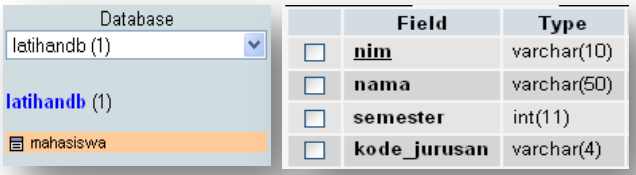
# BAB XI

# KONEKSI JAVA – MYSQL(2)

Kalau belum baca yang jilid pertamanya anda di haruskan membaca yang jilid pertamanya terlebih dahulu

## A. Buat Database



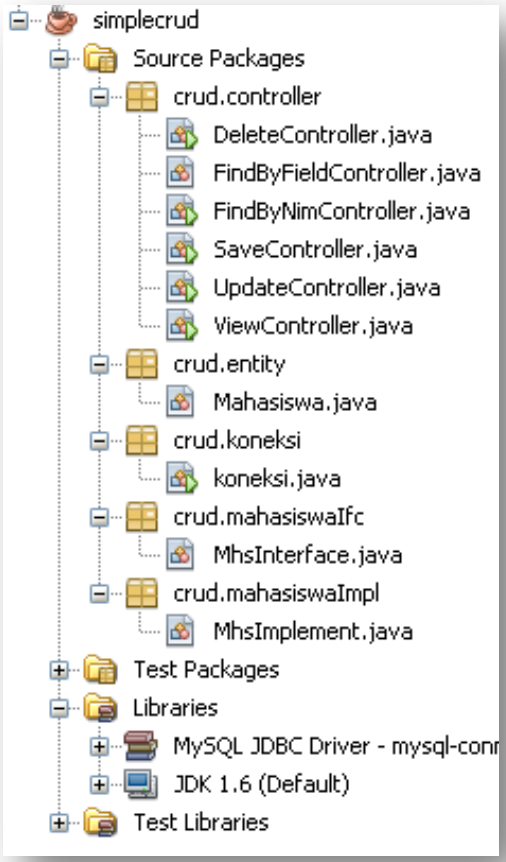

| | Field | Type |
|---|---|---|
| | nim | varchar(10) |
| | nama | varchar(50) |
| | semester | int(11) |
| | kode_jurusan | varchar(4) |

Nah, pada contoh kali ini saya membuat database **latihandb** dengan nama tablenya mahasiswa

| | | | nim | nama | semester | kode_jurusan |
|---|---|---|---|---|---|---|
| ☐ | ✎ | ✕ | 020410016 | Akhmad Bilyyasif | 4 | 0204 |
| ☐ | ✎ | ✕ | 020410017 | Doni Aliandi | 4 | 0204 |

Kemudiam isi table tersebut beberapa record

## B. Buat Projek Java




**keterangan :**

1. MhsInterface : kelas interface yang membuat method – method yang ada dalam crud ex: insert, update delete, view, find.
2. MhsImplements : merupakan kelas yang mengimplementasikan kelas interface MhsInterface
3. Mahasiswa : merupakan kelas yang berisikan variable yang akan menampung data-data mahasiswa, isinya adalah setter – getter
4. Koneksi : merupakan kelas yang mengkoneksikan antara database dan program java
5. Sedangkan kelas yang ada dipackage controller, merupakan main dari method – method yang ada dalam kelas MhsImplements

Berikut adalah isi dari kelas2 diatas :

1. **Kelas Koneksi**

```
package crud.koneksi;

import java.sql.Connection;
import java.sql.DriverManager;

public class koneksi {

   private Connection connection;

   public Connection openCon() {
      try {
         // Class.forName("com.mysql.jdbc.Driver");
         DriverManager.registerDriver(new com.mysql.jdbc.Driver());
         connection = DriverManager.getConnection("jdbc:mysql://localhost/latihandb", "root", "");
         //System.out.println("Koneksi Ok");
      } catch (Exception e) {
         System.err.println("gagal " + e);
      }
      return connection;
   }

   public Connection closCon(){
      try {
         if(connection !=null){
            connection.close();
          //  System.out.println("KOneksi Terputus");
         }
      } catch (Exception e) {
         System.err.println("gagal2s" + e);
      }
      return connection;
   }
   public static void main(String[] args) {
      koneksi k =  new koneksi();
      k.openCon();
      k.closCon();
   }
}
```

**2. Kelas Mahasiswa**

```
package crud.entity;

public class Mahasiswa {
        private int semester;
        private String nim, nama, kode_jurusan;
        private boolean NimBool; //PENGECEKAN NIM

        public Mahasiswa() {
        }

        public boolean isNimBool() {
            return NimBool;
        }
        public void setNimBool(boolean NimBool) {
            this.NimBool = NimBool;
        }

        public String getKode_jurusan() {
             return kode_jurusan;
        }
        public void setKode_jurusan(String kode_jurusan) {
            this.kode_jurusan = kode_jurusan;
        }

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

        public String getNim() {
            return nim;
        }
        public void setNim(String nim) {
            this.nim = nim;
        }

        public int getSemester() {
            return semester;
        }
        public void setSemester(int semester) {
            this.semester = semester;
        }

}
```

### 3. Kelas MhsInterface

```
package crud.mahasiswaIfc;

import crud.entity.Mahasiswa;
import java.util.List;

public interface MhsInterface {

        List<Mahasiswa> GetAll();

        String SaveData(Mahasiswa m);

        String UpdateData(Mahasiswa m);

        String DeleteData(String nim);

        Mahasiswa GetByNim(String nim);

        List<Mahasiswa> GetByField(String Field, String isi);
}
```

### 4. Kelas MhsImplements

Dalam hal ini sebenarnya saya memisah2 methode yang ada di kelas MhsImplements. Tapi saat mencoba kalian satuin aja.

```
package crud.mahasiswaImpl;

import crud.entity.Mahasiswa;
import crud.koneksi.koneksi;
import crud.mahasiswaIfc.MhsInterface;
import java.sql.*;
import java.util.*;

public class MhsImplement implements MhsInterface {

   String Q;
   koneksi k = new koneksi();
```

Ingat yaaa... iomportnya semuanya memakai **java.sql.*** jangan menggunakan **com.***

Ini merupakan bagian atas dari kelas Mhs Interface, “Q” merupakn variable yang natinya akan menampung Query dari tabel.

Dibawah ini merupakan penjabaran dari methode yang diimplementasikan dari kelas interface :

a. **Get All()**

```
public List<Mahasiswa> GetAll() {
    List<Mahasiswa> list = new ArrayList<Mahasiswa>();
        try {
            Q = "SELECT * FROM mahasiswa";
            Connection c = k.openCon();
            Statement st = c.createStatement();
            ResultSet rs = st.executeQuery(Q);
            while (rs.next()) {
                Mahasiswa mhs = new Mahasiswa();
                mhs.setNim(rs.getString("nim"));
                mhs.setNama(rs.getString("nama"));
                mhs.setSemester(rs.getInt("semester"));
                mhs.setKode_jurusan(rs.getString("kode_jurusan"));
                list.add(mhs);
            }

        } catch (SQLException e) {
            System.err.println("GAGAL TAMPIL " + e);
        }
        return list;
}
```

Dalam tampil data tipe datanya tidak harus list, memakai vector juga bisa, baca tutorial saya bab IX atau bab V.

b. **Save Data()**

```
public String SaveData(Mahasiswa m) {
    String hasil = "";
        try {
            Q = "INSERT into mahasiswa VALUES("
                    + " '" + m.getNim() + "' , "
                    + " '" + m.getNama() + "' , "
                    + " " + m.getSemester() + " , "
                    + " '" + m.getKode_jurusan() + "' ) " ;
            Connection c = k.openCon();
            PreparedStatement ps = c.prepareStatement(Q);
            ps.executeUpdate();
            hasil = "SUKSES INSERT";
        } catch (SQLException e) {
            hasil = "GAGAL INSERT " + e;
        }
        return hasil;
}
```

c. **Update Data()**

```
public String UpdateData(Mahasiswa m) {
    String hasil = "";
    try {
        Q = "UPDATE mahasiswa SET nama = '" + m.getNama() + "',
semester = " + m.getSemester() + " , kode_jurusan = '" +
m.getKode_jurusan() + "' WHERE nim = '" + m.getNim() + "'";

        Connection c = k.openCon();
        PreparedStatement ps = c.prepareStatement(Q);
        ps.executeUpdate();
        hasil = "SUKSES UPDATE";
    } catch (SQLException e) {
        hasil = "GAGAL UPDATE" + e;
    }
    return hasil;
}
```

d. **Delete Data()**

```
public String DeleteData(String nim) {
    String hasil = "";
    try {
      Q = "DELETE FROM mahasiswa WHERE nim = '" + nim + "'";
      Connection c = k.openCon();
      PreparedStatement ps = c.prepareStatement(Q);
      ps.executeUpdate();
      hasil = "SUKSES DELETE";
    } catch (SQLException e) {
      hasil = "GAGAL DELETE " + e;
    }
    return hasil;
  }
```

Dalam insert,update,delete juga banyak caranya, baca tutorial crud java menggunakan mysql yang sebelum - sebelumnya

Seperti yang sering saya bolang, bahwa insert update delete itu merupakan satu kesatuan dimana code untuk crudnya hapir mirip beda di Querynya saja.
Sedikit mengingatkan, kalau insert update delete menggunakan **preparestatement** dan eksekusinya menggunakan **executeupdate()** karena dia merubah data**.** Sedangkan tampil menggunakan **Statement** dan **resultSet** eksekusinya menggunakan **ExecuteQuery()** karena ia sifatnya hanya menampilkan data**.** Dan ingat importnya memakai java.sql.*.

**e. Get By Nim()**

```
public Mahasiswa GetByNim(String nim) {
    Mahasiswa mhs = new Mahasiswa();
    mhs.setNimBool(false);
    try {
      Q = "SELECT * FROM mahasiswa WHERE nim = '" + nim + "'";
      Connection c = k.openCon();
      Statement st = c.createStatement();
      ResultSet rs = st.executeQuery(Q);
      if (rs.next()) {
        mhs.setNim(rs.getString("nim"));
        mhs.setNama(rs.getString("nama"));
        mhs.setSemester(rs.getInt("semester"));
        mhs.setKode_jurusan(rs.getString("kode_jurusan"));
        mhs.setNimBool(true);
      }
    } catch (SQLException e) {
      System.err.println("GAGAL TAMPIL " + e);
    }
    return mhs;
  }
```

getBynim maksudnya adalah, menampilkan data berdasarkan nim. Untuk source kodenya anda bisa cari sendiri yang lebih efisien, ia menggunakan **if** karena data yang akan ditampilkan Cuma ada satu, karena yang dicari berdasarkan primary keynya

**f. Get By Field()**

```
public List<Mahasiswa> GetByField(String Field, String isi) {
    List<Mahasiswa> list = new ArrayList<Mahasiswa>();
    if (Field.equalsIgnoreCase("semester")) {
        int sms = Integer.parseInt(isi);
        Q = "SELECT * FROM mahasiswa WHERE " + Field + " = " + sms;
    } else {
        Q = "SELECT * FROM mahasiswa WHERE " + Field + " like  '%" + isi + "%'";
    }
    try {
            Connection c = k.openCon();
            Statement st = c.createStatement();
            ResultSet rs = st.executeQuery(Q);
            while (rs.next()) {
                Mahasiswa mhs = new Mahasiswa();
                mhs.setNim(rs.getString("nim"));
                mhs.setNama(rs.getString("nama"));
                mhs.setSemester(rs.getInt("semester"));
                mhs.setKode_jurusan(rs.getString("kode_jurusan"));
                list.add(mhs);
                mhs.setNimBool(true);
            }
    } catch (SQLException e) {
            System.err.println("GAGAL TAMPIL " + e);
    }
    return list;
}
```

Sedangkan getbyField menampilkan data berdasarkan nama fieldnya dan isinya

## 5. Package Controller

Berisikan kelas2 yang mengeksekusi methode yang ada di kelas MhsImplements

### a. View Controller

```
public class ViewController {
   public ViewController() {
      MhsImplement mhsImplement = new MhsImplement();
      List<Mahasiswa> list = mhsImplement.GetAll();
      System.out.println("\tDATA MAHASISWA");
      System.out.println("\t=-=-=-=-==-=-=");
      for (Iterator<Mahasiswa> it = list.iterator(); it.hasNext();) {
            Mahasiswa mahasiswa = it.next();
            System.out.println("NIM        : " + mahasiswa.getNim());
            System.out.println("Nama       : " + mahasiswa.getNama());
            System.out.println("Semester   : " + mahasiswa.getSemester());
            System.out.println("Kode Jurusan : " + mahasiswa.getKode_jurusan());
            System.out.println("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");
      }
}
```

Kalau yang belum tau for iterator baca tutorial saya sebelumnya.

Methode Mainnya, tinggal panggil konstraktornya

### b. Save Controller

```
public class SaveController {
      public SaveController() {
            Mahasiswa m = new Mahasiswa();
            MhsImplement mi = new MhsImplement();
            m.setNim("020410015");
            m.setNama("Nurelah");
            m.setSemester(5);
            m.setKode_jurusan("0203");
            System.out.println("Hasil : " + mi.SaveData(m));
      }
}
```

### c. Update Controller

```
public class UpdateController {
        public UpdateController() {
                Mahasiswa m = new Mahasiswa();
                MhsImplement mi = new MhsImplement();
                m.setNim("020410015");
                m.setNama("Ria Rohana");
                m.setSemester(4);
                m.setKode_jurusan("0204");
            System.out.println("Hasil : " + mi.UpdateData(m));
        }
}
```

### d. Delete Controller

```
public class DeleteController {
        public DeleteController() {
                MhsImplement mi = new MhsImplement();
                System.out.println("Hasil : " + mi.DeleteData("020410015"));
        }
}
```

Nih bagi yang belum paham methode mainnya kaya gimana mah...

```
public static void main(String[] args) {
    DeleteController dc = new DeleteController();
}
```

Tinggal anda ubah saja instansiasinya,

Jangan lupa anda pelajari dan jangan lupa anda juga harus mencobanyaa...

### e. FindByNim Controller

```
public class FindByNimController {
   public FindByNimController() {
      MhsImplement mhsImplement = new MhsImplement();
      System.out.println("\tDATA MAHASISWA");
      System.out.println("\t=-=-=-=-==-=-=");
      Mahasiswa mahasiswa = mhsImplement.GetByNim("020410016");
      if (mahasiswa.isNimBool()) {
         System.out.println("NIM        : " + mahasiswa.getNim());
         System.out.println("Nama       : " + mahasiswa.getNama());
         System.out.println("Semester   : " + mahasiswa.getSemester());
         System.out.println("Kode Jurusan : " + mahasiswa.getKode_jurusan());
         System.out.println("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");
      } else {
         System.out.println("DATA KOSONG");
      }
   }
}
```

### f. FindByField Controller

```
public FindByFieldController() {
      MhsImplement mhsImplement = new MhsImplement();
      List<Mahasiswa> list = mhsImplement.GetByField("nama", "an");
      System.out.println("\tDATA MAHASISWA");
      System.out.println("\t=-=-=-=-==-=-=");
      if (list.isEmpty()) {
         System.out.println("\tDATA TIDAK ADA ");
      } else {
         for (Iterator<Mahasiswa> it = list.iterator(); it.hasNext();) {
            Mahasiswa mahasiswa = it.next();
            System.out.println("NIM        : " + mahasiswa.getNim());
            System.out.println("Nama       : " + mahasiswa.getNama());
            System.out.println("Semester   : " + mahasiswa.getSemester());
            System.out.println("Kode Jurusan : " + mahasiswa.getKode_jurusan());
            System.out.println("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");
         }
      }
   }
}
```

# ☺ SAMPAI KETEMU DIEDISI BERIKUTNYA ☺

**Nama : Akhmad Bilyyasif**

**Email : asief.asief@yahoo.com**

**BANDUNG, 13 MEI 2012**