JDBC DRIVER

|  |
| --- |
| JDBC Driver is a software component that enables java application to interact with the database.  There are 4 types of JDBC drivers:   1. JDBC-ODBC bridge driver 2. Native-API driver (partially java driver) 3. Network Protocol driver (fully java driver) 4. Thin driver (fully java driver) |

### **JDBC-ODBC bridge driver**

|  |
| --- |
| The JDBC-ODBC bridge driver uses ODBC driver to connect to the database.  The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls.  **This is now discouraged because of thin driver.** |



#### **In Java 8, the JDBC-ODBC Bridge has been removed.**

Oracle does not support the JDBC-ODBC Bridge from Java 8. Oracle recommends that you use JDBC drivers provided by the vendor of your database instead of the JDBC-ODBC Bridge.

### **Advantages:**

* easy to use.
* can be easily connected to any database.

### **Disadvantages:**

* Performance degraded because JDBC method call is converted into the ODBC function calls.
* The ODBC driver needs to be installed on the client machine.

### **Native-API driver**

|  |
| --- |
| The Native API driver uses the client-side libraries of the database.  The driver converts JDBC method calls into native calls of the database API.  It is not written entirely in java.  Native-API driver **Advantage:**  * performance upgraded than JDBC-ODBC bridge driver.  **Disadvantage:**  * The Native driver needs to be installed on the each client machine. * The Vendor client library needs to be installed on client machine.  **3) Network Protocol driver** The Network Protocol driver uses middleware (application server) that converts JDBC calls  directly or indirectly into the vendor-specific database protocol.  It is fully written in java. |



### **Advantage:**

* No client-side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.

### **Disadvantages:**

* Network support is required on client machine.
* Requires database-specific coding to be done in the middle tier.
* Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.

### **Thin driver**

|  |
| --- |
| The thin driver converts JDBC calls directly into the vendor-specific database protocol.  That is why it is known as thin driver. It is fully written in Java language. |



### **Advantage:**

* Better performance than all other drivers.
* No software is required at client side or server side.

### **Disadvantage:**

* Drivers depend on the Database.

**Java Database Connectivity with 5 Steps**

5 Steps to connect to the database in java

* 1. Register the driver class
  2. Create the connection object
  3. Create the Statement object
  4. Execute the query
  5. Close the connection object

### **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**

1. **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.

1. Class.forName("oracle.jdbc.driver.OracleDriver");

### **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**

1. Connection con=DriverManager.getConnection(
2. "jdbc:oracle:thin:@localhost: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**

1. **public** Statement createStatement()**throws** SQLException

### **Example to create the statement object**

1. 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**

1. **public** ResultSet executeQuery(String sql)**throws** SQLException

### **Example to execute query**

1. ResultSet rs=stmt.executeQuery("select \* from emp");
3. **while**(rs.next()){
4. System.out.println(rs.getInt(1)+" "+rs.getString(2));
5. }

### **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**

1. **public** **void** close()**throws** SQLException

### **Example to close connection**

1. con.close();

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/testdb** 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 testdb is the database name.
3. We may use any database, in such case, we need to replace the testdb with our database name.
4. **Username:**The default username for the mysql database is **root**.
5. **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.

**RESULTSET**

1. **Scrollability:**Determines whether you can move back and forth in the ResultSet
   * **TYPE\_FORWARD\_ONLY:**Can only move forward through the rows
   * **TYPE\_SCROLL\_INSENSITIVE:**Can move forward and backward but changes are not reflect ResultSet
   * **TYPE\_SCROLL\_SENSITIVE:**Can move forward and backward but changes are affect the ResultSet
2. **Concurrency:**Determines whether you can update the ResultSet
   * **CONCUR\_READ\_ONLY:**Can only read data
   * **CONCUR\_UPDATABLE:**Allows updates to the ResultSet