JDBC

1. JDBC stands for **J**ava **D**ata**B**ase **C**onnectivity
2. Using the JDBC APIs you can perform the operations on the Database.
3. These JDBC APIs are the part of **java.sql package**.
4. In this you can connect you java application with the database servers and perform the operations on DB.



1. To connect with the Database form the java application, you will required a Driver which help you to interact with DB.
2. Driver is a java class which is provided by the Database vendors in the form of jar file.

Example : For Mysql you required mysql connector jar file

For Oracle you required ojdbc14 jar file. (You can find out it into your oracle installation directory)

**Why Database**

1. Can store data into file, memory, DB
2. Memory can be use for a small information and for the temporary time. In Java it can be achieved using **Object creation and Heap memory**.
3. File can be use to store data for a longer time. And can store a large information into file. You can have a backup of the file. Data will be store in the unstructured way. This data is difficult to use for performing operation. To achieve this in java you can use **java.io APIs.**
4. Database can be use to store large information into the structure manner which can be store for a longer time and also you can perform operation easily using SQL query. In java to achieve this you can use **JDBC APIs.**

**SQL Queries**

1. **Create a Table (DDL)**

**Create table employee(id int, name varchar(20), email varchar(20), dept varchar(15));**

**Desc employee;**

1. **Insert records(DML)**

Insert into employee values(1, ‘A’, ‘a@gmail.com’,’Dev’);

**Retrieve records**

Select \* from employee;

**Update records**

Update employee set email=’b@yahoo.com’ where id = 2;

**Delete record**

Delete from employee where id = 1;

**Driver jar**

Oracle Jar file (dirver)

C:/oracleexe/app/oracle/product/10.2.0/server/jdbc/lib/**ojdbc14.jar**

Mysql

Mysql connector jar.

Eclipse

Workspace creation

Set the “Java” perspective

Create a “java Project”

File -> New -> Java Project ->

set the Project name and optout the Module creation check box -> next -> Finish

Create a Java class

Right click on the Src

Go to “New” option -> Select “class” option

Set the class name -> Click on “Finish”

Add jar files into Core Java Project

1. Right click On Project
2. Build Path -> Configure Build Path
3. Go to “libraries” tab in the new window
4. Click on “Classpath” Option
5. Click “Add External Jar” button
6. Browse and select jar file and click on “Apply and close” button

**Steps to connect With Database**

1. **Register Driver**
   1. Driver has to register from the jar file.
   2. Internally the Driver class will be loaded and instantiate (Object) into a memory.
   3. This driver is use for all the database interaction.
   4. Syntax:

**Class.forName(“DriverName”);**

* 1. Driver Name

Oracle: **oracle.jdbc.driver.OracleDriver**

MySql-8: **com.mysql.cj.jdbc.Driver**

1. **Create Connection**
   1. To connect with Database you have to provide following details into the java application
      1. URL
         1. Protocol : **mysql- “jdbc:mysql”, Oracle-“jdbc:oracle”**
         2. IP address/HostName: **localhost**
         3. Port Number: **mysql: 3306, Oracle: 1521/1522**
         4. Resource: **mysql: DBName, Oracle: seviceName (XE)**
         5. **MySql :** jdbc:mysql://localhost:3306/<DB>
         6. **Oracle :** jdbc:oracle:thin:@localhost:1521:xe
      2. User Name
      3. Password
   2. Syntax:

**Connection obj = DriverManager.getConnection(“URL”,”UserName”,”password”);**

1. **Create Statement**
   1. It used to hold the sql queries.
   2. Before this step make sure that Connection Object is created.
   3. In this step there are 3 options provided to us.
      1. **Statement**
         1. You can execute DML, DDL and DQL queries using this statement.
         2. You cannot execute the parameterized queries.
         3. It is not secure. It may leads to sql injection.
         4. Is slower than the other options.
         5. Syntax:

**Statement stmt = conObj.createStatement();**

* + 1. **PreparedStatement**
       1. You can execute DML, DDL and DQL queries using this statement.
       2. You can use a parameterized query.
       3. There are no chances of sql injection in this.
       4. It is faster than the Statement.
       5. Syntax:

**PreapredStatement stmt = conObj.prepareStatement(“SQL”);**

* + 1. **CallableStatement**
       1. Are used to execute Procedure in PL/SQL
       2. It is faster than above options.
       3. Syntax:

**CallableStatement stmt = conObj.prepareCall(“{call procedure()}”);**

1. **Execute Statement**
   1. You can execute the query and get the result back into java application.
   2. In this step you have a methods to get query executed and to get result into java program.
   3. In this step there are 3 options(method)
      1. **executeUpdate**()
         1. Is use to execute DML, DDL type of queries except Select query.
         2. This method returns the int value which is the count of records affected after query execution.
         3. Syntax:

**int count = stmtObj.executeUpdate();**

* + 1. **executeQuery()**
       1. is use to execute the Select type of query (DQL).
       2. This method return the object of **ResultSet**. In which you can find all the data selected by query.
       3. Syntax:

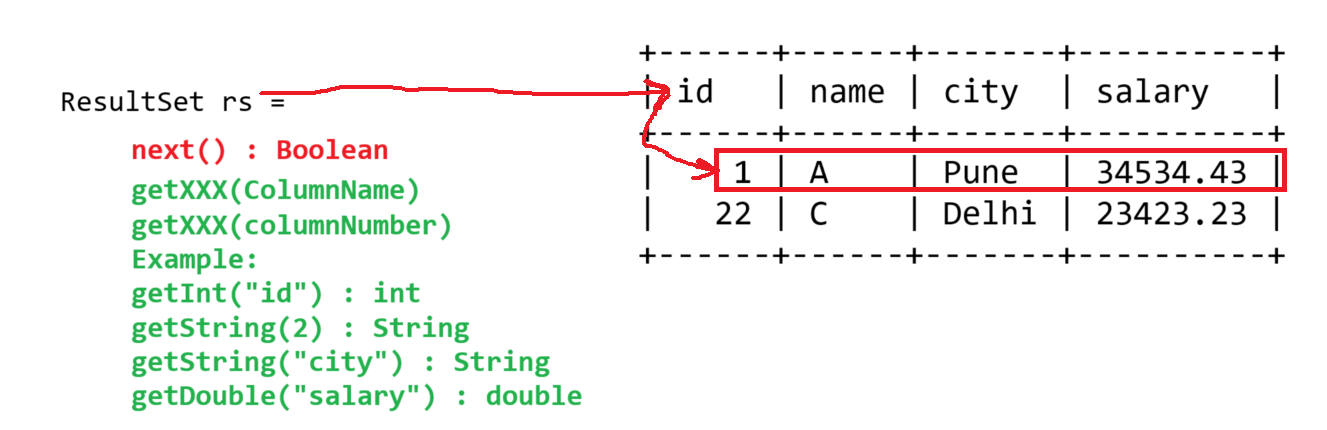
**ResultSet rs = stmtObj.executeQuery();**

* + 1. **execute()** 
       1. Is use to execute all type of queried.
       2. This method returns the Boolean value as a result. If it return true so you can access a ResultSet from it and if returns false then you can get the int value as a response.
       3. Syntax:

**boolean flag = stmtObj.execute();**

1. **Close Connection**
   1. Is use to release the memory acquire by DB connection.
   2. As a part of this step you just have to call close method.

**Handling a ResultSet**



Task-1

Create a table product with column id, name, price, qty

**Create a java program which accept the product details from user and insert the same into DB**

Task-2

**Get the product details from the product tables based on ID and print the billing details into console.**

mysql> select \* from product where id IN (2,4);

+----+--------------+----------+------+

| id | name | price | qty |

+----+--------------+----------+------+

| 2 | SmartTV | 63221.22 | 2 |

| 4 | SmartSpeaker | 5221.22 | 5 |

+----+--------------+----------+------+

**Output**

Name Price Quantity Total Price

--------------------------------------------------------------------------------------

SmartTV 63221.22 2 126,442.44

- - - - - - -

---------------------------------------------------------------------------------------

Bill Amount : <Sum of all the Product total price>