

Implementasi MVC (Model-View-Controller) Dengan DAO (Data Access Object) Pada Java Desktop Application

Mudafiq R. Pratama

me@mudafiqriyan.net

http://www.mudafiqriyan.net

Lisensi Dokumen:

Copyright © 2003-2007 IlmuKomputer.Com

Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.

1. Dasar Teori

MVC merupakan sebuah konsep dalam membangun sebuah aplikasi dengan memisahkan antara data dari tampilan dan aksi pemrosesannya. Model berfungsi sebagai sumber data. View berfungsi sebagai desain interface yang berinteraksi langsung dengan user. Controller berguna sebagai “otak” atau “*business logic*” yang memproses data yang dilakukan oleh user.

Sekilas MVC kelihatannya merepotkan karena kita harus menulis kode lebih banyak tapi keun-tungan-nya adalah kode kita lebih main-tainable karena kita bisa meng-ubah salah satu bagian tanpa harus mengubah bagian yang lain.

2. Peralatan

Peralatan yang digunakan penulis:

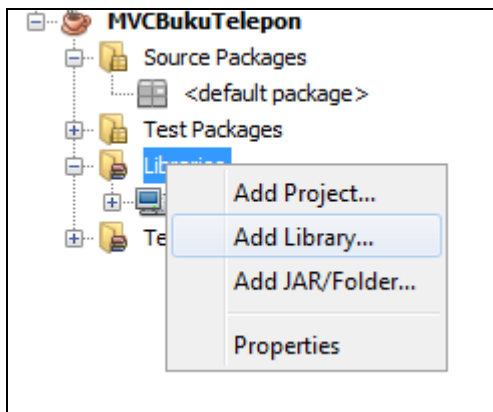
- Netbeans 6.9.1
- JDK 6 update 24
- Database MySQL

3. Pembahasan

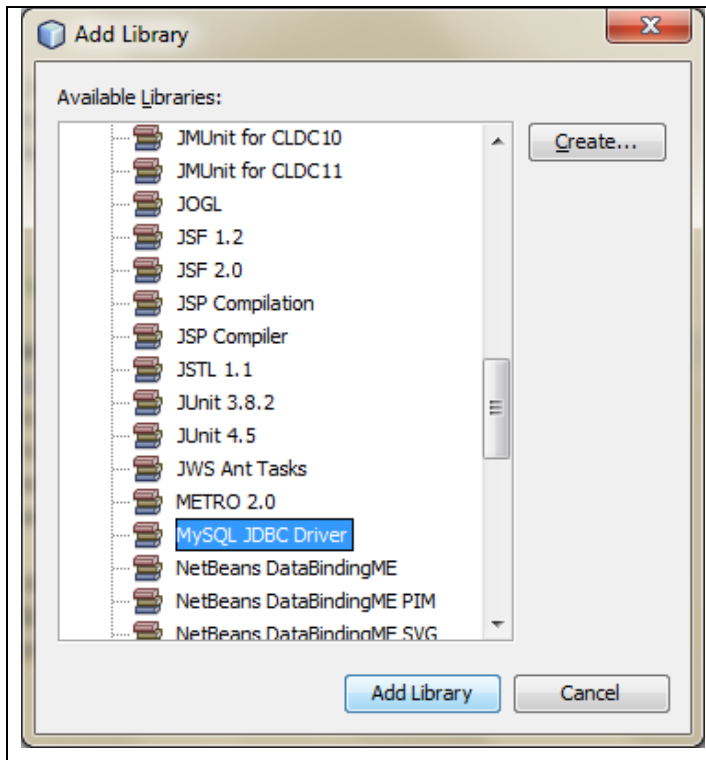
Pada contoh konsep MVC ini menggunakan studi kasus pembuatan Buku Telepon dengan database MySQL. Ikuti tahap-demi-tahap dari proses pembuatan Buku Telepon dengan konsep MVC-DAO dengan fungsi CRUD (Create-Read-Update-Delete) dan fungsi pencarian.

2.1. Import library MySQL

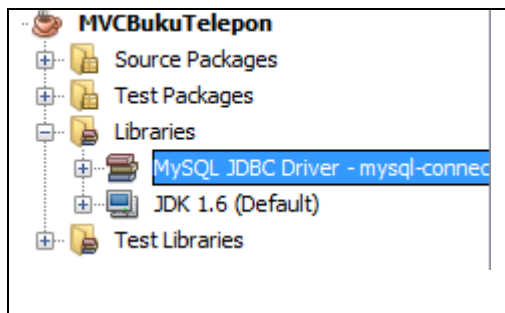
Klik kanan pada Libraries dari project anda, kemudian pilih “Add Library”



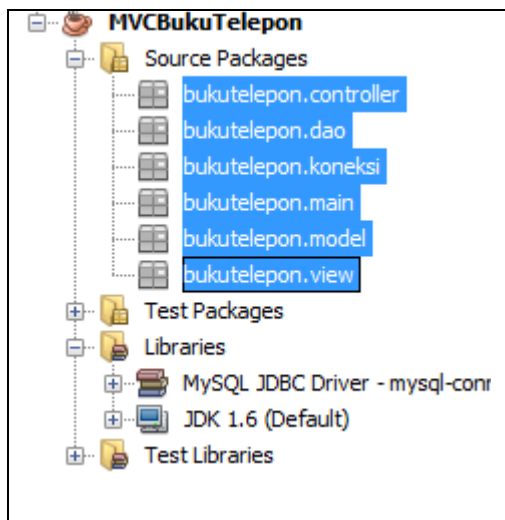
Pilih MySQL JDBC Driver, kemudian tekan tombol “Add Library”. Pada netbeans telah menyediakan JDBC Driver untuk MySQL, sehingga tidak perlu meng-import manual menggunakan file .jar.



Sehingga pada libraries project anda telah tertanam MySQL JDBC Driver, yang berfungsi sebagai connector Java dan MySQL



- 2.2. Buatlah package-package untuk model, view, controller, dao, koneksi, dan main. Yang bertujuan agar lebih terstruktur dalam peng-konsep-an MVC.



2.3. Membuat Koneksi

Buatlah Java class dengan nama “koneksi.java” dan tempatkan pada package “koneksi”, kemudian isikan dengan code berikut.

```
package bukutelepon.koneksi;

import com.mysql.jdbc.jdbc2.optional.MysqlDataSource;
import java.sql.Connection;
import java.sql.SQLException;

public class koneksi {

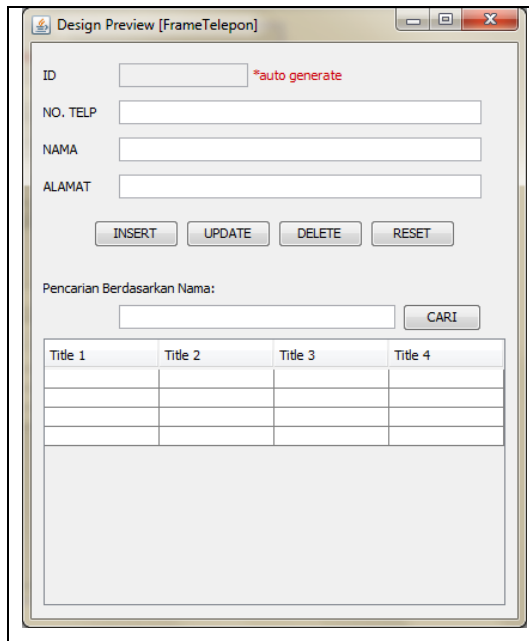
    static Connection con;

    public static Connection connection() {
        if (con == null) {
            MysqlDataSource data = new MysqlDataSource();
            data.setDatabaseName("buku_telepon");
            data.setUser("root");
            data.setPassword("");
            try {
                con = data.getConnection();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
        return con;
    }
}
```

Pada `setDatabaseName()` merupakan keterangan nama database pada project BukuTelepon tersebut, yaitu nama databasenya: “buku_telepon”. `setUser()` merupakan nama user MySQL yaitu “root”, sedangkan `setPassword()` dikosongkan karena MySQL tidak menggunakan password sehingga hanya ditandai dengan string kosong.

2.4. Membuat Desain Interface

Buatlah frame desain aplikasi yang ditempatkan pada package “view”



Kemudian tambahkanlah code berikut pada source frame desain tersebut. Nama seperti txtID, txtNoTelp, buttonInsert, tabelData, dan lain-lain merupakan nama variabel dari komponen yang ada pada frame tersebut.

```
public JTextField getTxtID() {
    return txtID;
}
public JTextField getTxtNoTelp() {
    return txtNoTelp;
}
public JTextField getTxtNama() {
    return txtNama;
}
public JTextField getTxtAlamat() {
    return txtAlamat;
}
public JTable getTabelData() {
    return tabelData;
}
public JButton getButtonInsert() {
    return buttonInsert;
}
public JButton getButtonUpdate() {
    return buttonUpdate;
}
public JButton getButtonDelete() {
    return buttonDelete;
}
public JButton getButtonReset() {
    return buttonReset;
}
public JButton getButtonCari() {
    return buttonCariNama;
}
public JTextField getTxtCariNama() {
    return txtCariNama;
}
}
```

Tujuan pembuatan method tersebut, agar komponen di frame dapat dipanggil dan diterapkan oleh class lain, oleh karena itu dibuat public.

2.5. Membuat Table Model

Buatlah file “bukutelepon.java” pada package “model” yang berisi method set dan get

```
package bukutelepon.model;

public class bukutelepon {

    private Integer id;
    private String nomer;
    private String nama;
    private String alamat;

    public String getAlamat() {
        return alamat;
    }

    public void setAlamat(String alamat) {
        this.alamat = alamat;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public String getNomer() {
        return nomer;
    }

    public void setNomer(String nomer) {
        this.nomer = nomer;
    }
}
```

id, nomer, nama, alamat merupakan nama dari attribute yang ada di database. Fungsi setter and getter tersebut untuk mengambil data dari database dan memanggilnya, sebagai perantara penyaluran data.

Kemudian buatlah file “tableModelBukuTelepon.java” pada package “model” dan isikan dengan code berikut:

```
package bukutelepon.model;

import java.util.List;
import javax.swing.table.AbstractTableModel;

public class tableModelBukuTelepon extends AbstractTableModel {
    List<bukutelepon> lb;

    public tableModelBukuTelepon(List<bukutelepon> lb) {
        this.lb = lb;
    }

    @Override
    public int getColumnCount() {
        return 4;
    }

    public int getRowCount() {
        return lb.size();
    }

    @Override
    public String getColumnName(int column) {
        switch (column) {
            case 0:
                return "ID";
            case 1:
                return "Nomer";
            case 2:
                return "Nama";
            case 3:
                return "Alamat";
            default:
                return null;
        }
    }

    @Override
    public Object getValueAt(int row, int column) {
        switch (column) {
            case 0:
                return lb.get(row).getId();
            case 1:
                return lb.get(row).getNomer();
            case 2:
                return lb.get(row).getNama();
            case 3:
                return lb.get(row).getAlamat();
            default:
                return null;
        }
    }
}
```

Tabel model berguna untuk mengambil data dari database yang akan ditampilkan pada JTable yang ada di Frame. Data disimpan dalam List. Yang kemudian dapat dipanggil untuk ditampilkan di JTable.

2.6. Membuat Fungsi DAO

Data Access Object (DAO) merupakan sebuah object yang menyediakan sebuah abstract interface terhadap beberapa database atau mekanisme persistence, menyediakan beberapa operasi tertentu tanpa mengekspos detail database. Penerapan konsep ini sering disebut dengan *separation of concern* dimana setiap kode dipisahkan berdasarkan fungsinya sehingga kode di atasnya hanya perlu mengetahui secara abstrak cara mengakses data tanpa perlu mengetahui bagaimana akses ke sumber data diimplementasikan. DAO sering dikaitkan dengan Java EE dan akses ke relational database melalui JDBC API, karena memang DAO berasal dari pedoman praktek Sun Microsystems. Kebanyakan penggunaan DAO adalah satu objek DAO untuk satu objek entity.

Buatlah Class Interface dengan nama “implementBukuTelepon.java” yang diletakkan pada package “dao”. Kemudian isikan code berikut:

```
package bukutelepon.dao;

import java.util.List;
import bukutelepon.model.*;

public interface implementBukuTelepon {

    public void insert(bukutelepon b);

    public void update(bukutelepon b);

    public void delete(int id);

    public List<bukutelepon> getALL();

    public List<bukutelepon> getCariNama(String nama);
}
```

Terdapat method-method insert, update, delete, dan getALL(), getCarinama(). Method pada class interface digunakan sebagai method inti yang wajib dideklarasikan oleh subclass yang meng-implement class interface tersebut.

Buatlah class “daoBukuTelepon.java” pada package “dao” dan isikan dengan code berikut untuk menampilkan data:

```
package bukutelepon.dao;

import bukutelepon.koneksi.koneksi;
import bukutelepon.model.bukutelepon;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
```

Import beberapa api yang dibutuhkan untuk class dao tersebut.

```
public class daoBukuTelepon implements implementBukuTelepon {

    Connection connection;
    final String insert = "INSERT INTO bukutelepon (nomer, nama, alamat) VALUES (?, ?, ?)";
    final String update = "UPDATE bukutelepon set nomer=?, nama=?, alamat=? where id=? ";
    final String delete = "DELETE FROM bukutelepon where id=? ";
    final String select = "SELECT * FROM bukutelepon;";
    final String carinama = "SELECT * FROM bukutelepon where nama like ?";

    public daoBukuTelepon() {
        connection = koneksi.connection();
    }
}
```

Class “daoBukuTelepon” meng-implements class interface “implementBukuTelepon”. Kemudian mendeklarasikan query insert, update, delete, dan select. Constructor daoBukuTelepon() berisi koneksi database.

Fungsi untuk insert data ke dalam database.

```
public void insert(bukutelepon b) {
    PreparedStatement statement = null;
    try {
        statement = connection.prepareStatement(insert);
        statement.setString(1, b.getNomer());
        statement.setString(2, b.getNama());
        statement.setString(3, b.getAlamat());
        statement.executeUpdate();
        ResultSet rs = statement.getGeneratedKeys();
        while (rs.next()) {
            b.setId(rs.getInt(1));
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    } finally {
        try {
            statement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

Fungsi untuk update data ke dalam database

```
public void update(bukutelepon b) {
    PreparedStatement statement = null;
    try {
        statement = connection.prepareStatement(update);
        statement.setString(1, b.getNomer());
        statement.setString(2, b.getNama());
        statement.setString(3, b.getAlamat());
        statement.setInt(4, b.getId());
        statement.executeUpdate();

    } catch (SQLException ex) {
        ex.printStackTrace();
    } finally {
        try {
            statement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

Fungsi delete data dari database

```
public void delete(int id) {
    PreparedStatement statement = null;
    try {
        statement = connection.prepareStatement(delete);

        statement.setInt(1, id);
        statement.executeUpdate();

    } catch (SQLException ex) {
        ex.printStackTrace();
    } finally {
        try {
            statement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

Fungsi menampilkan data ke tabel

```
public List<bukutelepon> getAll() {
    List<bukutelepon> lb = null;
    try {
        lb = new ArrayList<bukutelepon>();
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(select);
        while (rs.next()) {
            bukutelepon b = new bukutelepon();
            b.setId(rs.getInt("id"));
            b.setNomer(rs.getString("nomer"));
            b.setNama(rs.getString("nama"));
            b.setAlamat(rs.getString("alamat"));
            lb.add(b);
        }
    } catch (SQLException ex) {
        Logger.getLogger(daoBukuTelepon.class.getName()).log(Level.SEVERE, null, ex);
    }

    return lb;
}
```

Fungsi menampilkan data ke tabel berdasarkan pencarian

```
public List<bukutelepon> getCariNama(String nama) {  
    List<bukutelepon> lb = null;  
    try {  
        lb = new ArrayList<bukutelepon>();  
        PreparedStatement st = connection.prepareStatement(carinama);  
        st.setString(1, "%" + nama + "%");  
        ResultSet rs = st.executeQuery();  
        while (rs.next()) {  
            bukutelepon b = new bukutelepon();  
            b.setId(rs.getInt("id"));  
            b.setNomer(rs.getString("nomer"));  
            b.setNama(rs.getString("nama"));  
            b.setAlamat(rs.getString("alamat"));  
            lb.add(b);  
        }  
    } catch (SQLException ex) {  
        Logger.getLogger(daoBukuTelepon.class.getName()).log(Level.SEVERE, null, ex);  
    }  
    return lb;  
}
```

2.7. Membuat Controller

```
package bukutelepon.controller;

import bukutelepon.dao.daoBukuTelepon;
import bukutelepon.dao.implementBukuTelepon;
import bukutelepon.model.bukutelepon;
import bukutelepon.model.tableModelBukuTelepon;
import bukutelepon.view.FrameTelepon;
import java.util.List;
import javax.swing.JOptionPane;

public class controllerBukuTelepon {

    FrameTelepon frame;
    implementBukuTelepon implBukuTelepon;
    List<bukutelepon> lb;

    public controllerBukuTelepon(FrameTelepon frame) {
        this.frame = frame;
        implBukuTelepon = new daoBukuTelepon();
        lb = implBukuTelepon.getAll();
    }

    //mengosongkan field
    public void reset() {
        frame.getTxtID().setText("");
        frame.getTxtNoTelp().setText("");
        frame.getTxtNama().setText("");
        frame.getTxtAlamat().setText("");
    }

    //menampilkan data ke dalam tabel
    public void isiTable() {
        lb = implBukuTelepon.getAll();
        tableModelBukuTelepon tmb = new tableModelBukuTelepon(lb);
        frame.getTabelData().setModel(tmb);
    }

    //merupakan fungsi untuk menampilkan data yang dipilih dari tabel
    public void isiField(int row) {
        frame.getTxtID().setText(lb.get(row).getId().toString());
        frame.getTxtNoTelp().setText(lb.get(row).getNomer());
        frame.getTxtNama().setText(lb.get(row).getNama());
        frame.getTxtAlamat().setText(lb.get(row).getAlamat());
    }

    //merupakan fungsi untuk insert data berdasarkan inputan user dari textfield di frame
    public void insert() {
        bukutelepon b = new bukutelepon();
        b.setNomer(frame.getTxtNoTelp().getText());
        b.setNama(frame.getTxtNama().getText());
        b.setAlamat(frame.getTxtAlamat().getText());

        implBukuTelepon.insert(b);
    }

    //berfungsi untuk update data berdasarkan inputan user dari textfield di frame
    public void update() {
        bukutelepon b = new bukutelepon();
        b.setNomer(frame.getTxtNoTelp().getText());
        b.setNama(frame.getTxtNama().getText());
        b.setAlamat(frame.getTxtAlamat().getText());
        b.setId(Integer.parseInt(frame.getTxtID().getText()));
        implBukuTelepon.update(b);
    }

    //berfungsi menghapus data yang dipilih
    public void delete() {
        if (!frame.getTxtID().getText().trim().isEmpty()) {
            int id = Integer.parseInt(frame.getTxtID().getText());
            implBukuTelepon.delete(id);
        } else {
            JOptionPane.showMessageDialog(frame, "Pilih data yang akan di hapus");
        }
    }

    public void isiTableCariNama() {
        lb = implBukuTelepon.getCariNama(frame.getTxtCariNama().getText());
        tableModelBukuTelepon tmb = new tableModelBukuTelepon(lb);
        frame.getTabelData().setModel(tmb);
    }

    public void carinama() {
        if (!frame.getTxtCariNama().getText().trim().isEmpty()) {
            implBukuTelepon.getCariNama(frame.getTxtCariNama().getText());
            isiTableCariNama();
        } else {
            JOptionPane.showMessageDialog(frame, "SILAHKAN PILIH DATA");
        }
    }
}
```

Method-method yang ada di class controller berfungsi sebagai pengendali atau pemroses data ke frame. Fungsi-fungsi tersebut nantinya akan dipanggil ke frame atau view. Method **isiTable()** berfungsi untuk menampilkan data ke jTable yang ada di frame. **isiField()** memberikan fungsi menempatkan data yang dipilih oleh user berdasarkan dari tabel. **insert()** untuk input data ke database. **update()** untuk mengubah data. **delete()** untuk menghapus data. **isiTableCariNama()** menampilkan data ke tabel berdasarkan pencarian user. **carinama()** memberikan fungsi pada tombol pencarian.

2.8. Edit Source Desain Frame

```
public class FrameTelepon extends javax.swing.JFrame {  
  
    controllerBukuTelepon cbt;  
  
    /** Creates new form FrameTelepon */  
    public FrameTelepon() {  
        initComponents();  
        cbt = new controllerBukuTelepon(this);  
        cbt.isiTable();  
    }  
}
```

Dengan memanggil object controller, kemudian memanggil isi fungsi dari controller, maka ketika frame di run, maka akan meng-eksekusi fungsi dari isiTable().

Kemudian beri action atau event handling dari komponen button dan jTable

```
//action performed atau event handling dari button insert  
private void buttonInsertActionPerformed(java.awt.event.ActionEvent evt) {  
    cbt.insert();  
    cbt.isiTable();  
    cbt.reset();  
}  
  
//action performed atau event handling dari button reset  
private void buttonResetActionPerformed(java.awt.event.ActionEvent evt) {  
    cbt.reset();  
}  
  
//action performed atau event handling dari button update  
private void buttonUpdateActionPerformed(java.awt.event.ActionEvent evt) {  
    cbt.update();  
    cbt.isiTable();  
    cbt.reset();  
}  
  
//action performed atau event handling dari button delete  
private void buttonDeleteActionPerformed(java.awt.event.ActionEvent evt) {  
    cbt.delete();  
    cbt.isiTable();  
    cbt.reset();  
}  
  
//event handling mouse clicked dari tabel  
private void tabelDataMouseClicked(java.awt.event.MouseEvent evt) {  
    cbt.isiField(tabelData.getSelectedRow());  
}  
  
//action performed atau event handling dari button cari  
private void buttonCariNamaActionPerformed(java.awt.event.ActionEvent evt) {  
    cbt.carinama();  
}
```

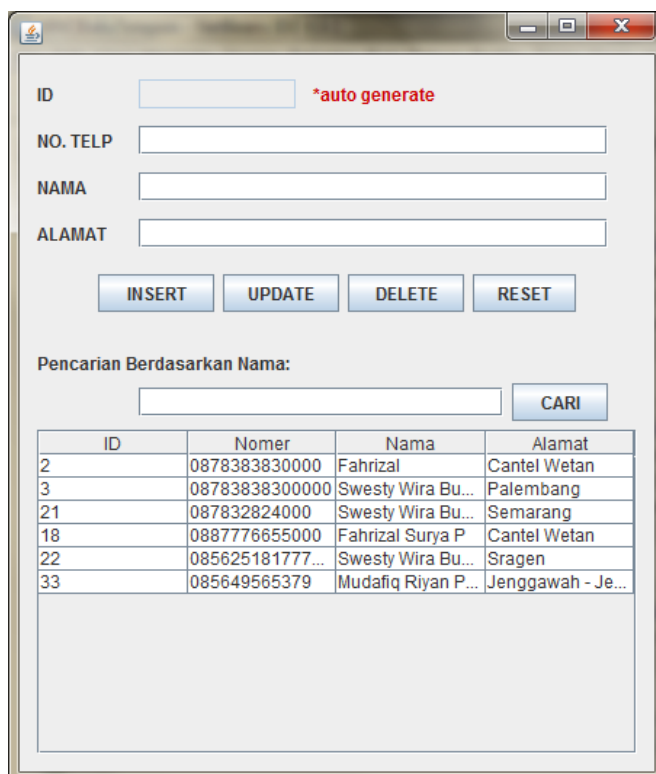
2.9. Fungsi Main

Pada package main, buatlah class “Main.java” yang difungsikan sebagai run active class.

```
package bukutelepon.main;  
  
import bukutelepon.view.FrameTelepon;  
  
public class Main {  
    public static void main(String[] args) {  
        new FrameTelepon().setVisible(true);  
    }  
}
```

Sehingga ketika class ini di run, maka yang akan dieksekusi adalah class FrameTelepon(). Dan pada class “Main” inilah anda bisa memberikan look and feel.

2.10. Run Main Project



ID	Nomer	Nama	Alamat
2	0878383830000	Fahrizal	Cantel Wetan
3	08783838300000	Swesty Wira Bu...	Palembang
21	087832824000	Swesty Wira Bu...	Semarang
18	0887776655000	Fahrizal Surya P	Cantel Wetan
22	085625181777...	Swesty Wira Bu...	Sragen
33	085649565379	Mudafiq Riyan P...	Jenggawah - Je...

=0=0=0=0=0=0=0=0=0= **Selamat Mencobá** =0=0=0=0=0=0=0=0=0=

Biografi Penulis



Mudafiq Riyan Pratama. Adalah seorang anak laki-laki dari Nurkholis (bapak) dan Susriyanti (ibu) ini terlahir di Jember pada tanggal 9 Mei 1989. Mengawali pendidikan TK dan SD di Jenggawah, Jember. Kemudian menempuh SMP di SMPN 6 Jember, yang kemudian dilanjutkan ke SMAN 2 Jember. Kuliah S1 di Universitas Muhammadiyah Malang dengan mengambil jurusan Teknik Informatika. Saat itulah penulis mengawali masuk ke dunia informatika yang sebelumnya tidak pernah tau tentang dunia tersebut. Saat itulah ketertarikan dengan informatika makin membesar. Penulis mulai meniti karir di bidang IT sebagai *programmer freelance*. Hingga akhirnya saat ini beliau berkarir sebagai tenaga pengajar (dosen) di Universitas Muhammadiyah Jember.