Java DataBase Connectivity

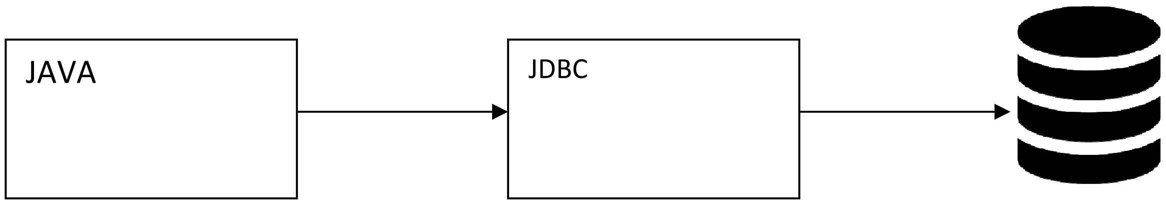
Java DataBase Connnectivity is an API, as the name implies, IT HELPS TO ACHIEVE THE CONNECTIVITY BETWEEN Java Programs & Database

Note: Servlets & JSPs are also java Programs

If we have a Web application & if it has aDB, then it needs to interact with DB to read/ modify the data

JDBC helps to do this & in the world of Java, JDBC is the "one and only" API that helps to interact with RDBMS (DB) Application.

Also JDBC is "DB independent" i.e. Using JDBC we can interact with any RDBMS Applications in the world.



JDBC Pre-Requirment:

i- Install Any RDBMS Appllication (MySQL)
ii- Create a "Database (Schema) " by any name.
iii- Create a table
iv- Insert some data into the table.

SQL Queries for MySQL RDBMS Application:-

1- create databse BECME89_DB;
2- use BECME89_DB;
3- create table students_info
   (
   regno Int(10) not null,
   firstname varchar(50),
   middlename varchar(50),
   lastname varchar(50),
   Primary key(regno)
   );
4- insert into table_name values(regno,’firstname’,’middlename’,’lastname’);

Sone UseFull Queries:

1- To Connect to database:
   use BECME89_DB;

2- To get the list of databases:
   show databases;
3- To get the list of tables:
   show tables;
4- To know the table structure:
   describe table_name;

Java ThumbRules :

A class which is declared with abstract keyword is called as abstract class.

A Class declare without abstract keyword is called as concrete class.

If LHS == RHS (A reg = new A();) then LHS Is always concrete class. If LHS != RHS , (B ref = new A();) then LHS can be an interface, concrete Class, or Abstract Class.

Simply having an abstract class is of no use and there must be at least one subclass.

Simply having an interface in java of no use and there must be at least one implementation class.

In java anything apart from primitive data types are called as object references. For example: Int I ,boolean isTrue("i " and "isTrue" is a primitive variable) , Object obj -> it's an Object ref varialble.

Anything is java which starts with lower case is eithere variable, if its with brackets its a method. And Anything which starts with a Caps is either a Class or an Interface.

In Java, "Super Class" can be either an "Abstract Class" or "Concrete Class".

Neccassary Steps to Work with JDBC

1- Loads the **Driver**
2- Get the **DB Connection** via **Driver**
3- Issue **SQL Queries** via **Connection**
4- **Process the results** returned by **SQL Queries**
5- Close All **JDBC Objects**

Note:-

>"java.sql.*" is the Package Representation of JDBC

> i.e Any Class / Interface belongs to this package means it's part of JDBC

Drivers

Drivers are additional software components required by JDBC to interact with RDBMS applications. Drivers are provided by DB vendors and they are DB depentdent, that means, using mySql driver we can only interact with my Sql RDBMS application and using Db2 drivers we can only interact with Db2 RDBMS application.

JAR (Java Archieve) File:

- It's a collection of ".class" files + others Necessary Resources (Text File ,XML, Property Files, Etc.)

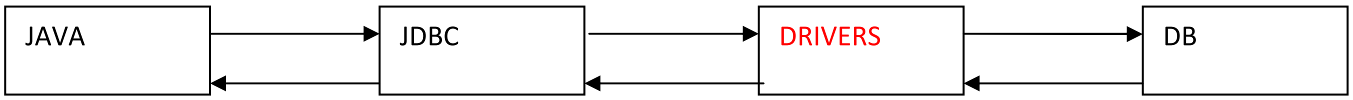
- JAR File helps us to transfer the “Java Files/ .class filse/ Java Application” From one location to an another location
- JAR File will have “.jar” file extension & Functionality wise it’s similar to “ZIP” file

Steps To Create JAR File:

1. Right Click on the Java Project, which we want to transfer , select “Export…”
2. Select “JAR file” option present under “Java” & click on “Next”
3. Provide the “Destination & File Name” , click on “Finish”

Steps To make use of JAR File:

1- Right Click on the Java Project, where we want to make use of JAR File, select “Build Path” click on “Add External Archieves…”
2- Select the “JAR File” & Click on “Open”
3- We see JAR File under “Referenced Libraries”



Driver Class

- “Driver Class” is a Concrete Class, present in driver JAR file, is the one that implements the “java.sql.Driver” interface

- This interface is present in JDBC API & every JDBC driver provider has to implement this Interface.
- “Driver” helps us to establish DB Connection, transfers the DB query and results between Java program and RDBMS Application.

Steps to Load the “Driver Class” ito the Program

1. By invoking “registerDriver()” method preseint in “java.sql.DriverManager” Class by passing an instance of “Driver Class”
   Syntax:
   public vod DriverManage.registerDriver(java.sql.Driver driverRef) throws SQLException

for MySQL Driver:

com.mysql.jdbc.Driver ref = new com.mysql.jdbc.Driver();

DdriverManager.registerDriver(ref);

Second approach

Class.forName(“com.mysql.jdbc.Driver”).newInstance();

- This is the most common approach to resister a Driver Class which helps us to pass “Driver Class Name at Runtime”
- But if we create an instance of a driver class using new operator, the driver class name can’t be passed at Runtime

Driver Types

There are 4 types of Drivers

1. Type 1: JDBC-ODBC Bridge
2. Type 2: Native-API Driver
3. Type 3: Network-Protocol Driver
4. Type 4: Native-Protocol Driver

Note: **In JDBC 4, the driver loads automatically, if the jar file is present in the project’s Classpath.**