PRAKTIKUM 2
MEMBUAT FUNGSI CRUD (Create, Read, Update, Delete) USER DENGAN DATABASE MYSQL

1.1 Tujuan

Tujuan praktikum ini yaitu mahasiswa mampu membuat fungsi CRUD data user menggunakan database MySQL, Adapun poin-poin praktikum yaitu :

- Mahasiswa mampu membuat table user pada database MySQL
- Mahasiswa mampu membuat koneksi Java dengan database MySQL
- Mahasiswa mampu membuat tampilan GUI CRUD user
- Mahasiswa mampu membuat dan mengimplementasikan interface
- Mahasiswa mampu membuat fungsi DAO (Data Access Object) dan mengimplementasikannya.
- Mahasiswa mampu membuat fungsi CRUD dengan menggunakan konsep Pemrograman Berorientasi Objek

1.2 Alat

- Computer / laptop yang telah terinstall JDK dan Eclipse
- MySQL
- MySQL connector atau Connector/J

1.3 Teori

MySQL adalah sebuah relational database management system (RDBMS) open-source yang digunakan dalam pengelolaan database suatu aplikasi, MySQL ini dapat digunakan untuk menyimpan, mengelola dan mengambil data dalam format table.

Logo MySQL

MySQL Connection/j adalah driver yang digunakan untuk menghubungkan aplikasi berbasis java dengan database MySQL sehingga dapat berinteraksi seperti menyimpan, mengubah, mengambil dan menghapus data. Beberapa fungsi MySQL connector yaitu :

- Membuka koneksi ke database MySQL
- Mengirimkan permintaan SQL ke server MySQL
- Menerima hasil dari permintaan SQL
- Menutup koneksi ke database MySQL

DAO (Data Access Object) merupakan object yang menyediakan abstract interface terhadap beberapa method yang berhubungan dengan database seperti mengambil data (read), menyimpan data(create), menghapus data (delete), mengubah data(update). Tujuan penggunaan DAO yaitu :

- Meningkatkan modularitas yaitu memisahkan logika akses data dengan logika bisnis sehingga memudahkan untuk dikelola
- Meningkatkan reusabilitas yaitu DAO dapat digunakan Kembali
- Perubahan pada logika akses data dapat dilakukan tanpa mempengaruhi logika bisnis.

Interface dalam Bahasa java yaitu mendefinisikan beberapa method abstrak yang harus diimplementasikan oleh class yang akan menggunakannya.

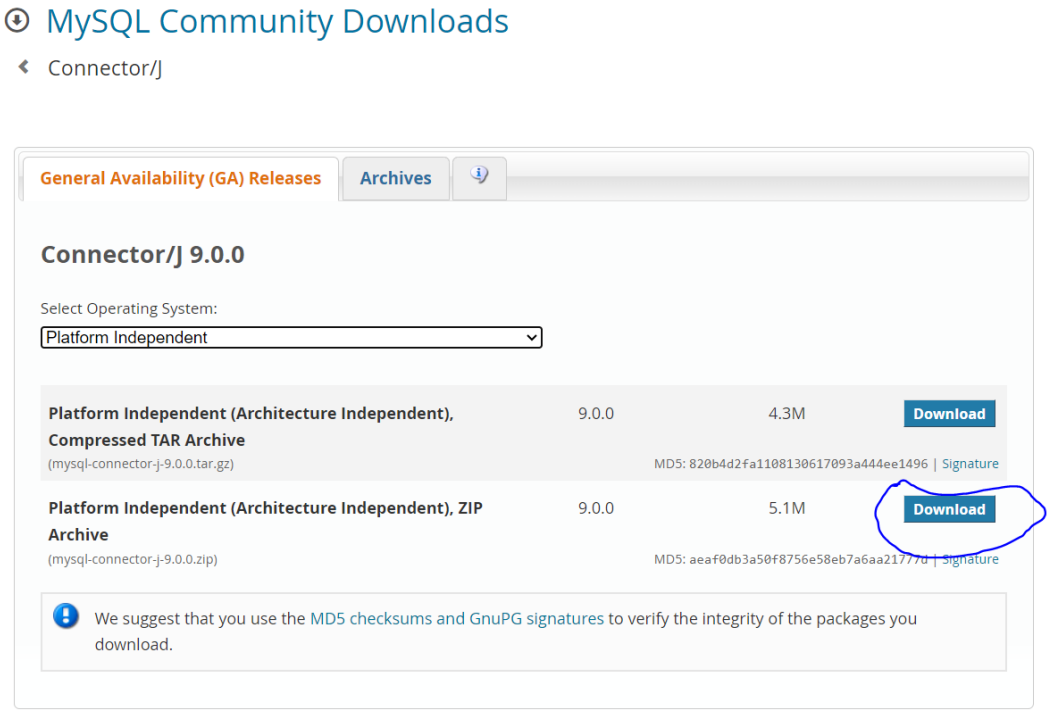
CRUD (Create, Read, Update, Delete) merupakan fungsi dasar atau umum yang ada pada sebuah aplikasi yang mana fungsi ini dapat membuat, membaca, mengubah dan menghapus suatu data pada database aplikasi.

1.4 Langkah-langkah

Menambahkan MySQL Connector

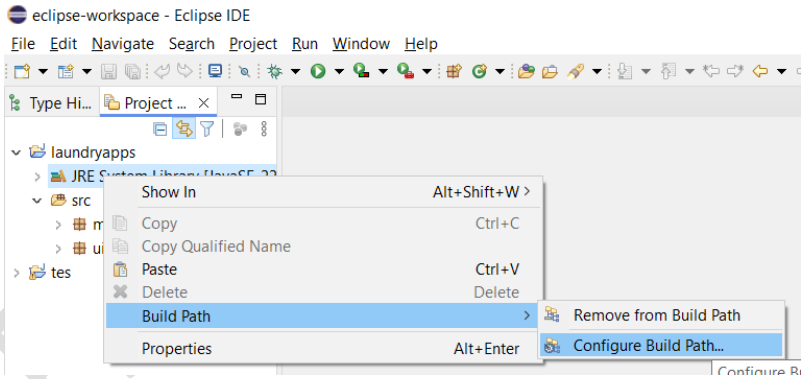
Aplikasi java agar dapat terhubung dengan database MySQL membutuhkan sebuah driver yaitu MySQL Connection, berikut ini Langkah-langkah membuat koneksi Database MySQL.

- Download MySQL connection pada link berikut
  https://dev.mysql.com/downloads/connector/j/

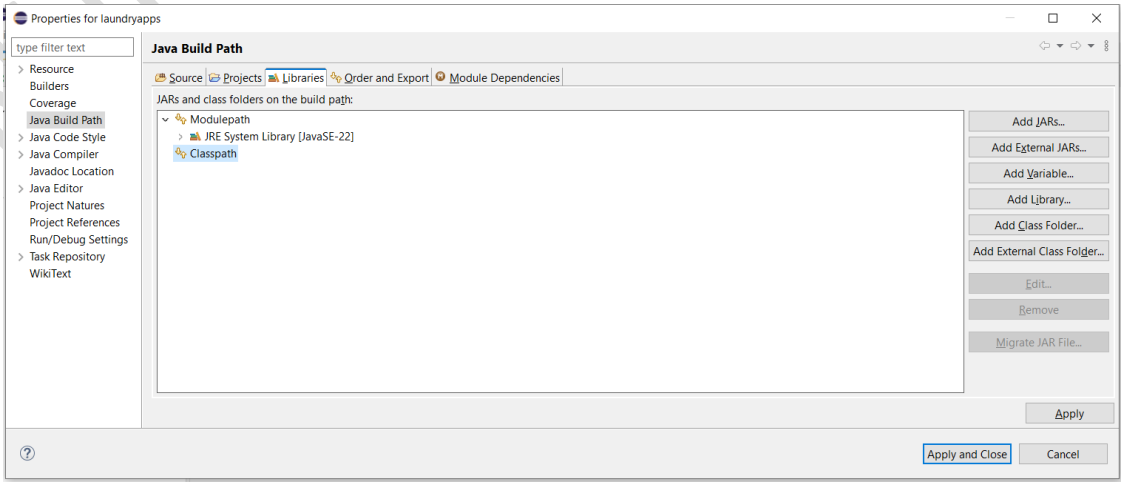


Pilih file yang berekstensi .zip

- Menambahkan MySQL Connector kedalam project dengan cara klik kanan directory **JRE System Library → Built Path → Configure Build Path**

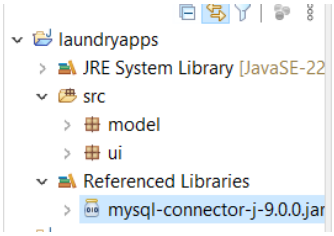


Selanjutnya pilih **Libraries → Classpath**



Tambahkan file MySQL Connector dengan cara klik **Add External JARs** dan pilih file yang telah didownlod dan pilih **Apply and Close**.

Jika berhasil menambahkan MySQL Connector maka akan generate folder Referenced Libraries pada project yang berisi MySQL Connector.



Membuat Database dan Table User

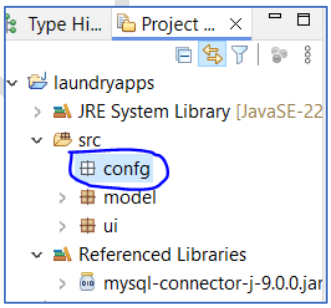
- Buat database dengan nama laundry_apps
- Buat table user dengan SQL sebagai berikut :

```sql
CREATE TABLE `user` (
  `id` int(11) NOT NULL,
  `name` varchar(128) NOT NULL,
  `username` varchar(64) NOT NULL,
  `password` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Membuat Koneksi ke Database MySQL

Setelah berhasil menambahkan MySQL Connector maka dapat membuat koneksi ke database MySQL, berikut Langkah-langkahnya.

- Buat package baru dengan nama config, package ini yang akan digunakan untuk membuat konfigurasi aplikasi yang akan dibuat termasuk dengan konfigurasi database.



- Buat class baru dengan nama Database, kemudian konfigurasi sesuai dengan kode program berikut.

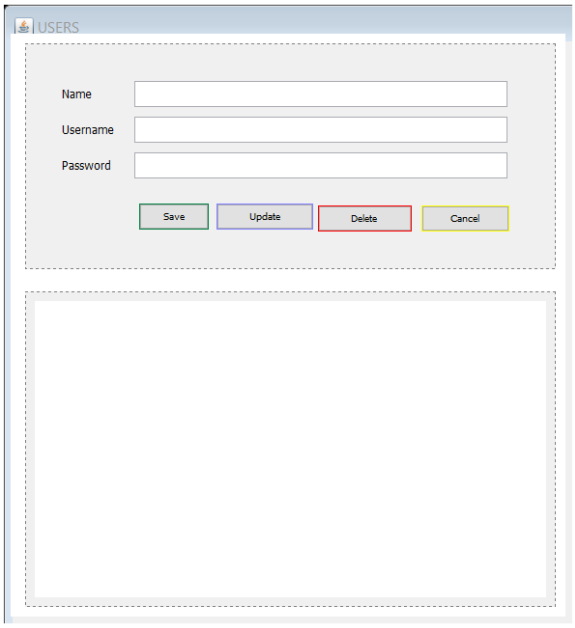
```java
package confg;

import java.sql.*;
import javax.swing.JOptionPane;

public class Database {
    Connection conn;
    public static Connection koneksi() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/laundry_apps",
                    "root","");
            return conn;
        }catch(Exception e) {
            JOptionPane.showMessageDialog(null, e);
            return null;
        }
    }
}
```

Penjelasan :

- Import java.sql.* digunakan untuk import seluruh fungsi-fungsi SQL
- Line 8 membuka method Connection dengan nama koneksi, yang mana method ini akan digunakan untuk membuka koneksi ke database
- Line 10-13 membuat koneksi database, jika koneksi berhasil maka akan mengembalikkan nilai Connection
- Line 15-16 jika koneksi gagal maka akan ditampilkan pesan error menggunakan JOptionPane.

Membuat Tampilan CRUD User

- Buat file baru menggunakan JFrame pada package ui dengan nama UserFrame seperti gambar berikut ini.



Keterangan :

Component	Variable	Keterangan
JTextField	txtName	Name
JTextField	txtUsername	Username
JTextField	txtPassword	Password
JButton	btnSave	Save
JButton	btnUpdate	Update
JButton	btnDelete	Delete
JButton	btnCancel	Cancel
JTable	tableUsers	Table Users

Membuat Table Model

Table model user ini berguna untuk mengambil data dari database dan ditampilkan kedalam table.

- Buat package baru dengan nama **table**
- Buat file baru didalam package table dengan nama **TableUser**, kemudian isikan dengan kode program berikut.

```
public class TableUser extends AbstractTableModel {
    List<User> ls;
    private String[] columnNames = {"ID", "Name", "Username","Password"};
    public TableUser(List<User> ls) {
        this.ls = ls;
    }

    @Override
    public int getRowCount() {
        // TODO Auto-generated method stub
        return ls.size();
    }

    @Override
    public int getColumnCount() {
        // TODO Auto-generated method stub
        return 4;
    }

    @Override
    public String getColumnName(int column) {
        // TODO Auto-generated method stub
         return columnNames[column];
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        // TODO Auto-generated method stub
        switch (columnIndex) {
        case 0:
            return ls.get(rowIndex).getId();
        case 1:
            return ls.get(rowIndex).getNama();
        case 2:
            return ls.get(rowIndex).getUsername();
        case 3:
            return ls.get(rowIndex).getPassword();
        default:
            return null;
        }
    }

}
```

Membuat Fungsi DAO

- Buat package baru dengan nama DAO
- Buat class Interface baru dengan nama **UserDAO**, kemudian isikan dengan kode program berikut.

```
public interface UserDao {
      void save(User user);
      public List<User> show();
      public void delete(String id);
      public void update(User user);

}
```

Terdapat method **save, show, delete dan update**. Method pada class interface digunakan sebagai method utama yang wajib diimplementasikan pada class yang menggunakannya.

Menggunakan Fungsi DAO

- Buat class baru pada package DAO dengan nama **UserRepo** yang mana akan digunakan untuk mengimplementasikan DAO yang telah dibuat.
- Implementasikan UserDao dengan kata kunci **implements**

```
public class UserRepo implements UserDao{
```

- Membuat instanisasi Connection, membuat constructor dan membuat String untuk melakukan manipulas database.

```
 private Connection connection;
 final String insert = "INSERT INTO user (name, username, password) VALUES (?,?,?);";
 final String select = "SELECT * FROM user;";
 final String delete = "DELETE FROM user WHERE id=?;";
 final String update = "UPDATE user SET name=?, username=?, password=? WHERE id=?;";

public UserRepo() {
    connection = Database.koneksi();
}
```

- Membuat method **save**, isikan dengan kode program berikut.

```
@Override
public void save(User user) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(insert);
        st.setString(1, user.getNama());
        st.setString(2, user.getUsername());
        st.setString(3, user.getPassword());
        st.executeUpdate();


    }catch(SQLException e) {
        e.printStackTrace();
    }finally {
        try {
            st.close();
        }catch(SQLException e) {
            e.printStackTrace();
        }
    }

}
```

- Membuat method **show** untuk mengambil data dari database

```
@Override
public List<User> show() {
    List<User> ls=null;
    try {
        ls = new ArrayList<User>();
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(select);
        while(rs.next()) {
            User user = new User();
            user.setId(rs.getString("id"));
            user.setNama(rs.getString("name"));
            user.setUsername(rs.getString("username"));
            user.setPassword(rs.getString("password"));
            ls.add(user);
        }

    }catch(SQLException e) {
        Logger.getLogger(UserDao.class.getName()).log(Level.SEVERE, null, e);
    }
    return ls;
}
```

- Membuat method **update** yang digunakan untuk mengubah data.

```
@Override
public void update(User user) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(update);
        st.setString(1, user.getNama());
        st.setString(2, user.getUsername());
        st.setString(3, user.getPassword());
        st.setString(4, user.getId());
        st.executeUpdate();
    }catch(SQLException e) {
        e.printStackTrace();

    }finally {
        try {
            st.close();

        }catch(SQLException e) {
            e.printStackTrace();
        }
    }

}
```

- Membuat method **delete** yang digunakan untuk menghapus data.

```
@Override
public void delete(String id) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(delete);
        st.setString(1, id);
        st.executeUpdate();

    }catch(SQLException e) {
        e.printStackTrace();

    }finally {
        try {
            st.close();

        }catch(SQLException e) {
            e.printStackTrace();
        }
    }

}
```

Menggunakan Fungsi CRUD DAO pada GUI

- method digunakan untuk menghapus value inputan Ketika suatu proses berhasil dilakukan, buat method **reset** pada JFrame seperti kode program dibawah ini.

```
public void reset() {
    txtName.setText("");
    txtUsername.setText("");
    txtPassword.setText("");
}
```

- Membuat instance pada UserFrame

```
UserRepo usr = new UserRepo();
List<User> ls;
public String id;
```

CREATE USER

- Klik kanan pada tombol **save → add event handlers → actionPerformed** kemdian isi dengan kode program berikut.

```
User user = new User();
user.setNama(txtName.getText());
user.setUsername(txtUsername.getText());
user.setPassword(txtPassword.getText());
usr.save(user);
reset();
```

READ USERS

- Buat method dengan nama **loadTable**() kemudian isikan dengna kode program berikut.

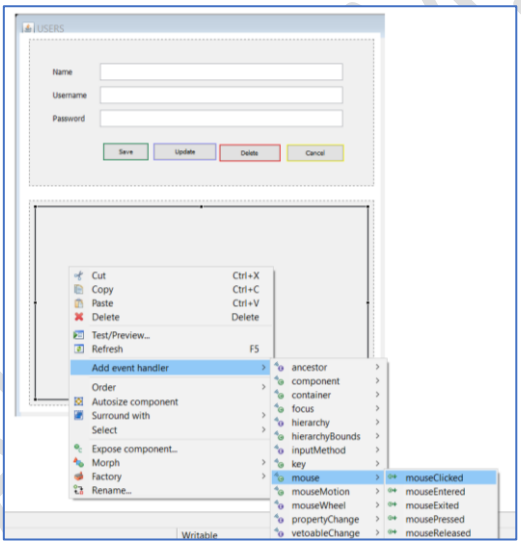
```
public void loadTable() {
    ls = usr.show();
    TableUser tu = new TableUser(ls);
    tableUsers.setModel(tu);
    tableUsers.getTableHeader().setVisible(true);
}
```

- Memanggil method pada class main, sehingga Ketika pertama kali program dijalankan maka **loadTable** akan dipanggil.

```
UserFrame frame = new UserFrame();
frame.setVisible(true);
frame.loadTable();
```

UPDATE USER

- Klik kanan pada **JTable → add event handler → mouse → mouseClicked**

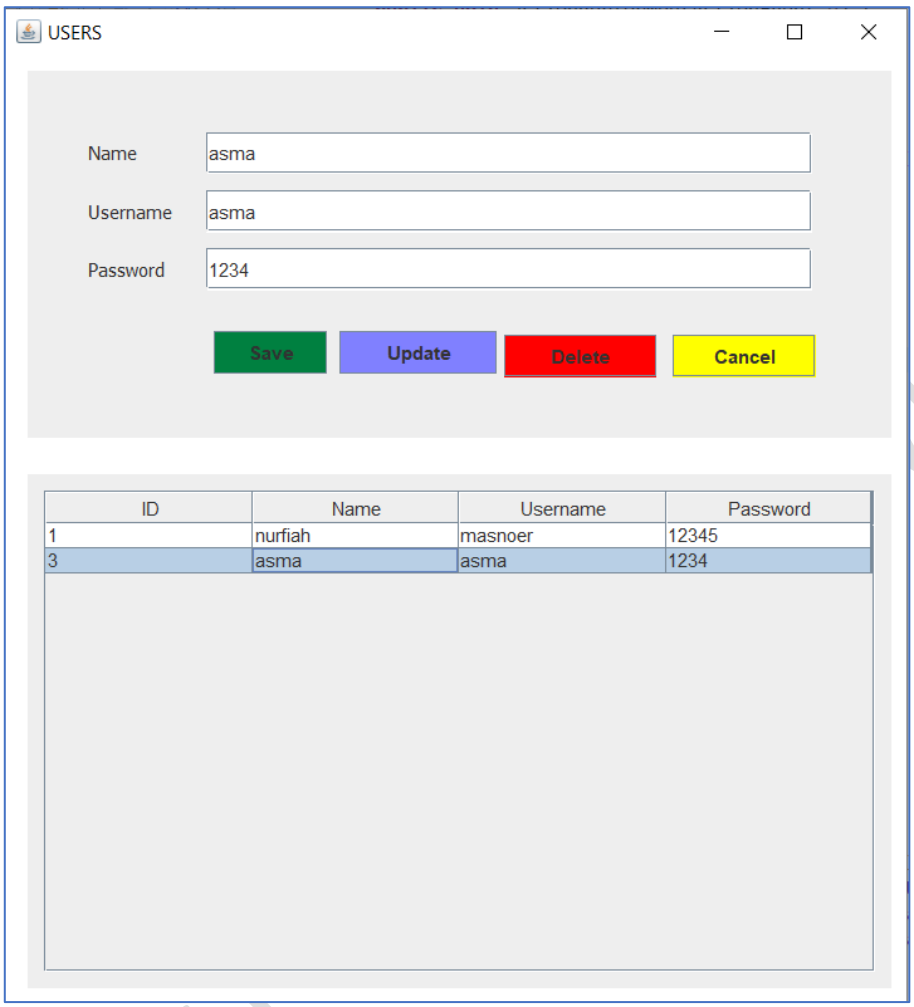


Isikan dengan kode program dibawha ini.

```
id = tableUsers.getValueAt(tableUsers.getSelectedRow(),0).toString();
txtName.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(),1).toString());
txtUsername.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(),2).toString());
txtPassword.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(),3).toString());
```

Kode program diatas berfungsi yaitu mengambil id user dan menyimpannya kedalam variable **id** kemudian mengambil data nama, username dan password dan ditampilkan kedalam form inputan.

- Klik salah satu isi table maka akan secara otomatis tampil pada form inputan.



- Klik kanan tombol **update → add event handler → action → actionPeformed** dan isikan dengan kode program berikut.

```
User user = new User();
user.setNama(txtName.getText());
user.setUsername(txtUsername.getText());
user.setPassword(txtPassword.getText());
user.setId(id);
usr.update(user);
reset();
loadTable();
```

DELETE USER

- Klik salah satu data pada JTable
- Klik kanan tombol **delete → add event handler → action → actionPerformed** dan isikan dengan kode program berikut.

```
if(id != null) {
    usr.delete(id);
    reset();
    loadTable();
}else {
    JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan di hapus");
}
```

1.5 Latihan /Tugas

- Buat fungsi CRUD untuk layanan dan pelanggan