Membuat Aplikasi Desktop *Client Server* Sederhana dengan Java RMI

Achmad Maulana
achmad.maulana.and@gmail.com
http://jakartait.com

Istilah client/server pertama kali digunakan pada tahun tahun 1980-an dalam mereferensikan suatu Personal Computer pada suatu jaringan. Model client/server yang sebenarnya mulai diterima pada akhir tahun 1980-an. Arsitektur perangkat lunak client/server adalah infrastruktur modular dan message-based yang sangat berguna, dimaksudkan untuk meningkatkan usability, flexibility, interoperability, dan scalability.[1]

RMI (*Remote Method Invocation*) menyediakan sarana dimana client dan server dapat berkomunikasi dan saling bertukar infomasi. RMI memungkinkan pengembang perangkat lunak untuk merancang aplikasi terdistribusi dimana methods dari remote object dapat dipanggil dari JVM (Java Virtual Mechine) lain, yang mungkin berjalan pada host yang berbeda. Remote object adalah obyek dalam Java yang dapat direferensikan secara remote. Pemrogram seakan- akan memanggil methods lokal dari file kelas lokal, sedang dalam kenyataannya semua argumen dikirimkan ke remote target dan diinter-pretasikan, kemudian hasilnya dikirimkan kembali ke pemanggil. Dalam RMI, server akan membuat remote objects, membuat referensi, dan menunggu client untuk memanggil methods dari remote object ini. Sedangkan client akan mendapatkan remote reference dari satu atau lebih remote object dan memanggil methods untuk remote object tersebut. [2]

PENDAHULUAN

Pada Tulisan kali ini, kita akan mengulas bagaimana membuat sebuah aplikasi desktop sederhana berbasis client server dengan Bahasa Program Java dan menggunakan teknologi RMI (Remote Method Invocation). Sebagai contoh sederhana kita akan membuat sebuah aplikasi yang dapat melakukan proses CRUD (Create, Read, Update, Delete) ke database MySQL.

PERSIAPAN

Sebelum memulai perancangan sistem kita harus siapkan dulu alat-alat perangnya:

1. Install Java+Netbeans , jika belum tersintall silahkan download: http://www.oracle.com/technetwork/articles/javase/jdk-netbeans-jsp-142931.html
2. Install xampp , jika belum tersintall silahkan download : https://www.apachefriends.org/download.html

Pada tulisan kali ini saya mengasumsikan pembaca sudah mengerti dan memahami bagaimana cara mengintall java Netbeans dan xampp serta bagaimana cara menggunakannya.

PERANG DIMULAI

A. MEMBUAT DATABASE DAN TABLE

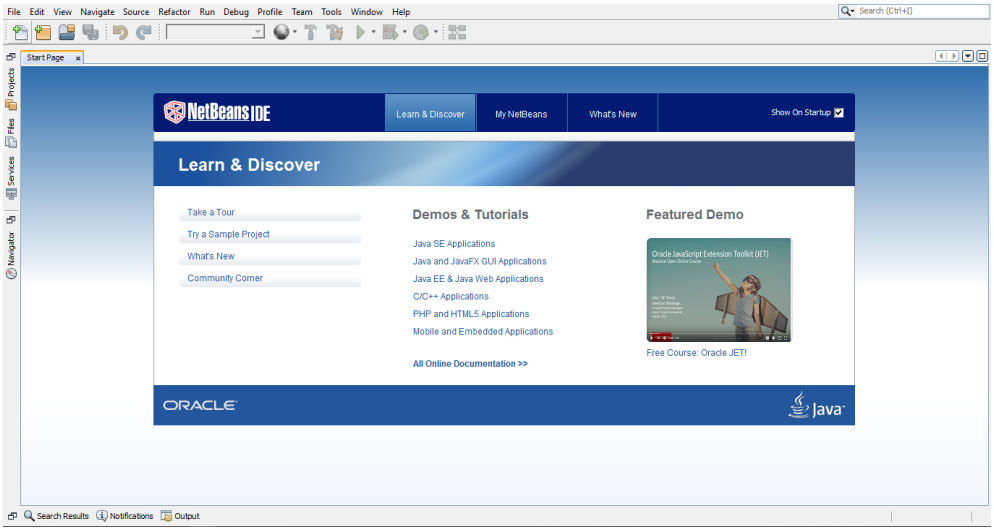
Pertama kita buat database dengan nama ***client_server_db*** dan table nya terlebih dahulu:

Jalankan query untuk buat table user :

```
CREATE TABLE IF NOT EXISTS `tbl_user` (
  `id` int(4) NOT NULL AUTO_INCREMENT,
  `username` varchar(100) NOT NULL,
  `email` varchar(50) NOT NULL,
  `password` varchar(200) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

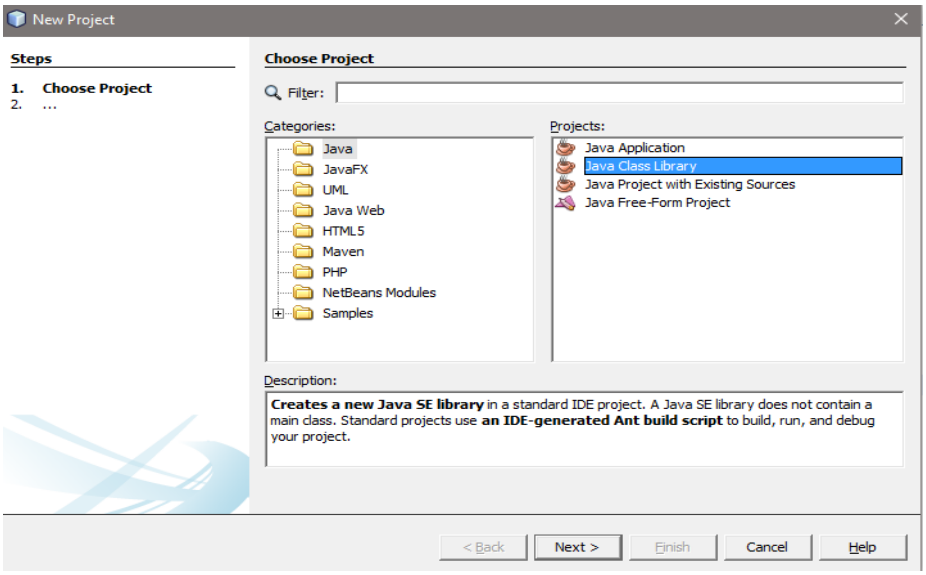
B. MEMBUAT PROJECT JAVA

Jalankan Netbeans yang sudah terinstall sehingga akan muncul tampilan seperti ini:



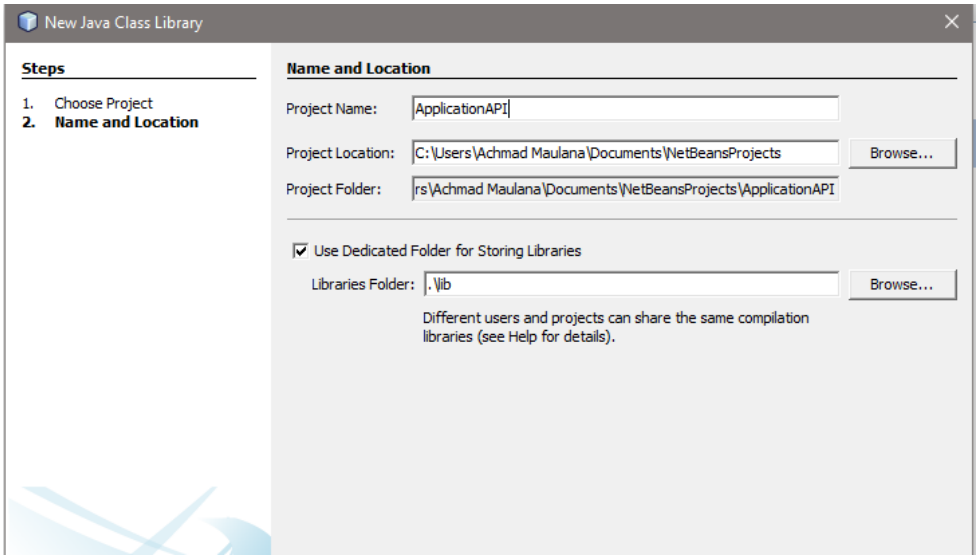
Selanjutnya kita akan membuat 3 project baru yaitu ***ApplicationAPI***., ***ApplicationServer*** dan ***ApplicationClient***.

- **YANG PERTAMA** kita buat project dengan nama ***ApplicationAPI*** dengan cara klik **File > New Project** akan muncul window seperti ini:

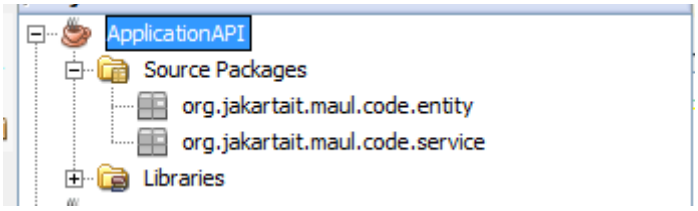


Selanjutnya pilih Java Class Library lalu klik **Next.**

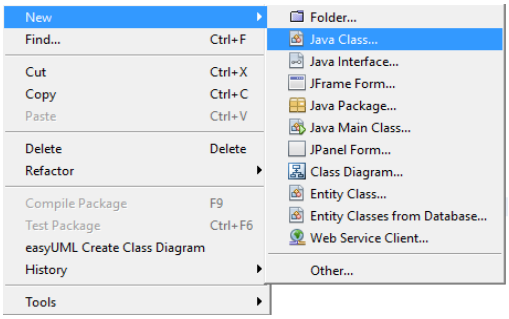
Setelah itu akan muncul window seperti dibawah, dan kita isikan nama project nya:



Selanjutnya buat 2 (dua) package seperti digambar dibawah ini:



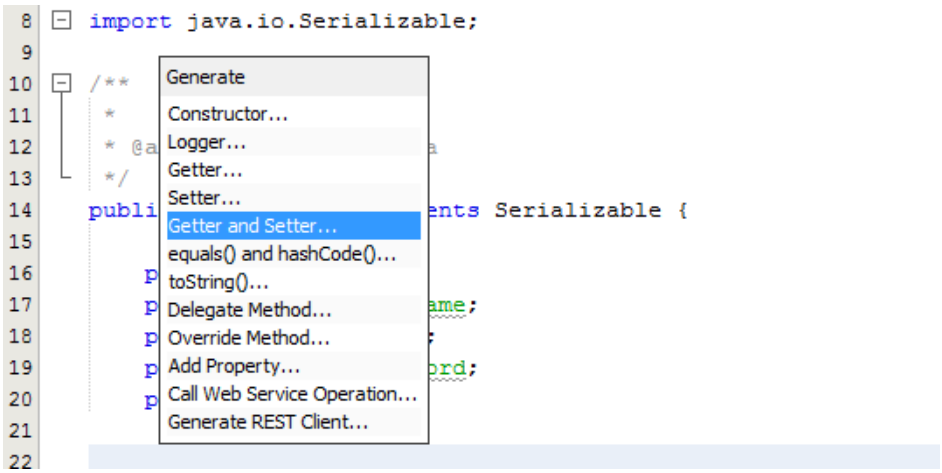
Setelah itu kita buat Java Class dengan nama User pada package entity dengan cara klik kanan pada package > **Java Class** lalu beri nama kelas.



Selanjutnya buatlah entitas-entitas sesuai dengan field-field yang ada pada table tbl_user yang sebelumnya kita telah buat, untuk lebih jelasnya lihat pada gambar dibawah:

```
8  import java.io.Serializable;
9
10 /**
11  *
12  * @author Achmad Maulana
13  */
14 public class User implements Serializable {
15
16     private Long id;
17     private String username;
18     private String email;
19     private String password;
```

Jangan lupa buat setter dan getter dengan cara menekan tombol **Alt+Insert** lalu pilih **Getter and Setter**



Lalu checklist pada semua entitasnya dan hasilnya seperti dibawah:

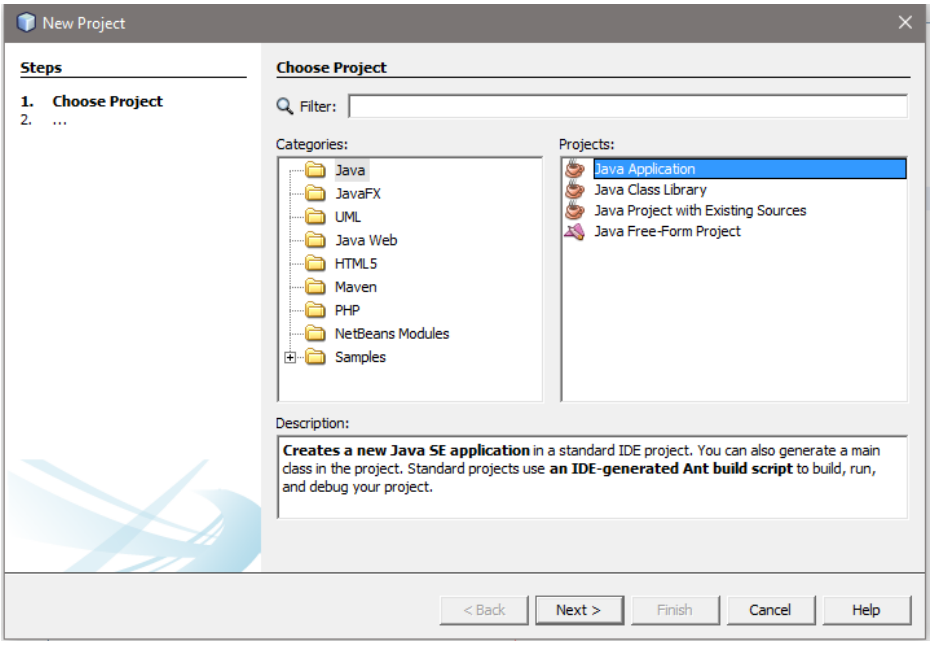
```
import java.io.Serializable;

/**
 *
 * @author Achmad Maulana
 */
public class User implements Serializable {

    private Long id;
    private String username;
    private String email;
    private String password;

    public Long getId() {
        return id;
    }

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

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
 }
```

Selanjutnya buat kelas interface dengan nama ***UserService*** pada Package service:

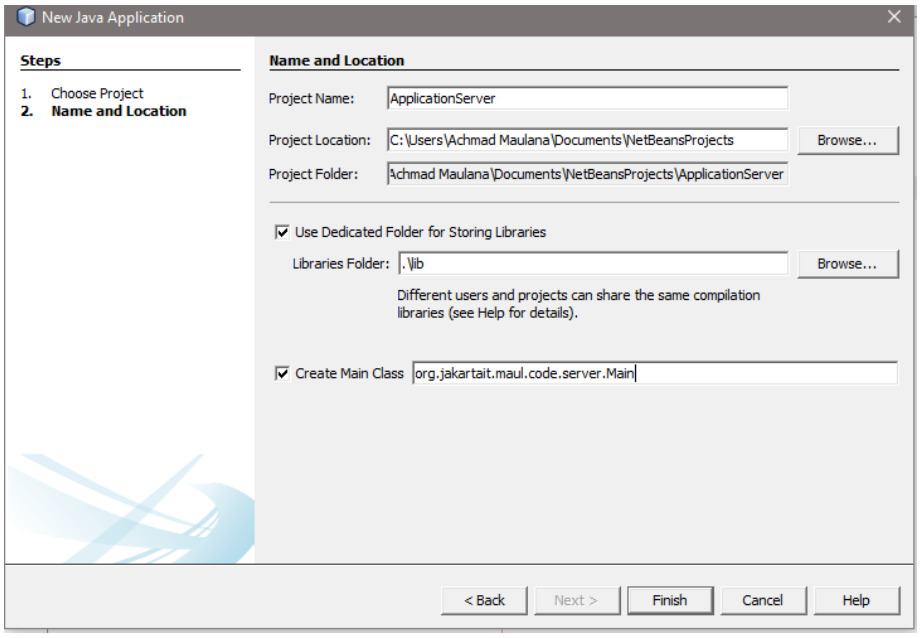
```
import java.rmi.Remote;
import java.rmi.RemoteException;
```

```
import java.util.List;
import org.jakartait.maul.code.api.entity.User;

/**
 *
 * @author Achmad Maulana
 */
public interface UserService extends Remote {

    User insert(User user) throws RemoteException;

    void update(User user) throws RemoteException;

    void delete(Long id) throws RemoteException;

    User getUser(Long id) throws RemoteException;

    List<User> getAllUser() throws RemoteException;
}
```

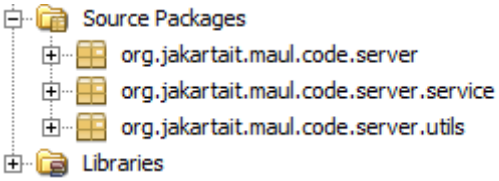
- **YANG KEDUA** kita buat project ***ApplicationServer*** dengan cara klik **File > New Project** akan muncul window seperti ini:



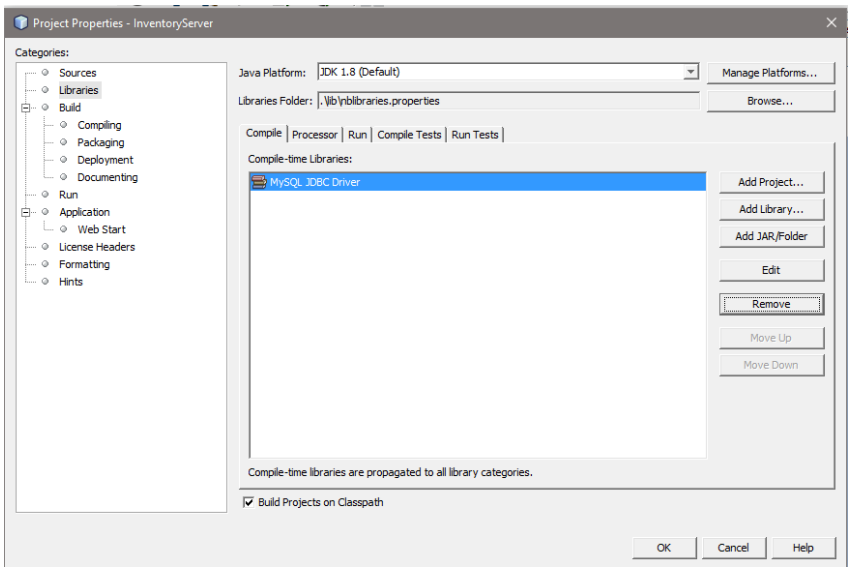
Setelah klik **Next** selanjutnya beli nama project ***ApplicationServer*** lalu klik tombol **Finish.**



Selanjutnya pembuatan 3 (dua) package seperti dibawah ini:

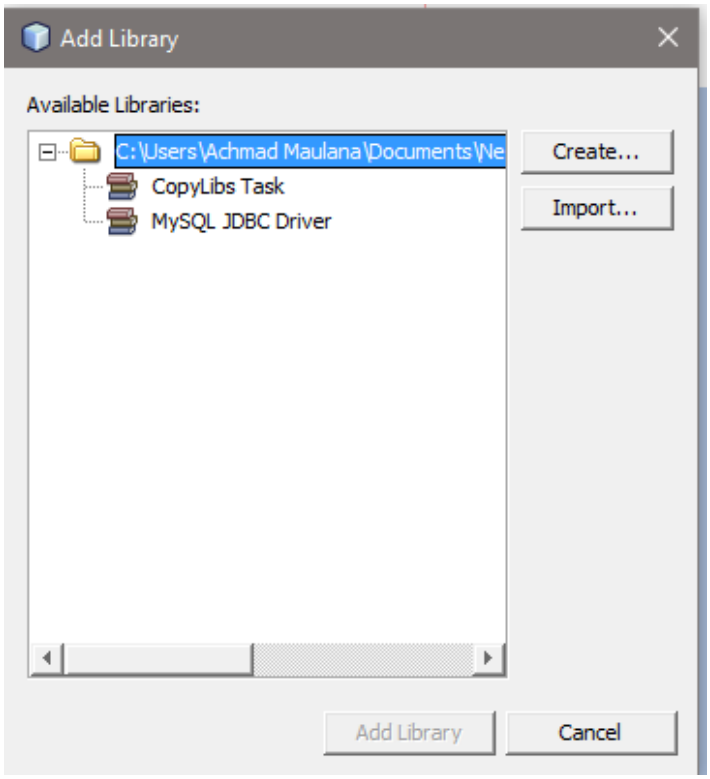


Jangan lupa import project ***ApplicationAPI*** pada project ***ApplicationServer*** terlebih dahulu:



Tekan tombol **Add Project ,** lalu arahkan pada project ***ApplicationAPI***

Selanjutnya import MySQL Drivernya .



Buat main kelas dengan nama ***MainClass*** package paling atas:

```
import java.rmi.AlreadyBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import org.jakartait.maul.code.server.service.UserServiceServer;
import org.jakartait.maul.code.server.utils.DatabaseUtilities;

/**
 *
 * @author Achmad Maulana
 */
public class MainClass {

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) throws RemoteException {
      // TODO code application logic here
      Registry server = LocateRegistry.createRegistry(6789);
      UserServiceServer userService = new UserServiceServer();
      server.rebind("service", userService);
      System.out.println("Server berjalan");
```

```
    }

}
```

Lalu buat kelas ***DatabaseUtilities*** pada package utils.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
/**
 *
 * @author Achmad Maulana
 */
public class DatabaseUtilities{

   private static Connection connection;

   public static Connection getConnection() {
      if (connection == null) {
         try {
            DriverManager.registerDriver(new com.mysql.jdbc.Driver());
            connection                                                    =
DriverManager.getConnection("jdbc:mysql://localhost:3306/client_server_db",      "root",
"");

         } catch (SQLException ex) {
            Logger.getLogger(DatabaseUtilities.class.getName()).log(Level.SEVERE, null, ex);
         }
      }
      return connection;
   }

   }
```

Selanjutnya buat kelas ***UserServiceServer*** pada package service

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
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.Logger;
import org.jakartait.maul.code.api.entity.User;
```

```
import org.jakartait.maul.code.server.utils.DatabaseUtilities;
import org.jakartait.maul.code.service.UserService;

/**
 *
 * @author Achmad Maulana
 */
public class UserServiceServer extends UnicastRemoteObject implements UserService {

   public UserServiceServer() throws RemoteException {
   }

   @Override
   public User insert(User user) throws RemoteException {
      System.out.println("Client calling insert process");

      PreparedStatement statement = null;
      try {
         statement = DatabaseUtilities.getConnection().prepareStatement(
               "INSERT INTO tbl_user (user_idx, username, email, password)"
               + " VALUES (null,'" + user.getUsername() + "', '" + user.getEmail() + "', MD5('"
+ user.getPassword() + "'))", Statement.RETURN_GENERATED_KEYS);
         Logger log = Logger.getLogger("Insert User Logger");
         DatabaseUtilities handler = new DatabaseUtilities();
         log.addHandler(handler);
         log.info("Client calling insert process");
         statement.executeUpdate();

         ResultSet resultSet = statement.getGeneratedKeys();
         if (resultSet.next()) {
            user.setId(resultSet.getLong(1));
         }

         resultSet.close();
         return user;
      } catch (SQLException e) {
         e.printStackTrace();
         return null;
      } finally {
         if (statement != null) {
            try {
               statement.close();
            } catch (Exception e) {
            }
         }
      }

   }
```

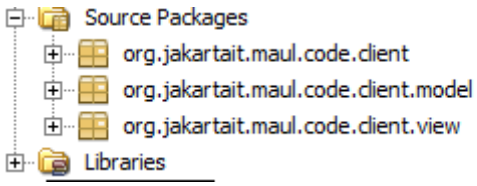
```
@Override
public void update(User user) throws RemoteException {
   System.out.println("Client calling update process");
   PreparedStatement statement = null;
   try {

      statement = DatabaseUtilities.getConnection().prepareStatement(
            " UPDATE tbl_user SET username=?"
            + ",email=?"
            + ",password=MD5(?)"
            + "WHERE user_idx=?"
      );

      statement.setString(1, user.getUsername());
      statement.setString(2, user.getEmail());
      statement.setString(3, user.getPassword());
      statement.setLong(4, user.getId());
      statement.executeUpdate();

   } catch (SQLException e) {
      e.printStackTrace();
   } finally {
      if (statement != null) {
         try {
            statement.close();
         } catch (SQLException e) {
            e.printStackTrace();
         }
      }
   }
}

@Override
public void delete(Long id) throws RemoteException {
   System.out.println("Client calling delete process");
   PreparedStatement statement = null;
   try {
      statement = DatabaseUtilities.getConnection().prepareStatement(
            "DELETE FROM tbl_user WHERE user_idx=?");
      statement.setLong(1, id);
      statement.executeUpdate();
   } catch (SQLException e) {
      e.printStackTrace();
   } finally {
      if (statement != null) {
         try {
            statement.close();
         } catch (SQLException e) {
            e.printStackTrace();
```

```
            }
        }
    }
}

@Override
public User getUser(Long id) throws RemoteException {
    System.out.println("Client calling getUser process");
    PreparedStatement statement = null;
    try {
        statement = DatabaseUtilities.getConnection().prepareStatement(
                "SELECT * FROM tbl_user WHERE user_idx= ? "
        );
        ResultSet resultSet = statement.executeQuery();
        User user = null;
        if (resultSet.next()) {
            user = new User();
            user.setId(id);
            user.setUsername(resultSet.getString("username"));
            user.setEmail(resultSet.getString("email"));
            user.setPassword(resultSet.getString("password"));
        }

        return user;
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    } finally {
        if (statement != null) {
            try {
                statement.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

}

@Override
public List<User> getAllUser() throws RemoteException {
    System.out.println("Client calling getAllUser process");
    Statement statement = null;
    try {
        statement = DatabaseUtilities.getConnection().createStatement();
        ResultSet resultSet = statement.executeQuery("SELECT * FROM tbl_user ");
        List<User> list = new ArrayList<>();
        while (resultSet.next()) {
            User user = new User();
```

```
            user.setId(resultSet.getLong("user_idx"));
            user.setUsername(resultSet.getString("username"));
            user.setEmail(resultSet.getString("email"));
            user.setPassword(resultSet.getString("password"));
            list.add(user);
        }
        return list;
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    } finally {
        if (statement != null) {
            try {
                statement.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
  }

}
```

- **YANG KETIGA** kita buat project ***ApplicationClient*** dengan cara seperti sebelumnya.

  Seperti biasa kita buat dulu package nya seperti dibawah ini:



  Jangan lupa **Add Project -> *ApplicationAPI*** terlebih dahulu seperti cara sebelumnya.

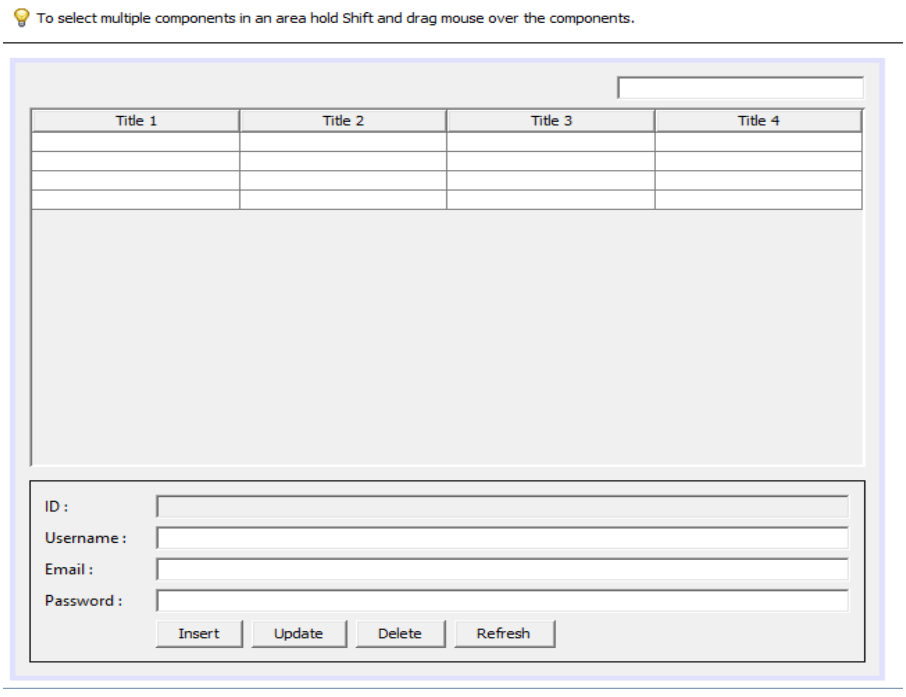
  Selanjutnya buat Kelas model ***TableModelUser*** :

```
import java.util.ArrayList;
import java.util.List;
import javax.swing.table.AbstractTableModel;
import org.jakartait.maul.code.api.entity.User;

/**
 *
 * @author Achmad Maulana
 */
```

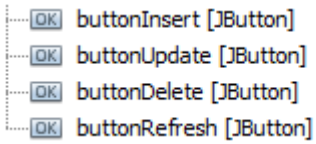
```
public class TableModelUser extends AbstractTableModel {

   private List<User> list = new ArrayList<User>();

   public TableModelUser() {
   }

   public User get(int row) {
      return list.get(row);
   }

   public void insert(User user) {
      list.add(user);
      fireTableDataChanged();
   }

   public void update(int row, User user) {
      list.set(row, user);
      fireTableDataChanged();
   }

   public void delete(int row) {
      list.remove(row);
      fireTableDataChanged();
   }

   public void setData(List<User> list) {
      this.list = list;
      fireTableDataChanged();
   }

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

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

   @Override
   public String getColumnName(int column) {
      switch (column) {
         case 0:
            return "Id";
         case 1:
            return "Username";
         case 2:
```

```
            return "Email";
        case 3:
            return "Password";
        default:
            return null;
    }
  }

  @Override
  public Object getValueAt(int rowIndex, int columnIndex) {
    switch (columnIndex) {
        case 0:
            return list.get(rowIndex).getId();
        case 1:
            return list.get(rowIndex).getUsername();
        case 2:
            return list.get(rowIndex).getEmail();
        case 3:
            return list.get(rowIndex).getPassword();
        default:
            return null;
    }
  }

}
```

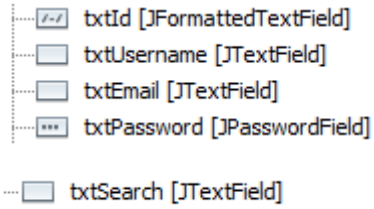
Selanjutnya buat JFrame dengan nama ***FrameUser*** pada packge view.



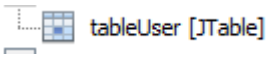
Penamaan variable button-buttonnya seperti ini:



Penamaan variable input text seperti ini:



Penamaan jtable seperti ini:



Setelah itu edit code pada ***FrameUser*** **:**

- Buat object TableModelUser dan buat variable UserService;

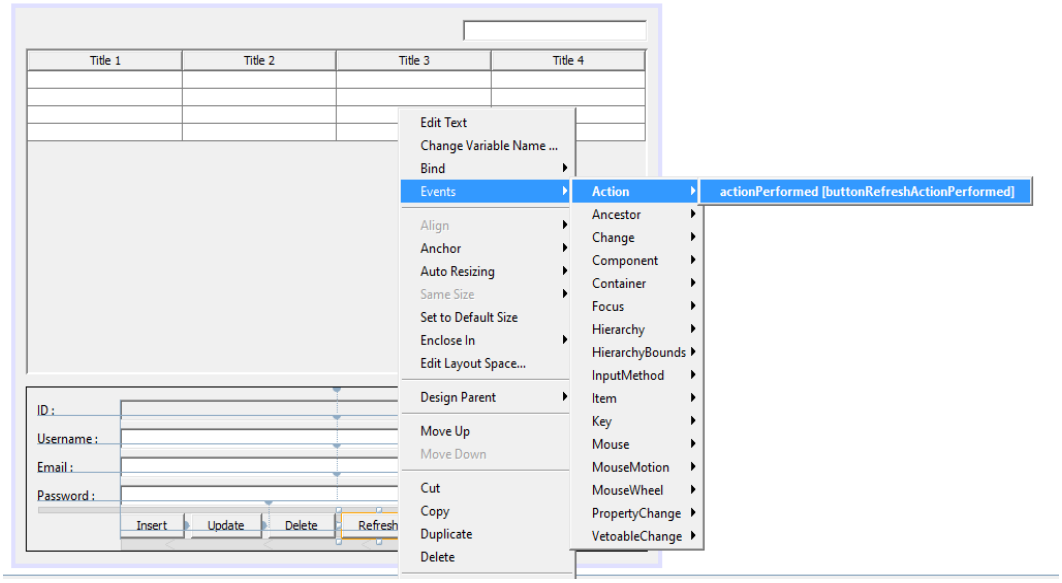
```
20  /**
21   *
22   * @author Achmad Maulana
23   */
24  public class FormUser extends javax.swing.JFrame {
25
26      private TableModelUser tableModelUser = new TableModelUser();
27      private UserService service;
28
```

- Selanjutnya edit konstuktor FormUser menjadi seperti dibawah

```
/**
    * Creates new form FormUser
    */
   public FormUser(UserService service) {
      this.service = service;

      try {
         tableModelUser.setData(service.getAllUser());

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

      initComponents();
```

```
    tableUser.setModel(tableModelUser);
    tableUser.getSelectionModel().addListSelectionListener(new  ListSelectionListener()
{

        @Override
        public void valueChanged(ListSelectionEvent e) {
            int row = tableUser.getSelectedRow();
            if (row !=-1) {
                User user = tableModelUser.get(row);
                txtId.setValue(user.getId());
                txtUsername.setText(user.getUsername());
                txtEmail.setText(user.getEmail());
                txtPassword.setText(user.getPassword());
            }
        }
    });

    TableRowSorter rowSorter = new TableRowSorter(tableModelUser);
    txtSearch.getDocument().addDocumentListener(new DocumentListener() {

        @Override
        public void insertUpdate(DocumentEvent e) {
            tableUser.setRowSorter(rowSorter);
            rowSorter.setRowFilter(RowFilter.regexFilter(txtSearch.getText()));
        }

        @Override
        public void removeUpdate(DocumentEvent e) {
            tableUser.setRowSorter(rowSorter);
            rowSorter.setRowFilter(RowFilter.regexFilter(txtSearch.getText()));
        }

        @Override
        public void changedUpdate(DocumentEvent e) {
            tableUser.setRowSorter(rowSorter);
            rowSorter.setRowFilter(RowFilter.regexFilter(txtSearch.getText()));
        }

    });
  }
```

Selanjutnya beri event pada tombol Refresh dengan cara klik kanan **Events > Action >actionPerformed**



Berikut kode pada action ***buttonRefresh*** :

```
    private void buttonRefreshActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        try {
            txtId.setValue(0);
            txtUsername.setText("");
            txtEmail.setText("");
            txtPassword.setText("");
            List<User> list = service.getAllUser();
            tableModelUser.setData(list);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
```

Selanjutnya beri event pada tombol ***buttonInsert***, berikut kodenya:

```
private void buttonInsertActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        try {
            User user = new User();
            user.setUsername(txtUsername.getText());
            user.setEmail(txtEmail.getText());
            user.setPassword(txtPassword.getText());
            User user1 = service.insert(user);

            tableModelUser.insert(user1);
            buttonRefreshActionPerformed(evt);
```

```
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
```

Event ***buttonUpdate*** **:**

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

        try {
            int row = tableUser.getSelectedRow();
            if (row ==-1) {
                return;
            }
            User user = tableModelUser.get(row);
            user.setUsername(txtUsername.getText());
            user.setEmail(txtEmail.getText());
            user.setPassword(txtPassword.getText());
            service.update(user);
            tableModelUser.update(row, user);
            buttonRefreshActionPerformed(evt);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
```

Event ***buttonDelete*** **:**

```
private void buttonDeleteActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        try {
            int row = tableUser.getSelectedRow();
            if (row ==-1) {
                return;
            }
            Long id = tableModelUser.get(row).getId();
            service.delete(id);
            tableModelUser.delete(row);
            buttonRefreshActionPerformed(evt);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
```

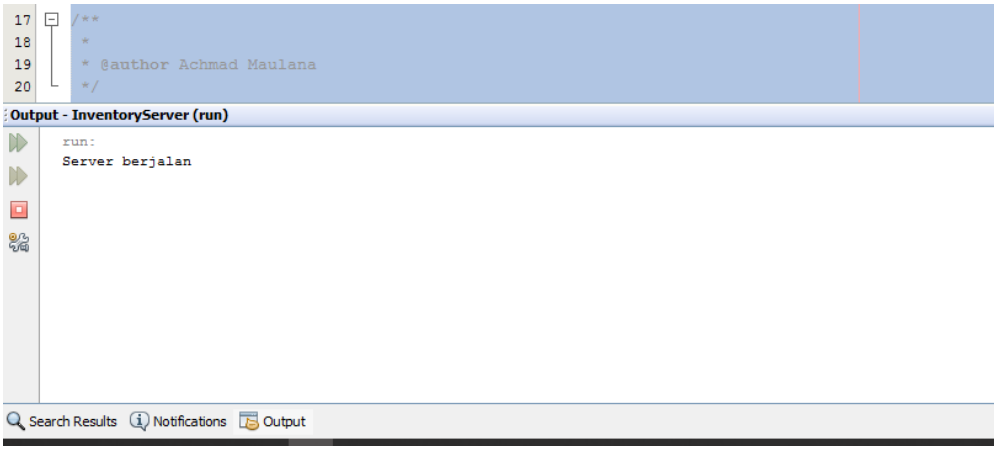
Yang terakhir kita buat kelas main dengan nama ***MainClass***

```
import java.rmi.NotBoundException;
```

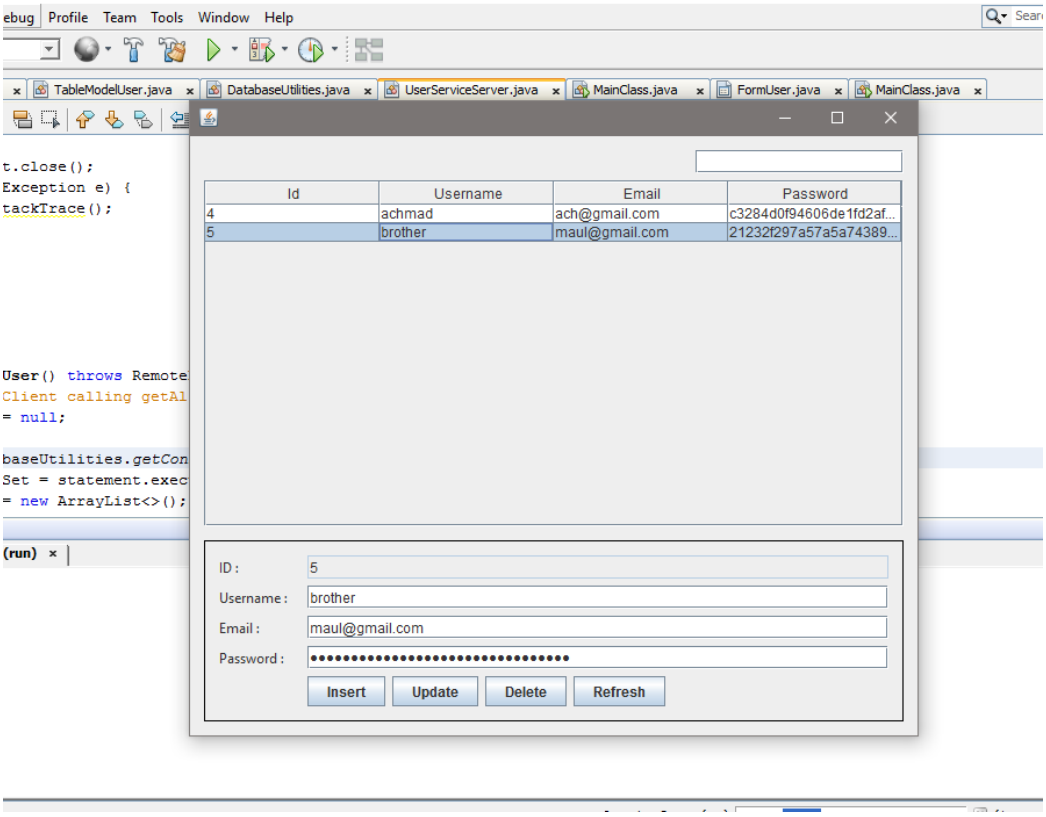
```
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import javax.swing.SwingUtilities;
import org.jakartait.maul.code.api.HelloInterface;
import org.jakartait.maul.code.client.view.FormUser;
import org.jakartait.maul.code.service.UserService;

/**
 *
 * @author Achmad Maulana
 */
public class MainClass {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws RemoteException, NotBoundException {
        Registry client = LocateRegistry.getRegistry("localhost", 6789);
        UserService service = (UserService) client.lookup("service");
        SwingUtilities.invokeLater(new Runnable() {

            @Override
            public void run() {
                FormUser formUser= new FormUser(service);
                formUser.setVisible(true);
            }
        });
    }
}
```

TEST APLIKASI

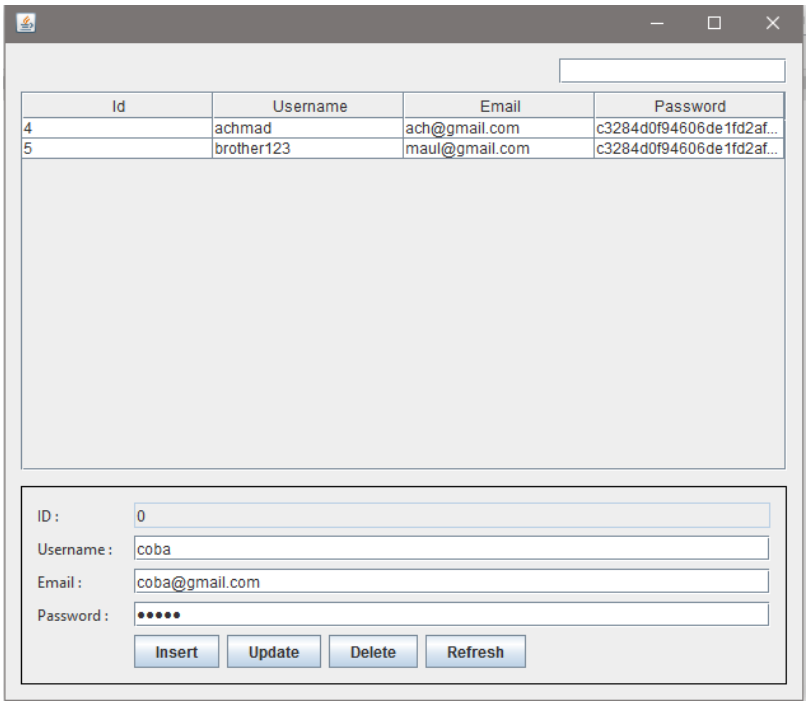
1. Jalankan ApplicationServer Terlebih dahulu



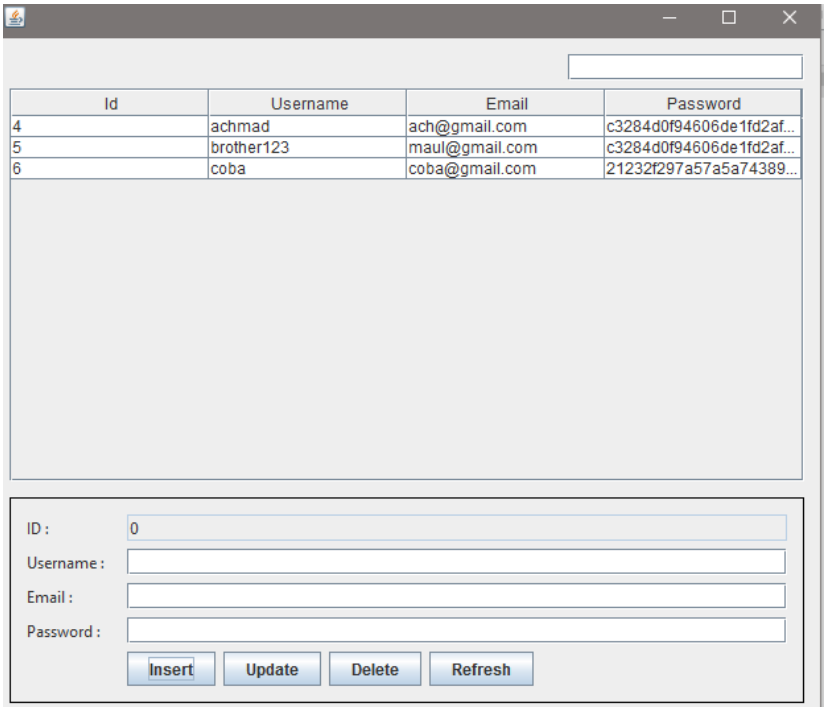
2. Selanjutnya Jalankan ApplicationClient



Test inpu user baru:



Hasilnya



Jika ingin sourcecode aplikasi bisa dengan cara mengemail saya. Skian smoga bermanfaat.

REFERENSI

[1] Pl, "Remote Method Invocation (RMI)," pp. 693–698, 2004.
[2] Ratnasari Nur Rohmah, "Client/Server dengan Java Remote Method Invocation (Java RMI)," *J. Tek. Elektro Dan Komput. Emit.*, vol. 3, p. 5, 2003.

BIOGRAFI PENULIS

Achmad Maulana, Lahir pada Agustus 1989. Penulis Menyelesaikan Pendidikan Sarjana Teknik Informatika (S1) di Universitas Indraprasta PGRI, Jakarta tahun 2011 dan Menyelesaikan pendirikan Magister Ilmu Komputer di Universitas Budi Luhur , Jakarta tahun 2016. Kompentensi inti pada bidang *software engineering*. Penulisa juga aktif dalam menulis blog pada website http://jakartait.com.