ACTIVITY PERTEMUAN 5

NAMA : **Muhammmad Reza Rahman**

NPM : **51421051**

Kelas : **4IA14**

Materi :

---

1. **Dependency Injection (DI)** adalah pola desain di mana objek-objek (dependencies) yang dibutuhkan oleh suatu kelas disediakan (injected) oleh kerangka kerja, bukan dibuat langsung oleh kelas tersebut. Dalam konteks Spring Framework, DI memungkinkan Spring untuk mengelola dependencies secara otomatis, menciptakan objek yang diperlukan, dan menyuntikkannya ke kelas yang memerlukannya melalui anotasi seperti `@Autowired`.

   Mengapa DI penting dalam Spring Boot?

   1. **Keterpisahan Tanggung Jawab**: DI memisahkan pembuatan objek dari logika bisnis, sehingga kode lebih bersih dan modular.
   2. **Pengujian Lebih Mudah**: Dengan DI, dependencies dapat disuntikkan dengan mudah, sehingga pengujian menjadi lebih mudah dengan memanfaatkan mock dependencies.
   3. **Pemeliharaan**: Mengurangi keterikatan antar komponen (low coupling), sehingga lebih mudah diperbarui dan dipelihara.

   Dengan DI, aplikasi Spring Boot lebih fleksibel, terstruktur, dan mudah untuk dikembangkan serta dikelola.

2. Kode Program

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>
    <groupId>com.mahasiswa</groupId>
    <artifactId>Pertemuan5</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>
```

```xml
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>18</maven.compiler.source>
        <maven.compiler.target>18</maven.compiler.target>
        <exec.mainClass>com.mahasiswa.app</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>
```

Pert5_51421051

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

 package me.reza;

 import me.reza.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 ASUS
  */
 @SpringBootApplication
 public class Pert5_51421051 implements CommandLineRunner{

     @Autowired
     private MahasiswaController mhsController;
     public static void main(String[] args) {
         SpringApplication.run(Pert5_51421051.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
 */
```

```java
package me.reza.controller;

import me.reza.model.ModelMahasiswa;
import me.reza.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.print("Pilih opsi: ");
            opsi = scanner.nextInt();
            scanner.nextLine(); // menangkap newline

            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.print("Masukkan NPM : ");
        String npm = scanner.nextLine();
        System.out.print("Masukkan Nama : ");
        String nama = scanner.nextLine();
        System.out.print("Masukkan Semester : ");
        int semester = scanner.nextInt();
        System.out.print("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.reza.model;

import jakarta.persistence.*;
```

```java
/**
 *
 * @author WINDOWS 10
 */
@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;
    }
```

```java
    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 +
                ", nama='" + npm + '\'' +
                ", nama='" + nama + '\'' +
                ", nama='" + semester + '\'' +
                ", jurusan='" + ipk + '\'' +
                '}';
    }
}
```

application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/spring_51421051?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
```

Output

```
Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih opsi: 2
Masukkan NPM : 51421051
Masukkan Nama : reza
Masukkan Semester : 7
Masukkan IPK : 3.2
Hibernate: insert into mahasiswa (ipk,nama,npm,semester) values (?,?,?,?)
Mahasiswa berhasil ditambahkan.

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, nama='51421051', nama='reza', nama='7', jurusan='3.2'}

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.


Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih opsi: 4
Keluar dari program.
```