JDBC: Java Database connectivity

Java mostly used to develop the Business application.

--every business organizations provide their business services to the
customer, and to computerize those business services offered by a
business organization , whatever computer application we develop, is
known
as business application.

Business orgnizations are also of 2 types :

1.small scale business orgnization
2.Large scale business organization (Enterprise)

Main objective/goal of any business organization is : profit.

common and general things required in a business application :

1. maintaining business data permanently in a secure and easily retriaval
manner.

2.processing the business data according to the business rule.

3. presenting the data to end-user in user understandable format.

Note: In realtime, the business data will be stored in RDBMS s/w

ex:

Mysql
Oracle
Postgres
DB2
Sql-Server

Why Java DB communication is required:
==============================

--Our Java application is very efficient in Processing the data and
presenting the data,
but it is very poor in storing the data in a secure and easily retrieval
manner.

--with the java using Serialization and desirialization process we can
store/save business
data(holded in the objects).

--RDBMS s/w are very much excellent in storing/maintaining the data, but
they are very poor in presenting the data and processing the data.

--to develop the robust and powerfull business application,we need to
integrate the power
of both hetregenious technology.

--Java application can perform any task through the method calls,
objects, these
method calls and objects are not understandable by RDBMS s/w directly.

--similarly, DB s/w performs any operations by using SQL, but these sql
commands is not understandable by the JVM.

---inspite of heteregenious env, we can communicate Java app with the DB
s/w using JDBC.

Note: JDBC enables any kind of Java application to communicate with any
kind of DB s/w in a standard manner.

Russian   translator
Japanese
Java<------------- Jdbc driver s/w ----------------------------->DB
s/w

JDBC:

--it is an open specification from sun-microsystem(Oracle), (a
documentation
which contains rules and guidelines to develop a perticular s/w (JDBC
driver s/w))

--DB vendor or any 3rd party vendor will implements this specification
and develop the
jdbc driver s/w.

mysql : oracle
db2 : IBM
sql_server : microsoft

Ratandb: Ratan   : JDBC driver according to the JDBC specification -----
-------> Java application.

--for each DB s/w, we have diff Jdbc driver s/w. but all driver s/w
devloped based on jdbc specification only.

--these driver s/w comes in the form of jar file.

mysql : mysql-connector.jar
oracle : ojdbc6.jar
postgres: postgres-version.jar

--this jdbc specification provides a jdbc api, to the
programmer/developer to perform the DB operation from the Java
application.

--this jdbc api comes in the form of 2 packages:

1.java.sql
2.javax.sql

--this jdbc api comes along with JDK installation. along with Java core
library.

--to make our java application to communicate with DB s/w we need to
downlaod the jdbc driver s/w jar file and we need to set that jar file to
the classpath of our application.

--in the context of Java DB communication, our Java application will act
as a client
because it needs the services of the DB s/w. so our Java application is
also called
as JDBC client.

responsibility of the JDBC client (Java application):
=======================================

1. requesting the connection from the DB.

2.submitting the appropriate sql statement to the DB server in the form
of String
object.

3. processing the result comming from the DB s/w.

4. dealing with exceptions if any. (most of jdbc exceptions are the
checked exception)

5.managing the transaction whenever it is required.

6.closing the connection once done with DB operation.

Jdbc driver :-

--it is a translation s/w written in Java according to the JDBC
specification.

responsibility of JDBC driver s/w:
----

1. establish the connection bt java application and DB s/w.

2.recieving the jdbc method calls from the java application, and
translate them
into the DB understandable format and forward them to the DB s/w.

3.translate the DB s/w given result to the Java format (some objects) and
returns
to the JDBC client (to the Java application)

Step to connect the Java application with the DB s/w:
=======================================

step1: download the jdbc driver related jar file and set that jar file in
the classpath
of our project/application.

step 2: load the jdbc driver (loading the jdbc driver related main class into the memory.)

step 3: prepare the connection URL.'

step 4: establish the connection

step 5: after performing the CRUD (insert, select, update, delete)operation close the connection.

Note: with the Java application even though we can perform DDL operation , but we should always perform DML, DRL ,TCL type of operations.

Javaapp:

A.java
B.Java
C.java
Demo.java

---after compiling we need to compess all the .class files inside a jar file.

app1.jar

Myapp1: here we need to set the app1.jar the classpath of Myapp1 application

class X {

A a1= new A();

}

adding the jar file to the project class path:
----

rightclick on the project ---->build path ----> configure build path -----> select libraries tab----> add external archieve ----->
select the downloaded jar file ---->apply and close.

Connection URL:

String url= "jdbc:mysql://localhost:3306/ratandb";

example app:

Demo.java:

```java
package com.masai;

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

public class Demo {

    public static void main(String[] args) {

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }


        String url= "jdbc:mysql://localhost:3306/ratandb";

        try {
            Connection conn= DriverManager.getConnection(url,
"root", "root");

            if(conn  != null)
                System.out.println("connected...");


        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

}
```

JDBC
JDBC specification and implementation
JDBC driver s/w

when Class.forName(-) method is executed following 3 things happens in
the backgroud.

1. Driver related main class will be loaded into the memory

2.Driver related main class object will be created.

3.that object will be registerd with the DriverManager class.


```
        try {
            Driver d= new Driver();

            DriverManager.registerDriver(d);


        } catch (SQLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
```


--the above code is written inside the static block of each Driver
related main class.

static block and non-static block:
-=========================
package com.masai;

public class A {

    public A() {
        System.out.println("inside the constrcutor of A");
    }

    void funA() {

        System.out.println("inside funA of A");
    }

    {
        System.out.println("inside non-static block");
    }

    static{
        System.out.println("inside static block");
    }

    public static void main(String[] args) {

        System.out.println("inside main method of A");
```

```
            A a1=new A();

            a1.funA();

        }

}


output:

inside static block
inside main method of A
inside non-static block
inside the constrcutor of A
inside funA of A
```

Note: before starting the execution from the main jvm will call all the static blocks one by one if they are defined.

--inside a static block we can write any kind of executable statement. for example we can create object of a class, we can call any method,etc..

--non-static block will be executed before executing the constructor of our class. so if we have some overloaded constructors are there and some statements are required in all the constrcutors then it is recomended to keep those statements inside the non-static block.

****Note: whenever we load any class into the memory then that class context will be created and its all static members will be loaded into the context area and then its all static blocks will be executed.

--to load any class into the memory we can make use of forName(-) method of Class class.

--this Class class belongs to java.lang package.

example

A.java:

```java
package com.masai;

public class A {

        void funA() {
                System.out.println("inside funA of A");
        }

        static {

                System.out.println("inside static block");
```

```
                A a1=new A();


                a1.funA();

        }
}



Demo.java:
--------------

package com.masai;

public class Demo {

     public static void main(String[] args) {


          try {
                Class.forName("com.masai.A");
          } catch (ClassNotFoundException e) {
                e.printStackTrace();
          }
     }

}
```

Note: from Java 1.6 loading the Driver related main class is optional ,
becoz once we add the driver related jar file inside the classpath of our
application, then automatically Driver related main class will be loaded
into the memory.

But it is recomended to keep the Class.forName() method inside our
application.

inserting a student record inside the db using Java application:
==

--once we get the Connection obj, then we can execute any type of sql
statements from our java application to the DB server.

for that we require a "java.sql.Statement(I)" object.

we get the Statement obj by using :

Statement st= conn.createStatement();

--with the Statement obj we can pass/supply any type of SQL statement to
the DB server from our java application for that we need to use following
methods:

1. public int executeUpdate(String dml)throws SQLException /// insert, update, delete
--this method return the number of row affected due to the DML statement.

2.public ResultSet executeQuery(String drl) throws SQLException //select statement
--it will return the result from the DB in the form of ResultSet object.

3.public boolean execute(String anysql)throws SQLException // any type of sql(including DDL also)

Demo.java:
=========

```java
package com.masai;

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

import com.mysql.cj.jdbc.Driver;

public class Demo {

     public static void main(String[] args) {

          try {
               Class.forName("com.mysql.cj.jdbc.Driver");
          } catch (ClassNotFoundException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
          }


          String url= "jdbc:mysql://localhost:3306/ratandb";

          try {
               Connection conn= DriverManager.getConnection(url,
"root", "root");

               Statement st= conn.createStatement();

               int x= st.executeUpdate("insert into student
values(10,'Ram','delhi',780)");


               if(x >0) {
                    System.out.println("record inserted sucessfully");
               }
               else
                    System.out.println("not inserted...");


          } catch (SQLException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
```

```
            }

        }

}


closing the connection solution 1:
============================

package com.masai;

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

import com.mysql.cj.jdbc.Driver;

public class Demo {

      public static void main(String[] args) {

            try {
                  Class.forName("com.mysql.cj.jdbc.Driver");
            } catch (ClassNotFoundException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

            String url= "jdbc:mysql://localhost:3306/ratandb";

            Connection conn=null;

            try {
                  conn = DriverManager.getConnection(url, "root", "root");

                  Statement st= conn.createStatement();

                  int x= st.executeUpdate("insert into student
values(20,'Ramesh','delhi',680)");


                  if(x >0) {
                        System.out.println("record inserted sucessfully");
                  }
                  else
                        System.out.println("not inserted...");



            } catch (SQLException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
            finally {
```

```java
			try {
					conn.close();
			} catch (SQLException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
			}


		}
	}

}
```

closing the connection solution 2: using try with resource:
==========================

```java
package com.masai;

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

import com.mysql.cj.jdbc.Driver;

public class Demo {

	public static void main(String[] args) {

			try {
					Class.forName("com.mysql.cj.jdbc.Driver");
			} catch (ClassNotFoundException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
			}

			String url= "jdbc:mysql://localhost:3306/ratandb";


			try(Connection conn = DriverManager.getConnection(url, "root",
"root")) {


					Statement st= conn.createStatement();

					int x= st.executeUpdate("insert into student
values(20,'Ramesh','delhi',680)");


					if(x >0) {
							System.out.println("record inserted sucessfully");
					}
					else
							System.out.println("not inserted...");
```

```
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }

}


inserting the student record by taking input from the user:
--------------------------------------------------------------------

package com.masai;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

import com.mysql.cj.jdbc.Driver;

public class Demo {

    public static void main(String[] args) {


            Scanner sc = new Scanner(System.in);

            System.out.println("Enter roll");
            int roll= sc.nextInt();

            System.out.println("Enter Name");
            String name= sc.next();

            System.out.println("Enter Address");
            String adr= sc.next();

            System.out.println("Enter Marks");
            int marks= sc.nextInt();




            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }


            String url= "jdbc:mysql://localhost:3306/ratandb";
```

```
            try(Connection conn = DriverManager.getConnection(url, "root",
"root")) {


                Statement st= conn.createStatement();

                int x= st.executeUpdate("insert into student
values("+roll+",'"+name+"','"+adr+"',"+marks+")");


                if(x >0) {
                    System.out.println("record inserted sucessfully");
                }
                else
                    System.out.println("not inserted...");



            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }



    }

}
```

Note: if we use the Statment(I) obj to perform any DB operation then this
Statement object has the following limitations:

1.complexities to concatenate dynamic variables.

2. whenever we pass any query to the DB using Statement obj, DB engine
will perform following task every time to execute that query.

a.query compilation
b.query plan generation
c.query optimization.

--for the same query doing this operation every time , it will degrede
the performance.

--to tell the DB engine to perform above 3 task only at first time with
the value or without the value, and put that query in the cache, next
time onwards just add the dynamic values and execute the query
--for that we need to use "java.sql.PreparedStatement(I)" obj instead of
Statement obj.

Statement(I)
        |
PreparedStatement(I)

Note: Statement is relativly faster than PreparedStatment.

--to get the PreparedStatment obj we need to call

prepareStatement(String sqlquery) method on the Connection obj, with the
value or without the value.

--in case of without value we need to use placeholders (?)

ex:

PreparedStatement ps= conn.prepareStatement("insert into student
values(?,?,?,?)");

--In case of placeholders, before executing the query we need to bind the
appropriate values to the corresponding placeholders by calling variaous
types of setXXX() methods on the PreparedStatement object.

ex:

ps.setInt(1,roll);
ps.setString(2,name);
ps.setString(3,adr);
ps.setInt(4,marks);

--after binding the appropriate placeholders we need to execute the query
using one of the following methods of PreparedStatement obj.

public int executeUpdate()throws SQLException;

public ResultSet executeQuery()throws SQLException;

public boolean execute()throws SQLException;

Demo.java:
----

package com.masai;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

import com.mysql.cj.jdbc.Driver;

public class Demo {

        public static void main(String[] args) {

                Scanner sc = new Scanner(System.in);

```java
System.out.println("Enter roll");
int roll= sc.nextInt();

System.out.println("Enter Name");
String name= sc.next();

System.out.println("Enter Address");
String adr= sc.next();

System.out.println("Enter Marks");
int marks= sc.nextInt();




try {
    Class.forName("com.mysql.cj.jdbc.Driver");
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}


String url= "jdbc:mysql://localhost:3306/ratandb";


try(Connection conn = DriverManager.getConnection(url, "root",
"root")) {

    PreparedStatement ps= conn.prepareStatement("insert into
student values(?,?,?,?)");

    ps.setInt(1, roll);
    ps.setString(2, name);
    ps.setString(3, adr);
    ps.setInt(4, marks);


    int x= ps.executeUpdate();


    if(x >0) {
        System.out.println("record inserted sucessfully");
    }
    else
        System.out.println("not inserted...");


} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}



    }

}
```

updating a record: giving grace marks to those students whose marks is
greater than 700.
===============


```java
package com.masai;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

import com.mysql.cj.jdbc.Driver;

public class Demo {

    public static void main(String[] args) {


        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the Grace marks");
        int gMarks= sc.nextInt();




        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }


        String url= "jdbc:mysql://localhost:3306/ratandb";


        try(Connection conn = DriverManager.getConnection(url, "root",
"root")) {

    PreparedStatement ps= conn.prepareStatement("update student set
marks=marks+? where marks > 700");

    ps.setInt(1, gMarks);


    int x= ps.executeUpdate();


            if(x >0) {
                System.out.println(x+" record updated
sucessfully");
            }
            else
                System.out.println("not updated...");
```

```
            } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }


        }

}


sql :

DDL (Data Defination language)  (create, alter, drop, rename) // table
schema

DML (Data manipulation language) (insert, update, delete) // on table
data   // executeUpdate()

DRL (Data retrival language) (select)    // executeQuery()

TCL(Transaction control language) (commit, rollback, savepoint)


example Deleting a record:
========================


package com.masai;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

import com.mysql.cj.jdbc.Driver;

public class Demo {

        public static void main(String[] args) {


                Scanner sc = new Scanner(System.in);

                System.out.println("Enter the roll to delete the student");
                int roll= sc.nextInt();




                try {
                        Class.forName("com.mysql.cj.jdbc.Driver");
                } catch (ClassNotFoundException e) {
                        // TODO Auto-generated catch block
```

```java
				e.printStackTrace();
			}


			String url= "jdbc:mysql://localhost:3306/ratandb";


			try(Connection conn = DriverManager.getConnection(url, "root",
"root")) {

		PreparedStatement ps= conn.prepareStatement("delete from student
where roll =?");

		ps.setInt(1, roll);


		int x= ps.executeUpdate();


				if(x >0) {
					System.out.println(x+" record deleted
sucessfully");
				}
				else
					System.out.println("Student does not exist...");



			} catch (SQLException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}



	}

}
```

```
Performing select DRL operation :
=========================

      PreparedStatement ps= conn.prepareStatement("select * from
student");

                ResultSet rs= ps.executeQuery();


--here we get the records from the DB, in the form of
java.sql.ResultSet(I) object.

--in order to get the details or data from the ResultSet object we need
to know the structure of the ResultSet object.

Note: ResultSet structure will depends upon the query whatever we are
executing. it does not depend upon on the table structure.

now to move the cursur from the BFR to the record area we need to use
following method of ResultSet obj.

public boolean next();

--this method will move the cursor from the BFR to the record area and if
it points to any record then it returns true other if no record is there
, i.e point to ALR then it returns false.

--to access/get the column fields value from the cursor pointed record we
need to use following method of the ResultSet obj.

public XXX getXXX(String columnName);  // here XXX will be replaced with
the appropriate data type :

ex:

public int getInt("roll");

public String getString("name");

public String getString("address");

public int getInt("marks");



Note: to move the cursor from the BFR to the record area we need to
follow following stategy:

1. if we sure that only 0 or 1 record will come (atmost 1) in that case
we need to use if-else condition.
(when our search condition is on PK)

2.if there may be a chance of getting more than one row then we need to
use "while" loop.


example:

Demo.java:
----------------
```

```java
package com.masai;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

import com.mysql.cj.jdbc.Driver;

public class Demo {

    public static void main(String[] args) {

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }


        String url= "jdbc:mysql://localhost:3306/ratandb";


        try(Connection conn = DriverManager.getConnection(url, "root",
"root")) {


            PreparedStatement ps= conn.prepareStatement("select *
from student");

            ResultSet rs= ps.executeQuery();

            while(rs.next()) {

                int r= rs.getInt("roll");
                String n= rs.getString("name");
                String a= rs.getString("address");
                int m= rs.getInt("marks");


                System.out.println("Roll is "+r);
                System.out.println("Name is "+n);
                System.out.println("Address is "+a);
                System.out.println("Marks is "+m);

    System.out.println("=============================");

            }


        } catch (SQLException e) {
            // TODO Auto-generated catch block
```

```
                e.printStackTrace();
            }



    }

}


Develop a searching application using JDBC:
==================================



package com.masai;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

import com.mysql.cj.jdbc.Driver;

public class Demo {

    public static void main(String[] args) {


        Scanner sc= new Scanner(System.in);

        System.out.println("Enter Roll to Search the Student");
        int roll= sc.nextInt();


        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }


        String url= "jdbc:mysql://localhost:3306/ratandb";


        try(Connection conn = DriverManager.getConnection(url, "root",
"root")) {


            PreparedStatement ps= conn.prepareStatement("select *
from student where roll =?");

            ps.setInt(1, roll);

            ResultSet rs= ps.executeQuery();
```

```java
                    if(rs.next()) {

                            int r= rs.getInt("roll");
                            String n= rs.getString("name");
                            String a= rs.getString("address");
                            int m= rs.getInt("marks");


                            System.out.println("Roll is "+r);
                            System.out.println("Name is "+n);
                            System.out.println("Address is "+a);
                            System.out.println("Marks is "+m);




                    }else
                            System.out.println("Student does not exist with
Roll :"+roll);





            } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }



    }

}


example 2:

package com.masai;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

import com.mysql.cj.jdbc.Driver;

public class Demo {

    public static void main(String[] args) {


            Scanner sc= new Scanner(System.in);
```

```java
            System.out.println("Enter Name to Search the Student");
            String name= sc.next();


            try {
                 Class.forName("com.mysql.cj.jdbc.Driver");
            } catch (ClassNotFoundException e) {
                 // TODO Auto-generated catch block
                 e.printStackTrace();
            }


            String url= "jdbc:mysql://localhost:3306/ratandb";


            try(Connection conn = DriverManager.getConnection(url, "root",
    "root")) {


                 PreparedStatement ps= conn.prepareStatement("select *
    from student where name =?");

                 ps.setString(1, name);

                 ResultSet rs= ps.executeQuery();

                 boolean flag=true;

                 while(rs.next()) {

                      flag=false;
                      int r= rs.getInt("roll");
                      String n= rs.getString("name");
                      String a= rs.getString("address");
                      int m= rs.getInt("marks");


                      System.out.println("Roll is "+r);
                      System.out.println("Name is "+n);
                      System.out.println("Address is "+a);
                      System.out.println("Marks is "+m);

                 }


                 if(flag)
                      System.out.println("record does not exist.");




            } catch (SQLException e) {
                 // TODO Auto-generated catch block
                 e.printStackTrace();
            }
```

```
        }

}
```

--all the above examples are not the reusable:

--to make our Data access logic reusable and efficient we need to use
a pattern called DAO (Data access object) pattern.