JAVA Notes

__/__/____

* JDBC :-

- JDBC Stands for Java Datebase Connectivity.
- JDBC is a Java API to Connect and execute the query with the database.
- It is a Part of JavaSE (Java Standard Edition).

Def :- JDBC is an API for Java Programming language that defines how a client may access a data base.

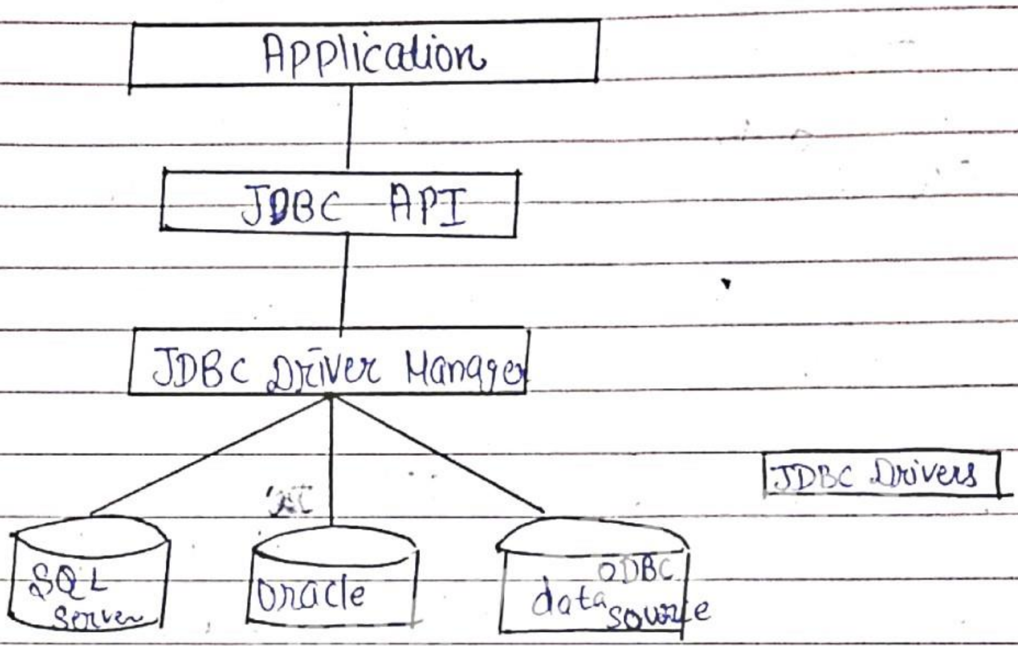
Tasks

- Making a connection to a data base.
- Creating a SQL or MySQL statements.
- Executing SQL Queries on database.
- Viewing & Modifying the results records.

API :- API Stands for "Application Programming Interface" is a document that contains a description of all the features of a software or Product.

__/__/__

* Architecture of JDBC :-



1. Application :- It is a Java applet or a servlet which communicates with a data source.

2. JDBC API :- The JDBC API all allows Java Programs to execute SQL Statements and reterive results.
It includes :-
Driver Manager.
Driver
Connection.
Statement
SQL data

_/_/_

3. Driver Manager:-

- It Plays an important role in the JDBC Architecture.
- It uses some data base specific drivers to effectively connect enterprise applications to data bases.

4 JDBC Drivers:-

- JDBC driver that intelligently communicates with the respective data source.

Types of JDBC Architecture

↓ Two-tier Model (A Java Application communicates directly to the data Source)

↓ Three-tier Model (In this, the user's queries are sent to middle tier services, from which the commands are again sent to data sourc

__/__/__

* JDBC Driver :-

- JDBC Driver is a software component that enables Java application to interact with the data base.

There are 4 types of JDBC Drivers :-

1. Type-1 :- JDBC-ODBC bridge driver.
2. Type-2 :- Native-API driver
3. Type-3 :- Network Protocol driver.
4. Type-4 :- Thin driver.

Type 1

1. JDBC-ODBC bridge driver

- The JDBC-ODBC bridge driver uses ODBC driver to connect to the data base.

- The JDBC-ODBC bridge converts JDBC method calls into the ODBC function calls.

- It is also called universal driver because it can be used to connect to any of the data base.

__/__/__

② $\boxed{\text{Type 2}}$

Native - API driver :-

- The Native API driver uses the client-side libraries of the database.

- This driver converts JDBC method calls into native calls of the data base API.

③ $\boxed{\text{Type 3}}$

Network Protocol driver :-

- The Network Protocol driver uses middleware that converts JDBC calls directly or indirectly into vendor specific database protocol.

- No client side library is required.

- Type -3 drivers are fully written in Java, hence they are Portable drivers.

④ $\boxed{\text{Type 4}}$ Thin driver :- It does not require any native database library, that is why it also known as Thin driver.

- It is fully written in Java language.

__/__/____

* Java·SQL package :-

- This Package include classes and interface to perform almost all JDBC operation such as Creating and executing SQL Queries.
- The Java-sql package is known as JDBC core API, to perform JDBC core operations.
- The classes in a java-sql package can be specified into the following categories based on different operations :-
  - Connection Management
  - Database Access
  - Data Types
  - Database metadata
  - Exception and warning.

* Some Important classes and interface of java.sql package

java·sql·connection :- It creates a connection with specified database.

java·Sql·Date :- It provide support for Date SQL type.

_/_/_

java.sql.Driver :- It create an instance of a driver with the Driver Manager.

java.sql.DriverManager:-This class manages database drivers.

java.sql.Statement :- This statement is used to execute SQL statements.

java.sql.SQLException :- It Encapsulate all JDBC related exception.

java.sql.SQLPermission :- The class manages the various SQL related which are provided to the accessing objects.

Types :- The class defines various SQL constants

The Java.sql package contains API for the following :-

1. Making a connection.
2. Sending SQL Parameter to database.
3. Updating and retervimg the results of query
4. Providing standard mapping for SQL type.
5. Meta data
6. Exceptions
7. Custom mapping an SQL user.

__/__/____

* SQL Data Types and their Corresponding JAVA DATA TYPE :-

SQL data type	Java data type
INT	int
DATE	java.sql.Data
TIME	java.sql.Time
VARCHAR	String
FLOAT	double
BOOLEAN	boolean
CHAR(N)	String
ARRAY	java.sql.Array

* Steps to Connect to database in Java

1. Register Driver class :-

Class.forName() is used to dynamically load the driver class.

Syntax: class.forName("oracle.jdbc.driver.OracleDriver");

2. Create Connection object :-

Connection con = DriverManager.getConnection ("—Path", "username", "password");

EMP < root

__/__/__

3. Create Statement object :-

- It is used to execute Queries with databas

Example :-

```
Statement stmt = con.createstatement();
```

4. Execute the Query :-

→ Execute the Query on the database and return the object of ResultSet type.

```
ResultSet rs = stmt.executeQuery ("select *
                        from Emp");
                              ↑
                          table name

while(rs.next())
 rs.getInt()
```

5. Close the Connection Object :-

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

__/__/____

* Performing basic database operations :-

1. Connecting to the database :-

```
import java.sql.*;
public class connect
 {
  public static void main(String args[])
   {
    try
     {
      Class.forName("oracle.jdbc.driver.    OracleDriver");
      Connection con = DriverManager.getConnection(
          "jdbc:oracle:thin:@localhost:1521:orcl",
                        "login1", "pwd1");
    if (con != null)
        System.out.println(" Connected");
    else
        System.out.println(" Not Connected");
    con.close();
  }
  catch (Exception e)
  {
    System.out.println(e);
  }
}
}
```

Insert Statement

```
// Java program to illustrate
 // inserting to the database
  import java.sql.*;


  Public class insert1
  {

  Public static void main (STRING ARGS[])
  {

  String id = "Id1";
  string pwd :- "pwd1"
  String fullname :- "abc";
  String email :- "abc @gmail.com";

  Try
  {

class.forName("oracle.jdbc.driver.Oracle
                    driver");
connection con :- DriverManager.get
                    connection ("
jdbc:oracle:thin:@localhost:1521:orcl", login1"
          , pwd1");
Statement stmt :- con.createStatement();
```

__/__/____

```
// Inserting data in database
String q1 :- "inserting into userid
            values ('"+ id+"', '"+pwd+
   "', '"+ fullname+"', '"+email+"')";
 int x :- stmt . execute update (q1);
   if (x>0)

 System . out println ("Successfully Inserted");
   else

 System . out . println ("Insert failed");


  con . close ();
 }
  catch (Expection e)
 {
  System . out . println (e);
 }
}
}
```

Update Statement

```
// Java program to illustrate
// updating the database
import java.sql.*;

Public class update 1
{
Public static void main (String args [])
{
  string id :- "id1";
  string pwd :- "pwd1";
  string new Pwd :- "newpwd";

  try
  {
  class.for name ("oracle.jdbc.driver.Oracle
                   driver");
Connection con :- driver Manager.get connectio("
jdbc:oracle:thin:@localhost:1521:orcl",
  "login1", pwd1");
  statement stmt :- con.create statement();

// updating database
string q1 :- "UPDATE userid set pwd :- '" +
              newpwd +
"' WHERE id :- '" + id + "' AND pwd :- '" +pwd+"'
  int x :- stmt.execute update (q1);
```

___/___/___

```
if (x>0)
system.out.println("Password successfully
 updated");
  else
system.out.println("ERROR OCCURED:(");

  con close();
}
 catch        (Exception e)
{
  system.out.println(e);
 }
}
}
```

Delete Statement

```
// Java program to illustrate
// deleting from database
import java.sql.*;

Public class delete
{
 string id :- "id2";
 string pwd :- 'pwd2";
 try
 {
```

__/__/____

```
class.for name("oracle.jdbc.driver.oracle
    driver");
connection con :- Driver Manager.get connection ("
jdbc:oracle:thin:@localhost:1521:orcl", "login
1", " pwd 1");
statement stmt :- con create statement();

// deleting from database
 string q1 :- "DELETE from userid WHERE id
    :- "" +id+ "" AND pwd :- "" +pwd+ """;

intx = stmt.execute update (q1);

if(x>0)
  system.out println (" One user successfully
     deleted");
 else
system.out.println("ERROR OCCURED: (");

  con.close();
}
catch (Expection e)
{
 system .out.println(e);
 }
}}
```

_/_/___

Select Statement

```
// Java program to illustrate
// selecting from database
import java.sql.*;

Public class select
{
Public static void main (STRING ARGS[])
{
string id :- "id 1";
string pwd :- "pwd1";
try
{
class.forName ("oracle.jdbc.driver.oracle
        driver");
connection con :- driver Manager.get
     connection ("
jdbc:oracle:thin:@localhost:1521:orcl",
 "login1", "pwd1");
statement stmt :- con:create statement ();

// select query
string q1 :- " select* from userid WHERE
id :- ''' + id +
   ''' AND pwd = ''' pwd + ''''';
Results Set rs : stmt.execute Query (q1)
 if (rs.next())
{
```

```
system.out println("User-ID:"+rs get
    string(1));
system.out.println("Full name:"+rs get
    string(3));
system.out println("E-mail:"+rs.get
    string(4));
}
else
{
system.out.println("No such user id is
    already registered");
}
con.close();
}
catch (Exception e)
{
system.out.println(e);
}
}
}
```