**PRACTICAL # 09**

**OBJECT:**

Java Connectivity with MySQL and CRUD Operations.

**THEORY:**

Java connects to different databases through the DB connectors or drivers. Such drivers are available for major DBMS like Oracle, MySQL, SQLServer etc.

In this lab, we’ll work with MySQL database connectivity and performing CRUD operations.

**Setting up DB Connector:**

Before using a database, you need to setup the database driver.

To connect Java application with the mysql database, the mysqlconnector.jar file needs to be loaded.

First download a connector driver for your database: **mysql-connector.jar** file.

To load this file, there are two alternatives:

1. Paste the mysqlconnector.jar file in **jre/lib/ext** folder:

Navigate to your Java installation folder and copy this jar file in the jre/lib/ext folder.

OR

2. Setup the classpath:

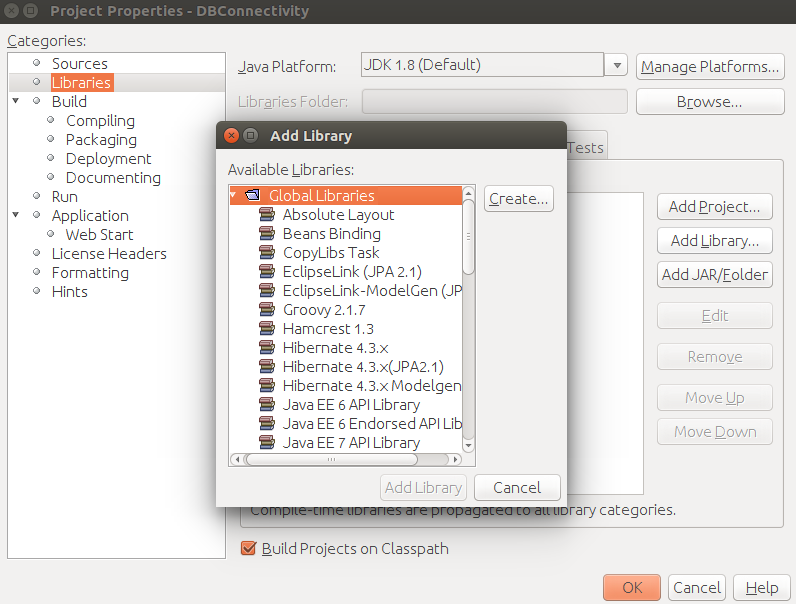
Go to environment variable then click on new tab. In variable name write classpath and in variable value paste the path to the mysqlconnector.jar file by appending mysqlconnector.jar;.; as C:\folder\mysql-connector-java-5.0.8-bin.jar;.;

You can set the classpath environment variable under Unix, Linux or Mac OS X either locally for a user within their .profile, .login or another login file. You can also set it globally by editing the global /etc/profile file.

OR add the Connector/J driver to your classpath using:

export CLASSPATH=/path/MySQL-connector-java-ver-bin.jar:$CLASSPATH

**Setting Connector in Netbeans IDE:**

Select project settings → From the properties side panel, select libraries → Select Add Library → Finally select MySQL JDBC Driver.

**Using Database:**

There usually are 5 steps to use a database in Java using JDBC.

1) Register the driver class:

This method is used to dynamically load the driver class. This method is not required in later versions of Java and JDBC connector, so you can safely ignore it if you JDBC v.4 or above, do not write it.

Example MySQL:

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

Example SQLServer:

**Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");**

2) Create the connection object

Next you need to create connection object as follows:

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

Exampe: MySQL Connectivity:

**Connection con=DriverManager.getConnection(**

**"jdbc:mysql://localhost:3306/JavaDB","username","password");**

Exampe: SQLServer Connectivity:

**Connection con=DriverManager.getConnection(**

**"sqlserver://localhost\\sqlexpress;user=sa;password=secret","username","password");**

3) Create the Statement object

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

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.

ResultSet rs=stmt.executeQuery("select \* from users");

while(rs.next()){

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

}

5) Close the connection object

The close() method of Connection interface is used to close the connection.

Example:

**con.close();**

**Program:**

This is an example program for CRUD operations.

public static void main(String[] args) {

Connection con=getConnection();

//insertUser(con);

//updateUser(con);

//deleteUser(con);

selectUsers(con);

con.close();

}

public static Connection getConnection(){

Connection con=null;

try{

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

con=DriverManager.getConnection(

"jdbc:mysql://localhost:3306/JavaDB","JavaUser","abcd1234");

}catch(Exception e){ System.out.println(e);}

return con;

}

public static void selectUsers(Connection con){

try{

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select \* from users");

while(rs.next())

System.out.println(rs.getInt(1)

+" "+rs.getString(2)+" "+rs.getString(3));

}catch(Exception e){ System.out.println(e);}

}

public static void insertUser(Connection con){

try{

Statement stmt=con.createStatement();

boolean res=stmt.execute("insert into users values "

+ "(4, \"shani\", \"niz\", \"user\")");

}catch(Exception e){ System.out.println(e);}

}

public static void updateUser(Connection con){

try{

Statement stmt=con.createStatement();

String query = "update users set "

+ "user\_name=\"shanniz\""

+ " where user\_name=\"shani\"";

boolean res=stmt.execute(query);

}catch(Exception e){ System.out.println(e);}

}

public static void deleteUser(Connection con){

try{

Statement stmt=con.createStatement();

String query = "delete from users where "

+ "user\_name=\"shanniz\"";

System.out.println(query);

boolean res=stmt.execute(query);

}catch(Exception e){ System.out.println(e);}

}

**ACTIVITIES**

**Activity 1**

Write a program to

**Activity 1**

Write a program to

**Activity 2**

Write a program to

**REVIEW QUESTIONS**

1. What is a database connector?
2. What information is specified, while establishing db connection?
3. What is Statement object used for?
4. How do we iterate the results of select query?
5. Why do we close the connection after use?