(1). What is java JDBC ?

🡪**JDBC** stands for **Java** Database Connectivity. **JDBC** is a **Java** API to connect and execute the query with the database.

🡪JAVA JDBC API uses java drivers to connect database.

(2). What is java drivers ?

🡪 JDBC Drivers is software components that enable java application to interact with the database

🡪There are four types of Drivers

1. JDBC-ODBC bridge drivers
2. Native drivers
3. Network protocol drivers
4. Thin drivers

(3). What is JDBC-ODBC bridge drivers ?

🡪 The JDBC-ODBC bridge driver converts JDBC methods calls in ODBC function calls , this is discouraged now.

Advantage

1. Easy to use
2. Can be easily connected with any database

Disadvantage

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

🡪In java 8 , the jdbc-odbc bridge has been removed

(4). What is Native drivers ?

🡪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.

Advantage

1. performance upgraded than JDBC-ODBC bridge driver.

Disadvantage

1. The Native driver needs to be installed on the each client machine.
2. The Vendor client library needs to be installed on client machine.

(5). What is Network protocol drivers ?

🡪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

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

Disadvantage

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

(6). What is Thin Drivers ?

🡪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

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

Disadvantage

1. Drivers depend on the Database.

(7). How to connect java application with data base ?

🡪 java database connectivity with five steps.

1. Register the driver
2. Get connection
3. Create statement
4. Execute query
5. Close connection

(8).Register the drivers class

🡪 the method forName() of Class class is used to register the drivers.

Syntax of forName() methods:-

🡪public static void forName(String className) throws ClassNotFoundException

Notes:- since JDBC 4.0 , explicitly registering the drivers 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.

Example:-

🡪Class.forName(“com.jdbc.mysql.Driver”); // registering driver of mysql

(9).Get connection

🡪 we need Connection object to establish connection between database and java application

🡪the getConnection() method of DriverManager class is used to obtain connection object.

Syntax:-

🡪public static Connection getConnection(String url) throws SQLException;

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

Example:-

🡪Connection con=DriverManager.getConnection

(“jdbc:mysql://localhost:3306/database\_name”,”username”,”password”);

(10). Create Statement

🡪the createStatement() method of connection interface return object of statement.

🡪Object of statement responsible for executing query

Syntax:-

🡪public Statement createStatement()throws SQLException

Example:-

Statement stmt = con.createStatement(); // con is reference variable of Connection interface

ResultSet set = stmt.executeQuery(“select \* from table1”); //storing data in set(object of ResultSet)

While(set.next()){

System.out.println(set.getString(2));

}

(11). Close connection

🡪By closing connection object statement and ResultSet will be closed automatically.

🡪he close() method of Connection interface is used to close the connection.

Syntax:-

🡪public void close()throws SQLException

Example:-

🡪con.close(); // con is reference object of connection

## Let’s learn in detail

**(1). DriverManager Class**

🡪 The DriverManager class acts as an interface between user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver

Methods:

1. public static void registerDriver(Driver driver);
   * is used to register the given driver with DriverManager.
2. public static void deregisterDriver(Driver driver);
   * Is used to deregister the given driver with Driver manager
3. public static Connection getConnection(String url);
   * is used to establish the connection with the specified url.
4. Public static Connection getConnection(String url, string username, String password);
   * is used to establish the connection with the specified url, username and password.

**(2). Connection interface**

🡪 when we get reference variable of Connection interface from getconnection(…) method of DriverManager

🡪Now we understand , what is functionality of this interface

Methods:

1. public Statement createStatement();
   * return Statement object which is used to execute a query
2. public Statement createStatement(int RestultSetType, int ResultSetConcurrency );
   * ResultSet object with given type and concurrency
3. public void setAutoCommit(Boolean status);
   * by default , it is true
4. public void commit()
   * saves the changes made since the previous commit/rollback permanent
5. public void rollback()
   * undo all changes made after the previous commit permanent
6. public void close()
   * used to close Connection

**(3). Statement objects**

Source:- https://www.javatpoint.com/Statement-interface