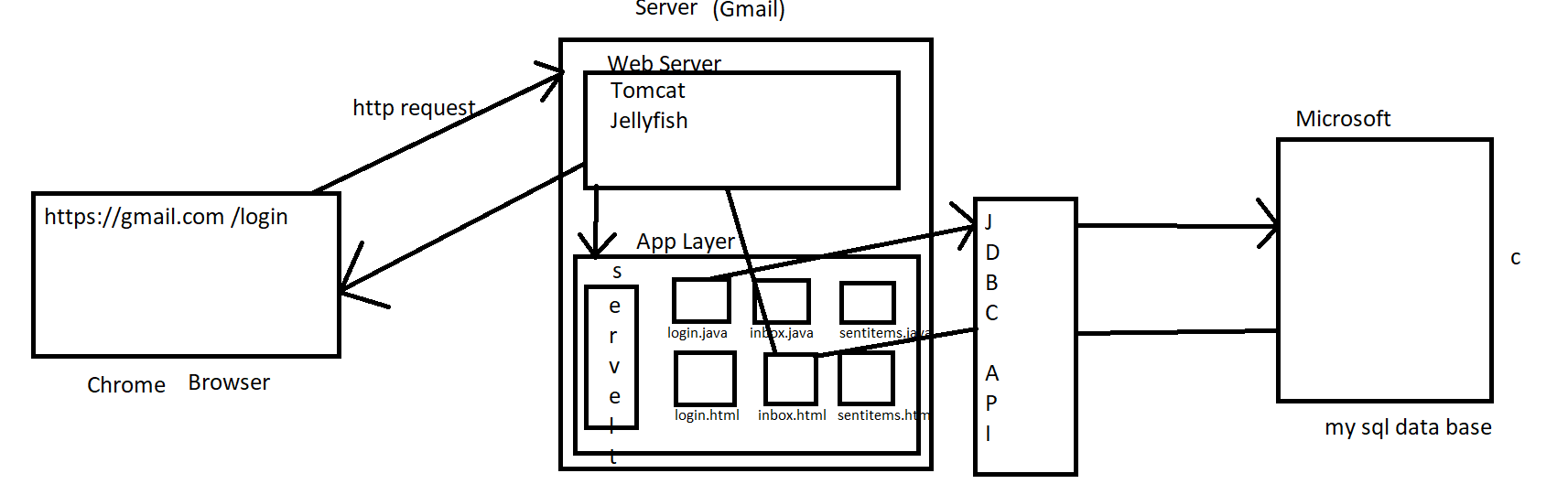
**What is JDBC?**

It’s an advance API Present inside J2EE, which is used to connect to any data base or talk to any data base from the java code.

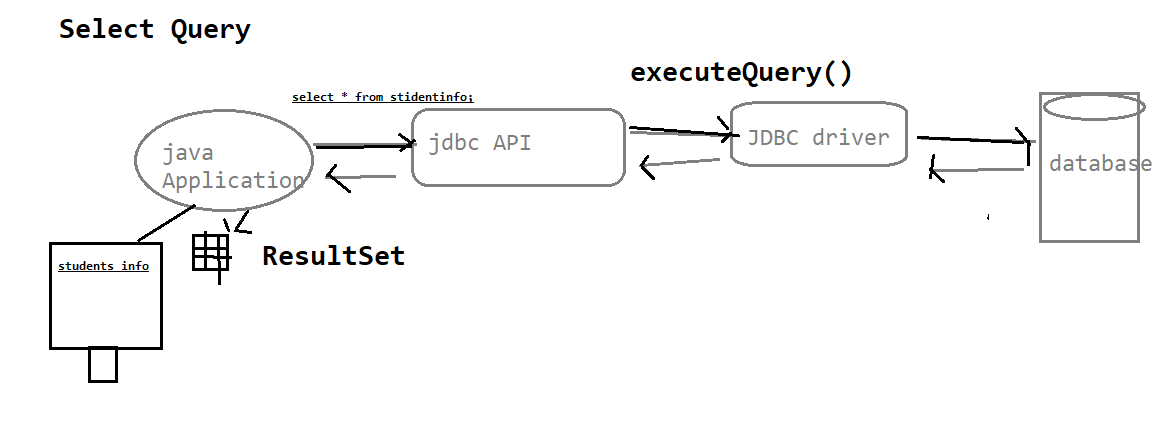




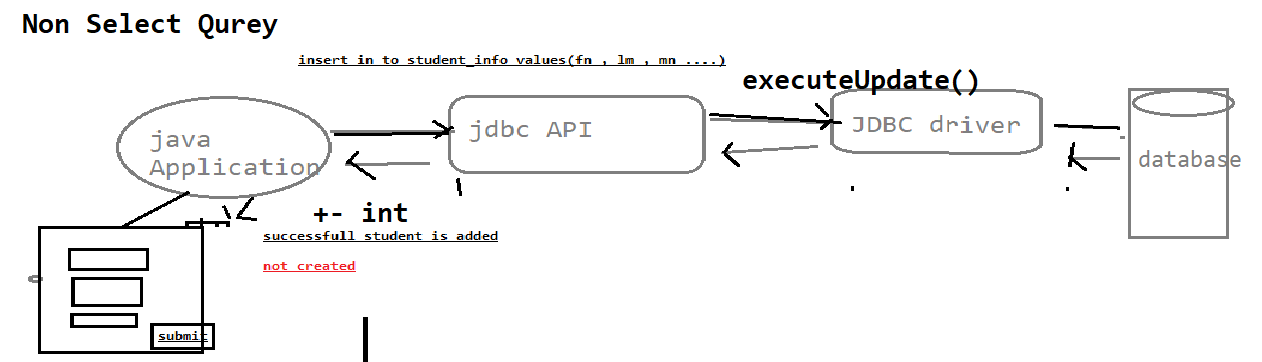
JDBC helps java programmer to store (insert query) / retrieve (Select Query) data from the physical database.

* Every operation we perform in the browser, that request goes to web application with the help of web server in the form of http request, then java program will connect to data base by taking help of JDBC API & JDBC Driver.

**JDBC Workflow?**

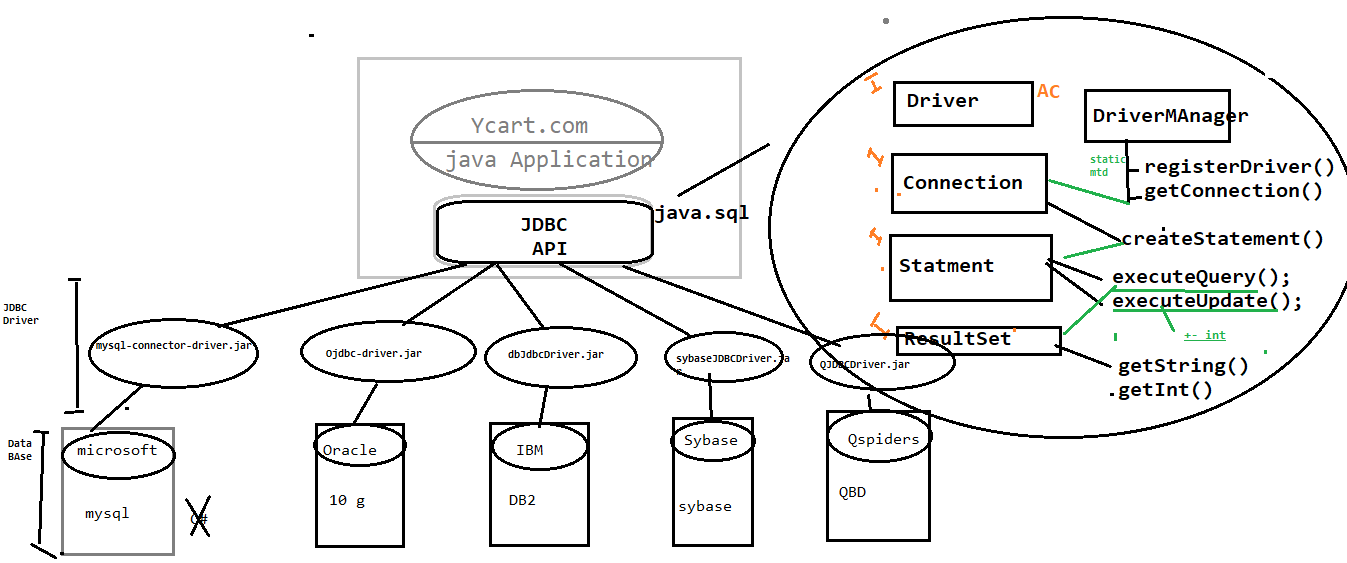


* Select Query – executeQuery()
* When we write the Select query our Java Application will talk to the JDBC API & Our JDBC API talks to the JDBC Driver provided by the Data Base vendor with the help of executeQuery().
* JDBC Driver connect to the DataBase and return the ResultSet (Table) to the user.



* Non Select Query – executeUpdate()
* When we write the Select query our Java Application will talk to the JDBC API & Our JDBC API talks to the JDBC Driver provided by the Data Base vendor with the help of executeQuery().
* JDBC Driver connects to the DataBase & returns integer value like( 1 row affected).

**What is JDBC Drivers & JDBC API?**



C Language

JDBC API: To connect to the Database Ycart.com people didn’t write any Java program so they released an Interface. It’s an API’s available in java.sql package, which is collection interfaces with abstract method, & it’s database independent.

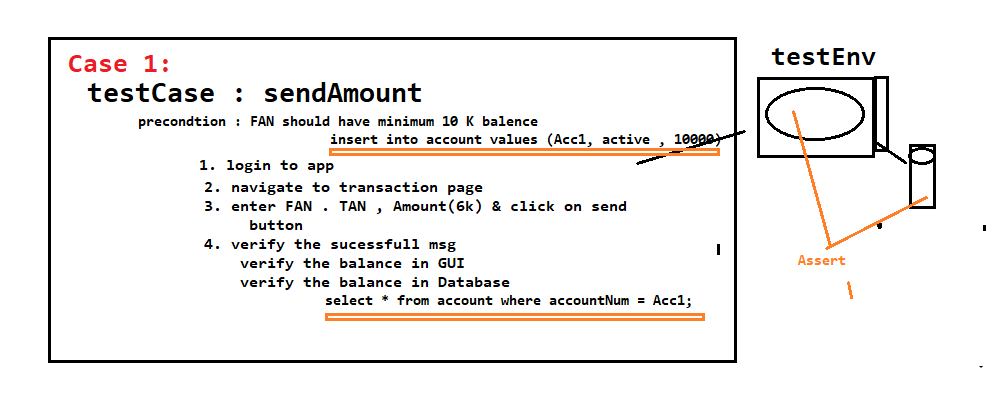
JDBC Driver: its implementation class of JDBC API provided by data base vender. it’s a database dependent.

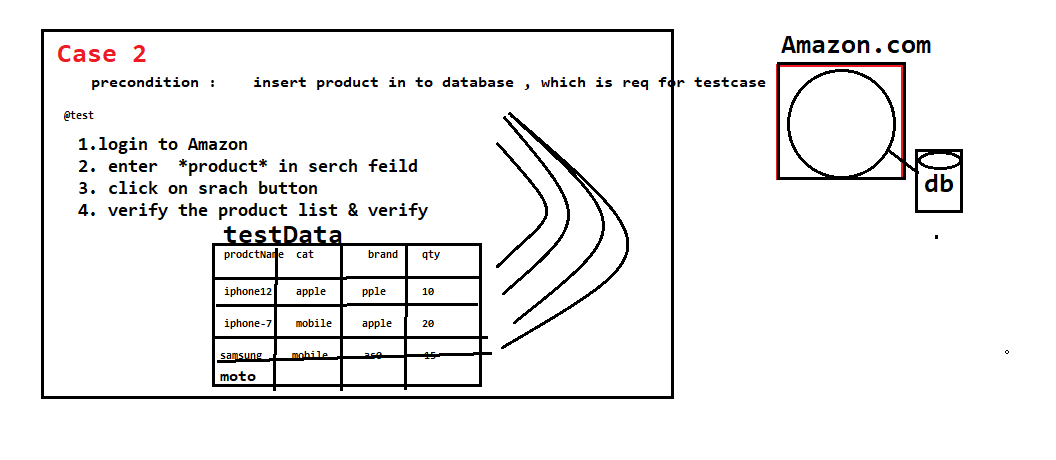
1. JDBC API contains Empty Interface & Abstract Classes.
2. JDBC API takes Help of JDBC Driver to connect to Different Data Base.
3. DriverManager is a Singleton Class which helps to establish connection with Data Base only once.
4. Singleton Class allows the user to create only one Object, we can call that object any no. of times. Ex: Antivirus.
5. registerDriver() helps to register the Driver File to the Data Base.
6. By Using getConnection() we can establish connection with the Data Base & it will return the connection.
7. createStatement() helps us to create Statement & returns statement, then only we can execute executeQuery() & executeUpdate().
8. getString() & getInt() of ResultSet Interface talks about the type of Data we will get from the Data Base.

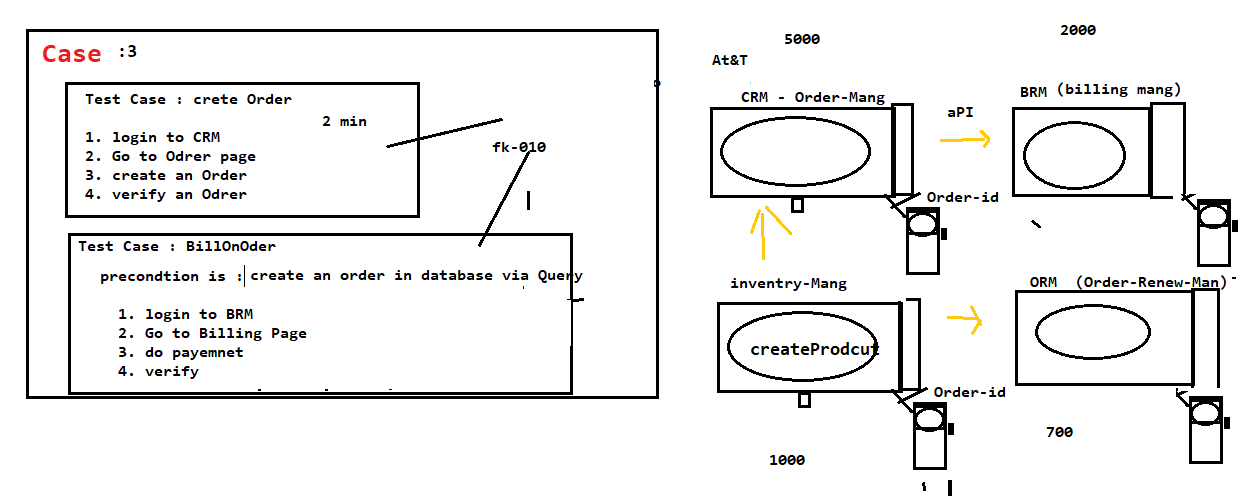
Why JDBC in Automation?

In Automation by using JDBC

1. We can automate any End-to-End Scenario.
2. We can automate Pre-conditions involving Data Base.
3. We can Fetch/Retrieve test data present in Data Base.
4. We can provide validation in GUI w.r.t. to Data Base.







**ExecuteQuery()**

@Test

public void getExecuteQuery() throws SQLException

{

//Step 1: Register the database

Driver driverref=new Driver();

DriverManager.registerDriver(driverref);

//Step 2: establish Connection with database

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

//Step 3: Issue create statement

Statement stat=con.createStatement();

//Step 4: Execute any query

ResultSet result = stat.executeQuery("select \* from project;");

while(result.next())

{ System.out.println(result.getString(1)+"\t"+result.getString(2)+"\t"+result.getString(3)+"\t"+result.getString(4));

}

//Step 5: Close database connection

con.close();

}

**ExecuteUpdate() With finally block**

@Test

public void getExecuteUpdate() throws SQLException

{

Connection con = null;

try {

//Step 1: Register the database

Driver driverref=new Driver();

DriverManager.registerDriver(driverref);

//Step 2: establish Connection with database --- database

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/projects", "root", "root");

//Step 3: Issue create statement

Statement stat=con.createStatement();

//Step 4: Execute any query -- table

int result = stat.executeUpdate("insert project values('TY\_PROJ\_999','RINCY','26/04/2021','SDET-13','ON-GOING',62);");

if(result==1)

{

System.out.println("Project is sucessfully created");

}

else

{

System.out.println("Query failed");

}

}

catch(Exception e) {

}

finally {

//Step 5: Close database connection

con.close();

System.out.println("Connection closed");

}

}