# LAPORAN HASIL PRAKTIKUM 1

# PEMROGRAMAN BERORIENTASI OBJEK LANJUT

# “ (Spring) ”

**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

Mahasiswa memahami arsitektur Spring dan konsep DI & IoC

## B. PELAKSANAAN PRAKTIKUM

### Sebelum Modifikasi

#### a. Capture Code Engine

```java
package com.example.spring.core.demo;

public class Engine {

    private String type;

    public void setType(String type) {
        this.type = type;
    }

    public void start() {
        System.out.println("Engine type " + type + " started!");
    }
}
```

**Analisis:** mendeklarasikan atribut type dengan tipe String yang bersifat private. Kemudian membuat method dengan nama setType dengan tipe void. Membuat method dengan nama start() yang akan menampilkan pesan.

#### b. Capture Code Car

```java
package com.example.spring.core.demo;

public class Car {

    private Engine engine;

    public void setEngine(Engine engine) {
        this.engine = engine;
    }

    public void drive() {
        engine.start();
        System.out.println(x: "Car is moving");
    }
}
```

**Analisis:** mendeklarasikan atribut engine bertipe Engine yang bersifat private. Kemudian terdapat method bernama setEngine dengan parameter engine yang bertipe Engine untuk mengatur nilai variabel engine. Setelah itu, method drive() yang bertipe void, memanggil method start() pada objek engine kemudian menampilkan pesan.

## c. Capture Code App

```java
package com.example.spring.core.demo;

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");

        Car car = (Car) context.getBean(string: "car");
        car.drive();
    }
}
```

**Analisis:** memuat file konfigurasi xml yang bernama applicationContext.xml menggunakan ClassPathXmlApplicationContext yang telah diimport. Kemudian, mendapatkan bean dengan nama “car” dan menyimpannya ke dalam variabel “car”. Lalu, memanggil method drive() pada objek car.

## d. Capture 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"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Konfigurasi bean untuk Engine -->
    <bean id="engine" class="com.example.spring.core.demo.Engine">
        <property name="type" value="V8"/>
    </bean>

    <!-- Konfigurasi bean untuk Car -->
    <bean id="car" class="com.example.spring.core.demo.Car">
        <property name="engine" ref="engine"/>
    </bean>
</beans>
```

**Analisis:** Terdapat konfigurasi bean untuk id engine yang merujuk ke class Engine. Bagian name digunakan untuk merujuk kepada apa yang ingin di set, yaitu “type” dan memberikan nilai “V8” pada value. Kemudian terdapat konfigurasi bean untuk id car yang merujuk ke class Car. Bagian name merujuk kepada “engine” yamg mengacu pada bean “engine” yang sudah didefinisikan sebelumnya.

## e. Capture Output

```
--- exec:3.1.0:exec (default-cli) @ sping-core-demo ---
Engine type V8 started!
Car is moving
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  2.823 s
Finished at: 2024-09-13T12:19:31+07:00
------------------------------------------------------------------------
```

## Setelah Modifikasi

### a. Capture Code Engine

```java
package com.example.spring.core.demo;

import org.springframework.stereotype.Component;

@Component
public class Engine {

    private String type = "V6";

    public void setType(String type) {
        this.type = type;
    }

    public void start() {
        System.out.println("Engine type " + type + " started!");
    }
}
```

**Analisis:** Perbedaan dari class car yang belum dimodifikasi dengan yang sudah dimodifikasi terdapat pada bagian import.Class ini meng-import Component dan menginisialisasi nilai dari variabel type.

### b. Capture Code Car

```java
package com.example.spring.core.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class Car {

    private Engine engine;

    @Autowired
    public void setEngine(Engine engine) {
        this.engine = engine;
    }

    public void drive() {
        engine.start();
        System.out.println(x: "Car is moving");
    }
}
```

**Analisis:** Perbedaan dari class car yang belum dimodifikasi dengan yang sudah dimodifikasi terdapat pada bagian import. Class ini meng-import Autowired dan Component. Component digunakan untuk menandai kelas tersebut adalah bean Spring. Sedangkan Autowired digunakan untuk menyediakan dan mengatur objek secara otomatis biasanya digunakan pada constructor dan setter.

### c. Capture 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"
       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">
    <!-- Aktifkan pencarian anotasi -->
    <context:component-scan base-package="com.example.spring.core.demo" />
</beans>
```

**Analisis:** <context:component-scan> digunakan untuk meng scan semua kelas yang terdapat pada package "com.example.spring.core.demo" untuk menemukan komponen-komponen yang terdapat di dalamnya secara otomatis

### d. Capture Output

```
--- exec:3.1.0:exec (default-cli) @ sping-core-demo ---
Engine type V6 started!
Car is moving
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.728 s
Finished at: 2024-09-13T11:39:05+07:00
------------------------------------------------------------------------
```

## C. DAFTAR PUSTAKA

-