JDBC

Jar file:

It is a file format based on zip file format which is used to compress many files into one single file.

Contents Of jar File:

- .java
- .class
- Config files

- ✓ .Xml file
- ✓ .properties

.xml:

Is used to configure the resources with the help of custom or user-defined tags.

.properties:

Used to provide set of properties in the form of key and value pair.

Need for jar file:

Needed to import the properties based on the requirement.

API(Application Programming Interface):

- It is used for inter-application communication.
- One application can communicate with another application with the help of abstraction API to achieve Loose Coupling.
- Backbone of API is ABSTRACTION.
- Output of abstraction is LOOSE COUPLING.

Example for API:

- ✓ Apache-POI
- ✓ Jaxcel - ***Java Excel used to create Excel Sheet.***
- ✓ JDBC
- ✓ Servlet

Contents of API:

- Interfaces
- Helper Classes
- Implementation Classes

Two Forms of API:

I form of API:

Single jar file

II form of API:

Two jar files. One contains Interfaces and Helper Classes.
One contains Implementation Classes.

JDBC API:

Contains,

Interfaces - Driver, Connection,Statement,Prepared Statement, Callable Statement, ResultSet.

Helper Class - DriverManager

Implementation Classes - provided by the respective database server in the form of jar file.

JDBC Driver is the implementation of JDBC API, because it contains the implementation classes for the interfaces present in the JDBC API.

All the Driver classes must mandatorily implements the java.sql.Driver Interface.

JDBC:

JAVA DATABASE CONNECTIVITY.

- It is the specification which is given in the form of Abstraction API to achieve Loose coupling between the java application and the database server.

ADVANTAGES:

- Platform Independent.
- We can achieve Loose coupling between the application and the database server.

COSTLY RESOURCES:

The resources which makes use of system properties in the form of Stream is known as Costly Resources.

All the Costly resources need to be closed within the finally block using IF CONDITION to avoid NULLPOINTEREXCEPTION.

In JDBC all the Interfaces of JDBC API are Costly resources.

SPECIFICATIONS OF JDBC:

- All the Driver classes must contain one Static block in it.
- All the Driver classes must mandatorily implements java.sql.Driver Interface present in JDBC API.
- All the Driver classes must mandatorily be registered with the DriverManager by using the method registerDriver().

STEPS OF JDBC:

LOAD AND REGISTER THE DRIVER:

Two ways are there,

- Manually (***Not a good practice, because it causes TIGHT COUPLING between Java application and the database server***).
- Automated way

AUTOMATED WAY:

- Using a Static Method called ***forName()*** method which is present in the class which is having name as ***Class(Present in java.lang Package)*** itself.
- It takes fully qualified classname.
- Which is used only for loading.
- Whenever we use this method it throws an Exception called ***ClassNotFoundException(Checked Exception)***.
- The Object of Driver class is created in order to Register the code in the ***RegisterManager*** Class with the help of a Static method called ***registerDriver()*** which is present in DriverManager() class.
- Inside the Driver class which is present inside the jar file provided by the database server contains one ***Static Block***,
- That ststic block is responsible for registering.
- That static block contains ***DriverManager.registerDriver(new Driver)***.

Establish a Connection between the Java application and the Database server:

- For this we use one static factory/Helper method called ***getConnection()*** method, which is present in ***DriverManager*** Class, which is used to create and return the Implementation Object of Connection Interface present in the JDBC API based on the URL.
- The ***return type of getConnection() method is Connection Interface***.

There are Three Overloaded variants of getConnection methods are there,

- getConnection(String URL)
- getConnection(String URL,Properties Info)
- getConnection(String URL,String username,String Password)

Whenever We use this method it throws an Exception called ***SQLException(Checked Exception).***

CREATE A STATEMENT OR PLATFORM:

- For creating a statement we use on Non-static Factory/Helper method called createStatement() method, which is present in Connection Interface.
- The return type of that method is Statement Interface.

EXECUTE THE SQL QUERIES:

To execute the SQL Queries we have three methods,

1. execute()
2. executeUpdate()
3. executeQuery()
4.

All these methods are present in Statement Interface, Which can be accessed by Prepared Statement as well as Callable Statement also, through INHERITENCE(Multi-Level Inheritence).

execute():

- It is a Generic method which is used to execute all types of SQL Queries.
- It returns True for DQL, False for DDL and DML.
- The return type of execute() method is BOOLEAN.

WHAT IS THE OUTCOME OF DML QUEIES/DML STATEMENTS?

The outcome of DML Queries is 0 to n Integer value which gives the Total Number of Records Affected in the database Server.

executeUpdate():

- It is a specialized method which is used to execute only DML Queries.
- The return type of this method is Integer, because The outcome of DML Queries is 0 to n Integer value , that's why the return type is Integer.
- Whenever we use this method it throws an Exception called SQLException.

We can't fetch the data directly from the table, Instead of that we have to fetch the data from the Cursor/Buffer Memory.

executeQuery():

- It is a specialized method which is used to execute only DQL Queries.
- The outcome of DQL Queries is the Processed/Resultant Data which is stored in the Cursor/Buffer Memory, which can be fetched with the help of ResultSet Interface, that's why the return type of executeQuery() method is ResultSet Interface.

STATEMENT INTERFACE:

- It is an interface which is present in JDBC API Where the implementation classes are provided by the respective Database server.
- In case od Statement Interface the Compilation takes place each time along with the Execution.
- It consumes more memory.
- That's why we go for Prepared Statement Interface(PLACE HOLDER CONCEPT).

By default the ResultSet Interface does not point towards any record in the Cursor/Buffer Memory, i.e>>> It pointed towards ZEROth RECORD AREA.

RESULTSET INTERFACE:

- It is an interface which is present in JDBC API Where the implementation classes are provided by the respective Database server.
- ResultSet Interface contains one method called getXXX() Method, that method is used to fetch the data from the cursor/Buffer Memory.
- It also contains other two methods,
- next() and absolute() methods.

We can create implementation object of ResultSet interface in two ways,

1) Using getResultSet() method, The return type is ResultSet Interface.
2) Another one is using executeQuery() method.

GetXXX():

This method is of two overloaded type,

- getXXX(int Column number)
- getXXX(String Column Name)

The return type of this method is the Respective Datatype.

Next():

This method is used to check whether the next record is present or not. It does not return the value.
The return type of next() method is Boolean.
This method is used only we have Minimum number of records, Because for higher number of records it starts from 0 for every checking process. It is a time consuming process.

Absolute():

This method is also used to check whether the particular record is present or not.
It takes Integer row number as a parameter.
Return type is Boolean.

PLACE HOLDER:

Place holder is a parameter which holds dynamic values at the run time by the user.

RULES TO SET VALUES FOR PLACEHOLDER:

- We have to set the values for the place holder before we execution.
- The number of datas should be matched with the number of placeholders present.
- We have to set the data for the placeholder by using setXXX() method.

setXXX():

- This method is present in Prepared Statement, Statement Interface cannot access it.
- So, Statement Interface does not supports Place holder Concept.
- The return type of setXXX() method is ***Void***.

PREPARED STATEMENT:

- It is an interface present in JDBC API where the implementation classes are provided by the database server in the form of jar file.
- In Prepared Statement Interface Compilation happens only one time during the object creation itself and the Execution happens many times (***EXECUTION PLANNING***).

prepareStatement() method is a Non-static method which is present in Connection Interface, Which is used to create and return the implementation object of Prepared Statement Interface.

Return type of prepareStatement() method is Prepared Statement Interface.

Why Prepared Statement Interface is called as Pre-Compiled Statement?

In Prepared Statement Interface we pass the query at the time of Implementation Object Creation. That's why it is called as Pre-Compiled Statement.