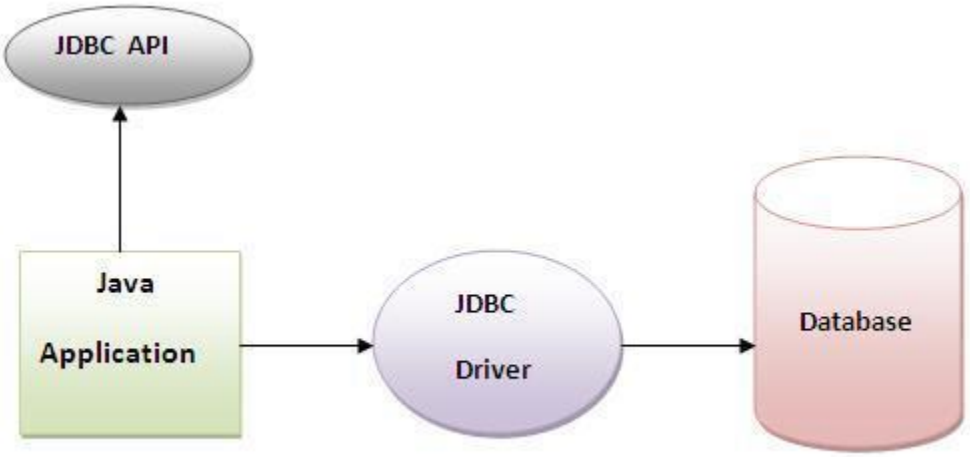
JDBC

Java JDBC

Java JDBC is a java API to connect and execute query with the database. JDBC API uses jdbc drivers to connect with the database.



What is API

API (Application programming interface) is a document that contains description of all the features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other. An API can be created for applications, libraries, operating systems, etc.

JDBC Driver

JDBC Driver is a software component that enables java application to interact with the database.

5 Steps to connect to the database in java

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

1. Register the driver class

2. Creating connection

3. Creating statement

4. Executing queries

5. Closing connection

1. Register the driver class

   The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class.

   Example: Class.forName("com.mysql.jdbc.Driver");

2. Create the connection object

   The getConnection() method of DriverManager class is used to establish connection with the database.

   public static Connection getConnection(String url)throws SQLException

   public static Connection getConnection(String url,String name,String password) throws SQLException

   Example:

   ```
   Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/studentdb","","");
   ```

3. Create the Statement object

   The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

   public Statement createStatement()throws SQLException

   Example: Statement stmt=con.createStatement();

4. Execute the query

   The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

   public ResultSet executeQuery(String sql)throws SQLException

   Example :

   ```
   ResultSet rs=stmt.executeQuery("select * from emp");
   while ( rs.next() ){
   ```

```
        System.out.println(rs.getInt(1)+" "+rs.getString(2));
    }
```

5. Close the connection object

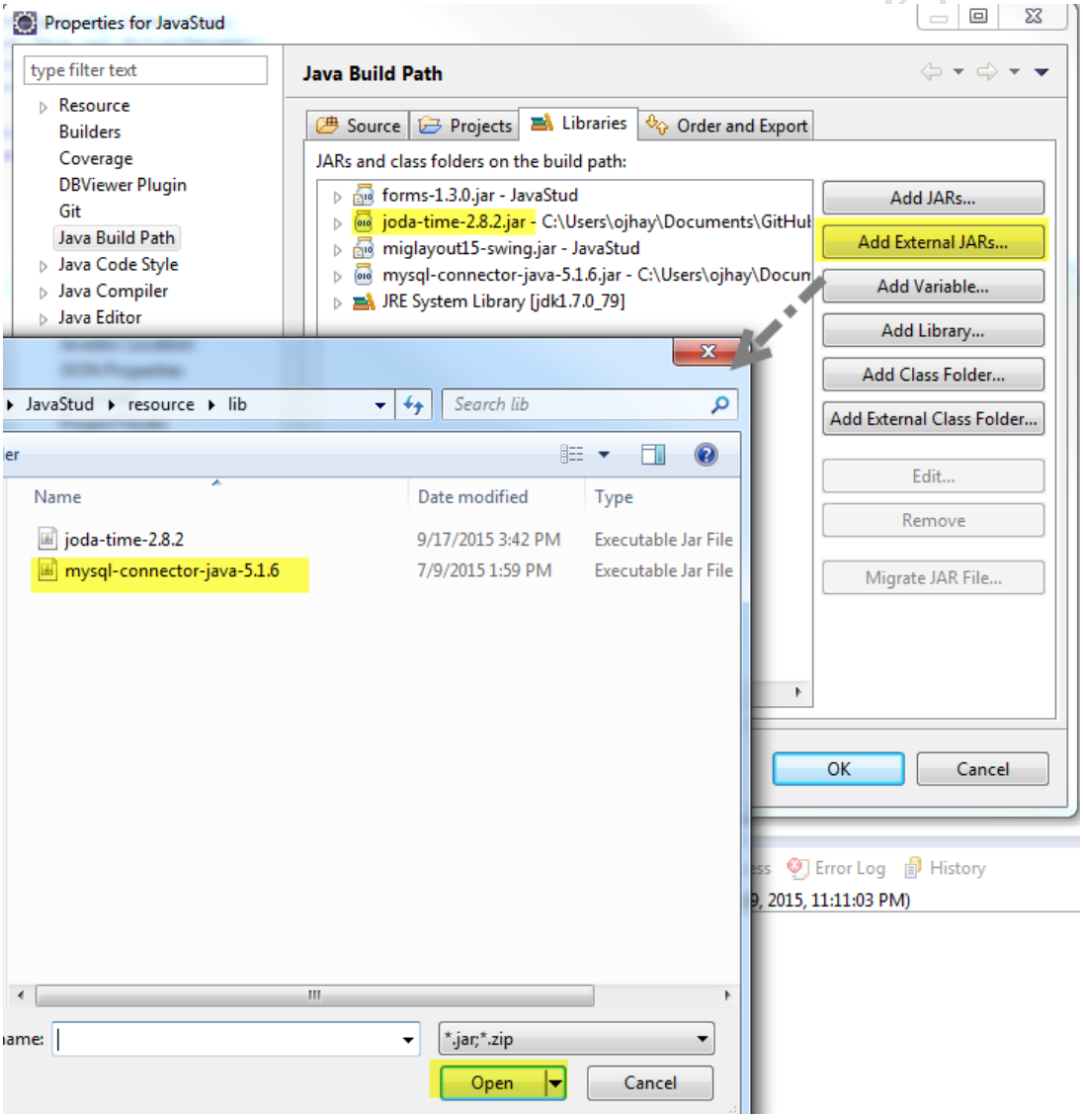
   By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

   public void close()throws SQLException

   Example:

   con.close();

Add Library to Project:



JDBC

Java Database Connectivity with MySQL

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

In this example we are using MySQL as the database. So, we need to know following information's for the MySQL database:

Driver class:

The driver class for the MySQL database is com.mysql.jdbc.Driver.

Connection URL:

The connection URL for the MySQL database is jdbc:mysql://localhost:3306/stud where jdbc is the API, MySQL is the database, localhost is the server name on which MySQL is running, we may also use IP address, 3306 is the port number and stud is the database name. We may use any database, in such case, we need to replace the stud with our database name.

Username:

The default username for the MySQL database is root.

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.

DriverManager class

The DriverManager class acts as an interface between user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. The DriverManager class maintains a list of Driver classes that have registered themselves by calling the method DriverManager.registerDriver()

Commonly used methods of DriverManager class

public static void registerDriver(Driver driver): is used to register the given driver with DriverManager.

public static void deregisterDriver(Driver driver): is used to deregister the given driver (drop the driver from the list) with DriverManager.

public static Connection getConnection(String url): is used to establish the connection with the specified url

public static Connection getConnection(String url,String userName,String password): is used to establish the connection with the specified url, username and password.

Connection interface

A Connection is the session between java application and database. The Connection interface is a factory of Statement, PreparedStatement, and DatabaseMetaData i.e. object of Connection can be used to get the object of Statement and DatabaseMetaData. The Connection interface provide many methods for transaction management like commit(), rollback() etc.

Commonly used methods of Connection interface:

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

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

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

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

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

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:

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

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

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

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

ResultSet interface

The object of ResultSet maintains a cursor pointing to a particular row of data. Initially, cursor points to before the first row.

Commonly used methods of ResultSet interface:

public boolean next(): is used to move the cursor to the one row next from the current position.

public boolean previous(): is used to move the cursor to the one row previous from the current position.

public boolean first(): is used to move the cursor to the first row in result set object.

public boolean last(): is used to move the cursor to the last row in result set object.

public boolean absolute(int row): is used to move the cursor to the specified row number in the ResultSet object.

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.

public int getInt(int columnIndex): is used to return the data of specified column index of the current row as int.

public int getInt(String columnName): is used to return the data of specified column name of the current row as int.

public String getString(int columnIndex): is used to return the data of specified column index of the current row as String.

public String getString(String columnName): is used to return the data of specified column name of the current row as String.

JdbcConnectionTest.java

```
1  package jdbc;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.ResultSet;
6  import java.sql.Statement;
7
8  public class JdbcConnectionTest {
9      public static void main(String[] args) {
10
11         try {
12             // 1. Load Driver
13 1           Class.forName("com.mysql.jdbc.Driver");
14
15             // 2.Connect to the Database: URL:database Path, username, password.
16             // here studentdb is database name, root is username and password
17 2           Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/studentdb", "root", "");
18
19             // 3. Create Statement
20 3           Statement stmt = con.createStatement();
21
22             // 4. Execute Query
23 4           ResultSet rs = stmt.executeQuery("select * from user");
24
25             // 5. Fetch the result.
26             while (rs.next()) {// Loop Each Row
27                 // Fetch columns
28 5               // System.out.println(rs.getInt(1) + "  " + rs.getString(2));
29                 System.out.println(rs.getInt("id") + "  " + rs.getString("username") + " : " + rs.getString("password"));
30             }
31
32             // 6. Close Connection
33 6           con.close();
34
35         } catch (Exception e) {
36             e.printStackTrace();
37
38         }
39
40     }
41
42 }
43
```

PreparedStatement interface

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

Let's see the example of parameterized query:

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

As you can see, we are passing parameter (?) for the values. Its value will be set by calling the setter methods of PreparedStatement.

The **prepareStatement()** method of Connection interface is used to return the object of PreparedStatement.

Commons methods of PreparedStatement interface are:

public void setInt(int paramIndex, int value) : sets the integer value to the given parameter index.

public void setString(int paramIndex, String value): sets the String value to the given parameter index.

public void setFloat(int paramIndex, float value): sets the float value to the given parameter index.

public void setDouble(int paramIndex, double value): sets the double value to the given parameter index.

public int executeUpdate(): executes the query. It is used for create, drop, insert, update, delete etc.

public ResultSet executeQuery(): executes the select query. It returns an instance of ResultSet.

void addBatch(String query): adds query into batch.

int[] executeBatch(): executes the batch of queries.

PreparedStatementTest.java

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class PreparedStatementTest {

    public static void main(String[] args) throws ClassNotFoundException, SQLException {
        //Load Driver
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/stud", "root", " your password ");

        //Create prepared statement
        PreparedStatement stmt = connection.prepareStatement("insert into user(id,name) values(?,?)");
        stmt.setInt(1,101);//1 specifies the first parameter in the query
        stmt.setString(2,"Prabeen");  // 2 specifies the second parameter in the query

        int i=stmt.executeUpdate();  // execute statement
        System.out.println(i+" records inserted");
        connection.close();
    }

}
```