Java Platform

Java platform consist of JVM and implementation of standard libraries.

To develop applications for different types of systems,there are now three types of

Platforms.

1. **J2SE**
   Java 2 platform – standard edition
   To build applications lime applet and client applications.

2. **J2EE**
   Java 2 platform – Enterprise edition
   Used for building server side applications.
3. **J2ME**
   Java platform – Micro edition
   It enables the building of java applications for 'micro-devices' that include hand – held devices
   like phone
   ==

Java API (Application Programming Interface)
The Java API is a set of .class files available at run time.
A java API is a library that contains methods,which offers way of accessing resources on the computer.
Any machine that have JVM has the java API.

==

Components of J2EE
J2EE specifies both the service APIs for building the application infrastructure for managing the application.
1) JDBC
2) Servlets
3) JSP (Java Server Pages)
4) Java mail
5) JNDI

JDBC(Java Database Connectivity)

JDBC API

The java database connectivity application programming interface is a set of specifications that defines how a Java Program can communicate with database.
The JDBC API is composed of methods defined by a set of class and interfaces that enable java applications to communicate with database management system (DBMS) or relational database management system (RDBMS).

A java program interact with a database. The program invokes the methods of the JDBC API. The JDBC API then calls the JDBC Driver and submits the queries to it.

JDBC-

It enable java application to connect to and access databases. In JDBC ,the java classes are designed to provide access to any database. In Java.sql package, JDBC consist of set of class and interface files.0

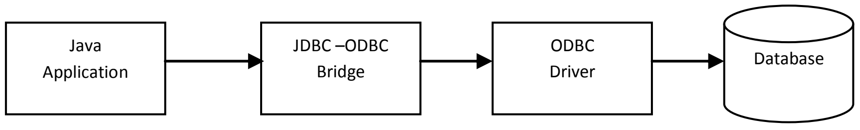
JDBC Drivers

A JDBC driver enables java application to connect to a database to particular DBMS /RDBMS , and allows programmers to manipulate that database, using JDBC API.

Type1 $\longrightarrow$	JDBC-ODBC bridge driver
Type2 $\longrightarrow$	Native – API / partly java driver
Type3 $\longrightarrow$	Net – protocol / All java drivers
Type4 $\longrightarrow$	Pure java client to server

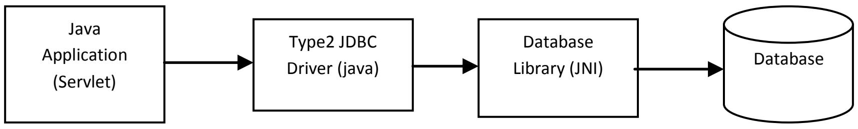
1) JDBC-ODBC bridge driver

The JDBC-ODBC bridge driver converts all JDBC calls into ODBC calls and sends them to the ODBC driver.The ODBC driver then forwards the calls to the database server.
There are some database, such as MSAccess that do not provide a JDBC driver. Instead they provide a Open Database Connectivity (ODBC) drivers.
This driver is mixture of Java code and native code.

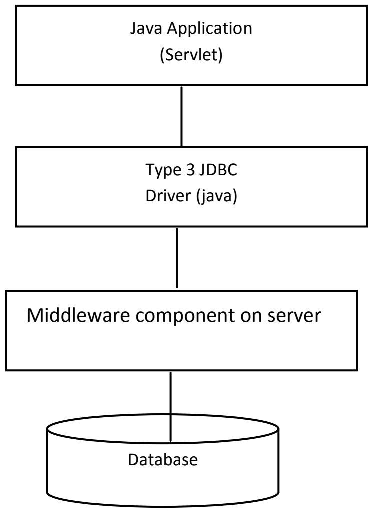


2) **Java to Native API / partly java driver**
Type 2 driver use the Java Native Interface (JNI) to make calls to a local database library API.
The driver converts JDBC calls into native calls (database specific calls)
This driver communicate directly with the database server. It requires some native code to connect to the database.

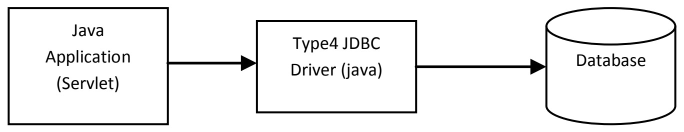
Type2 driver requires native database client libraries to be installed and configured on the client machine.
Type2 driver usually faster than the type1 driver



3) **Java to Network protocol / All java drivers**
Pure java client to server driver take JDBC requests and translates them into a network protocol that is not a database specific.
These requests are sent to a server, which translates the database request into database specific protocol.
Type3 drivers do not requires native database libraries on the client and can connect to many different database on the back end.


Java Application (Servlet)
Type 3 JDBC Driver (java)
Middleware component on server
Database

4) **Pure java client to server**
Pure java drivers implements database specific protocols, so that Java platform can connect directly to a database specific.



Steps in program

1. Import necessary packages
   Import java.sql package

2. Load the driver
   public static void forName(String) method of class Class.
   e.g. Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

3. **Connect to the database**
   public static Connection getConnection(String url) method of class DriverManager used to establish connection
   e.g. Connection con= DriverManager.getConnection("jdbc:odbc:DSN");
   URL:
   Jdbc
   Indicates that JDBC is being used to establish the database connection.

   Odbc (Sub protocol)
   Is the name of the database the developer wants to connect to .
   (e.g. odbc,oracle,mysql and so on.)

   DSN (Sub name)
   Is logical name which provides additional information on how and where to connect. DSN is a data structure contains information in the database which is used by driver to connect to database..

4. Create statement
   Types of statement
   a) Statement
   b) PreparedStatement
   c) CallableStatement

5. Execute the statement
6. Retrieve / Iterate the ResultSet
7. Close the statement and the connection
8.

==

Important classes and interfaces of java.sql packages

JDBC

java.sql package

DriverManager

public static Connection getConnection(String url) throws SQLException

Attempts to establish a connection to the given database URL. The DriverManager attempts to select an appropriate driver from the set of registered JDBC drivers.

e.g. String url=”jdbc:odbc:DSN”

==

1) Connection interface

connection (session) with a specific database. SQL statements are executed and results are returned within the context of a connection.

1. Statement createStatement()
2. PreparedStatement prepareStatement(String sql)
3. CallableStatement prepareCall(String sql)
4. DatabaseMetaData getMetaData()

   All above methods throws SQLException

==

2) Statement interface

Mehtods of it are used to execute sqlstatement.
Mehtods:-

1. ResultSet executeQuery(String sql)

   Executes the given DDL SQL statement, which returns a single ResultSet object.

2. int executeUpdate(String sql)
   Executes the given DML SQL statement, which may be an <code>INSERT</code>, UPDATE or DELETE statement or an SQL statement that returns nothing, such as an SQL DDL statement.

===

3. void execute()

4. ResultSet getResultSet()

   All above methods throws SQLException

Retrieves the current result as a ResultSet object. This method should be called only once per result.

==

3) ResultSet interface

ResultSet object maintains a cursor pointing to its current row of data. Initially the cursor is positioned before the first row. The next method moves the cursor to the next row, and because it returns false when there are no more rows in the ResultSet object, it can be used in a while loop to iterate through the result set.

Methods:

boolean next() ,boolean first(), boolean previous() , boolean last()

String getString(int columnIndex)

int getInt(int columnIndex) ,

float getFloat(int columnIndex)

String getString(String columnLabel),

int getInt(String columnLabel),

float getFloat(String columnLabel)

boolean getBoolean(int columnIndex) etc.

All above methods throws SQLException

==

4) PreparedStatement interface

An object that represents a precompiled SQL statement. A SQL statement is precompiled and stored in a PreparedStatement object. This object can then be used to efficiently execute this statement multiple times.

Methods of PreparedStatement are used to execute dynamic sql statement.In the normal sql statement , values are not known at the time of creation of query. In the dynamic sql statement values are known at the time of creation of query.

PreparedStatement extends Statement

Methods:

void setBoolean(int parameterIndex, boolean x)

void setByte(int parameterIndex, byte x)

void setInt(int parameterIndex, int x)

void setLong(int parameterIndex, long x)

void setFloat(int parameterIndex, float x)

void setDouble(int parameterIndex, double x)

void setString(int parameterIndex, String x)

void setBytes(int parameterIndex, byte x[])

void setDate(int parameterIndex, java.sql.Date x)

ResultSetMetaData getMetaData()

Above all methods throws SQLException

===

5) CallableStatement interface

The interface used to execute SQL stored procedures. The JDBC API provides a stored procedure SQL escape syntax that allows stored procedures to be called in a standard way for all RDBMSs.

CallableStatement extends PreparedStatement

Registers the OUT parameter in ordinal position parameterIndex to the JDBC type

sqlType All OUT parameters must be registered before a stored procedure is executed.

Methods:

void registerOutParameter(int parameterIndex, int sqlType) throws SQLException

void setInt(String parameterName, int x)

void setLong(String parameterName, long x)

void setFloat(String parameterName, float x)

void setDouble(String parameterName, double x)

void setString(String parameterName, String x)

void setDate(String parameterName, java.sql.Date x)

String getString(String parameterName)

boolean getBoolean(int parameterIndex)

byte getByte(int parameterIndex)

short getShort(int parameterIndex)

int getInt(int parameterIndex)

long getLong(int parameterIndex)

float getFloat(int parameterIndex)

double getDouble(int parameterIndex)

byte[] getBytes(int parameterIndex)

java.sql.Date getDate(int parameterIndex)

Above all methods throws SQLException

---

6) ResultSetMetaData

An object that can be used to get information about the types and properties of the columns in a ResultSet object.
Method of ResultSet returns the object of ResultSetMetaData
ResultSetMetaData getMetaData()

Methods of ResultSetMetaData

Returns the number of columns in this ResultSet object.

int getColumnCount()

String getColumnName(int)

All above methods throws SQLException

----

7) **DatabaseMetaData interface**

A user for this interface is commonly a tool that needs to discover how to deal with the underlying DBMS.

ResultSet getTables(String catalog, String schemaPattern,

String tableNamePattern, String *types*[]) throws SQLException;

Retrieves a description of the tables available in the given catalog.

Only table descriptions matching the catalog, schema, table name and type criteria are returned. They are ordered by TABLE_TYPE ,TABLE_CAT,

TABLE_SCHEM and TABLE_NAME

===

Difference between:-

i) Statement and PreparedStatement
ii) Statement and CallableStatement
iii) PreparedStatement and CallableStatement

Sr. No.	Statement	PreparedStatement
1.	Methods of it are used to execute DDL and DML Sql Statement	Methods of it are used to execute DML Sql Statement
2.	An object that do not represents a precompiled SQL statement.	An object that represents a precompiled SQL statement.
3.	Staement interface is extends by PreparedStatement	PreparedStatement interface is extends to Statement
4.	Methods:- a) ResultSet executeQuery(String sql) b) int executeUpdate(String sql) c) void execute() d) ResultSetMetaData getMetaData()	Methods:- a) Methods of Statement interface b) void setBoolean(int parameterIndex, boolean x) c) void setByte(int parameterIndex, byte x) d) void setInt(int parameterIndex, int x) g) void setLong(int parameterIndex, long x) h) void setFloat(int parameterIndex, float x) i) void setDouble(int parameterIndex, double x) j) void setString(int parameterIndex, String x) k) void setBytes(int parameterIndex, byte x[]) l) void setDate(int parameterIndex, java.sql.Date x)
5.	How to get Statement [Method of Connection] Statement createStatement()	How to get Statement [Method of Connection] PreparedStatement prepareStatement(String sql)

Sr. No.	Statement	CallableStatement
1.	Methods of it are used to execute DDL and DML Sql Statement	Methods of it are used to execute store procedure.
2.	Statement interface is extends by PreparedStatement and PreparedStatement extended by CallableStatement	CallableStatement extends to PreparedStatement and PreparedStatement interface is extends to Statement
3.	Methods:- e) ResultSet executeQuery(String sql) f) int executeUpdate(String sql) g) void execute() h) ResultSetMetaData getMetaData()	Methods:- e) Methods of Statement and PreparedStatement interface f) void registerOutParameter(int parameterIndex, int sqlType) throws SQLException g) void setInt(String parameterName, int x) h) void setLong(String parameterName, long x) i) void setFloat(String parameterName, float x) j) void setDouble(String parameterName, double x) k) void setString(String parameterName, String x) l) void setDate(String parameterName, java.sql.Date x) m) String getString(String parameterName) n) boolean getBoolean(int parameterIndex) o) byte getByte(int parameterIndex) p) short getShort(int parameterIndex) q) int getInt(int parameterIndex)
4	How to get Statement [Method of Connection] Statement createStatement()	How to get CallableStatement[Method of Connection] CallableStatement prepareCall(String sql)

Sr. No.	**PreparedStatement**	**CallableStatement**
1.	Methods of it are used to execute DML Sql Statement	Methods of it are used to execute store procedure.
2.	An object that represents a precompiled SQL statement.	An object that do not represents a precompiled SQL statement
3.	PreparedStatement interface is extends to Statement	CallableStatement extends to PreparedStatement and PreparedStatement interface is extends to Statement
4.	Methods:- r) Methods of Statement interface s) void setBoolean(int parameterIndex, boolean x) t) void setByte(int parameterIndex, byte x) u) void setInt(int parameterIndex, int x) g) void setLong(int parameterIndex, long x) h) void setFloat(int parameterIndex, float x) ii) void setDouble(int parameterIndex, double x) j) void setString(int parameterIndex, String x) k) void setBytes(int parameterIndex, byte x[]) l) void setDate(int parameterIndex, java.sql.Date x)	Methods:- a) Methods of Statement and PreparedStatement interface b) void registerOutParameter(int parameterIndex, int sqlType) throws SQLException c) void setInt(String parameterName, int x) d) void setLong(String parameterName, long x) e) void setFloat(String parameterName, float x) f) void setDouble(String parameterName, double x) g) void setString(String parameterName, String x) h) void setDate(String parameterName, java.sql.Date x) i) String getString(String parameterName) j) short getShort(int parameterIndex) k) int getInt(int parameterIndex)
	How to get Statement [Method of Connection] PreparedStatement prepareStatement(String sql)	How to get CallableStatement[Method of Connection] CallableStatement prepareCall(String sql)

ResultSet static type constant

- **TYPE_FORWORD_ ONLY**

Specifies that a ResultSet's cursor can move only in the forward direction (i.e. from frist record to the last in the ResultSet)

- **TYPE_SCROLL_SENSITIVE**

Specifies that ResultSet can scroll in the either direction and the changes made to the ResultSet during ResultSet processing are **not** reflected in the ResulltSet unless the program queries the data again.

- **TYPE_SCROLL_INSENSITIVE**

Specifies that ResultSet can scroll in the either direction and the changes made to the ResultSet during ResultSet processing are reflected immediately in the ResulltSet.

- **CONCUR_READ_ONLY**

Specifies that ResultSet cannot be updated (i.e. changes to the ResulltSet contents cannot be reflected the database with ResulltSets update method)

- **CONCUR_UPDATABLE**

Specifies that ResultSet can be updated (i.e. changes to the ResulltSet contents can be reflected the database with ResulltSets update method)

e.g.

```
Statement stmt = con.createStatement
                        (
                          ResultSet.TYPE_SCROLL_INSENSITIVE,
                          ResultSet. CONCUR_READ_ONLY
                        );
```

==