JDBC Driver

# What is JDBC?

* **JDBC(Java Database connectivity)** allows programs to be linked to a database.
* ‘Driver Software’ is responsible for converting java application calls to database calls and database calls to java application.

**JDBC Driver**

**Java Application**

**JDBC**



Database

JDBC API has two drivers

* Java.sql
* Javax.sql

## Interfaces and Classes of JDBC API

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

## JDBC Driver types

There are four types of JDBC driver

* Type 1 or JDBC-ODBC driver
* Type 2 or Native API driver
* Type 3 or Network protocol driver
* Type 4 or Thin driver

Driver is selected based on the database type

|  |  |
| --- | --- |
| Database | Driver |
| MySQL | com.mysql.jdbc.Driver |

For mySQL, we need to download a JAR file named ‘*mysql.connector*‘

https://dev.mysql.com/downloads/connector/j/

## Steps for connecting to a database

1. Import the required package for the corresponding database.
2. Load and register the JDBC drivers.
3. Establish the connection
4. Create a statement
5. Execute the query
6. Process the results
7. Close the connections

Example:

* Consider you want to talk to somebody far away. For which we need a device known as cell-phone/telecom which signifies our Step 1.
* Now user needs a network and a SIM card which is equivalent to the loading and registering process signifying Step 2.
* Now with a working SIM card, user is required to dial numbers and pressing the call button signifying step 3 as we are trying to establish the connection to commute.
* Whatever message we need to deliver is equivalent to creating a statement.
* Now once connected we need to speak the message as we have thought off, which is equivalent to the execution of the query.
* Now a response will be received which was our goal signifies step 6 of processing results technically in the case of JDBC.
* Lastly, we need to press the button on the device to close communication, similarly, it is a good practice of closing the connections in the technical aspect which here is JDBC.

# Implementing JDBC

There are three types of statements that can be executed

* Normal statements – static sql statements without parameters
* Prepared statements – SQL statements that can be parameterized
* Callable statements – Stored procedures

Steps to implement JDBC

* Download mysql.connector jar file
* Apply the jar file for the project
* Create a class
* Import java.sql.\*
* Write code to establish connect
* Execute the query
* Close the connection

**try** {

Class.*forName*("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.*getConnection*(

"jdbc:mysql://localhost:3306/mydb",

"root", "password");

Statement st = con.createStatement();

ResultSet rs = st.executeQuery(

"select \* from country");

**while** (rs.next()) {

System.***out***.println(rs.getInt("CountryID")+"-----"+rs.getString("CountryName"));

}

con.close();

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

}

For Inserting data into a database:

st.execute("Insert into country values (5,'Australia','AU')");

# prepared Stements

The PreparedStatement interface is a subinterface of Statement. It is used to execute parameterized query.

Example:

String sql="insert into emp values(?,?,?)";

## Methods available in Prepared Statements interface

|  |  |
| --- | --- |
| Methods | Description |
| setInt() | sets the integer value to the given parameter index. |
| setString() | sets the string value to the given parameter index. |
| setFloat() | sets the float value to the given parameter index. |
| setDouble() | sets the double value to the given parameter index. |
| executeUpdate() | executes the query. It is used for create, drop, insert, update, delete etc. |
| executeQuery() | executes the select query. It returns an instance of ResultSet. |

--------------------------For Date and Time--------------------------

prst.setDate(4, Date.*valueOf*("2024-08-21"));

prst.setTimestamp(2, Timestamp.*valueOf*("2024-08-21 13:30:00"));

Example of a prepared statement :

PreparedStatement stmt=con.prepareStatement("insert into country values(?,?,?)");

stmt.setInt(1,5);//1 specifies the first parameter in the query

stmt.setString(2,"Australia");

stmt.setString(3, "AUS");

**int** i=stmt.executeUpdate();

System.***out***.println(i+" records inserted");

con.close();

**Prepared statement – creating Insert operation by taking input from user**

**try** {

Class.*forName*("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.*getConnection*(

"jdbc:mysql://localhost:3306/mydb",

"root", "Meghu\*\*8288");

PreparedStatement stmt=con.prepareStatement("insert into country values(?,?,?)");

Scanner inp = **new** Scanner(System.***in***);

Scanner inp2 = **new** Scanner(System.***in***);

**do** {

System.***out***.println("Enter country ID");

**int** countryID = inp.nextInt();

System.***out***.println("Enter country Name");

String countryName = inp2.nextLine();

System.***out***.println("Enter country Code");

String countryCode = inp2.nextLine();

stmt.setInt(1,countryID);

stmt.setString(2,countryName);

stmt.setString(3,countryCode);

**int** i=stmt.executeUpdate();

System.***out***.println(i+" records affected");

System.***out***.println("Do you want to continue: y/n");

String input = inp2.nextLine();

**if**(input.equalsIgnoreCase("n"))

**break**;

}**while**(**true**);

}

**catch**(Exception ex)

{

ex.printStackTrace();

}

# ResultSet MetaData

* Metadata is data about data i.e. we can get further information from the data.
* Example metadata of a table like total number of columns, column name, column type etc
* ResultSetMetaData interface is useful because it provides methods to get metadata from the ResultSet object.

|  |  |
| --- | --- |
| Method | Descriotion |
| public int getColumnCount() | It returns the total number of columns in the ResultSet object |
| public String getColumnName(int index) | It returns the column name of the specified column index. |
| public String getColumnTypeName | It returns the column type name for the specified index. |
| public String getTableName(int index) | It returns the table name for the specified column index. |

PreparedStatement stmt=con.prepareStatement("Select \* from country");

ResultSet rs = stmt.executeQuery();

ResultSetMetaData metadata = rs.getMetaData();

System.***out***.println("Total columns: "+metadata.getColumnCount());

System.***out***.println("Column Name of 1st column: "+metadata.getColumnName(1));

System.***out***.println("Column Type Name of 1st column: "+metadata.getColumnTypeName(1));

System.***out***.println("Column Type Name of 1st column: "+metadata.getTableName(1));

con.close();

# Database Metadata

DatabaseMetaData interface provides methods to get meta data of a database such as database product name, database product version, driver name, name of total number of tables, name of total number of views etc.

|  |  |
| --- | --- |
| Method | Description |
| public String getDriverName() | it returns the name of the JDBC driver. |
| public String getDriverVersion() | it returns the version number of the JDBC driver. |
| public String getUserName() | it returns the username of the database. |
| public String getDatabaseProductName() | it returns the product name of the database. |
| public String getDatabaseProductVersion() | it returns the product version of the database. |
| public ResultSet getTables | it returns the description of the tables |

DatabaseMetaData dbmd=con.getMetaData();

System.***out***.println("Driver Name: "+dbmd.getDriverName());

System.***out***.println("Driver Version: "+dbmd.getDriverVersion());

System.***out***.println("UserName: "+dbmd.getUserName());

System.***out***.println("Database Product Name: "+dbmd.getDatabaseProductName());

System.***out***.println("Database Product Version: "+dbmd.getDatabaseProductVersion());

--------------Getting table names------------------

String table[]={"TABLE"};

ResultSet rs1=dbmd.getTables(**null**,**null**,**null**,table);

**while**(rs1.next()){

System.***out***.println(rs1.getString(3));

}