ACTIVITY PERTEMUAN 5

NAMA **: Arya Wahyu Wijaya**

NPM **: 50421223**

KELAS **: 4IA14**

MATERI **: SPRING**

MATA PRAKTIKUM **: REKAYASA PERANGKAT LUNAK 2**

---

1. Apa itu Dependency dan Mengapa DI sangat penting dalam pengembangan aplikasi berbasis Spring Boot!

2. Screenshot Code Dan Output

JAWAB

1.

Dependency dalam konteks pemrograman adalah hubungan antar objek atau komponen di mana satu objek memerlukan layanan atau data dari objek lain untuk menjalankan fungsinya. Dependency ini berarti suatu kelas bergantung pada kelas lain untuk melakukan pekerjaannya.

Mengapa Dependency Injection (DI) Sangat Penting?

Meningkatkan Reusability dan Modularitas

Dengan DI, komponen atau kelas dalam aplikasi lebih modular karena dependency dapat disuntikkan dari luar dan tidak terikat langsung pada kelas yang bersangkutan. Misalnya, Service tidak harus tahu bagaimana Repository dibuat; hanya perlu interface yang tepat, sehingga Repository dapat diganti dengan implementasi lain jika diperlukan, tanpa mengubah kode Service.

Mendukung Testing dan Mocking yang Mudah

DI sangat mendukung pengujian unit dan mocking. Dengan dependency yang dapat disuntikkan, kita dapat mengganti dependency asli dengan mock atau objek tiruan saat pengujian, sehingga pengujian bisa fokus pada fungsionalitas kode tanpa bergantung pada komponen eksternal seperti database atau API.

Mempermudah Pengelolaan Dependency yang Kompleks

Dalam aplikasi yang besar, dependency antar komponen bisa sangat rumit. DI membantu Spring untuk mengatur dan mengelola dependency ini dengan lebih efisien, sehingga memudahkan pengembang dalam memelihara kode dan mencegah tight coupling antar komponen.

Mengurangi Redundansi dan Kode Boilerplate

Spring Boot secara otomatis akan membuat dan menyuntikkan dependency yang diperlukan sehingga kita tidak perlu membuat objek secara manual, mengurangi kode berulang (boilerplate code), dan memudahkan pengelolaan.

Memudahkan Skalabilitas dan Ekstensibilitas

Dengan DI, kita dapat menambahkan fitur atau memperbarui komponen tanpa perlu memodifikasi kode yang sudah ada. DI memungkinkan pengembangan aplikasi yang lebih skalabel dan fleksibel terhadap perubahan, karena dependency dapat dikelola dari luar kelas.

2.

- Pert5_50421223

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 */

package me.arya;

import me.arya.controller.MahasiswaController;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
/**
 *
 * @author arya
 */
@SpringBootApplication
public class Pert5_50421223 implements CommandLineRunner {

    @Autowired
```

```
    private MahasiswaController mhsController;

    public static void main(String[] args) {
        SpringApplication.run(Pert5_50421223.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        mhsController.tampilkanMenu();
    }
}
```

- MahasiswaController

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package me.arya.controller;

import me.arya.model.ModelMahasiswa;
import me.arya.repository.MahasiswaRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;

import java.util.List;
import java.util.Scanner;

@Controller
public class MahasiswaController {
```

```
@Autowired
private MahasiswaRepository mahasiswaRepository;

public void tampilkanMenu() {
    Scanner scanner = new Scanner(System.in);
    int opsi;

    do {
        System.out.println("\nMenu: ");
        System.out.println("1. Tampilkan semua mahasiswa");
        System.out.println("2. Tambah mahasiswa baru");
        System.out.println("3. Cek koneksi database");
        System.out.println("4. Keluar");
        System.out.println("Pilih Opsi: ");
        opsi = scanner.nextInt();
        scanner.nextLine();

        switch (opsi) {
            case 1:
                tampilkanSemuaMahasiswa();
                break;
            case 2:
                tambahMahasiswa(scanner);
                break;
            case 3:
                cekKoneksi();
                break;
            case 4:
```

```
            System.out.println("Keluar dari program");
            break;
        default:
            System.out.println("Opsi tidak valid, coba lagi");
      }
   } while (opsi != 4);
}

private void tampilkanSemuaMahasiswa() {
   List<ModelMahasiswa> mahasiswaList = mahasiswaRepository.findAll();
   if (mahasiswaList.isEmpty()) {
      System.out.println("Tidak ada data mahasiswa.");
   } else {
      mahasiswaList.forEach(mahasiswa -> System.out.println(mahasiswa));
   }
}

private void tambahMahasiswa(Scanner scanner){
   System.out.println("Masukkan NPM: ");
   String npm = scanner.nextLine();
   System.out.println("Masukkan Nama: ");
   String nama = scanner.next();
   System.out.println("Masukkan Semester: ");
   int semester = scanner.nextInt();
   System.out.println("Masukkan IPK: ");
   float ipk = scanner.nextFloat();

   ModelMahasiswa mahasiswa = new ModelMahasiswa(0, npm, nama, semester, ipk);
   mahasiswaRepository.save(mahasiswa);
```

```
        System.out.println("Mahasiswa Berhasil ditambahkan.");
    }

    private void cekKoneksi() {
        try {
            mahasiswaRepository.findAll();
            System.out.println("Koneksi ke database berhasil");
        } catch (Exception e) {
            System.out.println("Gagal terhubung ke database");
        }
    }
}
```

- ModelMahasiswa

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package me.arya.model;

import jakarta.persistence.*;
/**
 *
 * @author arya
 */
@Entity
@Table(name = "mahasiswa")
public class ModelMahasiswa {
```

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Column(name = "id")
private int id;

@Column(name = "npm", nullable = false, length = 10)
private String npm;

@Column(name = "nama", nullable = false, length = 55)
private String nama;

@Column(name = "semester")
private int semester;

@Column(name = "ipk")
private float ipk;

public ModelMahasiswa(){

}

public ModelMahasiswa(int id, String npm, String nama, int semester, float ipk){
   this.id = id;
   this.npm = npm;
   this.nama = nama;
   this.semester = semester;
   this.ipk = ipk;
}
```

```
public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getNpm() {
    return npm;
}

public void setNpm(String npm) {
    this.npm = npm;
}

public String getNama() {
    return nama;
}

public void setNama(String nama) {
    this.nama = nama;
}

public int getSemester() {
    return semester;
}

public void setSemester(int semester) {
```

```
        this.semester = semester;
    }

    public float getIpk() {
        return ipk;
    }

    public void setIpk(float ipk) {
        this.ipk = ipk;
    }
    @Override
    public String toString(){
        return "Mahasiswa{"+
                "id="+ id +
                ", npm='" + npm +'\'' +
                ", nama='" + nama +'\'' +
                ", semester='" + semester +'\'' +
                ", ipk='" + ipk +'\'' +
                '}';
    }
}
```

- MahasiswaRepository

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package me.arya.repository;
```

```
import me.arya.model.ModelMahasiswa;

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

import org.springframework.stereotype.Repository;

/**

 *

 * @author arya

 */

@Repository

public interface MahasiswaRepository extends JpaRepository<ModelMahasiswa, Long> {


}
```

- application

```
# Konfigurasi MySQL Hibernate

spring.datasource.url=jdbc:mysql://localhost:3306/spring_50421223?useSSL=false&serverTimezone=
UTC

spring.datasource.username=root

spring.datasource.password=

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver


# Hibernate settings

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true
```

- pom

```
<?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>me.arya</groupId>
    <artifactId>Pert5_50421223</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.release>17</maven.compiler.release>
        <exec.mainClass>me.arya.Pert5_50421223</exec.mainClass>
    </properties>
    <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.3.3</version>
    <relativePath/>
</parent>

<dependencies>
    <!-- Hibernate + Spring Data JPA -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <!-- MySQL Connector -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.33</version>
    </dependency>
```

```
    <!-- Spring Boot Web dependency (for MVC if needed) -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- Testing dependencies -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
</project>
```

OUTPUT

Output - **Run (Pert5_50421223)**

```
Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
2
Masukkan NPM:
50421223
Masukkan Nama:
Arya
Masukkan Semester:
7
Masukkan IPK:
4
Hibernate: insert into mahasiswa (ipk,nama,npm,semester) values (?,?,?,?)
Mahasiswa Berhasil ditambahkan.
```

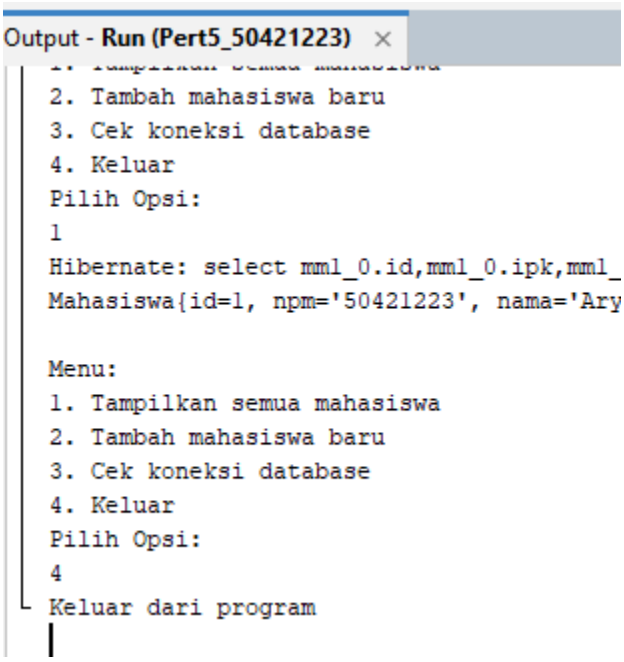
Output - **Run (Pert5_50421223)**

```
Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
3
Hibernate: select mml_0.id,mml_0.ipk,mml_0.nama,mml_0.npm,mml_0.semester from mahasiswa mml_0
Koneksi ke database berhasil
```

Output - **Run (Pert5_50421223)**

```
Hibernate: select mml_0.id,mml_0.ipk,mml_0.nama,mml_0.npm,mml_0.semester from mahasiswa mml_0
Koneksi ke database berhasil

Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mml_0.id,mml_0.ipk,mml_0.nama,mml_0.npm,mml_0.semester from mahasiswa mml_0
Mahasiswa{id=1, npm='50421223', nama='Arya', semester='7', ipk='4.0'}
```


Output - Run (Pert5_50421223)
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mml_0.id,mml_0.ipk,mml_
Mahasiswa{id=1, npm='50421223', nama='Ary
Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
4
Keluar dari program