**Why Should We Use JDBC:**

Before JDBC, ODBC API was the database API to connect and execute the query with the database.  It is Open Database Connectivity (ODBC) provided by **Microsoft.** But, ODBC API uses ODBC driver which is **written in C language** (i.e. platform **dependent and unsecured**). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (**written in Java language**).

JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

What is JDBC Driver?

**JDBC Driver is a software components that enables java application to interact with databases.**

* JDBC-ODBC Bridge Driver,
* Native Driver(partially java driver)
* Network Protocol Driver(fully java driver)
* Thin Driver(fully java driver)

The **java.sql** package contains **classes and interfaces** for JDBC API.A list of popular ***classes* of JDBC API** are given below:

* DriverManager class
* Blob class
* Clob class
* Types class

A list of popular *interfaces* of JDBC API are given below:

* Driver interface
* Connection interface
* Statement interface
* PreparedStatement interface
* CallableStatement interface
* ResultSet interface
* ResultSetMetaData interface
* DatabaseMetaData interface

**The current version of JDBC is 4.3.** It is the stable release since 21st September, **2017**. It is based on the X/Open SQL Call Level Interface.

## **What is API**

API (Application programming interface) is a document that contains a description of all the features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other.

### 1) JDBC-ODBC bridge 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.The ODBC driver needs to be installed on the client machine. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls.

### 2) Native-API driver

It is not written entirely in java. **The Native driver needs to be installed on the each client machine.**  The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java.

### 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.Requires database-specific coding to be done in the middle tier.

### 4) 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. Better performance than all other drivers.No software is required at client side or server side.(Disadvatage) Drivers depend on the Database.

# **Java Database Connectivity with 5 Steps**

### 1) Loading Driver(**forName())**

The **forName()** method is used to dynamically load the driver class.

#### **Note: Since JDBC 4.0, explicitly registering the driver is optional. We just need to put vender's Jar in the classpath, and then JDBC driver manager can detect and load the driver automatically.**

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

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

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

### 2) Established Connection(**getConnection()** )

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

Example for Establishing Connection with Oracle Database.

Connection con=

DriverManager.getConnection(  "jdbc:oracle:thin:@localhost:1521:xe","system","password");

DriverManager.getConnection("jdbc:mysql://localhost:3306/db1","root","")

### 3) Create the Statement object (createStatement())

The **createStatement() method of Connection interface** is used to create statement. The object of statement is responsible to execute queries with the database.

Example: Statement stmt=con.createStatement();

### Why use PreparedStatement?

**Improves performance**: The performance of the application will be faster if you use PreparedStatement interface because **query is compiled only once.** The PreparedStatement interface is a **subinterface of Statement**. **It is used to execute parameterized query.**

**4) Execute the query (**executeQuery())

|  |
| --- |
| 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. |

Example:

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

**while**(rs.next()){

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

}

### 5) Close the connection object

con.close();

We can Use for all operation e.g CREATE TABLE etc in **Oracle, mysql, or DB2 database**. So you can learn this topic only for knowledge.

Msql require msqlConnector driver for database access.There some difference in ORAClE,MYSQL

What is MetaData (Ans: data about data, we can get further information from the data.)

**1. ResultSetMetaData:**

ResultSetMetaData is an interface in java.sql package of JDBC API which is used to get the metadata about a ResultSet object. Whenever you query the database using SELECT statement, the result will be stored in a ResultSet object. Every ResultSet object is associated with one ResultSetMetaData object. This object will have all the meta data about a ResultSet object like schema name, table name, number of columns, column name, datatype of a column etc.

The **getMetaData()** method of ResultSet interface.

Example:

{

ResultSet rs=ps.executeQuery();

**ResultSetMetaData rsmd=rs.getMetaData();**

System.out.println("Total columns: "+rsmd.getColumnCount());

("Column Name of 1st column: "+rsmd.getColumnName(1));

("Column Type Name of 1st column: "+rsmd.getColumnTypeame(1));

  con.close();

}

**2. DatabaseMetaData:**

DatabaseMetaData interface provides methods to get meta data of a database such as name of total number of tables, name of total number of views, database product name, database product version, driver name, etc.

Example:

{ **DatabaseMetaData dbmd=con.getMetaData()**;

System.out.println("Driver Name: "+dbmd.getDriverName());

System.out.println("Driver Version: "+dbmd.getDriverVersion());

System.out.println("UserName: "+dbmd.getUserName());

System.out.println("Database Product Name: "+dbmd.getDatabaseProductName();

System.out.println("Database Product Version: "+dbmd.getDatabaseProductVersion());

con.close();

}

# **Example to store image in Oracle database**

<https://www.javatpoint.com/storing-image-in-oracle-database>

# **Example to retrieve image from Oracle database**

<https://www.javatpoint.com/retrieving-image-from-oracle-database>

# **Example to store file in Oracle database:**

<https://www.javatpoint.com/storing-file-in-oracle-database>

# **Example to retrieve file from Oracle database:**

<https://www.javatpoint.com/retrieving-file-from-oracle-database>

# **JDBC RowSet**

The instance of **RowSet** is the java bean component because it has properties and java bean notification mechanism. It is introduced since JDK 5.

It is the wrapper of ResultSet. It holds tabular data like ResultSet but it is easy and flexible to use.

The implementation classes of RowSet interface are as follows:

* JdbcRowSet
* CachedRowSet
* WebRowSet
* JoinRowSet
* FilteredRowSet

Let's see how to create and execute RowSet.

1. JdbcRowSet rowSet = RowSetProvider.newFactory().createJdbcRowSet();
2. rowSet.setUrl("jdbc:oracle:thin:@localhost:1521:xe");
3. rowSet.setUsername("system");
4. rowSet.setPassword("oracle");
6. rowSet.setCommand("select \* from emp400");
7. rowSet.execute();

There are many new features that have been added in java. There are major enhancement made in Java5, Java6, Java7 and Java8 like **auto-boxing**, **generics**, **var-args**, **java annotations**, **enum**, **premain method** , **lambda expressions**, **functional interface**, **method references** etc.

<https://www.javatpoint.com/New-features-in-java>