Session (02.01)
Interfacing MS Access Database using Java (1)

Dr. Ahmed M. ElShafee

Introduction

- Java uses something called JDBC (Java Database Connectivity) to connect to databases.
- There's a JDBC API, which is the programming part, and a JDBC Driver Manager, which your programs use to connect to the database.
- JDBC allows you to connect to a wide-range of databases (Oracle, MySQL, etc),
- We are going to build a simple MicroSoft Access database ".mdb" or ".accdb" .
- Then access it through java using jdbc driver

Connecting DB using java code

- Note that the main methods “throws SQLException”

```
public static void main(String[] args) throws SQLException {
```

- Create a string variable contains the database file path as follows

```
String database = "jdbc:odbc:Driver={Microsoft Access Driver
(*.accdb)};DBQ=D:\\Employees.accdb";
```

- Note that “DBQ=“ refers to database file path.

- Java provide two classes called “Connection” , and “DriverManager” that used in connecting databses as follows

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

```
Connection conn = DriverManager.getConnection(database, "", "");
```

- The blanks followed by database parameter is the username and password if required to connect to the database file.
- Now create an object from connection class that returns from connection object created in the last step

```
Statement s = conn.createStatement();
```

- Now you are connected to the database
- The next step ot to brows database files tables

Add SQL statements to your java code

- Now we agreed that to display all fields and all records on the table we can use the following statement

```
SELECT * FROM students
```

- To execute that statement, you will use the statement object created in the last section to execute the sql statement

```
String sqlstr= "SELECT * FROM students";
 s.execute(sqlstr);
```


ResultSets

- A **ResultSet** is a way to store and manipulate the records returned from a SQL query.
- **ResultSets** come in three different types.
- The type you use depends on what you want to do with the data:

```
ResultSet rs = s.getResultSet();
```

- The ResultSet also has methods you can use to identify a particular column (field) in a row.
- You can do so either by using the name of the column, or by using its index number.
- For our Workers table we set up four columns.
- They had the following names: **ID, First_Name, Last_Name, and Job_Title**.
- The index numbers are therefore 1, 2, 3, 4.

- We set up the ID column to hold Integer values. The method you use to get at integer values in a column is **getInt**:

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

- We could use the Index number instead:

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

- For the other three columns in our database table, we set them up to hold Strings.
- We, therefore, need the getString method:

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

- Or we could use the Index number:

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


- Because the ResultSet Cursor is pointing to just before the first record when the data is loaded, we need to use the **next** method to move to the first row.
- The following code will get the first record from the table:

```
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");
```

- You can add a print line to your code

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

- Now it's the time to fetch the results, which in this case the database table contents, fields X rows
- Create an object from "ResultSet" class, which is responsible of fetching data from database tables

```
ResultSet rs = s.getResultSet();
```

- Now get data from records as follows

```
while ((rs != null) && (rs.next())) {
            System.out.println(rs.getString(1) + " : " + rs.getString(2)+" : " +
rs.getString(3)+" : " + rs.getString(4)+" : " + rs.getString(5)+" : " +
rs.getString(6));
        }
```


- Now close the connection and statement to free database

```
s.close();  // Close the statement
        conn.close(); // Close the database. Its no more required
```

javadbexample01

- Check lab manual

Thanks,..

See you next week (ISA),…