**JDBC Programming**

**MySQL Connector JAR file download:**

<https://jar-download.com/artifacts/mysql/mysql-connector-java/8.0.30/source-code>

<https://jar-download.com/?search_box=mysql-connector-j-8.0.32>

JDBC (Java Database Connectivity)

JDBC is an API used to connect java application to database such as Oracle, Mysql any databases.

It’s a Platform independent interface between rdbms and java programming.

It allows java programs to execute sql statements to interact with the db.

JDBC Drivers

JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand. JDBC drivers are the software components which implements interfaces in JDBC APIs to enable java application to interact with the database. Now we will learn how many JDBC driver types does Sun defines? There are four types of JDBC drivers defined by Sun Microsystem that are mentioned below:

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

 Thin driver – Type 4 driver (fully Java driver)

Type-4 driver is also called native protocol driver. This driver interacts directly with database. It does not require any native database library, that is, why it is also known as Thin Driver.

**Steps to Connect to DB**

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

* Register the Driver class
* Create connection
* Create statement
* Execute queries
* Close connection

The **java.sql** package contains classes and interfaces for JDBC API. A list of popular *interfaces* of JDBC API are given below:

* Driver interface
* Connection interface
* Statement interface
* PreparedStatement interface
* CallableStatement interface
* ResultSet interface
* ResultSetMetaData interface
* DatabaseMetaData interface
* RowSet interface

A list of popular *classes* of JDBC API are given below:

* DriverManager class
* Blob class
* Clob class
* Types class

**Difference between Statement and PreparedStatement**

**1. Statement :**   
It is used for accessing your database. Statement interface cannot accept parameters and useful when you are using static SQL statements at runtime. If you want to run [SQL](https://www.geeksforgeeks.org/sql-tutorial/) query only once then this interface is preferred over PreparedStatement.

**Example –**

//Creating The Statement Object

Statement prep\_statement = con.createStatement();

//Executing The Statement

prep\_statement.executeUpdate("CREATE TABLE STUDENT(ID NUMBER NOT NULL, NAME VARCHAR)");

**2. PreparedStatement :**   
It is used when you want to use SQL statements many times. The PreparedStatement interface accepts input parameters at runtime. 

**Example –**

//Creating the PreparedStatement object

PreparedStatement prep\_statement = con.prepareStatement("update STUDENT set NAME = ? where ID = ?");

//Setting values to place holders

//Assigns "RAM" to first place holder

prep\_statement.setString(1, "RAM");

//Assigns "512" to second place holder

prep\_statement.setInt(2, 512);

//Executing PreparedStatement

prep\_statement.executeUpdate();

**Connection Interface**

**Commonly used methods of Connection interface:**

1) public Statement createStatement(): creates a statement object that can be used to execute SQL queries.

2) public Statement createStatement(int resultSetType,int resultSetConcurrency): Creates a Statement object that will generate ResultSet objects with the given type and concurrency.

3) public void setAutoCommit(boolean status): is used to set the commit status. By default, it is true.

4) public void commit(): saves the changes made since the previous commit/rollback is permanent.

5) public void rollback(): Drops all changes made since the previous commit/rollback.

6) public void close(): closes the connection and Releases a JDBC resources immediately.

**Statement interface**

The Statement interface provides methods to execute queries with the database. The statement interface is a factory of ResultSet i.e. it provides factory method to get the object of ResultSet.

**Commonly used methods of Statement interface:**

The important methods of Statement interface are as follows:

1) public ResultSet executeQuery(String sql): is used to execute SELECT query. It returns the object of ResultSet.

2) public int executeUpdate(String sql): is used to execute specified query, it may be create, drop, insert, update, delete etc.

3) public boolean execute(String sql): is used to execute queries that may return multiple results.

4) public int[] executeBatch(): is used to execute batch of commands.

**Commonly used methods of ResultSet interface**

|  |  |
| --- | --- |
| **1) public boolean next():** | is used to move the cursor to the one row next from the current position. |
| **2) public boolean previous():** | is used to move the cursor to the one row previous from the current position. |
| **3) public boolean first():** | is used to move the cursor to the first row in result set object. |
| **4) public boolean last():** | is used to move the cursor to the last row in result set object. |
| **5) public boolean absolute(int row):** | is used to move the cursor to the specified row number in the ResultSet object. |
| **6) public boolean relative(int row):** | is used to move the cursor to the relative row number in the ResultSet object, it may be positive or negative. |
| **7) public int getInt(int columnIndex):** | is used to return the data of specified column index of the current row as int. |
| **8) public int getInt(String columnName):** | is used to return the data of specified column name of the current row as int. |
| **9) public String getString(int columnIndex):** | is used to return the data of specified column index of the current row as String. |
| **10) public String getString(String columnName):** |  |

package JDBC;  
  
import java.sql.\*;  
import java.sql.DriverManager;  
import java.util.Scanner;  
  
public class ConnectDatabase {  
 public static void main(String[] args) throws ClassNotFoundException, SQLException {  
 try {  
 Connection connection = null;  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 connection = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/wipro\_java\_jdbc", "root", "rps@123");  
 System.*out*.println("Connected");  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("Enter name");  
 String name = sc.nextLine();  
 System.*out*.println("Enter password");  
 String password = sc.nextLine();  
 System.*out*.println("Enter id");  
 int id = sc.nextInt();  
  
 String insertQuery = "insert into student(student\_id,student\_name,student\_password) values(?,?,?)";  
 PreparedStatement pstmt = connection.prepareStatement(insertQuery);  
 pstmt.setInt(1, id);  
 pstmt.setString(2, name);  
 pstmt.setString(3, password);  
 int x = pstmt.executeUpdate();  
 if (x > 0)  
 System.*out*.println("Inserted Successfully");  
 else  
  
 System.*out*.println("Could Not able to Insert");  
 } catch (Exception e) {  
 System.*out*.println(e);  
 }  
 }  
}

**Transaction management in JDBC**

package JDBC;  
  
import java.sql.\*;  
import java.sql.DriverManager;  
import java.util.Scanner;  
  
public class TransactionManagement {  
 public static void main(String[] args) throws ClassNotFoundException, SQLException {  
 try {  
 Connection connection = null;  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 connection = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/wipro\_java\_jdbc", "root", "rps@123");  
 System.*out*.println("Connected");  
 Scanner sc = new Scanner(System.*in*);  
 connection.setAutoCommit(false);  
 String insertQuery = "insert into student1(student\_id,student\_name,student\_password) values(?,?,?)";  
 PreparedStatement pstmt = connection.prepareStatement(insertQuery);  
 while (true) {  
 System.*out*.println("Enter id");  
 int id = Integer.*parseInt*(sc.nextLine());  
 System.*out*.println("Enter name");  
 String name = sc.nextLine();  
 System.*out*.println("Enter password");  
 String password = sc.nextLine();  
  
 pstmt.setInt(1, id);  
 pstmt.setString(2, name);  
 pstmt.setString(3, password);  
 pstmt.executeUpdate();  
 System.*out*.println("Want to Commit or Rollback");  
 String commit = sc.nextLine();  
 if (commit.equals("Yes") || commit.equals("yes"))  
 connection.commit();  
 if (commit.equals("Rollback") || commit.equals("rollback"))  
 connection.rollback();  
 System.*out*.println("Want to insert values ");  
 String a = sc.nextLine();  
 if (a.equals("Yes") || a.equals("yes"))  
 continue;  
 else  
 break;  
 }  
 }  
 catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}