LAPORAN AKHIR PRAKTIKUM

Mata Praktikum : Rekayasa Perangkat Lunak 2

Kelas : 4IA06

Praktikum ke- : 6

Tanggal : 19 November 2024

Materi : Implementasi Aspect Oriented Programming (AOP) dan Depedency Injection pada project spring dan hibernate

NPM : 50421130

Nama : Alvi Haikal Farwiza

Ketua Asisten : Gilbert Jefferson Faozato Mendrofa

Paraf Asisten :

Nama Asisten :

Jumlah Lembar : 10 Lembar

LABORATORIUM TEKNIK INFORMATIKA
UNIVERSITAS GUNADARMA
2024

Jelaskan satu per satu codingan kalian dari hasil screenshot activity!

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.

```
<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>
```

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>
</project>
```

2. Code File mahasiswaController.java :

addMahasiswa() : Fungsi ini menerima objek modelMahasiswa melalui request body, kemudian memanggil metode addMhs() dari service mahasiswaService untuk menambahkan data mahasiswa ke dalam database. Setelah itu, mengembalikan pesan bahwa mahasiswa berhasil ditambahkan.

```
public String addMahasiswa(@RequestBody modelMahasiswa mhs){
    mahasiswaService.addMhs(mhs);
    return "Mahasiswa Added Successfully";
}
```

getMahasiswa() : Fungsi ini menerima id mahasiswa sebagai parameter dari path URL. Kemudian, menggunakan mahasiswaService untuk mengambil data mahasiswa berdasarkan ID yang diberikan. Fungsi ini mengembalikan objek modelMahasiswa yang ditemukan berdasarkan ID.

```
public modelMahasiswa getMahasiswa(@PathVariable int id){
    return mahasiswaService.getMhs(id);
}
```

updateMahasiswa() : Fungsi ini menerima objek modelMahasiswa melalui request body dan memanggil metode updateMhs() pada service mahasiswaService untuk memperbarui data mahasiswa dalam database.

```
public String updateMahasiswa(@RequestBody modelMahasiswa mhs){
    mahasiswaService.updateMhs(mhs);
    return "Mahasiswa Updated Successfully";
}
```

deleteMahasiswa(): Fungsi ini menerima id mahasiswa dari path URL, kemudian memanggil deleteMhs() pada service mahasiswaService untuk menghapus data mahasiswa dengan ID yang diberikan.

```
public String deleteMahasiswa(@PathVariable int id){
    mahasiswaService.deleteMhs(id);
    return "Mahasiswa Deleted Successfully";
}
```

getAllMahasiswa(): Fungsi ini memanggil metode getAllMahasiswa() dari service mahasiswaService untuk mengambil daftar semua mahasiswa yang ada dalam database dan mengembalikannya dalam bentuk list.

```
public List<modelMahasiswa> getAllMahasiswa(){
    return mahasiswaService.getAllMahasiswa();
}
```

3. Code File MahasiswaService.java :

addMhs(modelMahasiswa mhs): Fungsi ini menerima objek modelMahasiswa dan menyimpannya ke dalam database dengan memanggil repository.save(mhs).

```
public void addMhs(modelMahasiswa mhs){
    repository.save(mhs);
}
```

getMhs(int id): Fungsi ini mencari mahasiswa berdasarkan ID yang diberikan dengan memanggil repository.findById(id) yang mengembalikan data mahasiswa jika ditemukan, atau null jika tidak ada data dengan ID tersebut.

```
public modelMahasiswa getMhs(int id){
    return repository.findById(id).orElse(null);
}
```

updateMhs(modelMahasiswa mhs): Fungsi ini menerima objek modelMahasiswa dan menyimpannya kembali ke database dengan memanggil repository.save(mhs). Fungsi ini berfungsi untuk memperbarui data mahasiswa jika sudah ada di database.

```
public void updateMhs(modelMahasiswa mhs){
    repository.save(mhs);
}
```

deleteMhs(int id): Fungsi ini menghapus mahasiswa berdasarkan ID yang diberikan dengan memanggil repository.deleteById(id).

```
public void deleteMhs(int id){
    repository.deleteById(id);
}
```

getAllMahasiswa(): Fungsi ini mengembalikan daftar semua mahasiswa dengan memanggil repository.findAll() yang mengembalikan list dari semua data mahasiswa dalam database.

```
public List<modelMahasiswa> getAllMahasiswa(){
    return repository.findAll();
}
```

4. modelMahasiswa.java

Annotations: Anotasi @Entity menandai kelas ini sebagai entitas Hibernate, dan @Table(name = "mahasiswa") menandai tabel yang terkait. Setiap kolom di tabel ini diwakili oleh atribut kelas yang memiliki anotasi seperti @Column.

```
6   @Entity
7   @Table(name = "mahasiswa")
8   public class modelMahasiswa {
9       @Id
10      @GeneratedValue(strategy = GenerationType.IDENTITY)
11      @Column(name = "id")
12      private int id;
13
14      @Column(name = "npm", nullable = false, length = 8)
15      private String npm;
16
17      @Column(name = "nama", nullable = false, length = 50)
18      private String nama;
19
20      @Column (name = "semester")
21      private int semester;
22
23      @Column (name = "ipk")
24      private float ipk;
25
26      public modelMahasiswa(){
27
28      }
```

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

```
28      }
29
30      public modelMahasiswa(int id, String npm, String nama, int semester, float ipk){
31          this.id = id;
32          this.npm = npm;
33          this.nama = nama;
34          this.semester = semester;
35          this.ipk = ipk;
36      }
37
38      public int getId() {
39          return id;
40      }
41
42      public void setId(int id) {
43          this.id = id;
44      }
45
46      public String getNpm() {
47          return npm;
48      }
49
50      public void setNpm(String npm) {
51          this.npm = npm;
52      }
53
54      public String getNama() {
```

5. File Code mahasiswaView.java

mahasiswaView(mahasiswaController controller): Konstruktor kelas ini yang menerima parameter mahasiswaController. Konstruktor ini menginisialisasi tampilan GUI dan memuat data mahasiswa ke dalam tabel menggunakan loadMahasiswaTable().

```
public mahasiswaView(mahasiswaController controller){
this.controller = controller;
initComponents();
loadMahasiswaTable();
}

public mahasiswaView() {
    throw new UnsupportedOperationException("Not Supported Yet.");
}
```

loadMahasiswaTable(): Fungsi ini memuat data mahasiswa ke dalam tabel dengan memanggil controller.getAllMahasiswa() yang akan mengembalikan daftar mahasiswa. Data ini kemudian dimasukkan ke dalam modelTableMahasiswa, yang digunakan sebagai model untuk dataTable.

```
public void loadMahasiswaTable(){
    List<modelMahasiswa> listMahasiswa = controller.getAllMahasiswa();
    modelTableMahasiswa tableModel = new modelTableMahasiswa(listMahasiswa);
    dataTable.setModel(tableModel);
}
```

jButton1ActionPerformed(java.awt.event.ActionEvent evt): Fungsi ini dijalankan ketika tombol "Save" diklik. Fungsi ini mengambil input dari kolom-kolom di formulir, membuat objek modelMahasiswa baru dengan data tersebut, dan kemudian mengirimkannya ke controller untuk ditambahkan ke dalam database. Setelah itu, data di-refresh dengan memanggil loadMahasiswaTable() dan kolom-kolom formulir dibersihkan.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String npm = jTextField1.getText();
    String nama = jTextField2.getText();
    int semester = Integer.parseInt(jTextField3.getText());
    float ipk = Float.parseFloat(jTextField4.getText());
    modelMahasiswa mahasiswa = new modelMahasiswa(0, npm, nama, semester, ipk);
    System.out.println(mahasiswa.getIpk());
    System.out.println(mahasiswa.getNama());
    System.out.println(mahasiswa.getSemester());
    System.out.println(mahasiswa.getNpm());

    controller.addMahasiswa(mahasiswa);
    loadMahasiswaTable();

    jTextField1.setText("");
    jTextField2.setText("");
    jTextField3.setText("");
    jTextField4.setText("");

}
```

jButton2ActionPerformed(java.awt.event.ActionEvent evt): Fungsi ini dijalankan ketika tombol "Refresh" diklik. Fungsi ini memanggil loadMahasiswaTable() untuk memperbarui data yang ditampilkan pada tabel mahasiswa.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    loadMahasiswaTable();
}
```

jButton3ActionPerformed(java.awt.event.ActionEvent evt): Fungsi ini dijalankan ketika tombol "Delete" diklik. Fungsi ini meminta input ID mahasiswa yang ingin dihapus melalui dialog input, dan jika ID yang dimasukkan valid, maka fungsi ini akan memanggil controller.deleteMahasiswa(id) untuk menghapus mahasiswa yang bersangkutan dari database. Setelah itu, tabel data mahasiswa diperbarui.

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JTextField idField = new JTextField(10);

    JPanel panel = new JPanel();
    panel.add(new JLabel("Masukan ID yang ingin dihapus"));
    panel.add(idField);

    int result = JOptionPane.showConfirmDialog(null, panel, "Hapus Mahasiswa",JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAI

    if(result == JOptionPane.OK_OPTION){
        try{
            int id = Integer.parseInt(idField.getText());
            controller.deleteMahasiswa(id);
            JOptionPane.showMessageDialog(null,"Data berhasil dihapus", "Sukses", JOptionPane.INFORMATION_MESSAGE);
            loadMahasiswaTable();
        }catch(NumberFormatException e){
            JOptionPane.showMessageDialog(null, "ID harus berupa Angka", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

6. File Code modelTableMahasiswa.java

getRowCount(): Mengembalikan jumlah baris dalam tabel (jumlah elemen mahasiswaList).

```
public class modelTableMahasiswa extends AbstractTableModel {
    private List<modelMahasiswa> mahasiswaList;
    private String[] columnNames = {"ID", "NPM", "NAMA", "SEMESTER", "IPK"};

    public modelTableMahasiswa(List<modelMahasiswa> mahasiswaList){
        this.mahasiswaList = mahasiswaList;
    }

    @Override
    public int getRowCount(){
        return mahasiswaList.size();
    }
```

getColumnCount(): Mengembalikan jumlah kolom dalam tabel (columnNames.length).

```
@Override
public int getColumnCount(){
    return columnNames.length;
}
```

getValueAt(int rowIndex, int columnIndex): Mengambil nilai pada baris dan kolom tertentu dalam tabel, berdasarkan atribut modelMahasiswa.

```
@Override
public Object getValueAt(int rowIndex, int columnIndex){
    modelMahasiswa mahasiswa = mahasiswaList.get(rowIndex);
    switch (columnIndex){
        case 0:
            return mahasiswa.getId();
        case 1:
            return mahasiswa.getNpm();
        case 2:
            return mahasiswa.getNama();
        case 3:
            return mahasiswa.getSemester();
        case 4:
            return mahasiswa.getIpk();
        default:
            return null;
    }
}
```

getColumnName(int column): Mengembalikan nama kolom sesuai dengan columnNames.

```
@Override
public String getColumnName(int column){
    return columnNames[column];
}
```

isCellEditable(): Mengatur agar sel tabel tidak dapat diedit.

```
@Override
public boolean isCellEditable(int rowIndex, int columnIndex){
    return false;
}
```

setMahasiswaList(List<modelMahasiswa> mahasiswaList): Mengubah data yang ditampilkan di tabel dan memperbarui tabel.

```
public void setMahasiswaList(List<modelMahasiswa> mahasiswaList){
    this.mahasiswaList = mahasiswaList;
    fireTableDataChanged();
}
```

7. File Code mahasiswaRepository.java

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

```
1
2   package com.mahasiswa.repository;
3
4   import com.mahasiswa.model.modelMahasiswa;
5   import org.springframework.data.jpa.repository.JpaRepository;
6   import org.springframework.stereotype.Repository;
7
8   @Repository
9   public interface mahasiswaRepository extends JpaRepository<modelMahasiswa, Integer> {
10
11  }
12
```

8. File Code application.properties

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/pertemuan6_db?useSSL=false&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.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
```