LAPORAN AKHIR PRAKTIKUM

Mata Praktikum : Rekayasa Perangkat Lunak 2

Kelas : 4IA06

Praktikum ke- : 5

Tanggal : 12 November 2024

Materi : : Konsep Framework Spring, Pembuatan Project Spring dan Hibernate.

NPM : 50421130

Nama : Alvi Haikal Farwiza

Ketua Asisten : Gilbert Jefferson Faozato Mendrofa

Paraf Asisten :

Nama Asisten :

Jumlah Lembar : 7 Lembar

LABORATORIUM TEKNIK INFORMATIKA
UNIVERSITAS GUNADARMA
2024

1. Jelaskan Apa itu Springboot dan bagaimana perbedaanya dengan kode pertemuan sebelumnya. Apa keuntungan utama yang ditarwan spring boot bagi pengembang aplikasi?

Springboot adalah salah satu framework untuk aplikasi berbasais java, tepatnya JEE. Springboot merupakan sebuah framework (kerangka Kerja) yang digunakan untuk membangun sebuah aplikasi Enterprise. Spring termasuk framework yang ringan untuk mendukung secara penuh dalam pengembangan enterprise siap pakai.

Code Pertemuan 4 : Menggunakan Hibernate secara manual untuk mengelola database dan kode lebih kompleks. Menggunakan file hibernate.cfg.xml untuk konfigurasi Hibernate secara manual.

Code Pertemuan 5 : Menggunakan Spring Boot dengan Hibernate (Spring Data JPA) yang otomatis mengelola pengaturan Hibernate, sehingga kode lebih ringkas dan lebih mudah dibaca. Menggunakan application.properties Spring Boot untuk konfigurasi database dan Hibernate.

Keuntungan Utama :

- Mengurangi waktu pengembangan dan meningkatkan produktivitas
- Mengintegrasikan aplikasi dalam rangkaian proyek Spring
- Menawarkan plugin dan alat untuk mempermudah pengembangan
- Mengurangi perlunya menulis kode boilerplate, anotasi, dan konfigurasi XML
- Menyediakan alat pengembangan/pengujian

2.Jelaskan kode dan langkah-langkah program yang telah dibuat!

1. Code File Pom.xml :

dependencies: Daftar library atau framework yang digunakan dalam proyek:

- spring-boot-starter-data-jpa: Untuk integrasi Hibernate dan JPA.
- mysql-connector-java: Untuk koneksi ke MySQL.
- spring-boot-starter-web: Untuk membangun aplikasi web berbasis Spring Boot.
- spring-boot-starter-test: Untuk kebutuhan testing.

```
<!-- Hibernate + Spring Data JPA -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactI
    <version>3.3.3</version>
</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>
```

build: Konfigurasi untuk proses build:

- spring-boot-maven-plugin: Plugin Maven untuk menjalankan aplikasi Spring Boot

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

2. Code File Pertemuan5_50421130.java :

@SpringBootApplication: Annotation untuk menandai kelas sebagai entry point aplikasi Spring Boot. CommandLineRunner: Interface yang memungkinkan menjalankan logika tambahan ketika aplikasi Spring Boot dimulai.

```
@SpringBootApplication
public class Pertemuan5_50421130 implements CommandLineRunner{
```

main(String[] args): Fungsi utama yang menjalankan aplikasi.

```
public static void main(String[] args) {
    SpringApplication.run(Pertemuan5_50421130.class, args);
}
```

run(String... args): Method dari CommandLineRunner yang digunakan untuk memanggil method tampilkanMenu() di MahasiswaController.

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

3. Code File MahasiswaController.java :

@Controller: Annotation untuk mendefinisikan kelas sebagai controller dalam Spring MVC.

```
@Controller
public class MahasiswaController {
```

tampilkanMenu(): Menampilkan menu pengguna dan menggunakan Scanner untuk membaca input.

- Opsi 1: Menampilkan semua data mahasiswa dari database.
- Opsi 2: Menambahkan data mahasiswa baru ke database.
- Opsi 3: Mengecek koneksi database.
- Opsi 4: Keluar dari program.

```
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;
```

tampilkanSemuaMahasiswa(): Mengambil dan menampilkan daftar semua mahasiswa dari repository.

```
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));
    }
}
```

tambahMahasiswa(Scanner scanner): Membaca input dari pengguna, membuat objek ModelMahasiswa, dan menyimpannya ke database menggunakan repository.

```
private void tambahMahasiswa(Scanner scanner){
    System.out.println("Masukkan NPM: ");
    String npm = scanner.nextLine();
    System.out.println("Masukkan Nama: ");
    String nama = scanner.nextLine();
    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");

}
```

cekKoneksi(): Mengecek apakah koneksi ke database berhasil dengan mencoba operasi findAll() pada repository.

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

4. Code File ModelMahasiswa.java :

@Entity untuk menandakan bahwa ini adalah entitas JPA dan @Table(name = "mahasiswa") untuk menghubungkan entitas dengan tabel mahasiswa, dengan atribut **id**, **npm**, **nama**, **semester**, **ipk**

```java
@Entity
@Table(name = "mahasiswa")
public class ModelMahasiswa {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    @Column(name = "npm", nullable = false, length = 8)
    private String npm;

    @Column(name = "nama", nullable = false, length = 50)
    private String nama;

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

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

    public ModelMahasiswa(){

    }
```

Getter dan Setter: Memberikan akses ke atribut kelas secara langsung. Misalnya, getNama() dan setNama(String nama) digunakan untuk mengambil dan menetapkan nama mahasiswa.

```java
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;
}
```

5. Code File MahasiswaRepository.java :

JpaRepository<ModelMahasiswa, Long>: Interface bawaan Spring Data JPA untuk operasi CRUD (Create, Read, Update, Delete).

```java
/*
 * 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.alvi.repository;
import me.alvi.model.ModelMahasiswa;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;


@Repository
public interface MahasiswaRepository extends JpaRepository<ModelMahasiswa, Long>{

}
```

6. Code File Application.properties.java :

Database Configuration:

- spring.datasource.url: URL koneksi database MySQL.
- spring.datasource.username: Username database.
- spring.datasource.password: Password database.
- spring.datasource.driver-class-name: Driver JDBC untuk MySQL.

```
#konfigurasi Mysql Hibernate
spring.datasource.url = jdbc:mysql://localhost:3306/spring_50421130?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: Mengatur bagaimana Hibernate menangani schema database. Nilai update berarti Hibernate akan menyesuaikan schema tanpa menghapus data.
- spring.jpa.show-sql: Menampilkan SQL yang dieksekusi oleh Hibernate.

```
#Hibernate settings
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```