12th August 2023

JDBC - Java Database Connectivity

file - data may get repeated

JDBC - native API

JDBC - to connect java application to relational database

We should have external .jar of the database software.

~~for JDK 8~~ → Step 0 → Add external .jar

for JDK 5, there's no cj

Step - ① Load driver class

cj

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

Step - ② Establish connection

```
Connection con = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/test",
Username → "root", "root" ← password);
```

database name

Date: 12/08/23 YOUVA

Step-③ Construct SQL Query

String sql = "insert into Student values (1, 'XYZ', 'Karvenagar');

Step-④ create reference of required jdbc sql statement, (Create Statement's reference)

Statement smt = con.createStatement();

Step-⑤ submit sql query

smt.execute(sql);

Step-⑥ close all resources

smt.close();

con.close();

Statement interface

PreparedStatement interface

CallableStatement interface

Eclipse IDE - adding JAR file in project -

Eclipse - Project name → right click → Build path → Config build path → Libraries → Add external JARs.

Select query साठी

ResultSet rs = smt.executeQuery(sql);

```
public class Test1
{
public static void main(String[] args) throws Exception
{
    Class.forName("com.mysql.cj.jdbc.Driver");
Connection con = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/test",
            "root", "root");
```

"root" → Username of mysql

"root" → password of mysql

test → databasename

```
String sql = "select * from student";
Statement smt = con.createStatement();
ResultSet rs = smt.executeQuery(sql);
```

```
while (rs.next())                          1st column
{   System.out.println(rs.getInt(1));
    System.out.println(rs.getString(2));
}   System.out.println(rs.getString(3));

smt.close();   ← till JDK 1.6, हे करावे लागते.
con.close();
```

Autoclosable interface from JDK 1.7:

try with resources -

This feature is added in JDK 1.7 version. Resources get closed automatically. No. of lines of code will reduce. Memory leakage problem will be overcome. We can use those interfaces and classes which have "implements" or "extends" Autoclosable interface.

```
public class Test2
{   public static void main(String[] args) {
try {
    Class.forName("com.mysql.cj.jdbc.Driver");
    try (Connection con = DriverManager.getConnection
                ("jdbc:mysql://localhost:3306/test",
            username → "root", "root" ← password);
        Statement smt = con.createStatement();
        Resultset rs = smt.executeQuery("select * from student");
    )
    {
        while (rs.next())
        {   System.out.println(rs.getInt(1));
            System.out.println(rs.getString(2));
            System.out.println(rs.getString(3));
        }
    }
```

executeUpdate() - update, delete queries
executeQuery() - select query
execute() - insert query

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

executeUpdate() → delete query
update query

13th August 2023

JDBC - database Independent
- to connect java applic^ to relational database

```
public static void main(String[] args)
{
    Class.forName("com.mysql.cj.jdbc.Driver");
Connection con =
    DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test", "root", "root");
```

(test: database name; root: username; root: password)

```
String update = "update student set address='Mumbai'
                where rollno=1";
String delete = "delete from student where rollno=3";
```

Statement Interface's reference

```
Statement stmt = con.createStatement();

stmt.executeUpdate(update);
stmt.executeUpdate(delete);

stmt.close();
con.close();
}
```

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

Inner classes ↳ core or advanced ?

- Mysql server चा version प्रमाणे mysql connector चा .jar चे version पाहिजे.
- java.sql चे package वापरणे (JDBC साठी)

String name = sc.next() + sc.nextLine();

Statement interface – DDL operations

PreparedStatement – DML operations.
interface

PreparedStatement interface – child interface of Statement interface

Runtime ला fields चा values हवा असल्या तर त्या Scanner ने घेऊन query मध्ये concatenate करावा लागतील.

This is not feasible approach for multiple fields.
So, we've to use PreparedStatement interface.

Factory design pattern
- utility class बनवून commonly needed lines of code एका method मध्ये घालणे.

Positional parameters – ?

insert into student values(? ? ?)

Factory design pattern ← creating a utility class

```
public class DBUtil
{
    private static Connection con = null;
    public static Connection getConnection()
    {
        //Step 1 - Load driver class
        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
            try{
            con = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/test",
                    "root", "root");
```

```
        catch( SQLException e)
        {
            e.printStackTrace();
        }

    return con;
    }
    catch (ClassNotFoundException e)
    {
        e.printStackTrace();
    }

    return con;
    }
}
```

class written here

```
public class FetchData
{
    p.s.v.m.(String[] args) throws SQLException
    {
        Scanner sc = new Scanner(System.in);
        Connection con = DBUtil.getConnection();
        // step 3 :- Create query
        String fetch = "select * from student where rollno=?";
        // step 4 :- Create PreparedStatement reference
        PreparedStatement ps = con.prepareStatement(fetch);
        // step 5 :- add data in query at runtime
        System.out.println("Enter student rollno to get data");
        int rn = sc.nextInt();
        ps.setInt(1, rn);
        // step 6 :- Execute Query
        ResultSet rs = ps.executeQuery();
        while(rs.next())
        {
            System.out.println("student rollno" + rs.getInt(1));
            System.out.println("student name" + rs.getString(2));
            System.out.println("student address" + rs.getString(3));
        }
    }
}
```

```
public class InsertTest
{
    p.s.v.m.(String[] args) throws SQLException
    {   Scanner sc = new Scanner(System.in);
        Connection con = DBUtil.getConnection();
        // step 3 :- create query
        String Insert = "insert into student values(?,?,?)";
        // step 4 :- create PreparedStatement reference
        PreparedStatement ps = con.prepareStatement(insert);
        // step 5 :- provide data into the query at runtime
        Sysout.("Enter student rollno");
        ps.setInt(1, sc.nextInt());
        Sysout.("enter student name");
        ps.setString(2, sc.next() + sc.nextLine());
        Sysout.("Enter student's address");
        ps.setString(3, sc.next() + sc.nextLine());
        // step 6 :- execute query
        ps.execute();
        // step 7 :- close resources
        ps.close();
        con.close();
        Sysout.("Done!");
    }
}
```