JDBC

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

Jar: -

- ➔ Jar stands for Java archive.
- ➔ It is a zip file format, in this many files compressed into a single file.
- ➔ Jar contains .java, .class, .config files like .xml, .properties

There are Two types of jars.

→jar (Executable Jar)

→Runnable jar

1. **Executable jar: -**
   There is no class which contains main method.
   This is suitable to provide external library classes
2. **Runnable jar: -**

   There is a class which contains main method

   These suitable to execute the program without any other classes.

How to create a jar file: -

Step 1: - create a java project, create some packages and classes then write your code

Step 2: - Right click on project and then click on export

Step 3: - Later search jar and then select jar

Step 4: - Select a location where you want to store the jar file

Step 5: - Click on finish.

How to include jar files: -

We can include jar file into project in two ways.

1. External way
2. Internal way (recommended)

Adding jars external way: -

Step 1: - Right click on your project and select “build path” then select “configure build path”

Step 2: - Select “libraries” and then select class path later select add external jars

Step 3: - Select the jar file which you wanted to add to a project

Step 4: - click “finish” and then click “apply and close”

Adding jars Internal way: -

Step 1: - create a lib named folder inside the project

Step 2: - copy jar file and paste it into the lib folder

Step 3: - Right click on your project and select “build path” then select “configure build path”

Step 4: - Select “libraries” and then select class path later select add jars

Step 5: - Select the jar file which is present inside your lib folder

Step 6: - click “finish” and then click “apply and close”

(Right click on jar and then select build path and then select add to build path instead of Step 3, 4, 5, 6)

API: -

→ API stands for Application programming interface

→ API are used to connect one or more application/programs in a loosely coupled manner.

→ If API is developed by java, then API is given in the form of jar file.

→ API contains some classes and interfaces.

There are two types of API

→ I form of API

→ II form of API

I form	II form
• In this form implementation logic is given • As implementation logic is present “I form of API” is bulk in size • It always need updation if there is any changes at service provide side.	• There is no implementation logic • As it doesn’t contain implementation logic then are less in size. • It doesn’t update if there is any changes at service provider side.

Drivers: -

Drivers provides the implementation logic of API. Here the following are the 4 types of JDBC drivers.

1. Type-1 or JDBC-ODBC bridge driver.
2. Type-2 or Native-API driver
3. Type-3 or Network Protocol driver
4. Type-4 or Thin driver

JDBC-ODBC bridge driver: -

Type-1 driver or 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. Type-1 driver is also called Universal driver because it can be used to connect to any of the databases.

Advantages

- This driver software is built-in with JDK so no need to install separately.
- It is a database independent driver.

Disadvantages

- As a common driver is used in order to interact with different databases, the data transferred through this driver is not so secured.
- The ODBC bridge driver is needed to be installed in individual client machines.
- Type-1 driver isn't written in java, that's why it isn't a portable driver.

Native-API driver

The Native API driver uses the client -side libraries of the database. This driver converts JDBC method calls into native calls of the database API. In order to interact with different database, this driver needs their local API, that's why data transfer is much more secure as compared to type-1 driver. This driver is not fully written in Java that is why it is also called Partially Java driver.

Advantage

- Native-API driver gives better performance than JDBC-ODBC bridge driver.

Disadvantages

- Driver needs to be installed separately in individual client machines
- The Vendor client library needs to be installed on client machine.
- Type-2 driver isn't written in java, that's why it isn't a portable driver
- It is a database dependent driver.

Network Protocol Driver

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. Here all the database connectivity drivers are present in a single server, hence no need of individual client-side installation.

Advantages

- Type-3 drivers are fully written in Java, hence they are portable drivers.
- No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.
- Switch facility to switch over from one database to another database.

Disadvantages

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

Thin Driver: -

Type-4 driver is also called native protocol driver. This driver interact directly with database. It does not require any native database library, that is why it is also known as Thin Driver.

Advantages

- Does not require any native library and Middleware server, so no client-side or server-side installation.
- It is fully written in Java language; hence they are portable drivers.

Disadvantage

- If the database varies, then the driver will carry because it is database dependent.

JDBC: -

- JDBC stands for java data base connectivity.
- It contains two parts:
  - JDBC API
  - JDBC Driver

JDBC API

→ JDBC API is a specification given in the form of jar file

→ JDBC API helps to connect java program and dta base in a secure, organized and in a loosely coupled manner.

→ JDBC API is a II from of API

→ JDBC API is present in java.sql package.

JDBC Interfaces: -

Driver
Connection
Statement
PreparedStatement
CallableStatement
ResultSet
DatabaseMetaData
ResultSetMEtaData

Helper Classses: -

DriverManager

Steps of JDBC: -

- Load And Register the jdbc Driver
- Establish connection between java program and the data base.
- Create a platform to execute SQL queries.
- Execute the queries.
- Fetch the resultant dat(optional)
- Closing of costly resource.

Step 1: -Load and Register driver: -

- In this step we load and register the class which implements java.sql.Driver interface
- We can do this step in 2 ways
    - new keyword way
    - Class.forName() (Recommended)

New keyword way: -

➔ We create the object of driver class then we register with DriverManage.

➔ DriverManager provide a method which is used to register.

registerDriver(): -

→ It is a static method of DriverManager used to register driver class.

Public static void registerDriver(Driver t())

In mysql:

Java.sql.Driver p = new com.mysql.cj.jdbc.Driver();

Note:

New keyword way creates tight coupling between java program and data base, so this way of loading and registering is not recommended.

Class.forName way:

➔ In this way, we make use of forName method of class called as class

forName(): -

- It is a static method of class.
- It is used for loading the class.
- It gives checked exception ClassNotFoundExcecption
  public static Class forName(String fullyQualifiedName);

With the help of method loading as well as registering is completed.

Step 2: -Establish Connection between java program and Data base: -

In this step we are establishing the connection with data base.
We do this step with the help of getConnection().

getConnection(): -

It is a method of DriverManager class

It is a static method.

Public static Connection getConnection (String url)

Public static Connection getConnection(String url, String user, String password)

Public static Connection getConnection(String url, Properties p)

URI(Uniform resource Identifier):

- By URI we can locate the resource
  There are 2 types of URI
  - URN(uniform Resource Name): - It is used locate resource by resource name.
  - URL(uniform Resource Locator): - It is used to locate resource by special formate(path)

  Protocol:subprotocal://hostname:protnumber?data=value&data2=value2……..

Protocol: -

- These are set or rules to locate resource.
- These protocol helps to identify the resource in a secure and organized manner
  Ex: - http or https, ftp, tcp
  In Jdbc we use jdbc protocol

Sub protocol: -

- These are additional rules provided by service provider.
- These supports main protocals
  - Ex: mysql, oracle…..

Host:

- It is a machine which runs the programs and also manages the request.
  - There are 2 types of host
    - Localhost
    - Remotehost

Local host: -

Whenever client and application(resource) both are present and running in a same machine(system) then it is referred as localhost.
Ex: running mysql application and access in same system.

Remote Hose: -

Whenever application present/running in a system and accessed from client present in a different system, then the machine running the application can be referred as remote host.

→ domain name → Ip Address+portnumber

JDBC url for mysql: -

Jdbc:mysql://localhost:3306?user=root&password=root

Step 3: -Create Platform to execute sql queries: -

➔ Platform will help to compile the query, execute the query, get result of execution etc.,

➔ There are 3 different platforms
- Statement
- PreparedStatement
- CallableStatement

Statement:

➔ It is a general platform for executing sql queries.

➔ this platform can be created by using createStatement ()

createStatement() :

This is a non-static method of Connection interface.

Public Statement createStatement();

Step 4: - Execute SQL queries: -

To execute sql queries Statement interface has 3 methods

- execute()
- executeUpdate()
- executeQuery()

All the methods are non-static method

execute(): -

this is a generic method, used for executing all type of sql queries.

public Boolean execute(String query);

true: - DQL

false: - not DQL

executeUpdate(): -

this is a special type of method, it is used to execute only DML queries

public int executeUpdate(String query);

int -> it is a number which represent number of rows got affected by the execution of sql query.

executeQuery(): -

it is a special type of method used to execute only DQL queries.

public ResultSet executeQuery(String query)

Step 5: - fetching resultant data:

Whenever we execute DQL queries we get some data, to fetch the result we make use of ResultSet interface methods.

ResultSet: -

- it is a interface of jdbc api (java.sql.package)
- ResultSet object holds the tabular data by using resultset methods we can fetch data present in it.
- In other words result set contains the results of sql query executed.
- ResultSet cursor always points before the first row.
- By using next() method of ResultSet we can move the cursor to next positions.
    - next() is used o move the cursor to next record,(it moves the ursor with a step of 1).
    - It return true if net row is present, it gives false if next row is not present
        - public Boolean next();
- By using get****() method we can fetch data from specific column.

    public *** get***(int columnNumber)

    pubic *** get***(String columnName)

    here *** represents data type.

PreparedStatement: -

➔ It is a platform to execute sql queries

➔ PreparedStatement also an interface in java.sql package extends Statement interface

➔ We can create PreparedStatement platform by using prepareStatement() method

prepareStatement(): -

It is a method of connection interface

Method Declaration: -

Public PreparedStatement prepareStatement(String query)'

It supports placeholder concept

PlaceHolder in PreparedStatement:

→ PlaceHolder is indicated by the symbol '?'

→ It is representing the data at the time of query compilation.

→before executing the query data must be set for every placeholder.

→ we can set the placeholder value by using set***()

→ Every placeholder represent columns, each place holder must be set with proper type of value.

set***()

→ it is a method of PreparedStatement.

→ it is used for setting the place holder value.

***-> means data type.

public void set***(int placeHolderIndex, *** value);

Difference between Statement and PreparedStatement

Statement	PreparedStatement
• We create object of Statement by using createStatement()	• We create the oject of PreparedStatement by prepareStatement()
• We pass the query at the time of execution	• We pass the query at the time of Platform createion
• Statement doesn't support Placeholder concept	• It supports PlaceHolder Concept
• In case of executing same query for multiple time, each time compilation and each time execution will happen	• In case of executing same query multiple time, one compilation and multiple time execution will happens
• This is suitable for DDL type of queries	• This is suitable for DML type of queries
• Performance is low compared to PreparedStatement	• Performance is high