JDBC Tutorial

Java-jdbc is API to connect database.

Steps to connect with Oracle Database.

1. **Load Driver class**

   Class.forName("oracle.jdbc.driver.OracleDriver"); throws ClassNotFoundException.

2. Create connection object

   Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","system","oracle");

3. Create Statement/Prepared Statement object.
4. Then execute query or execute update as per requirement.
5. Close the connection

   Once connection object is created we use three interfaces to send and receive data from database.

   - Statement Interface : Used to execute normal SQL queries.(Hard Coded). It is preferred when a particular SQL query is to be executed only once
   - PreparedStatement Interface : Used to execute dynamic or parameterized SQL queries. It is preferred when a particular query is to be executed multiple times.
   - CallableStatement : Used to execute the stored procedures. It is preferred when the stored procedures are to be executed

   Statement Interface

   It supports static SQL Statements. It cannot accept parameters.

   Connection Utility Class

   This connection will be used throughout the application

```
public Connection getConnection()
      {
            try {

                  Class.forName(oracleDriverClass);
                  connection=DriverManager.getConnection(url, userID, password);
                  } catch (ClassNotFoundException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
            catch (SQLException e) {
                  // TODO: handle exception
                  e.printStackTrace();
            }

            return connection;

                                          }
```

Now a main class will consume it.

For Insert, Update and Delete

```
String query="insert into COMPANY values(003,'TCS','Mumbai')";//Write query to insert
                                                                    Update or Delete here.
              OracleSqlConnection orsql=new OracleSqlConnection();
              Connection connection=orsql.getConnection();
              Statement statement=connection.createStatement();
              int status=statement.executeUpdate(query);// executeUpdate/execute of
Statement interface is used for insert, update or delete purpose.
               System.out.println(status);//execute returns Boolean while executeUpdate
                                    returns integer 1.
```

For displaying/Fetching data

```
String query="select *from Company";
              OracleSqlConnection orsql=new OracleSqlConnection();
              Connection connection=orsql.getConnection();
              Statement statement=connection.createStatement();
              ResultSet rs=statement.executeQuery(query);
              while(rs.next())
              {
                     System.out.println(rs.getInt(1)+" "+rs.getString(2)+"
"+rs.getString(3));
                     Statement interface provides executeQuery method to fetch the
records
                                                     }
```

```
Consider Company table has three columns only int,String and String what will
happen?
Case : 01
```

System.***out***.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3)+" "+rs.getString(4));

As column 4 does not exists so it will be SQL exception saying that Invalid Column Index.

Case : 02

System.***out***.println(rs.getInt(2)+" "+rs.getString(1)+" "+rs.getString(3));

As column 2 contains String and Column 1 contains integer so here also SQL exception is encountered saying Fail to convert to internal representation

JDBC

PreparedStatement Interface

It is used to execute dynamic/parameterized queries.

Insert :

```
OracleSqlConnection orsql=new OracleSqlConnection();
            Connection connection=orsql.getConnection();
            String query="insert into company values(?,?,?)";//placeholder
            PreparedStatement ps=connection.prepareStatement(query);
//setter method of PreparedStatement is used to set values. Rest is same as Statement 
interface
            ps.setInt(1, 200);
            ps.setString(2, "Mahindra");
            ps.setString(3, "Chennai");
            int i=ps.executeUpdate();
            System.out.println(i);
```

Update :

```
OracleSqlConnection orsql=new OracleSqlConnection();
            Connection connection=orsql.getConnection();
            String query="update company set company_name=? where company_id=?";
            PreparedStatement ps=connection.prepareStatement(query);
            ps.setString(1, "Microsoft");
            ps.setInt(2, 3);
            //Here first placeholder indicates company_name and second indicates 
company_id
            //here 1 and 2 does not mean column 1 and column 2 of company table
            int i=ps.executeUpdate();
                                  System.out.println(i);
```

Fetching Displaying records

```
OracleSqlConnection orsql=new OracleSqlConnection();
            Connection connection=orsql.getConnection();
            String query="select *from company where company_id=?";
            PreparedStatement ps=connection.prepareStatement(query);

            ps.setInt(1, 3);
            //Here first placeholder indicates company_name and second indicates 
company_id

            //here 1 and 2 does not mean column 1 and column 2 of company table
            ResultSet rs=ps.executeQuery();

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

executeQuery()	executeUpdate()	execute()
This method is used to execute the SQL statements which retrieve some data from the database.	This method is used to execute the SQL statements which update or modify the database.	This method can be used for any kind of SQL statements.
This method returns a ResultSet object which contains the results returned by the query.	This method returns an int value which represents the number of rows affected by the query. This value will be the 0 for the statements which return nothing.	This method returns a boolean value. TRUE indicates that query returned a ResultSet object and FALSE indicates that query returned an int value or returned nothing.
This method is used to execute only select queries.	This method is used to execute only non-select queries.	This method can be used for both select and non-select queries.
Ex : SELECT	Ex : DML → INSERT, UPDATE and DELETE DDL → CREATE, ALTER	This method can be used for any type of SQL statements.

ResultSetMetaData In JDBC

ResultSetMetaData *is an interface in* java.sql *package of JDBC API which is used to get the metadata about a* ResultSet *object. It will have all the meta data about a* ResultSet *object like schema name, table name, number of columns, column name, datatype of a column etc. You can get this* ResultSetMetaData *object using* getMetaData() *method of* ResultSet.

```
OracleSqlConnection orsql=new OracleSqlConnection();
            Connection connection=orsql.getConnection();
            String query="select *from company";
            PreparedStatement ps=connection.prepareStatement(query);
            ResultSet rs=ps.executeQuery();
            ResultSetMetaData rsmd=rs.getMetaData();
            System.out.println(rsmd.getColumnCount());
            System.out.println(rsmd.getSchemaName(1));
            System.out.println(rsmd.getColumnName(1));
            System.out.println(rsmd.getColumnTypeName(3))
```

Important Methods of DriverManager class

public static void setLoginTimeout(int seconds) This method sets the maximum time in seconds that a driver will wait while attempting to connect to a database. If you pass zero as LoginTimeOut then driver will wait infinitely while attempting to connect to a database.

public static int getLoginTimeout()   This method returns maximum time in second that a driver can wait while attempting to connect to database.

public static Connection getConnection(String URL, String username, String password) throws SQLException   This method returns a Connection object after establishing the connection with the database at the specified URL with specified username and password. If the Driver class of the database is not registered with the DriverManager, it will throw SQLException.

public static Connection getConnection(String URL) throws SQLException   This method returns a Connection object after establishing the connection with the database at the specified URL. If the Driver class of the database is not registered with the DriverManager, it will throw SQLException.

Important Methods of Connection Interface

void setAutoCommit(boolean autoCommit) throws SQLException   This method sets the auto-commit mode of this Connection object. If the auto-commit mode of a Connection object is true, then all SQL statements will be executed and committed as individual transactions. If the auto-commit mode is false then all SQL statements will be grouped in transactions. By default, auto-commit mode of a Connection object is true.

boolean getAutoCommit() throws SQLException This method returns auto-commit mode of this Connection object.

void commit() throws SQLException   This method makes all previous changes made to database since last commit() OR rollback() as permanent. This method should be used only when auto-commit mode of Connection object is false.

void rollback() throws SQLException   This method erases all changes made to database in the current transaction. This method also should be called when auto-commit mode of a Conncetion object is false.