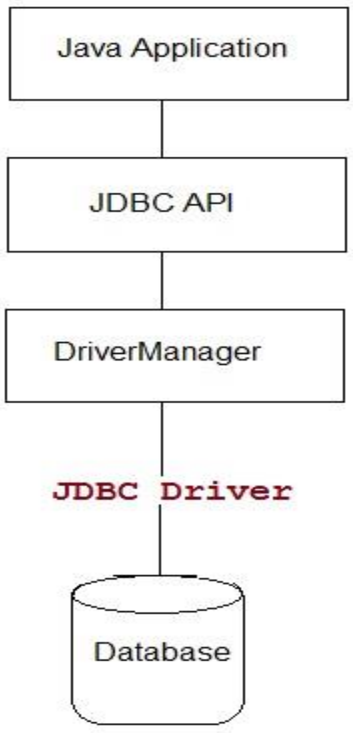
Introduction to JDBC

Java Database Connectivity(JDBC) is an Application Programming Interface(API) used to connect Java application with Database. JDBC is used to interact with various type of Database such as Oracle, MS Access, My SQL and SQL Server. JDBC can also be defined as the platform-independent interface between a relational database and Java programming. It allows java program to execute SQL statement and retrieve result from database.



JDBC Driver

JDBC Driver is required to process SQL requests and generate result. The following are the different types of driver available in JDBC.

Type-1 Driver or JDBC-ODBC bridge

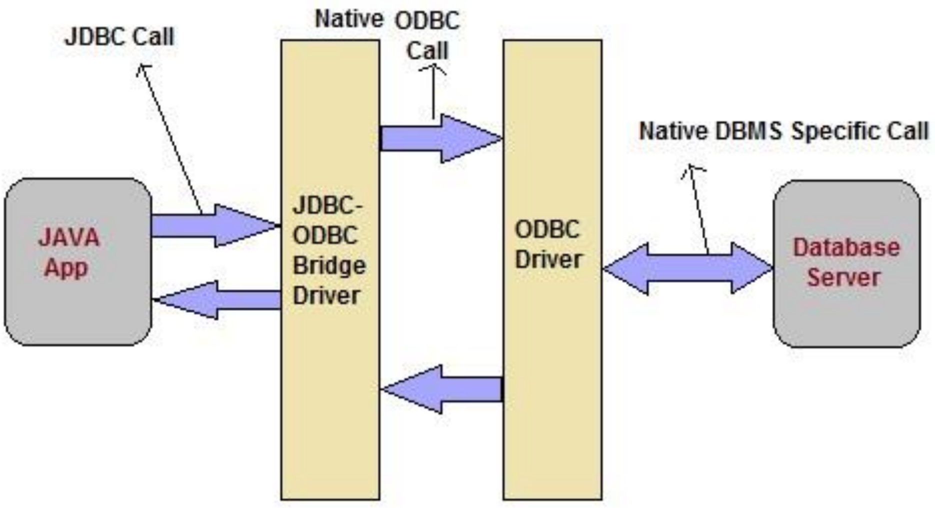
Type-2 Driver or Native API Partly Java Driver

Type-3 Driver or Network Protocol Driver

Type-4 Driver or Thin Driver

JDBC-ODBC bridge

Type-1 Driver act as a bridge between JDBC and other database connectivity mechanism(ODBC). This driver converts JDBC calls into ODBC calls and redirects the request to the ODBC driver.



Advantage

Easy to use

Allow easy connectivity to all database supported by the ODBC Driver.

Disadvantage

Slow execution time

Dependent on ODBC Driver.

Uses Java Native Interface(JNI) to make ODBC call.

Steps to connect a Java Application to Database

The following 5 steps are the basic steps involve in connecting a Java application with Database using JDBC.

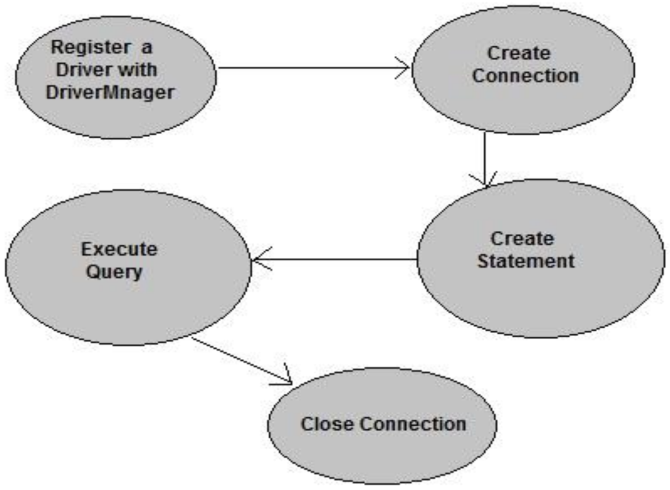
Register the Driver

Create a Connection

Create SQL Statement

Execute SQL Statement

Closing the connection



Register the Driver

Class.forName() is used to load the driver class explicitly.

Example to register with JDBC-ODBC Driver

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Create a Connection

getConnection() method of DriverManager class is used to create a connection.

Syntax

getConnection(String url)

getConnection(String url, String username, String password)

getConnection(String url, Properties info)

Example establish connection with Oracle Driver

Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","username","password")

Create SQL Statement

createStatement() method is invoked on current Connection object to create a SQL Statement.

Syntax

public Statement createStatement() throws SQLException

Example to create a SQL statement

Statement s=con.createStatement();

Execute SQL Statement

executeQuery() method of Statement interface is used to execute SQL statements.

Syntax

public ResultSet executeQuery(String query) throws SQLException

Example to execute a SQL statement

```
ResultSet rs=s.executeQuery("select * from user");
  while(rs.next())
  {
   System.out.println(rs.getString(1)+" "+rs.getString(2));
 }
```

Closing the connection

After executing SQL statement you need to close the connection and release the session. The close() method of Connection interface is used to close the connection.

Syntax

public void close() throws SQLException

Example of closing a connection

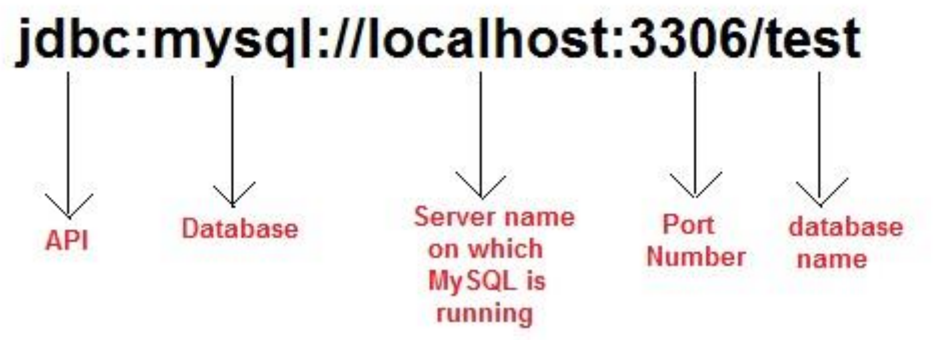
con.close();

Connecting to MySQL Database using Thin Driver

To connect a Java application with MySQL database using Thin Driver. You need to follow the following steps

Load Driver Class: The Driver Class for MySQL database is com.mysql.jdbc.Driver and Class.forName("com.mysql.jdbc.Driver") method is used to load the driver class for MySQL database.

Create Connection: For creating a connection you will need a Connection URL. The Connection URL for MySQL is


jdbc:mysql://localhost:3306/test
API
Database
Server name on which MySQL is running
Port Number
database name

You will also require Username and Password of your MySQL Database Server for creating connection.

Loading jar file: To connect your java application with MySQL, you will also need to load mysql-connector.jar file. This file can be loaded into 2 ways.

Copy the jar file into C:\Program Files\Java\jre7\lib\ext folder.

or,

Set it into classpath. For more detail see how to set classpath

Download mysql-connector.jar file

Example

Create a table in MySQL Database

create table Student(sid int(10),name varchar(20));

Insert some record into the table

insert into Student values(102,'adam');

insert into Student values(103,'abhi');

Accessing record from Student table in Java application

import java.sql.*;

class Test

```
{
public static void main(String []args)
{
try{
//Loading driver
Class.forName("com.mysql.jdbc.Driver");
//creating connection
Connection con = DriverManager.getConnection
                ("jdbc:mysql:/ /localhost:3306/test","username","password");
Statement s = con.createStatement();   //creating statement
ResultSet rs = s.executeQuery("select * from Student");   //executing statement
while(rs.next()){
System.out.println(rs.getInt(1)+" "+rs.getString(2));
}
con.close();  //closing connection
}catch(Exception e){
e.printStacktrace();
}
}
}
```

```
102 adam
103 abhi
```

<u>Inserting record into a table using java application</u>

```
import java.sql.*;
```

```
class Test

{

public static void main(String []args)

{

try{

//Loading driver

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

//creating connection

Connection con = DriverManager.getConnection

                ("jdbc:mysql:/ /localhost:3306/test","username","password");

PreparedStatement pst=con.prepareStatement("insert into Student values(?,?)");

         pst.setInt(1,104);

         pst.setString(2,"Alex");

               pst.executeUpdate();

con.close();  //closing connection

}catch(Exception e){

e.printStacktrace();

}

}

}
```