

Activity Pertemuan 5

Nama : Simon Praja H P Pasaribu

Kelas : 4IA14

NPM : 51421422

1. Code:

```
1  /*  
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  
3   */  
4  
5   package me.simon;  
6  
7   import me.simon.controller.MahasiswaController;  
8   import org.springframework.beans.factory.annotation.Autowired;  
9   import org.springframework.boot.CommandLineRunner;  
10  import org.springframework.boot.SpringApplication;  
11  import org.springframework.boot.autoconfigure.SpringBootApplication;  
12  
13  /**  
14   * @author totes  
15   */  
16  @SpringBootApplication  
17  public class Pertemuan5_51421422 implements CommandLineRunner {  
18  
19      @Autowired  
20      private MahasiswaController mhsController;  
21  
22      public static void main(String[] args) {  
23          SpringApplication.run(Pertemuan5_51421422.class, args);  
24      }  
25  
26      @Override  
27      public void run(String... args) throws Exception {  
28          mhsController.tampilkanMenu();  
29      }  
30  }
```

```
Pertemuan5_51421422.java x ModelMahasiswa.java x MahasiswaRepository.java x MahasiswaController.java x application.properties x pom.xml [Pertemuan5_51421422] x
Source History
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package me.simon.model;
6
7  import jakarta.persistence.*;
8
9  @Entity
10 @Table(name = "mahasiswa")
11 public class ModelMahasiswa {
12
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     @Column(name = "id")
16     private int id;
17
18     @Column(name = "npm", nullable = false, length = 10)
19     private String npm;
20
21     @Column(name = "nama", nullable = false, length = 55)
22     private String nama;
23
24     @Column(name = "semester")
25     private int semester;
26
27     @Column(name = "ipk")
28     private float ipk;
29
30     public ModelMahasiswa() {}
31
32     }
33
34     public ModelMahasiswa(int id, String npm, String nama, int semester, float ipk){
35         this.id = id;
36         this.npm = npm;
37         this.nama = nama;
38         this.semester = semester;
39         this.ipk = ipk;
40     }
41
42     public int getId() {
43         return id;
44     }
45
46     public void setId(int id) {
47         this.id = id;
48     }
49 }
```

```
Pertemuan5_51421422.java x ModelMahasiswa.java x MahasiswaRepository.java x MahasiswaController.java x application.properties x pom.xml [Pertemuan5_51421422] x
Source History
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package me.simon.repository;
6
7  import me.simon.model.ModelMahasiswa;
8  import org.springframework.data.jpa.repository.JpaRepository;
9  import org.springframework.stereotype.Repository;
10 /**
11 *
12 * @author totes
13 */
14 @Repository
15 public interface MahasiswaRepository extends JpaRepository<ModelMahasiswa, Long> {
16
17 }
```

```
Pertemuan5_51421422.java x ModelMahasiswa.java x MahasiswaRepository.java x MahasiswaController.java x application.properties x pom.xml [Pertemuan5_51421422] x
Source History
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package me.simon.controller;
6
7  import me.simon.model.ModelMahasiswa;
8  import me.simon.repository.MahasiswaRepository;
9  import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.stereotype.Controller;
11
12 import java.util.List;
13 import java.util.Scanner;
14
15 @Controller
16 public class MahasiswaController {
17
18     @Autowired
19     private MahasiswaRepository mahasiswaRepository;
20
21     public void tampilkanMenu() {
22         Scanner scanner = new Scanner(System.in);
23         int opsi;
24
25         do {
26             System.out.println("\nMenu: ");
27             System.out.println("1. Tampilkan semua mahasiswa");
28             System.out.println("2. Tambah mahasiswa baru");
29             System.out.println("3. Cek koneksi database");
30             System.out.println("4. Keluar");
31             System.out.println("Pilih Opsi: ");
32             opsi = scanner.nextInt();
33             scanner.nextLine();
34
35             switch (opsi) {
36                 case 1:
37                     tampilkanSemuaMahasiswa();
38                     break;
39                 case 2:
40                     tambahMahasiswa(scanner);
41                     break;
42                 case 3:
43                     cekKoneksi();
44                     break;
45                 case 4:
46                     System.out.println("Keluar dari program");
47                     break;
48                 default:
49                     System.out.println("Opsi tidak valid. coba lagi");
50             }
51         } while (opsi != 4);
52     }
53
54     private void tampilkanSemuaMahasiswa() {
55         List<ModelMahasiswa> mahasiswaList = mahasiswaRepository.findAll();
56         if (mahasiswaList.isEmpty()) {
57             System.out.println("Tidak ada mahasiswa ditemukan");
58         } else {
59             System.out.println("Daftar Mahasiswa:");
60             for (ModelMahasiswa mahasiswa : mahasiswaList) {
61                 System.out.println(mahasiswa.toString());
62             }
63         }
64     }
65
66     private void tambahMahasiswa(Scanner scanner) {
67         System.out.println("Masukkan NPM:");
68         String npm = scanner.nextLine();
69         System.out.println("Masukkan Nama:");
70         String nama = scanner.nextLine();
71         System.out.println("Masukkan Semester:");
72         int semester = scanner.nextInt();
73         scanner.nextLine();
74         System.out.println("Masukkan IPK:");
75         double ipk = scanner.nextDouble();
76         scanner.nextLine();
77
78         MahasiswaRepository mahasiswaRepository;
79         MahasiswaRepository mahasiswaRepository = mahasiswaRepository;
80         mahasiswaRepository.save(new ModelMahasiswa(npm, nama, semester, ipk));
81         System.out.println("Mahasiswa berhasil ditambahkan");
82     }
83
84     private void cekKoneksi() {
85         boolean koneksi = true;
86         try {
87             DriverManager.getConnection("jdbc:mysql://localhost:3306/spring_514211422?useSSL=false&serverTimezone=UTC", "root", "");
88         } catch (SQLException e) {
89             koneksi = false;
90             System.out.println("Koneksi database gagal: " + e.getMessage());
91         }
92         if (koneksi) {
93             System.out.println("Koneksi database berhasil");
94         } else {
95             System.out.println("Koneksi database gagal");
96         }
97     }
98 }
application.properties x pom.xml [Pertemuan5_51421422] x
Source History
1 # Konfigurasi MySQL Hibernate
2 spring.datasource.url=jdbc:mysql://localhost:3306/spring_514211422?useSSL=false&serverTimezone=UTC
3 spring.datasource.username=root
4 spring.datasource.password=
5 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
6
7 # Hibernate settings
8 spring.jpa.hibernate.ddl-auto=update
9 spring.jpa.show-sql=true
```

OUTPUT:

```
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
2
Masukkan NPM:
51421422
Masukkan Nama:
Simon
Masukkan Semester:
7
Masukkan IPK:
```

2. Dependency Injection (DI) adalah sebuah pola desain yang digunakan untuk mengelola dan menyuntikkan (inject) dependensi antar komponen dalam aplikasi. Dalam konteks Spring Framework, DI mengacu pada cara Spring mengelola objek-objek dalam aplikasi dan menyediakan objek-objek tersebut ke dalam komponen lain sesuai dengan kebutuhan.