16/10/2023

Application

Stand Alone Applications

- -> internet not required
- -> No Database
- -> Present in Local Storage
- -> Installation is Mandatory

eg:- Calculator, Contacts, ...

Web based Application

- -> requires Internet
- -> Contains Database
- -> Installation is not mandatory
- -> Servers

Client-Server Application

Static web based

- -> Static resources
- -> Static response

Dynamic web based

- -> static + Dynamic resources
- -> Dynamic response

eg:- instagram, facebook....

Client Application

-> internet

Both can access in client application and also client server Application.

for example

GeeksofGeeks:-

Content is Static.

-> Type of Static web based

-> Will work only in internet Connection

Youtube:-

Content is Dynamic.

-> Type of Dynamic web based.

-> will work only in internet Connection.

17/10/23

BE ← JDBC

JDK Application

Servlets → FE

⇒ what is Console?

JDK Versions of JDK (Editions)

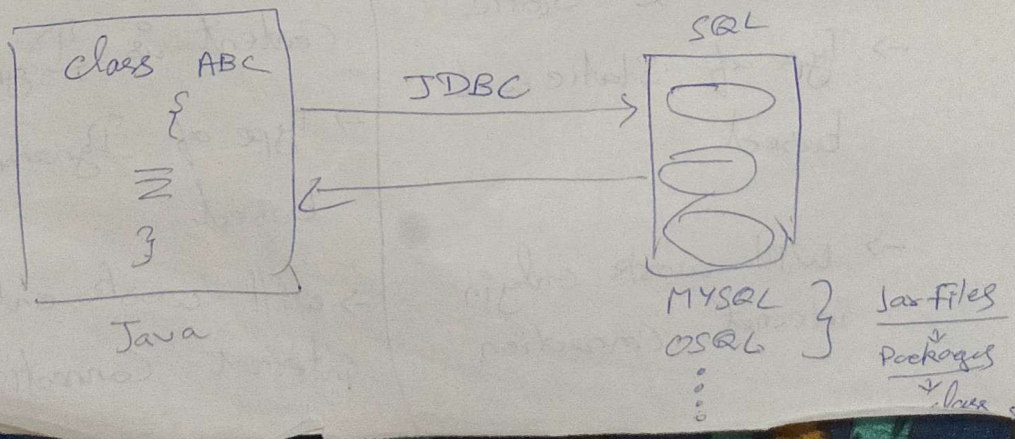
> J2SE Java Standard Edition
> J2EE Java Enterprise Edition.
> J2ME Java Micro Edition.

J2SE ⇒ Stand Alone Application. (only middleware)
J2ME ⇒ (frontend & middleware)
J2EE ⇒ Dynamic web Application
(Frontend, middleware, Backend.

To Instal ⇒ JEE
MySQL
Tomcat Server.

JDBC ⇒ Java DataBase Connectivity.

⇒ used to Connect ~~with~~ file Application with Database



API ⇒ Application Program Interface.

Server to Server.

⇒ Transfer Data to one Application to another Application

API (I)

Interface ⇒ Implementation Given by respective Vendors Database (MySQL, OSQL, ...)

Vendors ⇒ Jar files contains Packages (.class) (Database)

Components of JDBC

- Driver
- Connection
- Statement
- ResultSet
- DriverManager → class.

} Interface } Java.sql. Package

Interface Driver Java.SQL Package (Interface)

{
Abstract Methods;
}

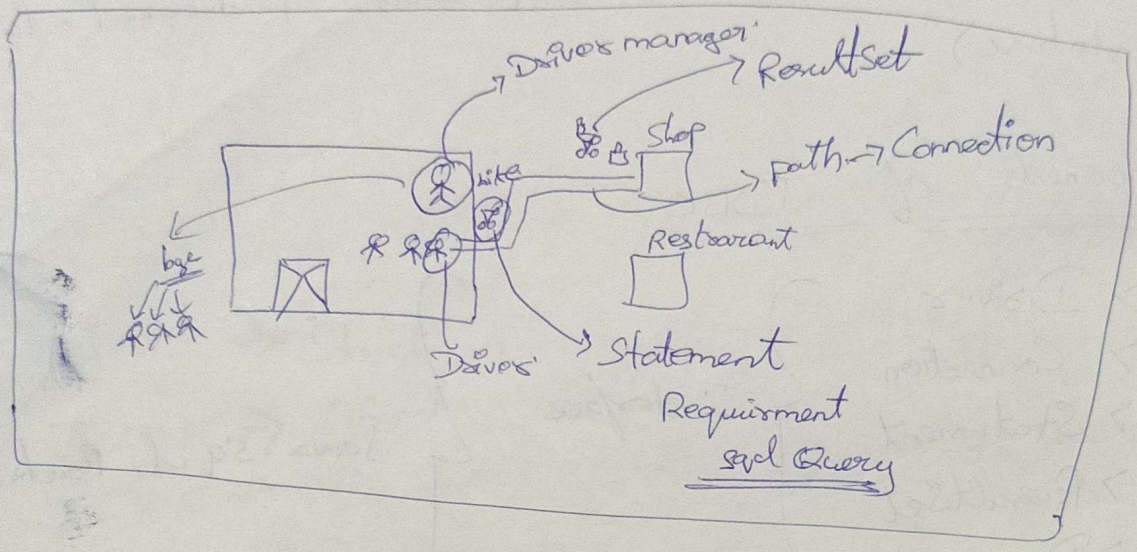
MySQL (Jar files) → com.mysql.cj.jdbc
↓
Driver.class.

class Driver Implements Driver

{
overriding of methods() { }.
}

Steps involved in JDBC

1.) Load and Register the driver (Choosing)
2.) Establish Connection.
3.) Create Statement and issue the Query
4.) Process the Result
5.) close the Connection.




```
Driver driver = new Com.mysql.cj.jdbc.Driver();
DriverManager.registerDriver(driver);
String path = "jdbc:mysql://localhost:3306/Studentdata";
String user = "root"; String pass = "root";
Connection conn = DriverManager.getConnection(path, user, pass);
Statement st = conn.CreateStatement();
String query = "insert into student value(1,'abc',
                'abc@gmail.com', 7676767676);
int result = st.executeUpdate(query);
Sysout(result);
Conn.close();
```

Maven Repository

Build path, Config B.P, → libraries → Add External jars.

DML Query

> Insert
> update
> delete

} Statement st;

int res = st.executeUpdate();

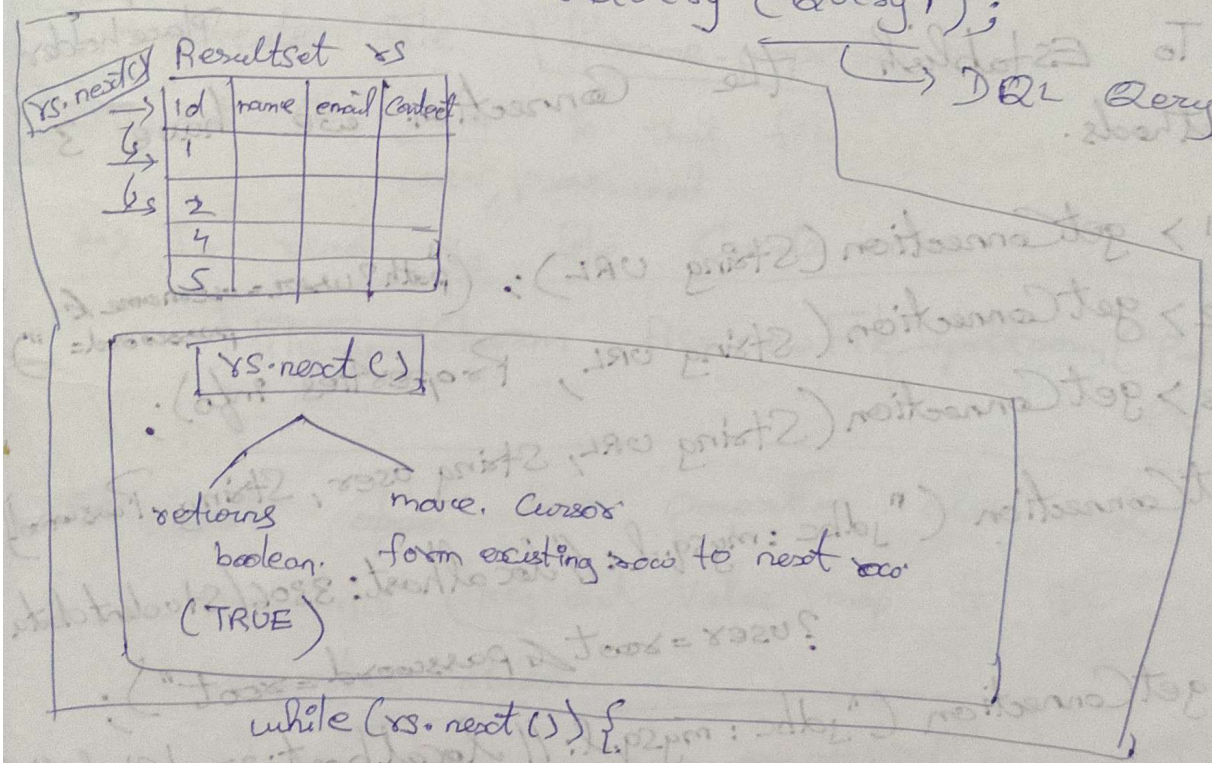
DQL Query

> Select * from student; (Query 1)
> Select id, name from Student (Query 2)

Statement st;

ResultSet rs = st.executeQuery(Query 1);

→ DQL Query




```
while(rs.next()){
    int id = rs.getInt(1);
    String name = rs.getString(2);
    String Email = rs.getString(3);
    long Contact = rs.getLong(4);
```

Syso (id + " " + name + " " + Email + " " + Contact);

}

Con.closecs;

26/10/2023

Query

(int) executeUpdate

(ResultSet) executeQuery

Static Query

assigning the Values in the Query itself.

eg:- delete fromemp where id=7;

Dynamic Query

assigning the Values in the RunTime.

int

Results

eg: delete from emp where id=?;

Place holder

=> To Establish the Connection we have 3 methods.

1> getConnection(String URL); ("path?user=username & password=")

2> getConnection(String URL, Properties info);

3> getConnection(String URL, String user, String Password);

1> getConnection ("jdbc:mysql://localhost:3306/Studentdata ?user=root & password=root");

3> getConnection ("jdbc:mysql://localhost:3306/Studentdata", "root", "root");

Connection path

path

better way Practice

"jdbc:mysql://localhost:3306/Studentdata"

(optional)

"jdbc:mysql://localhost

"jdbc:mysql:///Studentdata"

27 getConnection(String URL, Properties info);

class presentin java.util Package

Properties => used to get Configuration files

=> Internally using Map interface (key, value)

Steps

=> Create file filename.properties

=> put the data in that file

user, password

=> To read the file. Create use.

```
FileInputStream fis = new FileInputStream("Path of file");
```

=> Create object for properties class and pass the (key and value) map to the properties object

```
Properties p = new properties();
p.load(fis);
```

=> Now Properties class object P contains user and Password. you can Pass the (P) to the (>get Connection method)

```
Connection c = DriverManager.getConnection
        ("jdbc:mysql:///Studentdata", P);
```

```
FileInputStream f = new FileInputStream("Path of file");
Properties p = new Properties();
p.load(f);
Connection c = DriverManager.getConnection
        ("jdbc:mysql:///Studentdata", p);
```

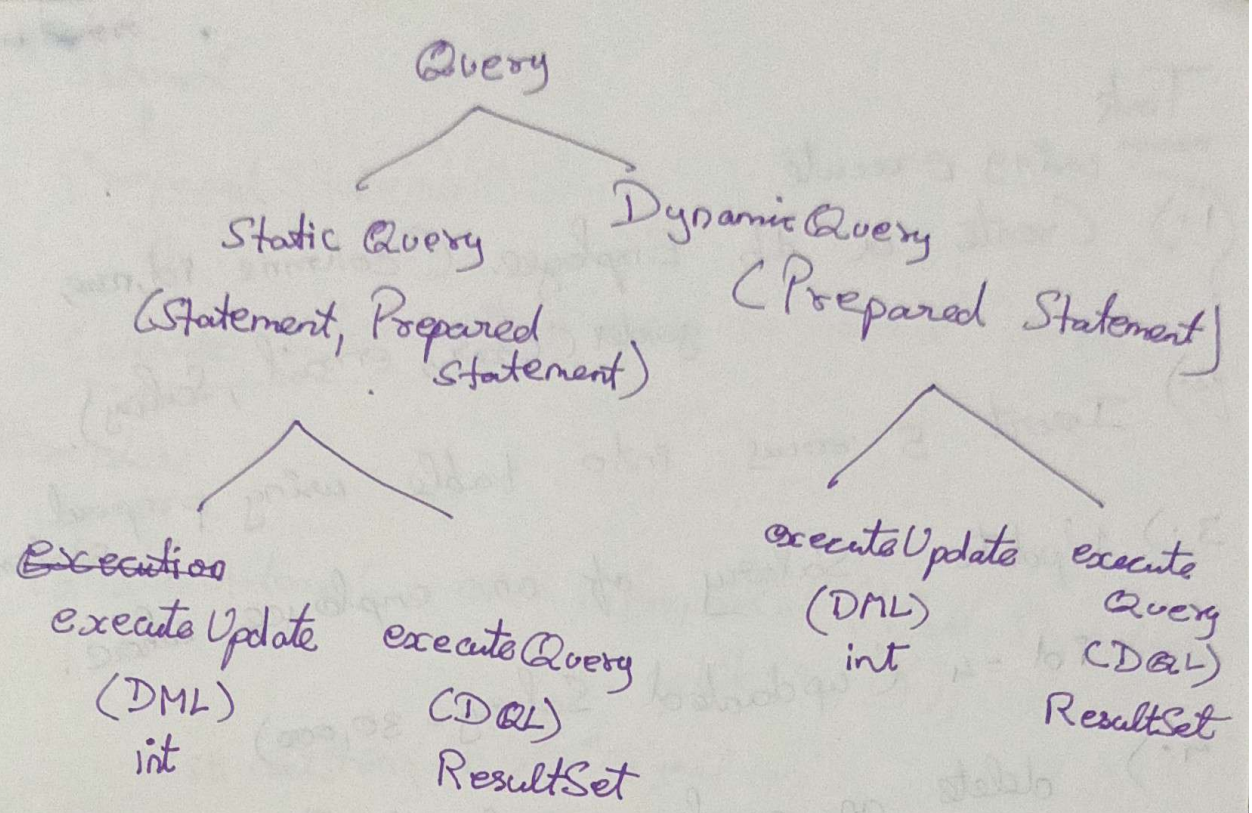
=> Task
insert 2 rows using preparedStatement (getConnection with 1 Para)
delete 1 row using Statement (getConnection with 2 Param)

28/10/23

Load and Register Can be done in another way also

=> `Class.forName("com.mysql.cj.jdbc.Driver");`
Automatically load and Register

Task => getConnect (2 param)
preparedStatement
Query => Select * from Student
display only name & Contact



=> if you don't know which Query is entered we use execute method it may Statement or PreparedStatement returns boolean

```
st :
pst : execute(Query);
```

[DML/DQL]

boolean

True [DQL]

false [DML]

=> DML if we want to see the no's of rows affected we can have method in Statement which is `st.getUpdateCount();` returns int value

=> if we using DQL Query we use method `st.getResultSet();` returns ResultSet

Task

using Execute

1.) Create a db Employee. (columns id, name, gender (char), email, Salary).

2.) Insert 5 rows into table using prepared statement

3.) Update Salary of an employee where id = 4 (updated Salary = 30,000)

4.) delete an employee where id = 2 using getConnection method with 2 parameter

31/10/2023

Store Procedure

=> Creating a Procedure which is used to perform set of Query repeatedly when it is called. (Same Concept like Methods in Java) => Mainly or only used to perform with database tables

```
Create procedure 'insert' (id int, name varchar(15),
                           procedure name
email varchar(20), Contact number(10))

<= Begin
insert into student employee values ('id, name, email, Contat')
end
3<
```

1/11/2023

Statement

⇓

Prepared Statement

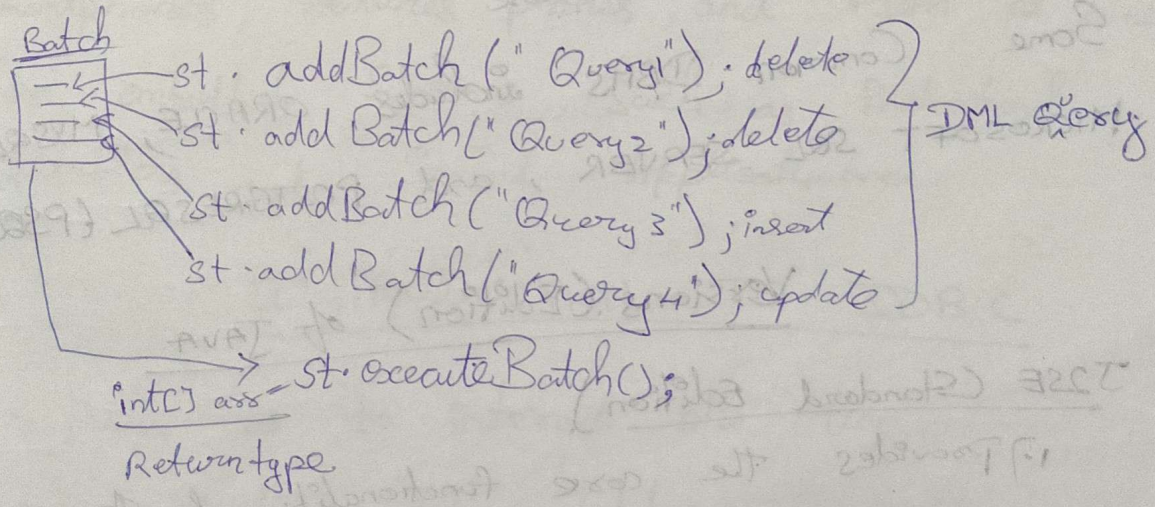
⇓

Callable Statement ⇒ used to execute Stored Procedure

```
CallableStatement cst = c.prepareCall("Call Studentdata-
                        update(?,?)");
cst.setString(1,"abc@gmail.com");
cst.setInt(2,30);
syso(cst.execute());  syso(cst.getUpdateCount());
```

Batch Execution

Batch is a Container which as multiple Queries to execute to Database.



Note we can't use DQL Query becaus its return type is Result set.

2/11/2023

Drawbacks of File Handling and Collection.

> Limited data Sharing using file Handling.
> Difficulty in accessing Data using file Handling
> Using Collection Interface, Stored data will be in temporary in nature.

DATABASES

* A Database is an Organized Collection of data that is Stored and Accessed Electronically
* Database can be managed using a Database Management System (DBMS), which is a Software which allows users to Create, Edit and Manage the data is the Database.
* Some Common DBMS includes ORACLE, MYSQL, MICROSOFT SQL SERVER, and POSTGRESQL (PSQL)

Versions (Edition) of JAVA

J2SE (Standard Edition)

1.) Provides the core functionality of the java programming Language.
2.) Using j2se Standalone Application can be built

J2EE (Enterprise Edition).

1.) J2EE is a platform for building enterprise-level applications. It includes a set of APIs and services that are used for developing web-based and distributed Applications

2.) Using j2ee web based Applications can be built

3.) J2EE is used for building large-scale, mission-Critical applications that require high availability, Scalability and Security

J2ME (Micro edition)

1.) J2ME is a platform for building mobile and embedded applications

2.) J2ME is used for developing Applications for mobile devices, Such as Smartphones, features phones, and PDAs as well as for embedded Systems, Such as industrial Controllers and Smart Appliance.

INTRODUCTION TO JDBC

> Helps us to interact b/w two application, java application and database

> JDBC Stands for Java Database Connectivity.

> JDBC is a Java API to Connect and execute the Query with the Database

> It is used to write programs required to access database

> It is a Standard interface present is java.sql package.

COMPONENTS OF JDBC

1.) DRIVER
(Translates API Calls Into Operations for Specific Database

2.) DRIVER MANAGR
(Loads Database Drivers and Manages Connection Between Application and Driver)

3.) Connection
(Connection b/w Application and Data Source)

4.) STATEMENT:
(To issue SQL Query to Database)

AJDBC DRIVER

1.) Is an Interpreter that Translates JDBC Methods Calls to Vendor-Specific Database Commands

2.) Implements Inteface in Java.sql packag

Steps in JDBC

1.) Load and Register the Driver
2.) Establish a Connection
3.) Create a Statement and issue the Query.
4.) Process the result
5.) Close the Connection.

1.) Load and Register the Driver

Method 1

> use the static DriverManager.RegisterDriver()

eg:-

```
Driver d = new com.mysql.cj.jdbc.Driver();
DriverManager.registerDriver(d);
```

Method 2

> use java's CLASS.FORNAME() Method.

eg:-

```
Class.forName("Com.mysql.cj.jdbc.Driver");
```

DriverManager.GetConnection() Methods

1> getConnection(String url);
2> getConnection(String url, properties prop);
3.) getConnection(String url, String user, String password);

HIERARCHY OF STATEMENT

Statement

use this general-purpose access to your database. Useful when you are using Static SQL Statements at ~~uniform~~ runtime. The Statement interface Cannot accept parameters.

PreparedStatement

Use this ~~for general~~ when you have dynamic Query. The Prepared Statement interface accepts input parameters at runtime (input Values for place holders)

Callable Statement

Use this when you want to access the database Stored procedures. The Callable Statement interface can also accept runtime input parameters.

Methods to Execute SQL Query

> executeQuery(query).
- Returns resultset
- DQL Queries are executed.

7 Executeupdate(Query)

- Returns integer type
- Returns number of rows / Column update
- DML Queries are executed.

7 Execute(Query)

- All types of Queries can be executed.
- Returns boolean result
- True - if Query is DQL
- False - if Query is DML
- Fails to returns how many rows are affected

4.) Process the Result.

=> Since each execute, executeupdate, executequery method returns different type result, it is necessary to process the result based on type of result.

=> For Integer result of executeupdate we can know the number of row altered. by Storing result in integer type Variable

=> For resultSet result of executequery we can store the result in resultset and access the data members that are fetched from database.

⇒ for boolean result of execute, we can only know the state of result, true if Query is dql and false if Query is dml.

5. Close the Connection

> closing the Connection in JDBC is Important Because it Ensure that the System Resource allocated to the Connection are Released and made Available for other Applications or processes to use

> If a JDBC Connection is Not closed properly, It Can Lead to issues Such as Memory Leaks, where System memory is not Freed up and Eventually Causes the Application to Crash.

> Additionally, It Can Cause Performance issues by keeping Database Connections open for a longer time Than Necessary, which Can Impact the Database Server's Ability to Handle other Requests

Batch Execution.

>> Instead of Executing a Single Query, we can Execute a Batch (Group) of Queries. It makes the performance fast. It is Because when one sends Multiple Statements of SQL at once to the Database, the Communication Overhead is Reduced. Significantly, as one is Not Communicating with the Database Frequently, which in Turn Results to Fast performance.

>> The addBatch() method of Statement, preparedstatement and CallableStatement is used to individual statement to the Batch.

Types of Drivers.

1> JDBC-ODBC driver (Type 1) → Platform dependent
2> Native API driver (Type 2) → Libraries (Platform dependent)
3.) Network Driver (Type 3) → Java.
4.) Thin Driver (Type 4) → Java - Platform independent

6/11/23

Servlet (API) (Generate Dynamic Response)
Used to Connect with FrontEnd

Servlet (Interface)

5 Methods (Abstract Methods)

→ init ();
→ service ( ); (generate dynamic response)
→ getServletInfo ( );
→ getServletConfig ( );
→ destroy ( );

service (ServletRequest req, ServletResponse res);

HTTP Errors

401 - Unauthorized.
400 - Bad Request
403 - Forbidden Errors
404 - Page Not found
500 - Internal Errors.
503 - Service Unavailable.

Server Installation Steps

1.) Search for Server in Eclipse Enterprise Edition
2.) click [Servers (Server)]
3.) click Apache (click [Tomcat v9.0 Server])
4.) click download and Install
5.) Select the Folder to Install the Server
6.) It takes few minutes to install in Eclipse
7) click Finish after installation.

14/11/23

Dynamic Web Page

```
Service (ServletRequest req, ServletResponse res);
    String name = req.getparameter("uname");
    String email = req.getParameter("mail");
    Syso("Name ::
```

→ getting and printing the Content (class java.io Pack)

```
PrintWriter pw = res.getWriter();
pw.println("Dynamic page");
pw.println("Name:-" + name);
pw.println("Mail:-" + mail);
```

→ print the content in the web page.

```
Syso("Name" + name + "Email" + email);
```

↓

will print the Content in the Console.

form.html < inside webapp html files.

<!- - - ->

```
<body>
  <form action = "login">
    Name : <input type = "text" name = "username"> <br>
    Email : <input type = "text" name = "mail" > <br>
    <input type = "submit">
</body>
```

Create class which implements Servlet Interface
(inside src/main/java)

Override only : service (ServletRequest req,
ServletResponse res) { }

}

web.xml

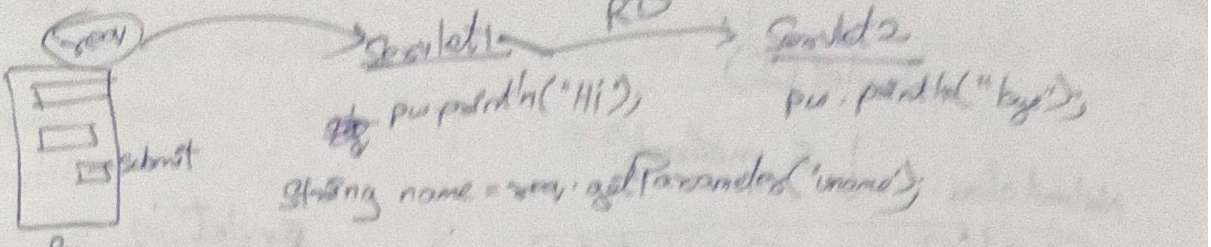
```
<servlet>
  <servlet-class> fully Qualified name </servlet-class>
  <servlet-name> eg:- S2 </servlet-name>
</servlet>
<servlet-mapping>
  <url-pattern> /login </url-pattern>
```

Specify this in form action

```
  <servlet-name> eg:- S2 </servlet-name>
```

↓ Same as <servlet-name>

```
</servlet-mapping>
```



String name = req.getParameter("name");

(I) RequestDispatcher disp = req.getRequestDispatcher("s2");

path

forward()
o/p → bye

include()
o/p → Hi bye

```
<body>
  <form action="d">
  ==
  </form>
</body>
```

Servlet1

```
RequestDispatcher rd =
  req.getRequestDispatcher(d1);
rd.include(req, res);
    or
rd.forward(req, res);
```

Servlet 2

```
pw.println("Today");
```

```
<servlet>
  <servlet-class> Servlet1
  <servlet-name> S1
</servlet>
<servlet-mapping>
  <url-pattern> d
  <servlet-name> S1
</servlet-mapping>
```

```
<servlet>
<servlet-class> servlet2
<servlet-name> S2
</servlet>
<servlet-mapping>
<url-pattern> d
<servlet-name> S2
</servlet-mapping>
```

Servlet (I)

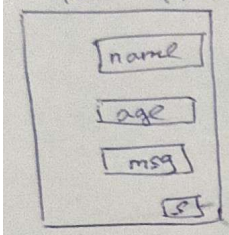
↓ 5 methods

Abstract class

AC | Generic Servlet (c) implements Servlet, 5 methods

- init()
- service()
- ~~Service()~~
- getServletInfo()
- getServletConfig()
- destroy().

16/11/2023



GServlet1 (GenericServlet)

```
service(ServletRequest req, ServletResponse res)
String name = (String) req.getParameter("name");
int age = Integer.parseInt(req.getParameter("age"));
String msg = req.getparameter("msg");

req.setAttribute("name", name);
                 key    value
req.setAttribute("age", age);
req.setAttribute("msg", msg);

RequestDispatcher rd = req.getRequestDispatcher("two");
rd.include(res, req);
```

url-pattern of 2nd servlet

Path

GServlet2 (GenericServlet

```
service(ServletRequest req, ServletResponse res)
String name = (String) req.getAttribute("name");
                                        key
String msg = (String) req.getAttribute("msg");
int age = (int) req.getAttribute("age");
PrintWriter pw = res.getWriter();
pw.println(name, age, msg);
```

<servlet>
<servlet-class> servlet.servlets.GServlet1 | servlet.servlets.GServlet2
<servlet-name> gs1 | gs2
</servlet>
<servlet-Mapping>
<url-pattern> /one | /two
<servlet-name> gs1 | gs2
</servlet Mapping>.

(I) Servlet (5)
↑
(AC) Generic Servlet (4+1)
↑
only for http protocol ← (AC) HTTP servlet → (5+n)
↳ but don't have any abstract methods
-> doget -> doput
-> dopost -> doOption
-> do delete -> service (HttpServletRequest, HttpServlet Response)
-> do Trace

html.
<form action=" " method="post" => will not display the content in url
method="get" => will display the content and also it is default

if you using "get" method in form use doget() (or) two Service(SReq, SRes) (or) Service(HReq, HRes)

if you using "post" method in form use doPost()

TEST QUESTIONS

1.) What is DriverManager and Driver

→ DriverManager manages the set of Java Database Connectivity (JDBC) drivers that are available for an application to use.

→ Loads Database Drivers and manages Connecting HW Application and Driver

→ Driver A JDBC Driver is a SW component Enabling a Java Application to interact with a Database

→ Translate API Calls to Operation for Specific

2.)

executeUpdate()	executeQuery()
→ This method is used to execute the SQL statements which update or modify the DB int	→ This method is used to execute the SQL statements which retrieve some data from the DB
→ ResultSet is the Returntype which represents the number of rows affected by the Query, "0" return nothing	→ ResultSet is the Returntype which contains the results returned by the Query
→ This method used to execute only Non Select Queries	→ This Method used to execute only Select Queries
eg:- Select	eg:- DML (INSERT, DELETE, UPDATE) DDL (CREATE, ALTER)

3.)

Statement	PreparedStatement
→ It is Base Interface	→ It extends Statement interface
→ It is used to execute normal SQL query	→ It is used to create Parameterized Query
→ It is used for DDL, DML statements	→ It is used for any SQL Query.

→ It is used, when a Particular SQL Query execute only once.

→ It is used, when a Particular SQL Query execute multiple times.

4) How Many ways we can load a class? Explain each

1.) Driver d = new com.mysql.cj.jdbc.Driver();
DriverManager.RegisterDriver(d);

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

5) What is Batch-Execution?

→ It allows you to group Related SQL Statements into Batch and Submit them with one Call to the DB. It will improving performance and you reduce the amount of Communication overhead.

→ JDBC Drivers are not required to Support this Features

6.)

Generic Servlet	HttpServlet
→ It is a Abstract class	→ It is a Abstract class
→ Package :- javax.servlet	→ java.servlet.http
→ Use to write protocol independent Servlets.	→ Use to write Http Specific Servlets
→ init(), service(), destroy(), getServletConfig(), getServletInfo(), log(), getServletContext(), getServletName(), getInitParameter(), getInitParameterNames().	→ doGet(), doPost(), doPut(), doDelete(), doHead(), doOptions(), doTrace(), getLastModified(), service(HttpServletReq, HttpServletRes)
→ Only service() method is abstract	→ No Abstract Methods

7)

GET	POST
→ Values are contained in the URL.	Values are contained in the message's body.
→ Has a length limitations of 255 characters.	Not Have a length limitations.
→ Supports only String Datatype	Supports different data types.
→ often Cacheable	Hardly Cacheable.
→ Get is a method that sends information by appending them to the page Request	→ Post is a Method that transfers information via HTTP header.
→ Get helps to send non-sensitive data	→ Post helps to sent sensitive data, Binary data and Uploading files.

8) How many ways we can Configure a Servlet

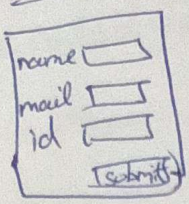
→ by web.xml <servlet> <servlet-class> <servlet-name> <servlet-Mapping> <url-pattern>

→ by Annotation (@WebServlet("/Path")

9)

Forward() Method	SendRedirect() Method
The forward() method works at Server Side.	This method works at client Side.
It Send Send request and response object to another Servlet	It always Sends a new request.
It Can work within the Server only	It Can be within & outside also

10) Frontend



Backend

Employee table

name	mail	id

form.html

```
<form action = "data" method = "post">
    Name = <input type="text" name="name">
    mail = <input type="text" name="mail">
    id = <input type="text" name="id">
    <input type="Submit">
</form>
```

Employee.java

```
public class Employee extends GenericServlet
{
public void service(ServletRequest req,
        ServletResponse res) throws
                ServletException, IOException{
String name = req.getParameter("name");
String mail = req.getParameter("mail");
int id = Integer.parseInt(req.getParameter("id"));

try {
    Class.forName("com.mysql.cj.jdbc.Driver");
    Connection c = DriverManager.getConnection("jdbc:mysql://
            localhost:3306/emptwo?user=root?Password=
            root");
    PreparedStatement pst=c.PreparedStatement
            ("insert into testdb Values(?,?,?)");
```

```
pst.setString(1, name);
pst.setString(2, mail);
pst.setInt(3, id);
PrintWriter pw = res.getWriter();
pw.println(pst.executeUpdate() + "row affected");
c.close()
}
Catch(ClassNotFoundException | SQLException e)
{
e.printStackTrace();
}
}
}
```

Servlets Test

1.) Diff blw SendRedirect and forword
2.) Diff blw GenericServlet and HttpServlet
3.) Diff blw ServletContext and ServletConfig
4.) what is Augumented Reality (AR)?
5.) what is Virtual Reality (VR)?
6.) What is Quantum Computing?
7.) What is CyberSecurity?
8.) What is Automation?
9.) What is encryption & decryption.

Web.xml

```
<context-Param>
<Param-name> dbname </Param-name>
<param-value> Employee </Param-value>
</context-Param>
```

```
Servlet Two {
ServletContext context = getServletContext();
String dbname = context.getInitParameter("db-name");
                                          ↑ Param-name
    s.o.pln(dbname)
        ↓
    Employee
}
```

getServletContext() is a method of Generic Servlet AbstractClass it is used to get the Context from the one Servlet to many Servlet by using it in Web.xml the tags are to write a context is

```
<context-param>
<Param-name>     </Param-name>
<Param-value>    </Param-value>
</context-param>
```

So, by this we can access the Data which is same and also use it where we need it (many Servlets)

to access data to a Particular Servlet we use ServletConfig

web.xml

```
<servlet>
<servlet-class>Config1</servlet-class>
<servlet-name>sft1</servlet-name>
<init-Param>
<param-name>db-name</Param-name>
<param-Value>Sys</Param-Value>
</init-param>
</servlet>
<Servlet-mapping>
<url-pattern>/config1</url-pattern>
<servlet-name>sft1</servlet-name>
</Servlet-mapping>
```

Config1.java (Generic Servlet)

```
Service(req, res){
    ServletConfig config = getServletConfig();
    String db = config.getInitParameter("db-name");
    PrintWriter pw = res.getWriter();
    pw.println("DB name is " + db);
}
```

ServletConfig is used to access the Values and data's on Perticular Servlets by using <init-Param> inside the <Servlet> in web.xml.

```
<init-Param>
  (<Param-name>
   <param-Value>)
</init-Param>
```

User Tracking

→ It is a method used to Record and Monitor the User Activities.

1.) Cookies
2.) Session
3.) URL re-writing
4.) Hidden form fields

Advantages of User Tracking:

1) Performance :- If Imagine that you are seeing some live Streams on webpages or in an Applications and now you turn off the Data or wifi to your device and again your trying to see same live stream without Internet and not you Screen will Show the Data where your stopped to watch and not display any images and you realised to turn on your data it will fetch where your left to watch the Live Stream and loads the images in the Screen which increase user performance with the help of user Tracking

2) Session :- Session Can able to store our data in the Server Side. Imagine that your are trying to Sign up to new Application or webpages, on that first time it will

ask you contact or some other details to Sign Up and it will be stored in. Server. After few day you are attempting to open the same App or webpage and it pro analyz or send the session what it has been stored in that Server.

3> Advertisements (or) Suggestions

When ever your are searching for some Products in the Shopping Applications. and you are Access some other Application which is Connected or accessing through the Internet and it will display the same Product what you have be searched for so, this is happening because of User Track

4) Security

=> When the data is storing in local Storage in the sence of Cookies. it is temporty and also no Security to the client Browser.

=> By that we see the ~~See~~ data which is stored in the server will maintain the Security.

Cookie is a small text file that is stored on the user's computer.

→ The maximum file size of a Cookie is **4KB**

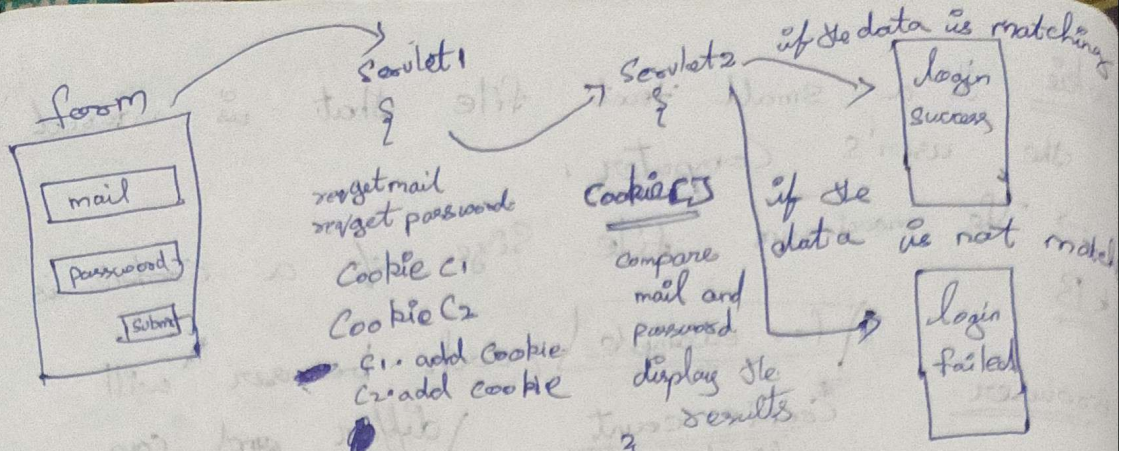
for example → each browser will differ and can have different number of cookie limits to be stored in it

Browser	Cookie count limit per domain
Chrome	180
Firefox	150
Opera	60
Safari	600

Session.

A Session is used to temporarily store the information on the Server to be used across multiple pages of the website.

Session	Cookies
→ Server Side	→ Client Side
→ Stores within a file in a temporary directory on the Server	→ Stored on user's Computer as a text file
→ Session will end when the user logout from the Application or closes his web browser	→ Cookies end on the lifetime set by the user
→ Stores Unlimited Data	→ Stores limited data 1 cookie → 4KB



form.html

```
<form action="Cookie1" method="post">
    Email:- <input type="text" name="mail">
    Password :- <input type="text" name="pass">
    <input type="Submit">
</form>
```

Cookie One Servlet

```
@WebServlet("/cookie1")
doPost(HttpServletRequest req, HttpServletResponse res)
{
String mail = req.getParameter("mail");
String pass = req.getParameter("pass");
Cookie c1 = new Cookie("mail", mail);
Cookie c2 = new Cookie("pass", pass);
c1.addCookie("mail", mail);
c2.addCookie("pass", pass);
resp.addCookie(c1);
resp.addCookie(c2);
RequestDispatcher rd = req.getRequestDispatcher("Cookie
rd.include(req, resp);
}
```

CookieTwoServlet.java

```
@WebServlet("/Cookie2")
doPost(HttpServletResponse resp, HttpServletRequest req)
{
Cookie[] Cookies = req.getCookie();
        String mail = null;
        String pass = null;
    if (Cookies.length>0) {
        for(Cookie c : Cookies) {
            if (c.getValue().equals("abc@gmail.com")
                mail = c.getValue();
            else if (c.getValue().equals("abc")
                pass = c.getValue();
        }
    }
    if(mail!=null && Pass!=null)
        PrintWrite pw = resp.getWriter();
        pw.println("login Success");
    else
        PrintWrite pw = resp.getWriter();
        pw.println("login failed");
}
```

Session

Email <input type="text" name="mail">
Password <input type="text" name="pass">

Sessionone.java

```
@WebServlet("/Session1")
dopost(req, resp){
String mail = req.getParameter("mail");
String pass = req.getParameter("pass");
RequestDispatcher rd = req.getRequestDispatcher
("Session2");
rd.include(req, resp);
HttpSession Session = req.getSession();
(I)
(I returns => HttpSession (I))
Session.setAttribute("mail", mail);
Session.setAttribute("pass", pass);
}
```

Sessiontwo.java

```
@WebServlet("/Session2")
dopost(req, resp){
HttpSession Session = req.getSession();
String mail = (String) Session.getAttribute("mail");
String pass = (String) Session.getAttribute("pass");
if(!mail.isBlank() && !pass.isBlank()){
RequestDispatcher rd = req.getRequestDispatcher
("Home.jsp");
rd.include(req, resp); }
else {
RequestDispatcher rd = req.getRequestDispatcher
("session.html");
rd.include(req, resp); } }
```

29/11/2023

JSP

Java Server Pages

Same as HTML But we can write/use Java Code with the help of JSP Tags

List of Tag Jat That are used to write Java Code in Java

1.) Sxciptlet tags (Act like a Service Method) <% %> (methods/constructors X)

2.) Declarative tags (Variable, Methods) <%! %>

3.) Expression tags (Printing Statements) <%= %>

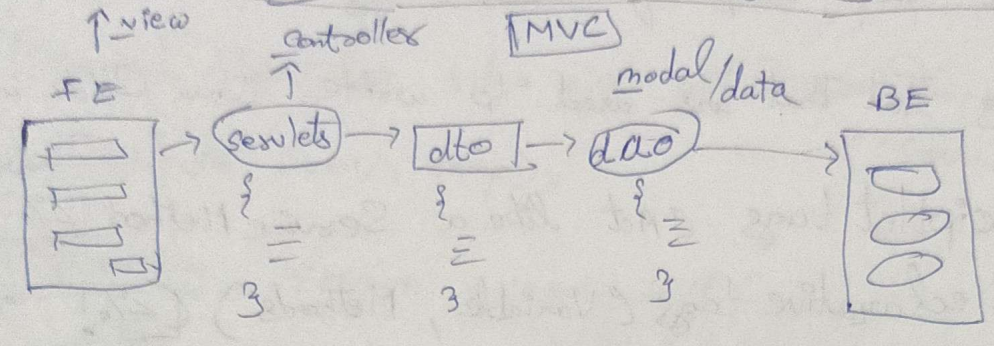
4.) Directive tags <%@ %> (ctrl+space) (to import)

home.jsp

```
<%@ import Student %> -> Directive
<%! int a=10%> -> Declarative.
<%= a %> =>Expression.
<% Student st = new Student();%> => Sxciptlet
<% List<Student> stud = request.getAttibute("Student")
                                                %>
<% for(Student S : stud) { %>
        <% int id = S.getId(); %>
<%= id %>
<% } %>
```

DTO ⇒ A classes which able to Transfer the Data is called [Data Transfer Object] Java Beans.

DAO ⇒ A classes which able to Access the Data is called [Data Access objects]



MVC => Modal View Controller

```
< form  action = "SObjects"  method = "post" >
    Id : < input  type = "text"  name = "id" >
    Name : < input  type = "text"  name = "name" >
    Contact : < input  type = "text"  name = "Contact" >
        < input  type = "Submit" >
</form>
```

```
@WebServlet("/SObject")
class ServletObjects extends HttpServlet
{
    doPost ( req, resp ) {
        String int id = Integer.parseInt ( req.getParameter(
        String name = req.getParameter("name");
        long Contact = req. Long.parseLong (
            req.getParameter ("Contact"))
```

```
Student s = new Student();
{
    s.setId(id);
    s.setName(name);
    s.setContact(contact);
```

```
}
Student
StudentJDBC sj = new StudentJDBC();
    sj.SaveStudent(Student s);
    sj.FindById(Student s);
    sj.deleteById(Student s);
    sj.findAll();   }
```

DTO

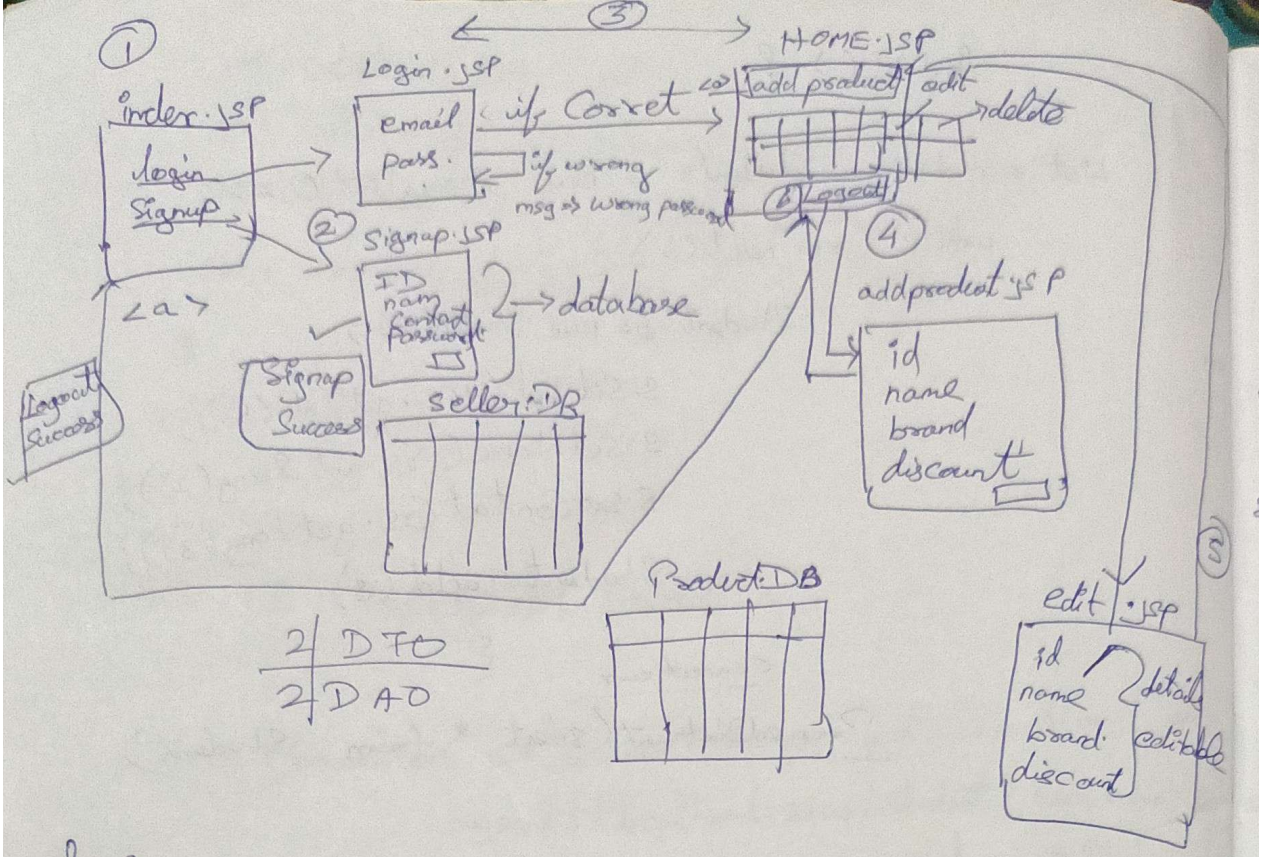
```
class : Student {
    private int id;
    private string name;
    private long contact;
    getters setters methods.
}
```

DAO

```
class StudentJDBC {
public int SaveStuden(Studen s).
    {
Class.forName("com.mysql.cj.jdbc.Driver");
Connection c = DriverManager.getConnection("jdbc:mysql://localhost:
                                            3306/student","root","root");
PreparedStatement pst = c.prepareStatement("insert into
                        student values(?,?,?)");
```

```
        pst.setInt(1, s.getid());
        pst.setString(2, s.getname());
        pst.setLong(3, s.getContact());
        pst.executeUpdate();
    }
    return SaveStudent;
}

public int findById(Student s)
    {
    Connection c=DriverManager.getConnection("jdbc:
            mysql://localhost:3306/Student?user=root&
                                password=root");

    PS    pst = c.prepareStatement("select * name from
                    Student where id=?");
            pst.setid(1, getid());
            pst.executeQuery();
        return
    }

public int ddeleteById(Student s)
    {
    Connection c=DriverManager.getConnection("jdbc:
            mysql://localhost:3306/Student?user=ro
                                Password=root");

    PS pst = c.prepareStatement("~~select~~ delete from
                    Student where id=?");
            pst.setid(1, getid());
            pst.executeUpdate();
        return
    }
```

```
findAll() {

List<Student> Students = new ArrayList<>();
while (rs.next()) {
    Student s = new Student();
    s.setId(rs.getInt(1));
    s.setName(rs.getString(2));
    s.setContact(rs.getLong(3));
    Student.add(s)
}
Connection
PreparedStatement(select * from Student)

}
return Students;

}
```



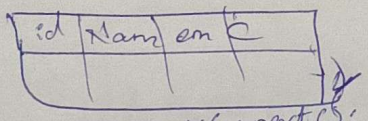
login

Seller findbyEmail(String email)

("select * from seller where email = ?")

pst: setString(1, email);

Resultset rs = pst.executeQuery();



rs.next();

rs.getInt(1);

int SaveSeller(SellerDto s){ }

SellerDto findByEmail (String email) { }

1.) DTO
- product_dto
- Seller_dto

2.) DAO
- product_dao
  - int saveProduct (Product_dto p) { }
  - int updateProduct (Product_dto p) { }
  - int deleteProduct(int id) { }
  - product_dto findById (int id { }
  - List<product_dto> findAll() { }
- seller_dao
  - int saveSeller (Seller_dto s) { }
  - Seller_dto findByEmail (String Email) { }

3.) DB
- SellerDb
- productDb

4.) Controllers -> Servlets

JSP

1.) index.jsp (SignUp, Login buttons)
2.) Login.jsp
3.) Sigup.jsp
4.) Home.jsp (Logout button)
5.) addproduct.jsp
6.) edit.jsp