**JDBC:-**

JDBC stands for Java database connectivity.

DataBase

JDBC req

Servlet Req

Front End

Middleware

Frf

**API:-**

Application programming interface. It is group of classes and interfaces which work together to provide additional feature to a programming language.

**JDBC:-**

JDBC stands for Java database connectivity.

JDBC is a JAVA API which works in Backend to establish connection with to database and to convert java object to database understandable value and Database values to java understandable value.

**There are 7 steps in JDBC.**

**7 steps of JDBC:-**

1. **Importing the packages**
2. **Load and register Driver**
3. **Establish the Connection**
4. **Create the statement**
5. **Execute to the statement**
6. **Process the result**
7. **Close the connection**

**Importing the Packages:-**

1. To import the required the files or packages required to run a JDBC program, we need to download jar file of database connector.
2. This Downloaded jar file should be added to java build path.

**Driver:-**

1. It is a third party software (DB software) Which is used to teach the type of database we are trying to access using java program.

In this example we are using type four driver i.e Database dependent driver.

1. To load and register the driver program with java class we will use **forName()** method form class **Class.**

**forName():-**

It is static method **from class Class** which is used to instantiate and retrieve the details of the class whose fully qualified name is passed as a string argument to the method.

**Example:-**

**Class.*forName*("com.mysql.cj.jdbc.Driver");**

**Establish the connection:-**

In this step the java program connected with database.

To get a connection method from **Drivermanger** class and provide String url,

String username, string password) as an argument **to getconnection().**

**Example:-**

Connection con= DriverManager.*getConnection*("jdbc:mysql://localhost:3306/m2e2jdbcday1","root","tiger");

Connection-: Interface

DriverManager -class

**1.getconnection():-**

It is static method from driver manger class it is a overloaded method where first get connection method.

1.getconnection(String url, String username, String Password).

**Example:**

Connection con= DriverManager.*getConnection*("jdbc:mysql://localhost:3306/firstdb","root","tiger");

**2.get connection(url):-**

method accept only string url and we need provide username and password as query String.

**Example:**

Driver driver=new Driver();

DriverManager.registerDriver(driver);

DriverManager.getConnection("jdbc:mysql://localhost:3306/firstdb?user=root&password=root");

**3.getconnection(String url, properties info):-**

getconnection() accept String url and properties info where username, password of the database is provided as an information in properties file. All of these get connection method returns object reference of implementing class of **Connection Interface.**

**Example:**

Driver driver = new Driver();

DriverManager.registerDriver(driver);

FileInputStream f=new FileInputStream("myDbInfo.properties");

Properties properties=new Properties();

properties.load(f);

-String url="jdbc:mysql://localhost:3306/firstdb";

Connection con=DriverManager.getConnection(url,properties);

DriverManager.*registerDriver*(driver);

FileInputStream f=**new** FileInputStream("myDbInfo.properties");

Properties properties=**new** Properties();

properties.load(f);

String url="jdbc:mysql://localhost:3306/firstdb";

Connection con=DriverManager.*getConnection*(url,properties);

Sjsjs

User:root

Password:tiger

myDbInfo.properties

**Statement:-**

* It is an interface from java.sql.package. Non-static methods from connection interface called create statement returns the object reference of implementing class of statement.
* Statement as function which are use to interact with database.

Example:-

Execute()

Statement statement=connnection.createStatement();

String query="INSERT INTO Student(sid,sname,email,age) VALUES (101,'Mirza','abc@gmail.com',22)";

boolean res=statement.execute(query);

**Excute method():-**

* It is non-static method of statement interface which accept database query as String argument.
* This execute() can be used for all four **CRUD** operation and for select query it will return **true** and for others it will return **false**.

**ExecuteUpdate():-**

It is non-static method from statement interface which accept database query as a string argument.

Assingnment:-

Create Employee table

Id,ename,email,phoneno,Designation,sal,Dept,Address

* Inserting
* Deleting
* Update
* Fetching
* This method can be only used for **insert, update and delete query.**
* The return type of this method is Integer where this **int** value shows number of rows that got updated.

Project name:- Employee Data management

**ExcuteQuery:-**

* Excute query is a non-static method from statement interface, It can only use with Select query.
* The return type of this method is ResultSet.

**ResultSet:-**

* It is interface from java.sql.package. An object reference of implementing class of interface will have data or values from database.
* By default result set object will be pointing towards 0th index of row no. in 0 in the table,and since we already know minimum row number in the table is 1 so we need to make pointer of ResultSet pointing 0th to next index. And to do this we have non-static method called **next().**
  + - **next():**
    - **Next** has two function

1. 1st pointer of ResultSet printing to next immediate row e.g.(0th to 1 index of the table.
2. 2nd function of resultset is to check cotain data or not.

Note:-next method return Boolean value.

**Accessing the data from Resultset Object:-**

* To access the data from ResultSet object we have method like get() followed by the data type of specific data which are trying to access. And we need to provide column int index or column position as a parameter of this method.

**Example:-**

resultSet.getInt(1);

* This getInt () method is accessing integer type data from column number 1.r

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Name | Deptno | address |
| 101 | Mirza | 10001 | Karnataka |
|  |  |  |  |

Object of implement class of ResultSet

* By default result set pointing to the column name
* resultSet.getInt(1); 🡪101
* resultSet.getString(2); 🡪Mirza
* resultSet.getInt(3); 🡪1000
* resultSet.getString(4); 🡪Karnataka

6. Processing the result

**7. Closing the connection**

Closing connection between java program and database.

* Connection is an resource odd link which is used to transfer data between java program and database. Is connection created will be an extra load on database server and if old connection is closed before performing new operation with the database than is connection will be treated as a new connection? Which will resulting unnecessary load or wastage of server resource (Expensive resources).
* Hence It is highly recommended to close the connection after each operation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Execute | | ExcuteUpdate | | ExcuteQuery | |
| Query | Result(Boolean) | Query | Result(Integer) | Query | ResultSet |
| Select | T | Insert | No.of rows | Select | Object reference of implement class of ResultSet containing data from |
| Insert | F | Update | No.of rows |  |  |
| Update | F | Delete | No.of rows |  |  |
| Delete | F |  |  |  |  |
|  |  |  |  |  |  |

**PreparedStatement**:-

To strore multiple data in the database ata a time we need to use two non static method for statement and prepared statement interface addBatch(),executeBatch().

Select Query:-Multiple rows

Update Query:-Multiple rows

Delete Query :Multiple rows

Insert query:- Single rows(Note:-Using excuteBatch() Mulptiple rows inserted)

**AddBatch():-**it store the values in temporary memory.

**executeBatch():-**The executeBatch() method begins the execution of all the grouped together statements. The method returns an integer array, and each of the element of the array represents the updated count for respective update statement.

Maven:-

1. new create ctlr+n
2. Maven Project
3. Crete simple project
4. Artifact id->project name
5. Group Id->Domain name
6. Goto webpage Mavenreprositary
7. Download mysql 8.26
8. Copy the xml code and open pom.xml in dependises tag paste
9. New folder genratrate maven dependencies in this folder mysql connector showing

**Native Query :-**

Database sppecific query is known as native query(directly dependent ).

**Native Driver(type 4):-**

Database specific driver are known as native driver.

**PreparedStatement:-**