JDBC – Java Database Connectivity

Why we need JDBC?

If we want to connect java program to the database then how can I do this? By using JDBC, we can achieve this.

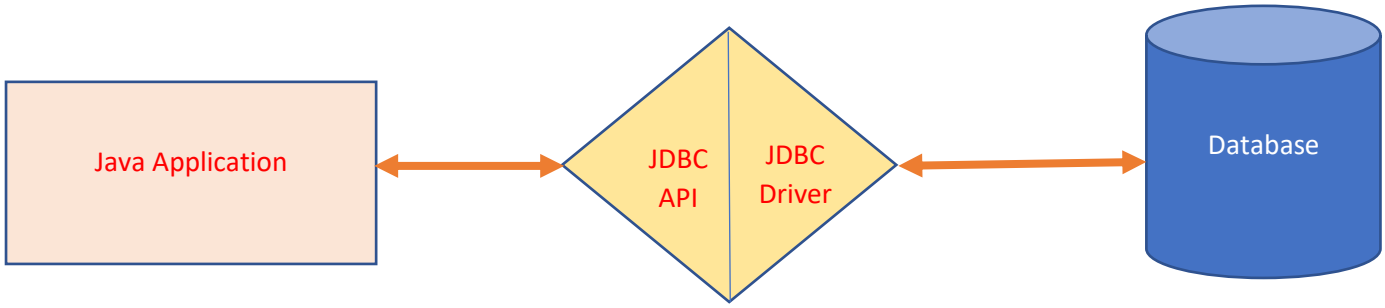
How to add the MySQL jar file into project.

Right click on Project->Build Path->Configure Build Path->Click on Libraries -> Add External Jar->Select Jar File-> Click on Apply and Close or Apply button.

JDBC stands for Java Database Connectivity.

It is an API for the Java programming language.

It allows the client to access the database. It can work on number of operating systems or platforms, like Windows, Mac, etc. It is basically a connection between the database and the application.



To connect Java application with the MySQL database, we need to follow 5 following steps.

1. **Driver class:** The driver class for the MySQL databaseis com.mysql.cj.jdbc.Driver.
2. **Connection URL:** The connection URL for the mysql databaseis jdbc:mysql://localhost:3306/demo","root","root
   jdbc is the API, mysql is the database, localhost is the server name on whichmysql is running, we may also use IP address, 3306 is the port number and test is the database name.
3. **Username:** The default username for the mysql database is **root**.
4. **Password:** It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.
5. **Create a table -**in the mysql database, but before creating table, we need to create database first.

Components of JDBC :

1)**Driver (Translator):**

To convert Java specific calls into Database specific calls and Database specific calls into Java calls.

2)**Connection (Road):**

By using Connection, Java Application can communicate with Database.

3)**Statement (Vehicle):**

By using Statement Object, we can send our SQL Query to the Database and we can get Results from Database.

4)**ResultSet:**

ResultSet holds Results of SQL Query.

Steps of connection-

Step 1: Load the Driver class.

Step 2: Establish the connection.

Step 3: Create the statement.

Step 4: Prepare the SQL statement.

Step 5: Submit the SQL statement to Database.

Step 6: Process the Results.

Step 7: Release the Resources.

MYSQL query for table creation- CREATE TABLE `user` (

`id` int(11) NOT NULL AUTO_INCREMENT,

`LastName` varchar(255) ,

`FirstName` varchar(255) ,

`Address` varchar(255),

`City` varchar(255) ,

`Salary` varchar(255) , PRIMARY KEY (`id`)

);

Example-1 Program for insert the student data using statement through JDBC.

```
package com.operation;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

//Insert the student data using statement
public class InsertData {

public static void main(String[] args) throws
ClassNotFoundException, SQLException {

try {
String sql = "insert into
user(lastName,firstName,Address,City,Salary)"
+ "values('pawar','ram','sangavi','pune',5000)";
// load the driver
Class.forName("com.mysql.cj.jdbc.Driver");
// establish the connection
Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/test",
"root", "root");
// create the sql statement
Statement statement = con.createStatement();

// submit the sql statement to database..
//statement.executeUpdate(sql);
 statement.execute(sql);

System.out.println("Insertion successfully...");

// close the resources.
con.close();
statement.close();
} catch (Exception e) {
 System.out.println(e);
}

}
}
```

Output-

Insertion successfully message displayed on screen and data will be stored into database.

MYSQL query for table creation-

```
CREATE TABLE `employee` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`username` varchar(255) ,
`password` varchar(255), PRIMARY KEY (`id`)
);
```

Example-2 Program for insert the student data using prepared statement.

```
1.  package com.operation;
2.
3.  import java.sql.Connection;
4.  import java.sql.DriverManager;
5.  import java.sql.PreparedStatement;
6.  //insert the data using preparedstatement
7.  public class InsertStudent {
8.
9.  public static void main(String[] args) {
10.
11. try {
12. Class.forName("com.mysql.jdbc.Driver");
13.  Connection con =
14. DriverManager.getConnection("jdbc:mysql://localhost:3306/tes t", "root", "root");
15.
16. PreparedStatement stmt = con.prepareStatement("insert into
    employee(username,password)values(?,?)");
17.
18. stmt.setString(1, "admin"); //1 first parameter in query.
19. stmt.setString(2, "guest");
20.
21. int i=stmt.executeUpdate();
22. System.out.println("Record is inserted."+i);
23.
24. con.close();
25. stmt.close();
26. } catch (Exception e) { e.getMessage();
27. }
28. }
29. }
30.
```

Static Query vs Dynamic Query:

- The sql query without positional parameter(?) is called static query.Eg: delete from employees where ename='vijay'
- The sql query with positional parameter(?) is called dynamic query.Eg: select * from employees where esal>?

Note:

Simple Statement can be used only for static queries where as PreparedStatement can used for both static and dynamic queries.

Statement	Prepared Statement
1) At the time of creating Statement Object, we are not required to provide any Query. Statement st = con.createStatement(); Hence Statement Object is not associated with any Query and we can use for multiple Queries.	1) At the time of creating PreparedStatement, we have to provide SQL Query compulsory and will send to theDatabase and will be compiled. PS pst = con.prepareStatement(query); Hence PS is associated with only one Query.
2) Whenever we are using execute Method, every time Query will be compiled and executed.	2) Whenever we are using execute Method, Query won't be compiled justwill be executed.
3) Statement Object can work only forStatic Queries.	3) PS Object can work for both Staticand Dynamic Queries.
4) Relatively Performance is Low.	4) Relatively Performance is High.
5) Best choice if we want to work withmultiple Queries.	5) Best choice if we want to work with only one Query but required to execute multiple times.