BAB XIII

KONEKSI DATABASE , TABEL MENGGUNAKAN *JAVA CODE*

A. CAPAIAN PEMBELAJARAN

1. Mahasiswa dapat dapat memahami Koneksi DATABASE
2. Mahasiswa dapat menggunakan Koneksi Database dengan Kode Java
3. Mahasiswa dapat Mempraktekan Koneksi Database dan Tabel dengan Kode Java

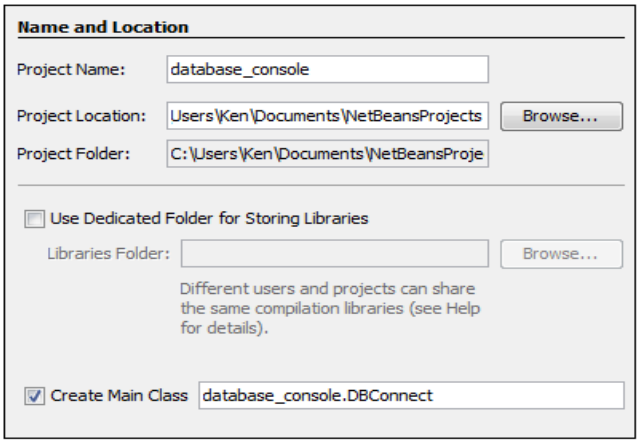
B. MATERI

1. Koneksi Database

Pada bagian selanjutnya, kita akan membuat Form Java yang memuat informasi dari database. Form akan memiliki tombol record selanjutnya dan Sebelumnya untuk menggulir data. Record individu kemudian akan ditampilkan di TextFields. Kita juga akan menambahkan tombol untuk Update Record, Hapus Record, dan buat Record baru di database.

Untuk memulai, cara sederhana, kita akan menggunakan jendela terminal/konsol untuk menampilkan hasil dari database.

Jadi mulailah proyek baru untuk dengan mengklik ***File*** > ***New Project*** dari menu NetBeans. Buat ***Java Application*** . Panggil *Package* ***database_console***, dan Main lass ***DBConnect*** :



Gambar 13. 1 Membuat Package Baru

Ketika kita mengklik finish, kode kita akan terlihat seperti ini:

```
package database_console;

public class DBConnect {

    public static void main(String[] args) {

    }
}
```

Untuk terhubung ke database, Anda memerlukan objek Connection. Objek *Connection* menggunakan *DriverManager*. *DriverManager* memasukkan nama pengguna basis data Anda, kata sandi Anda, dan lokasi basis data.

Tambahkan tiga pernyataan impor ini ke bagian atas kode Anda:

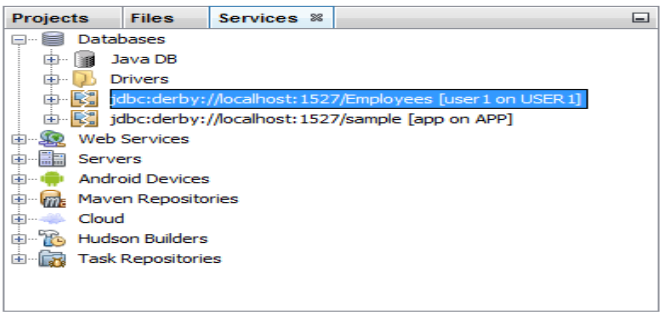
```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

Untuk mengatur koneksi ke database, kodenya adalah seperti dibawah ini:

```
Connection con =
DriverManager.getConnection( host,
username, password );
```

Jadi DriverManager memiliki metode yang disebut *getConnection*. Ini membutuhkan nama host (yang merupakan lokasi database Anda), nama pengguna, dan kata sandi. Jika koneksi berhasil, objek Connection dibuat, yang kita sebut con.

Kita bisa mendapatkan alamat host dengan melihat tab Layanan di sebelah kiri NetBeans:



Gambar 13. 2 Alamat Database terpilih

Alamat database yang disorot di atas adalah:

jdbc:derby://localhost:1527/Employees

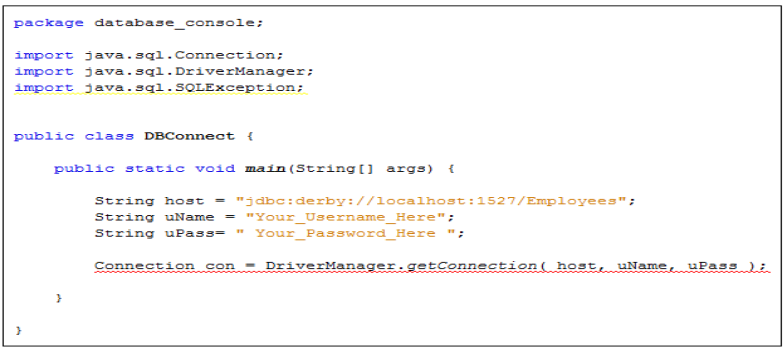
Bagian pertama, ***jdbc:derby://localhost***, adalah tipe database dan server yang kita gunakan. **1527** adalah nomor port. Databasenya adalah ***Employess***. Ini semua masuk dalam variabel String:

String host = "jdbc:derby://localhost:1527/Employees";

Dua string lagi dapat ditambahkan untuk username dan password :

String uName = "Nama_Pengguna_Anda_Di Sini";
String uPass= "Kata_Pass_Anda_Di Sini";

Penambahan tiga string ini sebelum objek koneksi dan kode kita akan terlihat seperti ini:



```
package database_console;

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


public class DBConnect {

    public static void main(String[] args) {

        String host = "jdbc:derby://localhost:1527/Employees";
        String uName = "Your_Username_Here";
        String uPass= " Your_Password_Here ";

        Connection con = DriverManager.getConnection( host, uName, uPass );

    }

}
```

Seperti yang kita lihat pada gambar di atas, ada garis bawah bergelombang berwarna merah untuk kode Koneksi. Alasan untuk ini adalah karena kita belum menjebak kesalahan spesifik yang akan muncul saat menghubungkan ke database - kesalahan SQLException.

Ini adalah DriverManager yang mencoba untuk terhubung ke database. Jika gagal (alamat host salah, misalnya) maka itu akan mengembalikan kesalahan SQLException. Kita perlu menulis kode untuk menangani potensi kesalahan ini. Dalam kode di bawah ini, kita menjebak kesalahan di bagian tangkapan dari pernyataan try … catch:

```
try {
}
catch ( SQLException err ) {
System.out.println( err.getMessage( ) );
}
```

Di antara tanda kurung siku, kita telah menyiapkan objek SQLException yang disebut err. Kita kemudian dapat menggunakan metode getMessage dari objek err ini.

```
package database_console;

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


public class DBConnect {

    public static void main(String[] args) {

        try {
            String host = "jdbc:derby://localhost:1527/Employees";
            String uName = "Your_Username_Here";
            String uPass= " Your_Password_Here ";

            Connection con = DriverManager.getConnection( host, uName, uPass );
        }
        catch ( SQLException err ) {
            System.out.println( err.getMessage( ) );
        }
    }

}
```

Tambahkan blok try …catch di atas ke kode, dan pindahkan empat baris kode koneksi kita ke bagian try. Kode kita kemudian akan terlihat seperti ini:

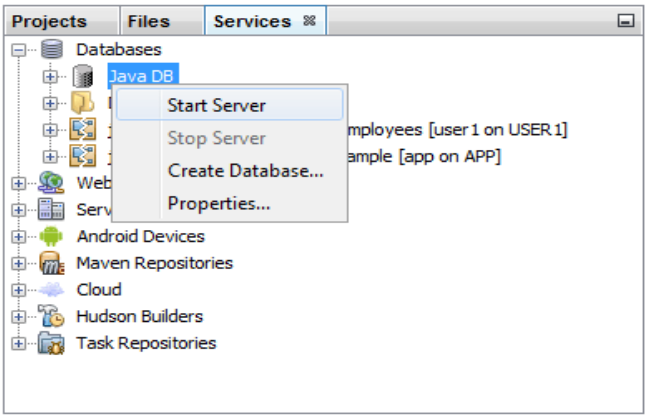
Uji coba run kode kita dan lihat apa yang terjadi.

mungkin saja mendapatkan pesan kesalahan ini di jendela konsol:

"java.net.ConnectException : Error connecting to server localhost on port 1527 with message Connection refused: connect."

Jika ya, itu berarti belum terhubung ke server database kita . Dalam hal ini, klik kanan pada Java DB

di ***Services Windows***. Dari menu yang muncul, klik ***Start Server***:



Gambar 13. 3 Start Server Database

Kita perlu memastikan bahwa *firewall* apa pun yang kita miliki tidak memblokir koneksi ke server. *Firewall* yang baik akan segera menampilkan pesan yang memperingatkan bahwa ada sesuatu yang mencoba untuk melewatinya, dan menanyakan apakah kita ingin mengizinkan atau menolaknya. Saat kita mengizinkan koneksi, jendela keluaran NetBeans kita akan mencetak pesan berikut:

"Apache Derby Network Server - 10.4.1.3 - (648739) started and ready to accept connections on port 1527 at DATE_AND_TIME_HERE"

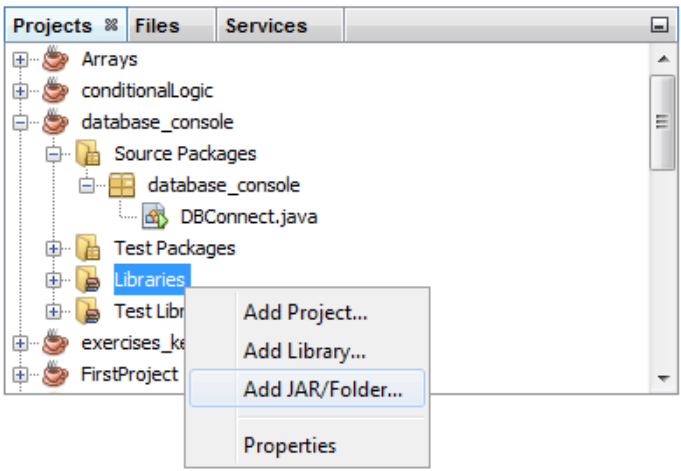
Setelah server kita dimulai, jalankan program lagi. Ada kemungkinan besar kita akan mendapatkan pesan kesalahan lain:

"No suitable driver found for jdbc:derby://localhost:1527/Employees"

Alasan kesalahan ini adalah bahwa ***DriverManager*** membutuhkan Driver untuk terhubung ke database. Contoh driver adalah Driver client dan Driver Tertanam. Anda dapat mengimpor salah satunya sehingga ***DriverManager*** dapat melakukan tugasnya.

Klik pada tab *Project* di sebelah kiri ***Services Windows*** di NetBeans. (Jika Anda tidak dapat melihat tab Proyek, klik ***Windows*** >***Project*** dari bilah menu di bagian atas NetBeans.)

Temukan proyek kita dan perluas entri. Klik kanan ***Library***. Dari menu yang muncul, pilih ***Add Jar/Folder***:



Gambar 13. 4 Libarary Jar

Ketika Anda mengklik Add Jar/Folder, sebuah kotak dialog akan muncul. Apa yang kita lakukan di sini adalah menambahkan file Java Archive ke proyek kita . Tetapi file JAR yang ditambahkan adalah untuk ***derby client Driver***. Jadi, kita perlu menemukan folder ini. Pada komputer yang menjalankan Windows, ini akan berada di lokasi berikut:

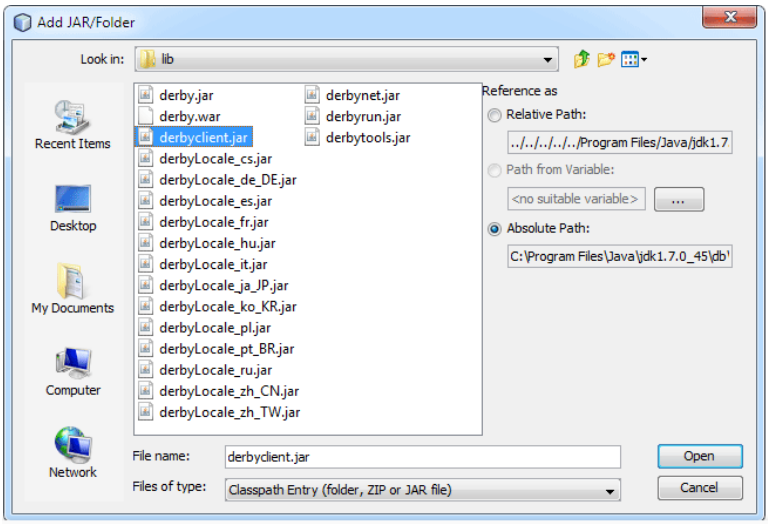
C:\Program Files\Java\jdk1.8.0_45\db\lib

File yang kita cari bernama derbyclient.jar. Jika kita tidak dapat menemukannya, atau menggunakan sistem operasi selain Windows, lakukan pencarian untuk file ini. Perhatikan lokasi file.

Jika file tidak ada di sistem kita, kita dapat mendownload pada link dibawah ini :

http://db.apache.org/derby/derby_downloads.html

Di kotak dialog untuk menambahkan file JAR, pilih file derbyclient.jar:



Gambar 13. 5 Library Derby Client

Klik Open dan file akan ditambahkan ke library project kita :

Sekarang setelah kita memiliki Client Driver yang ditambahkan ke project, jalankan kembali program kita. seharusnya sudah bebas dari kesalahan. (Jendela Output hanya akan menampilkan *Run, and Build Successful*..).

2. Koneksi ke Tabel Database

Sekarang kita telah terhubung ke database, langkah selanjutnya adalah mengakses tabel di database. Untuk ini, kita perlu menjalankan Pernyataan SQL, dan kemudian memanipulasi semua baris dan kolom yang dikembalikan.

Untuk mengeksekusi pernyataan SQL di Table kita, kita menyiapkan *Statement object*. Jadi tambahkan baris impor ini ke bagian atas kode Anda:

import java.sql.Statement;

Di bagian try dari try ... catch block tambahkan baris berikut (tambahkan tepat di bawah baris Connection kita):

Statement stmt = con.createStatement( );

Di sini, kita buat objek Pernyataan yang disebut stmt. Objek Statement membutuhkan objek Connection, dengan metode createStatment.

Kita juga membutuhkan Pernyataan SQL untuk *Statement Object* untuk dieksekusi. Jadi tambahkan baris ini ke kode kita :

String SQL = "SELECT * FROM Workers";

Pernyataan di atas memilih semua record dari tabel database *Employee*.

Kita dapat meneruskan kueri SQL ini ke metode *Statement Object* yang disebut *executeQuery*. Objek Statement kemudian akan bekerja mengumpulkan semua record yang cocok dengan query kita.

Namun, metode executeQuery mengembalikan semua record dalam sesuatu yang disebut ResultSet.

Sebelum dijelaskan apa itu ResultSet, tambahkan baris impor berikut ke bagian atas kode Anda:

import java.sql.ResultSet;

Sekarang tambahkan baris ini tepat di bawah baris SQL String:

ResultSet rs = stmt.executeQuery(SQL);

Jadi objek ResultSet disebut rs. Ini akan menampung semua record dari tabel database. Sebelum kita melangkah lebih jauh, berikut adalah penjelasan tentang apa itu ResultSets.

***ResultSets* pada Java**

ResultSet adalah cara untuk menyimpan dan memanipulasi record yang dikembalikan dari kueri SQL. ResultSets hadir dalam tiga tipe berbeda. Jenis yang kita gunakan bergantung pada apa yang ingin kita lakukan dengan data:

a. Apakah Anda hanya ingin bergerak maju melalui *record*, dari awal hingga akhir?
b. Apakah Anda ingin bergerak maju dan mundur melalui *record*, serta mendeteksi perubahan apa pun yang dibuat pada *record* ?

c. Apakah Anda ingin bergerak maju dan mundur melalui *record*, tetapi tidak peduli dengan perubahan apa pun yang dilakukan pada catatan?

Ketik nomor 1 pada daftar di atas disebut TYPE_FORWARD_ONLY ResultSet. Nomor 2 dalam daftar adalah TYPE_SCROLL_SENSITIVE ResultSet. Opsi ResultSet ketiga disebut TYPE_SCROLL_INSENSITIVE.

Jenis ResultSet berada di antara tanda kurung bundar createStement:

Statement stmt = con.createStatement( );

Kalau kita membiarkan kurung buk kosong, kita akan mendapatkan ResultSet default, yaitu TYPE_FORWARD_ONLY. Di bagian berikutnya, kita akan menggunakan salah satu jenis lainnya. Tetapi kita menggunakannya seperti ini:

Statement stmt = con.createStatement(
ResultSet.TYPE_SCROLL_SENSITIVE);

Jadi Anda ketik dulu kata ResultSet. Setelah titik, Anda menambahkan jenis ResultSet yang ingin Anda gunakan.

Namun, itu tidak berakhir di situ. Jika kita ingin menggunakan TYPE_SCROLL_SENSITIVE atau TYPE_SCROLL_INSENSITIVE Anda juga perlu menentukan apakah ResultSet adalah Read Only atau

apakah itu Dapat Diperbarui. Kita melakukan ini dengan dua konstanta bawaan: CONCUR_READ_ONLY dan CONCUR_UPDATABLE. Sekali lagi, ini muncul setelah kata ResultSet:

ResultSet.CONCUR_READ_ONLY

ResultSet.CONCUR_UPDATABLE

Ini mengarah ke baris kode yang agak panjang:

Statement stmt = con.createStatement(
ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);

Satu hal lagi untuk membiasakan diri dengan ResultSets adalah sesuatu yang disebut Kursor. Kursor sebenarnya hanyalah penunjuk ke baris tabel. Saat pertama kali memuat record ke dalam ResultSet, kursor menunjuk tepat sebelum baris pertama dalam tabel. kemudian menggunakan metode untuk memanipulasi kursor. Tetapi idenya adalah untuk mengidentifikasi baris tertentu di tabel kita.

Menggunakan *ResultSet*

Setelah kita memiliki semua record dalam kumpulan Hasil, ada metode yang dapat kita gunakan untuk memanipulasi record . Berikut adalah metode yang paling sering digunakan:

next	Moves the Cursor to the next row in your table. If there are no more rows in the table, a value of False will be returned.
previous	Moves the Cursor back one row in your table. If there are no more rows in the table, a value of False will be returned.
first	Moves the Cursor to the first row in your table
last	Moves the Cursor to the last row in your table
absolute	Moves the Cursor to a particular row in the table. So absolute(5) will move the Cursor to row number 5 in the table

ResultSet juga memiliki metode yang dapat digunakan untuk mengidentifikasi kolom (bidang) tertentu dalam satu baris. Anda dapat melakukannya dengan menggunakan nama kolom, atau dengan menggunakan nomor indeksnya. Untuk tabel *Worker* , kita menyiapkan empat kolom. Yaitu memiliki nama berikut: ID, First_Name, Last_Name, dan Job_Title. Oleh karena itu, nomor indeksnya adalah 1, 2, 3, 4.

Kita menyiapkan kolom ID untuk menyimpan nilai Integer. Metode yang digunakan untuk mendapatkan nilai integer dalam kolom adalah getInt:

int id_col = rs.getInt("ID");

Di sini, kita telah menyiapkan variabel integer yang disebut id_col. kemudian menggunakan metode getInt dari objek ResultSet, yang disebut rs. Di antara tanda

kurung buka, kita memiliki nama kolom. Kita bisa menggunakan nomor Indeks sebagai gantinya:

```
int id_col = rs.getInt(1);
```

Perhatikan bahwa nomor Indeks tidak memiliki tanda kutip, tetapi namanya memiliki.

Untuk tiga kolom lainnya di tabel database kita, kita atur untuk menahan Strings. Oleh karena itu, kita membutuhkan metode getString:

```
String first_name =
rs.getString("First_Name");
```

Atau kita bisa menggunakan nomor Indeks:

```
String first_name = rs.getString(2);
```

Karena Kursor ResultSet menunjuk tepat sebelum record pertama saat data dimuat, kita perlu menggunakan metode berikutnya untuk pindah ke baris pertama. Kode berikut akan mendapatkan catatan pertama dari tabel:

```
rs.next( );
int id_col = rs.getInt("ID");
String first_name =
rs.getString("First_Name");
String last_name =
rs.getString("Last_Name");
String job = rs.getString("Job_Title");
```

Perhatikan bahwa rs.next didahulukan dalam kode ini. Ini akan memindahkan kursor ke Record pertama dalam tabel.

Kita dapat menambahkan baris cetak ke kode Anda untuk menampilkan catatan di jendela Output:

System.out.println(id_col+" " +first_name +" "
+last_name+" " + job);

Berikut tampilan kode kita sekarang :

```
try {
    String host = "jdbc:derby://localhost:1527/Employees";
    String uName = "Your_Username_Here";
    String uPass= " Your_Password_Here ";
    Connection con = DriverManager.getConnection( host, uName, uPass );

    Statement stmt = con.createStatement();
    String SQL = "SELECT * FROM Workers";
    ResultSet rs = stmt.executeQuery( SQL );

    rs.next( );
    int id_col = rs.getInt("ID");
    String first_name = rs.getString("First_Name");
    String last_name = rs.getString("Last_Name");
    String job = rs.getString("Job_Title");

    System.out.println(id_col + " " + first_name + " " + last_name + " " + job);
}
catch ( SQLException err ) {
    System.out.println( err.getMessage( ) );
}
```

Jika kita ingin menelusuri semua record dalam tabel, kita dapat menggunakan perulangan *loop*. Karena metode berikutnya mengembalikan nilai true atau false, kita dapat menggunakannya sebagai kondisi untuk perulangan while:

while (rs.next( )) {
}

Di antara kurung buka *while* kita memiliki *rs.next*. Ini akan benar selama Kursor belum melewati record terakhir dalam tabel. Jika ya, rs.next akan mengembalikan nilai false, dan *loop while* akan berakhir. Menggunakan rs.next seperti ini juga akan memindahkan Kursor sepanjang satu record pada satu waktu. Berikut kode yang sama seperti di atas, tetapi menggunakan loop sementara sebagai gantinya. Ubah kode Anda agar sesuai:

```
try {
    String host = "jdbc:derby://localhost:1527/Employees";
    String uName = "Your_Username_Here";
    String uPass= " Your_Password_Here ";

    Connection con = DriverManager.getConnection( host, uName, uPass );

    Statement stmt = con.createStatement();
    String SQL = "SELECT * FROM Workers";
    ResultSet rs = stmt.executeQuery( SQL );

    while ( rs.next() ) {
        int id_col = rs.getInt("ID");
        String first_name = rs.getString("First_Name");
        String last_name = rs.getString("Last_Name");
        String job = rs.getString("Job_Title");

        System.out.println(id_col + " " + first_name + " " + last_name + " " + job);
    }
}
catch ( SQLException err ) {
    System.out.println( err.getMessage( ) );
}
```

Ketika kita *Run* kode di atas, jendela Output akan menampilkan berikut ini:

```
run:
1 Helen James IT Manager
2 Eric Khan Programmer
3 Tommy Lee Systems Analyst
4 Priyanka Collins Programmer
BUILD SUCCESSFUL (total time: 0 seconds)
```

Sekarang setelah kita memiliki kode cara menyambungkan ke tabel database dan menampilkan record, kita akan melanjutkan dan menulis program yang lebih kompleks menggunakan form dan tombol untuk menggulir rekaman.

C. LATIHAN

1. Apa yang dimaksud 3 pernyataan dibawah ini :

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

2. Apa yang dimaksud kode program dibawah ini :

   Connection con = DriverManager.getConnection(host, username, password);

3. Apa yang dimaksud kode program dibawah ini :
4. ***String SQL = "SELECT * FROM Workers";***

D. REFERENSI

Danny Poo Derek Kiong Swarnalatha Ashok, Object-Oriented Second edition Programming and Java, Springer 2008

https://www.homeandlearn.co.uk/java/connecting_to_a_database_table.html diakses pada diakses pada tanggal 10 Juli 2022.

https://www.w3schools.com/sql/, diakses pada diakses pada tanggal 30 Juli 2022

Romi satrio Wahono, Java Gui, 2016