

# **LAPORAN AKHIR PRAKTIKUM**

Mata Praktikum : Rekayasa Perangkat Lunak 2

Kelas : 4IA06

Praktikum ke- : 4

Tanggal : 5 November 2024

Materi : : Konsep Dasar Object Relational Mapping (ORM)  
dan Framework 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:** Bagian ini mengelola pustaka pihak ketiga yang diperlukan oleh proyek. Di sini, ada dua dependensi:

- hibernate-core: Untuk ORM (Object-Relational Mapping) dengan Hibernate.
- mysql-connector-java: Untuk menghubungkan aplikasi dengan database MySQL.

```
<dependencies>
  <dependency>
    <groupId>org.hibernate.orm</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>6.6.0.Final</version>
  </dependency>

  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.33</version>
  </dependency>
</dependencies>
```

**Build Settings:** Bagian build berisi konfigurasi untuk menentukan encoding dan mengarahkan ke file sumber daya (src/main/resources).

```
<build>
  <resources>
    <resource>
      <directory>src/main/resources</directory>
      <filtering>false</filtering>
    </resource>
  </resources>
</build>
```

## 2. Code File mahasiswaController.java :

**addMhs(modelMahasiswa mhs):** Fungsi untuk menambahkan objek modelMahasiswa ke database. Membuka sesi, memulai transaksi, menyimpan objek mhs, dan kemudian commit transaksi.

```
public class mahasiswaController {  
    public void addMhs (modelMahasiswa mhs) {  
        Transaction trx = null;  
  
        try (Session session = hibernateUtil.getSessionFactory().openSession()) {  
            trx = session.beginTransaction();  
            session.save(mhs);  
            trx.commit();  
        } catch (Exception e) {  
            if (trx != null) {  
                trx.rollback();  
            }  
            e.printStackTrace();  
        }  
    }  
}
```

**updateMhs(modelMahasiswa mhs):** Fungsi untuk memperbarui data mahasiswa dalam database. Sama seperti addMhs, tetapi dengan operasi update.

```
public void updateMhs (modelMahasiswa mhs) {  
    Transaction trx = null;  
  
    try (Session session = hibernateUtil.getSessionFactory().openSession()) {  
        trx = session.beginTransaction();  
        session.update(mhs);  
        trx.commit();  
    } catch (Exception e) {  
        if (trx != null) {  
            trx.rollback();  
        }  
        e.printStackTrace();  
    }  
}
```

**deleteMhs(int id):** Fungsi untuk menghapus data mahasiswa berdasarkan id. Mengambil data dari database berdasarkan id, dan jika ditemukan, data tersebut dihapus.

```
public void deleteMhs(int id){
    Transaction trx = null;

    try (Session session = hibernateUtil.getSessionFactory().openSession()){
        trx = session.beginTransaction();
        modelMahasiswa mhs = session.get(modelMahasiswa.class, id);
        if(mhs!= null){
            session.delete(mhs);
            System.out.println("Berhasil dihapus");
        }
        trx.commit();
    }catch (Exception e){
        if (trx !=null){
            trx.rollback();
        }
        e.printStackTrace();
    }
}
```

**getAllMahasiswa():** Mengambil seluruh data modelMahasiswa dari database menggunakan Query dan mengembalikannya sebagai List<modelMahasiswa>.

```
59 public List<modelMahasiswa> getAllMahasiswa(){
60     Transaction trx = null;
61     List<modelMahasiswa> listMhs = null;
62
63
64     try (Session session = hibernateUtil.getSessionFactory().openSession()){
65         trx = session.beginTransaction();
66         Query<modelMahasiswa> query = session.createQuery("from modelMahasiswa", modelMahasiswa.class);
67         listMhs = query.list();
68         trx.commit();
69     }catch (Exception e){
70         if (trx !=null){
71             trx.rollback();
72         }
73         e.printStackTrace();
74     }
75     return listMhs;
76 }
77 }
```

### 3. Code File hibernateUtil.java :

**getSessionFactory():** Fungsi yang mengembalikan objek SessionFactory, memungkinkan sesi Hibernate baru dapat dibuka.

```
public class hibernateUtil {  
    private static SessionFactory sessionFactory;  
  
    static{  
        try{  
            sessionFactory = new Configuration().configure().buildSessionFactory();  
        }catch(Throwable ex){  
            System.err.println("initial SessionFactory Creation failed" + ex);  
            throw new ExceptionInInitializerError(ex);  
        }  
    }  
  
    public static SessionFactory getSessionFactory(){  
        return sessionFactory;  
    }  
}
```

**testConnection():** Fungsi untuk menguji koneksi ke database dengan membuka sesi dan memberikan pesan keberhasilan jika terhubung.

```
    }  
  
    public static void testConnection(){  
        try (Session session = sessionFactory.openSession()){  
            System.out.println("Connection to the database was succesfull");  
        }catch (Exception e){  
            System.err.println("Failed connect to the database");  
            e.printStackTrace();  
        }  
    }  
}
```

#### 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`.

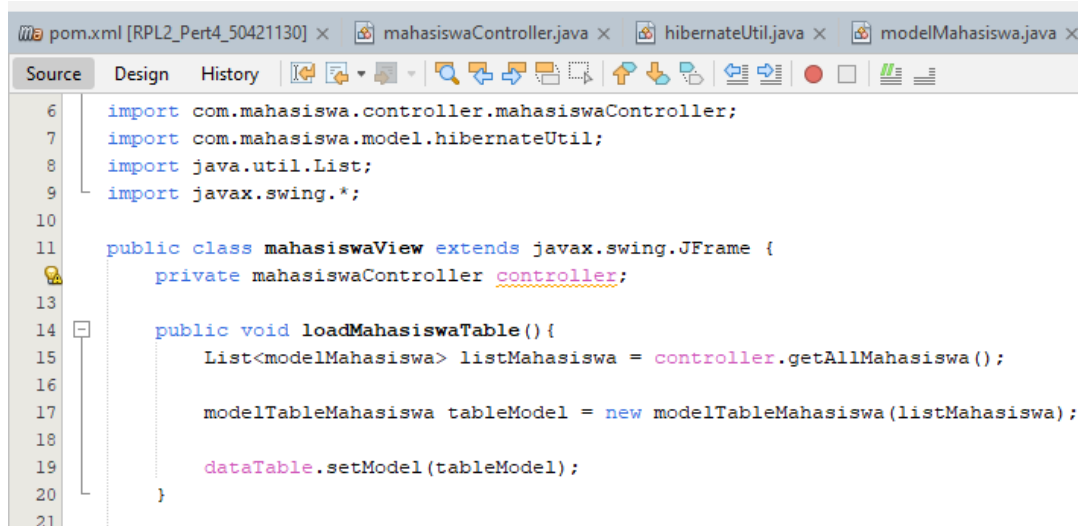
```
5
6  @Entity
7  @Table(name = "mahasiswa")
8  public class modelMahasiswa {
9      @Id
10     @GeneratedValue(strategy = GenerationType.IDENTITY)
11     @Column(name = "id")
12     private int id;
13
14     @Column(name = "npm", nullable = false, length = 8)
15     private String npm;
16
17     @Column(name = "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.

```
27
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

**loadMahasiswaTable():** Memuat data mahasiswa dari database ke dalam tabel GUI menggunakan modelTableMahasiswa.



```
6 import com.mahasiswa.controller.mahasiswaController;
7 import com.mahasiswa.model.hibernateUtil;
8 import java.util.List;
9 import javax.swing.*;
10
11 public class mahasiswaView extends javax.swing.JFrame {
12     private mahasiswaController controller;
13
14     public void loadMahasiswaTable() {
15         List<modelMahasiswa> listMahasiswa = controller.getAllMahasiswa();
16
17         modelTableMahasiswa tableModel = new modelTableMahasiswa(listMahasiswa);
18
19         dataTable.setModel(tableModel);
20     }
21 }
```

**clearTextField():** Menghapus konten dari semua bidang teks input.

```
public void clearTextField() {
    jTextField1.setText("");
    jTextField2.setText("");
    jTextField3.setText("");
    jTextField4.setText("");
}
```

**jButton1ActionPerformed():** Event handler untuk tombol "Simpan". Membaca data input dari teks, membuat objek modelMahasiswa, dan menambahkannya ke database menggunakan controller.addMhs.

```
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.addMhs(mahasiswa);
    loadMahasiswaTable();
    clearTextField();
}
```

**jButton2ActionPerformed():** Event handler untuk tombol "Hapus". Membaca id dari input, lalu memanggil `controller.deleteMhs` untuk menghapus data.

```
179 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
180     // TODO add your handling code here:  
181     JTextField idField = new JTextField(5);  
182  
183     JPanel panel = new JPanel();  
184     panel.add(new JLabel("Masukan ID yang ingin dihapus"));  
185     panel.add(idField);  
186  
187     int result = JOptionPane.showConfirmDialog(null, panel, "Hapus Mahasiswa", JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);  
188  
189     if(result == JOptionPane.OK_OPTION){  
190         try{  
191             int id = Integer.parseInt(idField.getText());  
192             controller.deleteMhs(id);  
193             JOptionPane.showMessageDialog(null, "Data berhasil dihapus", "Sukses", JOptionPane.INFORMATION_MESSAGE);  
194             loadMahasiswaTable();  
195         }catch(NumberFormatException e){  
196             JOptionPane.showMessageDialog(null, "ID harus berupa Angka", "Error", JOptionPane.ERROR_MESSAGE);  
197         }  
198     }  
199 }
```

## 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 hibernate.cfg.xml

**Database Connection Settings:** Menyimpan informasi koneksi ke database, seperti hibernate.connection.url (URL koneksi), username, dan password.

```

<hibernate-configuration>
  <session-factory>
    <!-- Database Connection Settings -->
    <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernate_db</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password"></property>

```

**Connection Pool Settings:** Mengatur parameter pool koneksi menggunakan C3P0, seperti jumlah minimum (min\_size) dan maksimum (max\_size) koneksi aktif.

```

<!-- JDBC connection pool settings -->
<property name="hibernate.c3p0.min_size">5</property>
<property name="hibernate.c3p0.max_size">20</property>
<property name="hibernate.c3p0.timeout">300</property>
<property name="hibernate.c3p0.max_statements">50</property>
<property name="hibernate.c3p0.idle_test_period">3000</property>

```

**SQL Dialect:** Menentukan hibernate.dialect untuk MySQL, yang membantu Hibernate menghasilkan query SQL yang kompatibel dengan MySQL.

```

<!-- SQL dialect -->
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<!-- Echo all executed SQL to stdout -->
<property name="hibernate.show_sql">true</property>

<!-- Drop and re-create the database schema on startup -->
<property name="hibernate.hbm2ddl.auto">update</property>

```