan buku baru ke dalam objek Book. Lalu, terdapat method findAll() yang berfungsi untuk mengembalikan daftar dalam bentuk List<Book>.

7. Capture code class BookRepositoryImpl

```java
package com.example.library.managment.repository;

import com.example.library.managment.model.Book;
import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Repository;

@Repository
public class BookRepositoryImpl implements BookRepository {

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

    @Override
    public void save(Book book) {
        books.add(e: book);
        System.out.println("Book saved: " + book);
    }

    @Override
    public List<Book> findAll() {
        return books;
    }
}
```

Penjelasan : class ini mengimplementasikan interface BookRepository. Pada class ini terdapat variabel books yang bertipe List<books> yang digunakan untuk menyimpan daftar buku dalam bentuk implementasi ArrayList<>. Kemudian, terdapat method save() dengan parameter book yang bertipe Book. Method ini akan memanggil method add pada objek books dengan parameter berisi book dan mencetak pesan jika buku tersimpan (“Book saved”). Method findAll() akan mengambil dan mengembalikan buku-buku yang ada setelah disimpan ke dalam objek Book.

8. Capture code interface BookService

```java
package com.example.library.managment.service;

import com.example.library.managment.model.Book;
import java.util.List;

public interface BookService {

    void addBook(Book book);

    List<Book> getAllBooks();
}
```

Penjelasan : interface ini memiliki method addBook() dengan parameter book yang bertipe Book. Method ini berfungsi untuk menambahkan objek Book baru. Method getAllBooks() akan mengambil dan mengembalikan buku-buku yang ada setelah disimpan ke dalam objek Book dalam bentuk List<Book>.

9. Capture code class BookServiceImpl

```java
package com.example.library.managment.service;

import com.example.library.managment.model.Book;
import com.example.library.managment.repository.BookRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
import org.springframework.transaction.annotation.Transactional;

@Service
public class BookServiceImpl implements BookService {

    @Autowired
    private BookRepository bookRepository;

    @Transactional
    public void addBook(Book book) {
        System.out.println("Adding book: " + book.getTitle());
        if ("Spring in Action".equals(book.getTitle())) {
            throw new RuntimeException("Error: Cannot add this book!");
        }
        bookRepository.save(book);
    }

    @Override
    public List<Book> getAllBooks() {
        return bookRepository.findAll();
    }
}
```

Penjelasan : @Service berfungsi untuk memproses logika bisnis. **@Autowired** berfungsi untuk melakukan Dependency Injection atau tidak. **@Transactional** untuk mengelola transaksi secara otomatis. Method addBook() berfungsi untuk menambahkan buku dengan memanggil method getTitle() pada objek book. Jika judul buku yang dicari tidak sama dengan “Spring in action”, maka akan menampilkan pesan. Lalu, memanggil mehod save() pada bookRepository. Method getAllBooks() akan mengambil dan mengembalikan buku-buku yang ada setelah disimpan ke dalam objek Book dalam bentuk List<Book>.

10. ApplicationContext.xml

```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"
       xmlns:tx="http://www.springframework.org/schema/tx"
       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
                           http://www.springframework.org/schema/tx
                           http://www.springframework.org/schema/tx/spring-tx.xsd">
    <!-- DataSource untuk H2 Database -->
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="org.h2.Driver"/>
    <property name="url" value="jdbc:h2:mem:testdb"/>
    <property name="username" value="sa"/>
    <property name="password" value=""/>
</bean>

    <!-- Transaction Manager -->
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"/>
</bean>
<tx:annotation-driven />

    <!-- Aktifkan scanning komponen -->
<context:component-scan base-package="com.example.library.managment" />

    <!-- Aktifkan AOP Auto Proxy -->
<aop:aspectj-autoproxy/>
<bean id="LoggingAspect" class="com.example.library.managment.aspect.LoggingAspect"/>
</beans>
```

Penjelasan : mengatur h2 database (DriverManagerDataSource), Transaction Manager (DataSourceTransactionManager), mengaktifkan scanning komponen pada com.example.library.managment, dan mengaktifkan AOP dengan <aop:aspectj-autoproxy/>.

11. Pom.xml

```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>
    <groupId>com.example</groupId>
    <artifactId>library-managment</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.10</version>
            <type>jar</type>
        </dependency>
        <!-- AspectJ -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>1.9.7</version>
            <type>jar</type>
        </dependency>
        <!-- Spring AOP -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>5.3.10</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>5.3.10</version>
        </dependency>
        <!-- H2 Database -->
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <version>2.3.232</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>5.3.32</version>
        </dependency>
    </dependencies>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <exec.mainClass>com.example.library.managment.LibraryManagment</exec.mainClass>
    </properties>
</project>
```

Penjelasan : pada bagian dependencies terdapat spring-context yang digunakan untuk mendukung fitur dari Spring, aspectjweaver untuk mengaktifkan fitur AOP menggunakan AspectJ, spring-aop untuk mendukung AOP dalam Spring, spring-tx untuk mendukung transaksi, H2 database, dan spring-jdbc mendukung operasi database.

C. DAFTAR PUSTAKA

-