3.4 DATABASE CONNECTIVITY

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

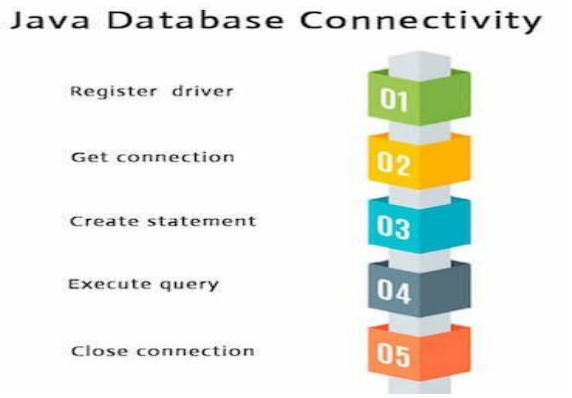

Fig 3.3 Java Db Connectivity

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.

Syntax of forName() method

public static void forName(String className)throws ClassNotFoundException

Example to register the OracleDriver class

Here, Java program is loading oracle driver to esteblish database connection.

```
Class.forName("oracle.jdbc.driver.OracleDri
                                        ver");
```

2) Create the connection object

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

Syntax of getConnection() method

1) public static Connection getConnection(String url)throws SQLException

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

Example to establish connection with the Oracle database Connection

```
con=DriverManager.getConnection( "jdbc:oracle:thin:@loca

lhost:1521:xe","system","password");
```

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.

Syntax of createStatement() method

```
public Statement createStatement()throws
                                SQLException
```

Example to create the statement object

```
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.

Syntax of executeQuery() method

```
public ResultSet executeQuery(String sql)throws SQLException
```

Example to execute query

```
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.

Syntax of close() method

public void close()throws

SQLException **Example to**

close connection

con.close();

3.4.1 Java Database Connectivity with Oracle

To connect java application with the oracle database, we need to follow 5 following steps. In this example, we are using Oracle 10g as the database. So we need to know following information for the oracle database:

- **Driver class:** The driver class for the oracle database is oracle.jdbc.driver.OracleDriver.
- **Connection URL:** The connection URL for the oracle10G database is jdbc:oracle:thin:@localhost:1521:xe where jdbc is the API, oracle is the database, thin is the driver, localhost is the server name on which oracle is running, we may also use IP address, 1521 is the port number and XE is the Oracle service name. You may get all these information from the tnsnames.ora file.
- **Username:** The default username for the oracle database is system.
- **Password:** It is the password given by the user at the time of installing the oracle database.

Create a Table

Before establishing connection, let's first create a table in oracle database. Following is the SQL query to create a table.

create table emp(id number(10),name varchar2(40),age number(3));

Example to Connect Java Application with Oracle database

In this example, we are connecting to an Oracle database and getting data from emp table. Here, system and oracle are the username and password of the Oracle database.

imp

ort

java

.sql.

*;

clas

s

```
Ora
cleC
on{
public static void
main(String args[]){
try{
//step1 load the driver class
Class.forName("oracle.jdbc.driver.Oracle
Driver");
  //step2 create  the connection
object Connection
con=DriverManager.getConnec
tion(
"jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
  //step3 create the
statement object
Statement
stmt=con.createStateme
nt();
  //step4 execute query
ResultSet rs=stmt.executeQuery("select *
from emp"); while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
  //step5 close the
connection object
con.close();
  }catch(Exception e){ System.out.println(e);}
  }
```

}

Two ways to load the jar file:

1. paste the ojdbc14.jar file in jre/lib/ext folder
2. set classpath

1) paste the ojdbc14.jar file in JRE/lib/ext folder:

Firstly, search the ojdbc14.jar file then go to JRE/lib/ext folder and paste the jar file here.

2) set classpath:

There are two ways to set the classpath:

- temporary
- permanent

How to set the temporary classpath:

Firstly, search the ojdbc14.jar file then open command prompt and write:

1. C:>set classpath=c:\folder\ojdbc14.jar;.;

How to set the permanent classpath:

Go to environment variable then click on new tab. In variable name write **classpath** and in variable value paste the path to ojdbc14.jar by appending ojdbc14.jar;.; as C:\oraclexe\app\oracle\product\10.2.0\server\jdbc\lib\ojdbc14.jar;.;

3.4.2 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 informations for the mysql database:

1. **Driver class:** The driver class for the mysql database is **com.mysql.jdbc.Driver**.
2. **Connection URL:** The connection URL for the mysql database is **jdbc:mysql://localhost:3306/sonoo** 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 sonoo is the database name. We may use any database, in such case, we need to replace the sonoo with our database name.
3. **Username:** The default username for the mysql database is **root**.
4. **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.

Let's first create a table in the mysql database, but before creating table, we need to create database first.

1. create database sonoo;

2. use sonoo;
3. create table emp(id **int**(10),name

varchar(40),age **int**(3)); **Example to Connect**

Java Application with mysql database

In this example, soon is the database name, root is the username and password both.

```
1. import java.sql.*;
2. class MysqlCon{
3. public static void main(String args[]){
4. try{
5. Class.forName("com.mysql.jdbc.Driver");
6. Connection con=DriverManager.getConnection(
7. "jdbc:mysql://localhost:3306/sonoo","root","root");
8. //here sonoo is database name, root is username and password
9. Statement stmt=con.createStatement();
10. ResultSet rs=stmt.executeQuery("select * from emp");
11. while(rs.next())
12. System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
13. con.close();
14. }catch(Exception e){ System.out.println(e);}
15. }
16. }
```

The above example will fetch all the records of emp table.

To connect java application with the mysql database, **mysqlconnector.jar** file is required to be loaded.

download the jar file

mysql-connector.jar Two

ways to load the jar file:

1. Paste the mysqlconnector.jar file in jre/lib/ext folder
2. Set classpath

1) Paste the mysqlconnector.jar file in JRE/lib/ext folder:

Download the mysqlconnector.jar file. Go to jre/lib/ext folder and paste the jar file here.

2) Set classpath:

There are two ways to set the classpath:

- temporary
- permanent

How to set the temporary classpathopen command prompt and write:

1. C:>set classpath=c:\folder\mysql-connector-java-5.0.8-bin.jar;.;

How to set the permanent 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;.;