http://www.science.smith.edu/dftwiki/index.php/Tutorial:_Accessing_a_MySql_database_in_Java_(Eclipse)

JDBC (Java Database Connectivity)

SELECT * from Customers;

Request from client > database (DB)

Response Database (DB) > client (SQL)

More versatile, open and compatible with countless different types of systems and architecture

JDBC (Java Database Connectivity) is a SQL-level API (methods) that allows us to execute SQL statements. It serves as a connectivity between the Java language and a wide range of databases including mySQL.

Connectors are simply libraries that contain methods and classes which help us to connect to DBs and run queries.

The following JDBC APIs provides the following classes and interfaces to interact wit h a SQL database:

- Driver class
    - Driver Manager
    - Connection
    - Statement
    - ResultSet
    - SQLException

1. Load the JDBC driver: `Class.forName("com.mysql.jdbc.Driver");`

2. Make a connection to the Database:

In order to make a connection to the database the syntax is

```
DriverManager.getConnection(URL, "userid", "password" )
```

- Userid is the username configured in the database
- Password of the configured user

URL of the database: Syntax: jdbc:< dbtype>://IPAddress:PortNumber/db_name"

Example: jdbc:mysql://localhost:3036/emp

Supported DB types: mySQL, Oracle, MSSQL

3. Create SQL statement: `Statement stmt=con.createStatement();`

4. Create a resultset to execute query and store the response sent by the SQL server:

```
ResultSet rs=stmt.executeQuery("select * from emp");
```

5. Process the results until the last record is available:

```
while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
```

6. Closing the connection to avoid memory leaks;

```
con.close();
```

7. Handle exceptions

Java JDBC provides many **methods** to process the results:

Method name	Description
String getString()	Method is used to fetch the string type data from the result set
int getInt()	Method is used to fetch the integer type data from the result set
double getDouble()	Method is used to fetch the double type data from the result set
Date getDate()	Method is used to fetch the Date type object from the result set
boolean next()	Method is used to move to the next record in the result set
boolean previous()	Method is used to move to the previous record in the result set
boolean first()	Method is used to move to the first record in the result set
boolean last()	Method is used to move to the last record in the result set
boolean absolute(int rowNumber)	Method is used to move to the specific record in the result set

Note:

Install MySQL Server and MySQL Workbench and please note the Username, Password and Port Number of the database

Download the MySQL JDBC connector (Mysql-connector-java-5.1.18-bin.jar) from here.

Or this link:

https://code.google.com/archive/p/find-ur-pal/downloads

Add the downloaded Jar to your Project

Right click on your Java File. Then click on Build Pathà Configure build path

Select the libraries

Click on add external JARs

You can see MySQL connector java in your library

Click on open to add it to the project

Write code for JDBC connectivity:

```
package kutub;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class JDBCKutub {

	public static void main(String[] args) throws Throwable {
		/*
		 * DB location: localhost:port number or any IP address:port number for a DB.
		 * DB name, user id, password Driver - to initialize
		 * MySQL workbench: local host or 127.0.0.1 and the port number:3306
		 * Statement Query - your request to DB Result set - DB response to your request
		 */

		String dburl = "jdbc:mysql://localhost:3306/world";
		// or using the IP address:	String dburl = "jdbc:mysql://127.0.0.1:3306/world";

		String userid = "root";
		String password = Password.password;
		String myQuery = "select * from city limit 100";

		Class.forName("com.mysql.jdbc.Driver");	// loads the class driver

		Connection con = DriverManager.getConnection(dburl, userid, password);

		// The object used for executing a static SQL statement and returning the
		// results it produces:
		Statement stmt = con.createStatement();
```

```
            // A table of data representing a database result set,
            // which is usually generated by executing a statement that queries the
            // database.

            ResultSet myrs = stmt.executeQuery(myQuery);

        while(myrs.next()) {
        System.out.println("CityID" +" " + (myrs.getInt(1) +"    CityName: " +" " + myrs.getString(2)));
            }
        }
}
```

JDBC Code from Dilruba apa's computer:

```
package jdbctest;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class MyJDBC {

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

		/*
```

*** we need 8 different credentials:**

connector type: **jdbc**

db type: **mysql**

location of db: localhost / 127.0.0.1 (or any IP address given by DBA)

name of the db: sakila

user name: root

password: 4444

port # 3306 (default for mySQL)

query: SELECT * FROM Customers;

*/

```
String dburl = "jdbc:mysql://localhost:3306/sakila"; // localhost =

                                        // 12.7.0.0.1

String username = "root";

String password = "4444";


String dbQuery = "SELECT * FROM customer limit 5";


// loading the JDBC driver class

Class.forName("com.mysql.cj.jdbc.Driver");


// creating connection:

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

```
                // creating sql statement to send the request/query to the DB:

                Statement stmt = con.createStatement();



                // creating a resultset to hold the values after running a query



                ResultSet rs = stmt.executeQuery(dbQuery);



                // n number rows: there is a next resultset:



                while (rs.next()) {

                        System.out.println("Customer ID: " + rs.getInt(1) + "
Customer First Name: " + rs.getString(3) + "  Email "

                                        + rs.getNString(5));

                }



        }
```

JDBC Interview Questions

1. What is JDBC Driver?

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)

3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

JDBC = Java Database (DB) Connectivity

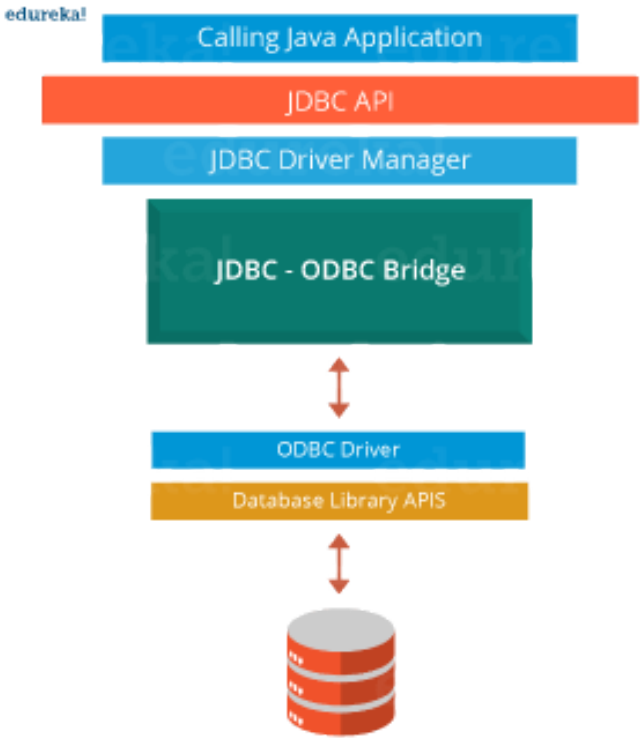
DB (server) ←---------- JDBC ----------> Java app (client that sends requests)

JDBC - a library of Java capabilities have been packed (zipped) as a JAR (Java ARchive) file , API (classes, interfaces and methods)

API - Application programming interface

We want to access the following DB and want to retrieve data. What are the things that we need to do so?

There must be a database, DB type, location (IP address), a communication channel = port number, and a user with username, password



2. What are the steps to connect to a database in java?

- Registering the driver class
- Creating connection
- Creating statement
- Executing queries
- Closing connection

3. What are the JDBC API components?

The java.sql package contains interfaces and classes for JDBC API.

Interfaces:

- Connection
- Statement
- PreparedStatement
- ResultSet
- ResultSetMetaData
- DatabaseMetaData
- CallableStatement etc.

Classes:

- DriverManager

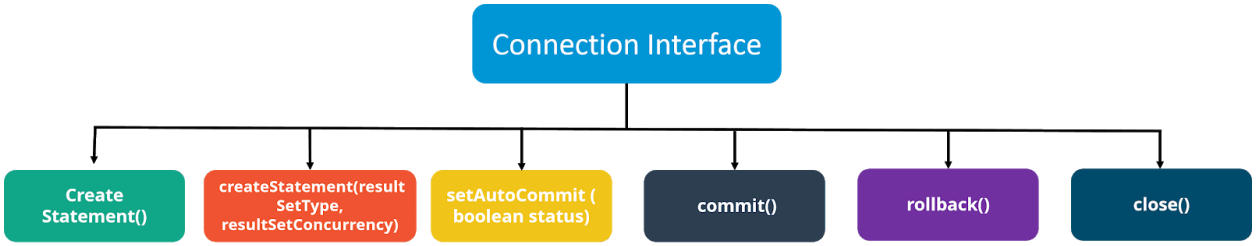
- Blob
- Clob
- Types
- SQLException etc.

4. What is the role of JDBC DriverManager class?

The DriverManager *class* manages the registered drivers. It can be used to register and unregister drivers. It provides factory method that returns the instance of Connection.

5. What is JDBC Connection interface?

The Connection interface maintains a session with the database. It can be used for transaction management. It provides factory methods that returns the instance of Statement, PreparedStatement, CallableStatement and DatabaseMetaData.



6. What is the purpose of JDBC ResultSet interface?

The ResultSet object represents a row of a table. It can be used to change the cursor pointer and get the information from the database.

7. What is JDBC ResultSetMetaData interface?

The ResultSetMetaData interface returns the information of table such as total number of columns, column name, column type etc.

8. What is JDBC DatabaseMetaData interface?

The DatabaseMetaData interface returns the information of the database such as username, driver name, driver version, number of tables, number of views etc.

9. What do you mean by batch processing in JDBC?

Batch processing helps you to group related SQL statements into a batch and execute them instead of executing a single query. By using batch processing technique in JDBC, you can execute multiple queries which makes the performance faster.

10. What is the difference between execute, executeQuery, executeUpdate?

Statement ***execute(String query)*** is used to execute any SQL query and it returns TRUE if the result is an ResultSet such as running Select queries. The output is FALSE when there is no ResultSet object such as running Insert or Update queries. We can use *getResultSet()* to get the ResultSet and *getUpdateCount()* method to retrieve the update count.

Statement ***executeQuery(String query)*** is used to execute Select queries and returns the ResultSet. ResultSet returned is never null even if there are no records matching the query. When executing select queries we should use executeQuery method so that if someone tries to execute insert/update statement it will throw java.sql.SQLException with message "executeQuery method can not be used for update".

Statement **executeUpdate(String query**) is used to execute Insert/Update/Delete (DML) statements or DDL statements that returns nothing. The output is int and equals to the row count for SQL Data Manipulation Language (DML) statements. For DDL statements, the output is 0.

You should use execute() method only when you are not sure about the type of statement else use executeQuery or executeUpdate method.

Q11. What do you understand by JDBC Statements?

JDBC statements are basically the statements which are used to send SQL commands to the database and retrieve data back from the database. Various methods like execute(), executeUpdate(), executeQuery, etc. are provided by JDBC to interact with the database.

JDBC supports 3 types of statements:

1. *Statement:* Used for general purpose access to the database and executes a static SQL query at runtime.
2. *PreparedStatement:* Used to provide input parameters to the query during execution.
3. *CallableStatement:* Used to access the database stored procedures and helps in accepting runtime parameters.