

Dt : 15/9/2022(day-1)

Java:

part-1 : CoreJava

part-2 : AdvJava

Note:

Hibernate with JPA, Spring, WebServices

Note:

Security, performance...

part-1 : CoreJava

1. Variables

2. Methods

3. Blocks

4. Constructors

5. Class

6. Interface

7. AbstractClass

**imp*

part-2 : AdvJava

=>AdvJava means "Application development".

=>AdvJava provide the following technologies to construct

WebApplications:

1.JDBC

2.Servlet

3.JSP

1.JDBC:

=>JDBC stands for 'Java DataBase Connectivity' and which is used to establish communication b/w JavaProgram and DataBase product.

2.Servlet:

=>The JavaProgram which is executed in Server environment is known as Server Program or Servlet Program.

3.JSP:

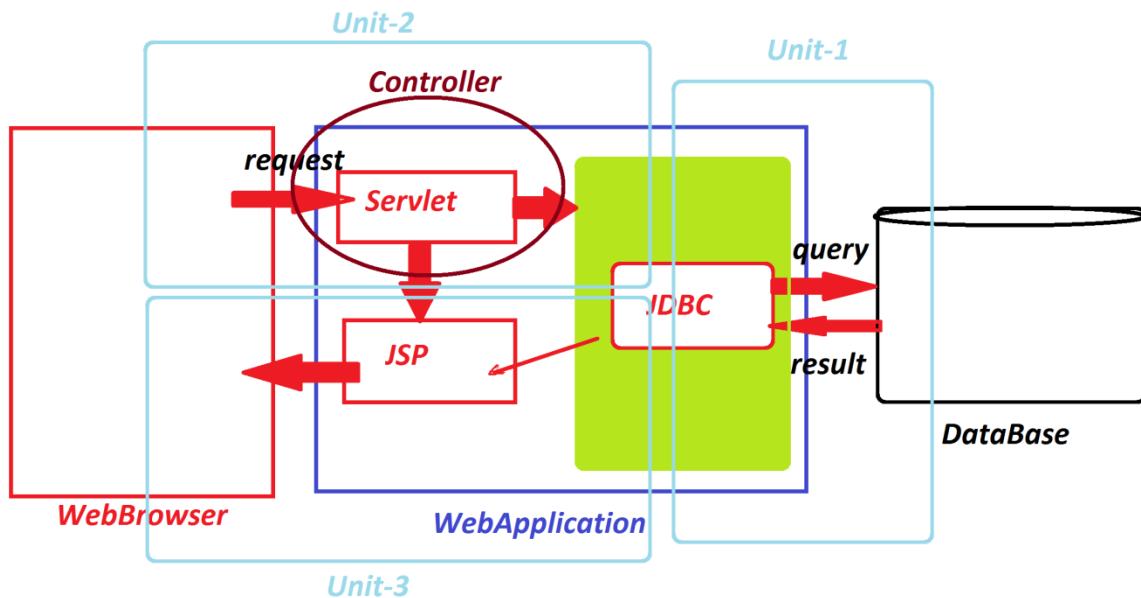
=>JSP stands for 'Java Server page' and which is response from WebApplication.

faq:

define WebApplications?

=>The applications which are executing in Web environment or Internet Environment are known as WebApplications.

Web Application Architecture:



faq:

define Storage?

=>*The memory location where the data is available to access is known as Storage.*

Types of Storages:

=>*According to Java application development the storages are categorized into four types:*

(a)Field Storage

(b)Object Storage

(c)File Storage

(d)DataBase Storage

(a)Field Storage:

=>The memory generated to hold single data value is known as

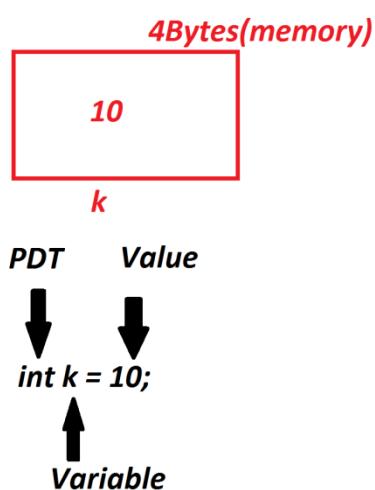
Field Storage.

Note:

=>**Field Storages are generated when we use Primitive datatypes.**

=>**The Field Storages can be in Class or Object or method.**

Ex:



(b)Object Storage:

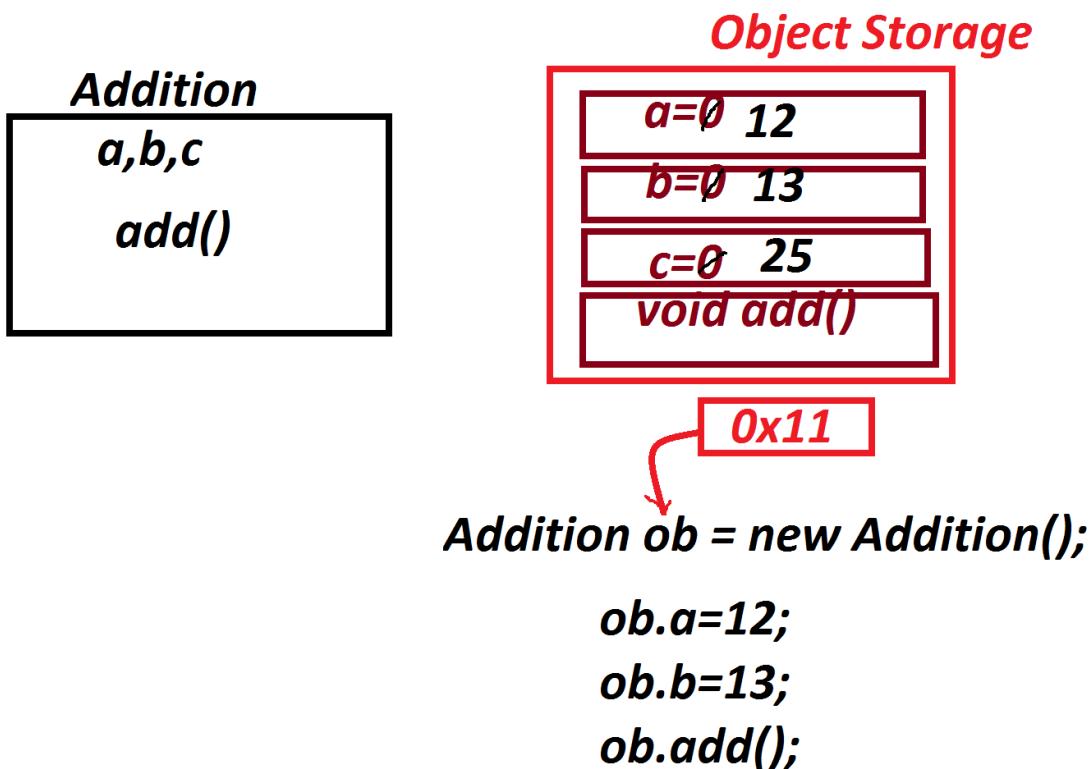
=>The memory generated to hold group members is known as

Object Storage.

Note:

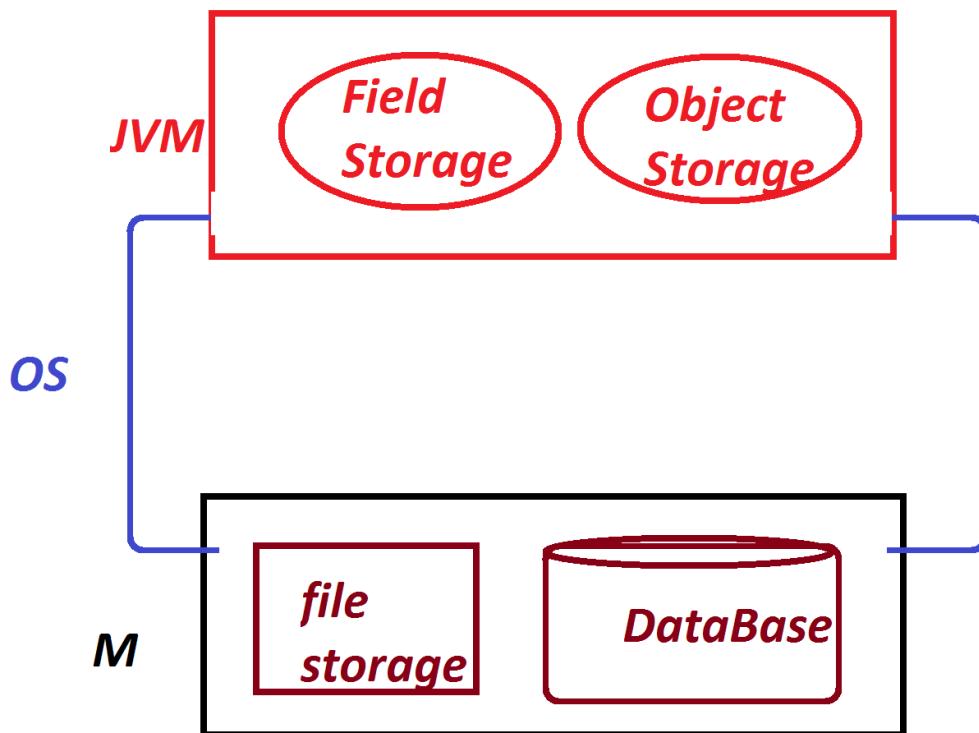
=>**Object Storages are generated when we use Non-Primitive datatypes.**

Ex:



Note:

=>The field Storages and Object Storages which are generated part of JVM will be destroyed automatically when JVM shutdowns
=>when we want to have permanent storage for JavaApplication then we must use File Storage or DataBase Storage.



Dt : 16/9/2022(day-2)

(c)File Storage:

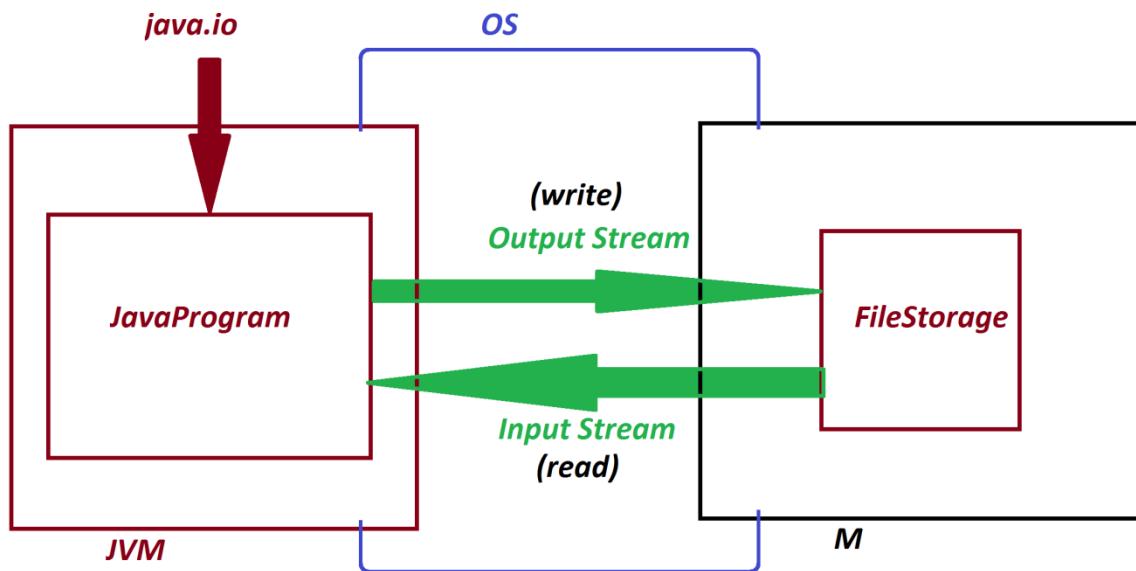
=>The smallest permanent storage of ComputerSystem which is controlled and managed by the OperatingSystem is known as File Storage.

Note:

=>In the process of establishing communication b/w JavaProgram and FileStorage, the JavaProgram must be constructed using 'Classes and

Interfaces' available from "java.io" package.

Diagram:



faq:

Dis-Advantages of File Storage:

- (a) Data Redundancy
- (b) Data Inconsistency
- (c) Data Integrity
- (d) Data Sharing
- (e) Data Security

(a) Data Redundancy:

=> In File Storages replication of data is generated, which means
duplicate data is generated.

(b) Data Inconsistency:

=> Because Data Redundancy the data may be Inconsistent.

(not same)

(c) Data Integrity:

=> Data Integrity is not possible in File Storages, because the data is available in Scatter form.

(d) Data Sharing:

=> Because of Scattered data, the data sharing is not possible.

(e) Data Security :

=> Security not available in File Storages

Note:

=> DisAdvantages of FileStorage can be Overcomed using DataBase Storage.

*imp

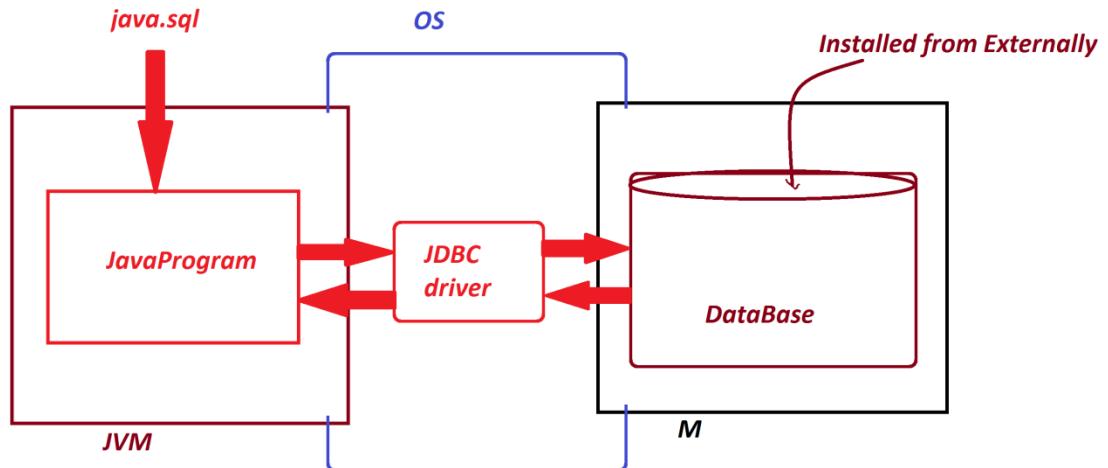
(d) DataBase Storage:

=> The Largest permanent Storage of ComputerSystem which is installed from externally is known as DataBase Storage.

Note:

=>In the process of establishing Communication b/w JavaProgram and DataBase product, the JavaProgram must be constructed using 'Classes and Interfaces' available from java.sql package and it must use "JDBC driver".

Diagram:



faq:

define API?

=>API stands for 'Application Programming Interface', and which provide 'Classes and Interfaces' used in constructing applications

=>API means package.

CoreJava:

java.lang - language package

java.util - Utility package

java.io - Input Output Stream package

java.net - Networking package

AdvJava:

java.sql - DataBase package

javax.servlet - Servlet Programming package.

javax.servlet.jsp - JSP Programming

=====

faq:

define driver?

=>*The small s/w program used by OperatingSystem to control the resources of ComputerSystem is known as driver.*

=>*driver will establish communication b/w two components.*

Ex:

audio driver

N/w driver

...

faq:

define JDBC-driver?

=>*JDBC-driver is used to establish connection b/w JavaProgram and DataBase product.(Java DataBase Connectivity Driver)*

Types of JDBC drivers:

=>JDBC drivers are categorized into four types:

1.JDBC-ODBC bridge driver(Type-1)

2.Native API driver(Type-2)

3.Network Protocol driver(Type-3)

4.Thin driver(Type-4)

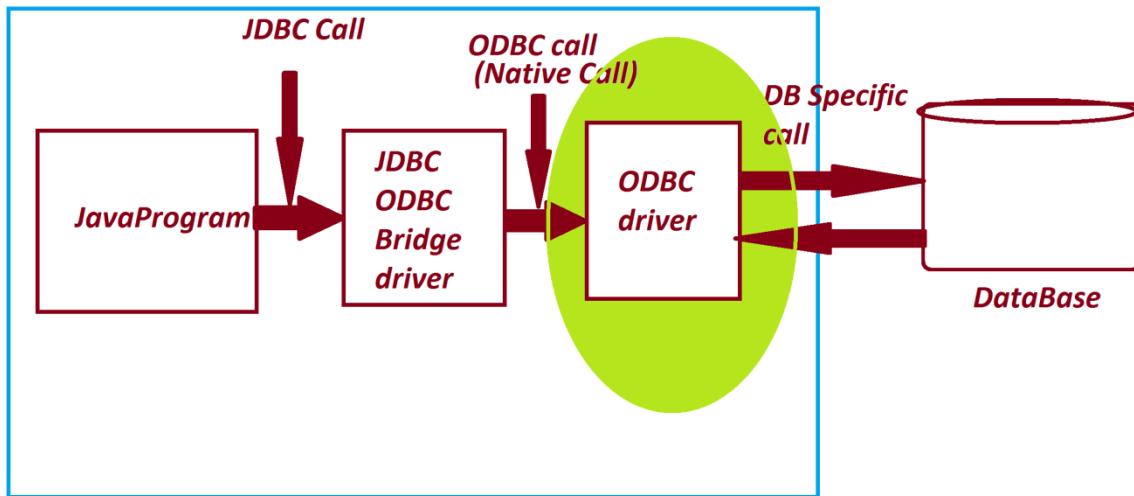
Dt : 17/9/2022

1.JDBC-ODBC bridge driver(Type-1):

=>This Type-1 driver will convert JDBC Call into ODBC call and

this ODBC will raise DB-Specific call for connection.

Diagram:



Dis-Advantages:

=>Number of internal conversions are increased and consumes more execution time, and degrades the performance of an application.

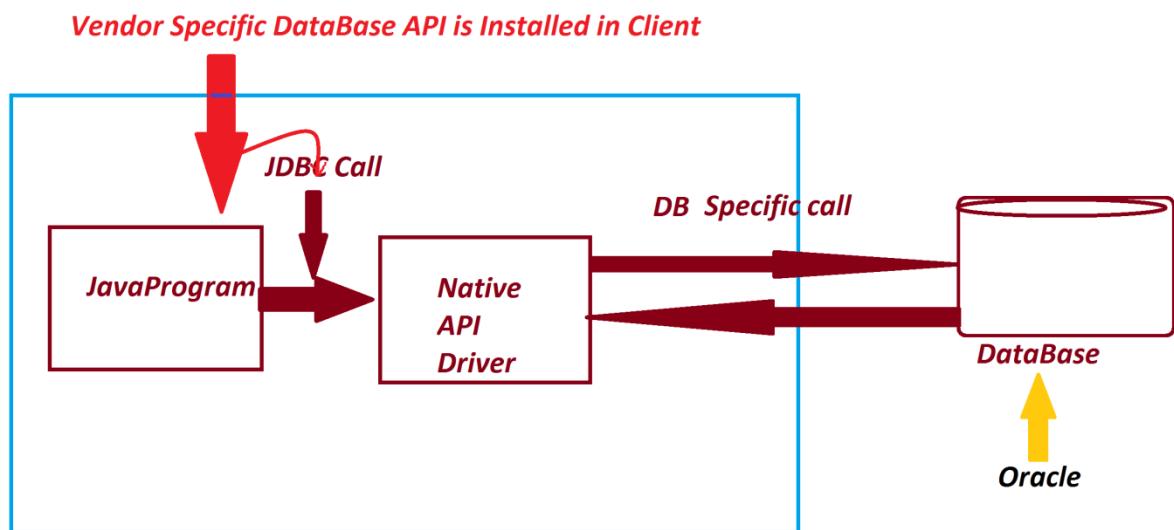
Note:

=>Type-1 driver is not available from Java8 version onwards.

2.Native API driver(Type-2):

=>In Type-2 driver 'Vendor Specific DataBase API' will Support JavaProgram to raise JDBC call, and this JDBC call is Converted into DB Specific call directly using Native API driver.

Diagram:



Note : DataBase Native API provided by Vendor

Advantage:

=>Type-2 driver is more efficient than Type-1 driver,because less number of Conventions and ODBC is not available.

DisAdvantage:

=>Type-2 driver will make application DataBase dependent,because the Client Computer must be installed with DataBase Specific API.

Note:

=>Type-1 and Type-2 are not Complete Java drivers.

3.Network Protocol driver(Type-3):

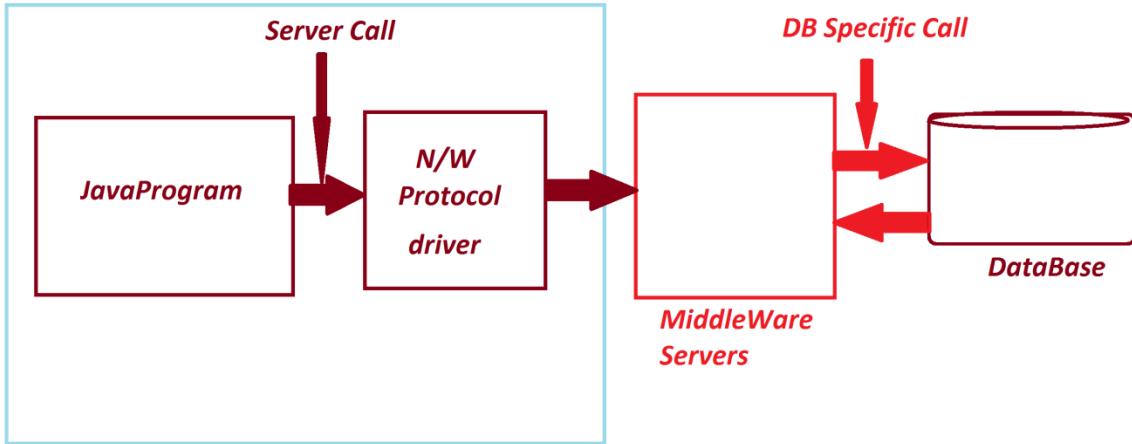
=>The Type-3 driver will establish connection b/w JavaProgram to MiddleWare Server, and this MiddleWare server internally having DB Specific Connection.

Advantage:

=>Type-3 is more efficient than Type-1 and Type-2,becuase OJDBC and DB-specific API is not involved.

Dis-Advantage:

=>In Type-3,Network Components are participated in execution process and increases the execution time, and which degrades the performance of an application.



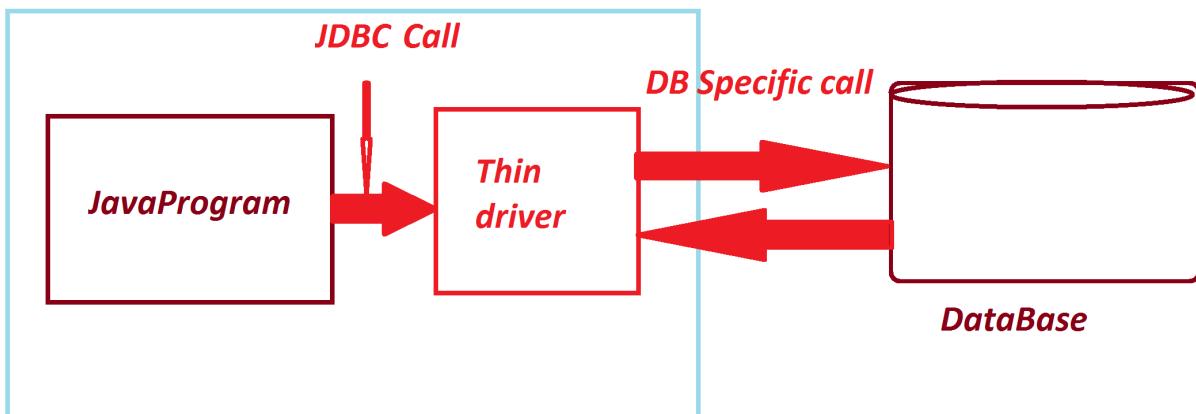
**imp*

4. Thin driver(Type-4):

=>*In Type-4 driver the JDBC call is converted into DB Specific call directly for communication.*

=>*Which is Complete Java Driver and HighPerformance driver.*

Diagram:



faq:

define ODBC driver?

=>*ODBC stands for 'Open DataBase Connectivity' driver and which is Native and Platform dependent driver.*

=>*ODBC driver internally having c or c++ code.*

Note:

=>*Type-5 driver is not officially accepted by Oracle Corporation.*

Dt : 19/9/2022

**imp*

Making Computer System ready to execute JDBC Application:

step-1 : Download and install Oracle DataBase(Any version)

step-2 : Perform Login process(Checking the Login process)

step-3 : Create table with name Product48

(code, name, price, qty)

primary key : code

```
create table Product48(code varchar2(10),name varchar2(15),
price number(10,2),qty number(10),primary key(code));
```

step-4 : Insert records into table 'Product48'

```
insert into Product48 values('A111','Mouse',234.56,12);
```

```
SQL> select * from Product48;
```

CODE	NAME	PRICE	QTY
A111	Mouse	234.56	12
A222	KBBB	232.56	12
A333	CDDR	134.56	10

```
SQL>
```

*step-5 : Find DB-jar and copy the DB-jar into one User defined
folder of any drive.*

DB-Jar file available at "lib" folder of Oracle Product:

C:\oraclexe\app\oracle\product\11.2.0\server\jdbc\lib

ojdbc6.jar

faq:

define JAR?

=>JAR means Java Archive and which is compressed format of
class files.

=>JAR is a ready executable file.

step-6 : Find PortNo and Service_name of Oracle Product

=>Open 'tnsnames.ora' file from 'ADMIN' folder of 'NETWORK' to find
PortNo and Service_name

PortNo : 1521

Service_name : XE

Available at :

C:\oraclexe\app\oracle\product\11.2.0\server\network\ADMIN

tnsnames.ora

=====

**imp*

JDBC API?

=>'java.sql' package is known as JDBC API.

=>'java.sql.Connection' interface is a root of JDBC API.

=>The following are some important methods of 'Connection':

1.createStatement()

2.prepareStatement()

3.PreparedCall()

4.setAutoCommit()

5.getAutoCommit()

6.setSavepoint()

7.getSavepoint()

8.commit()

9.rollback()

10.close()

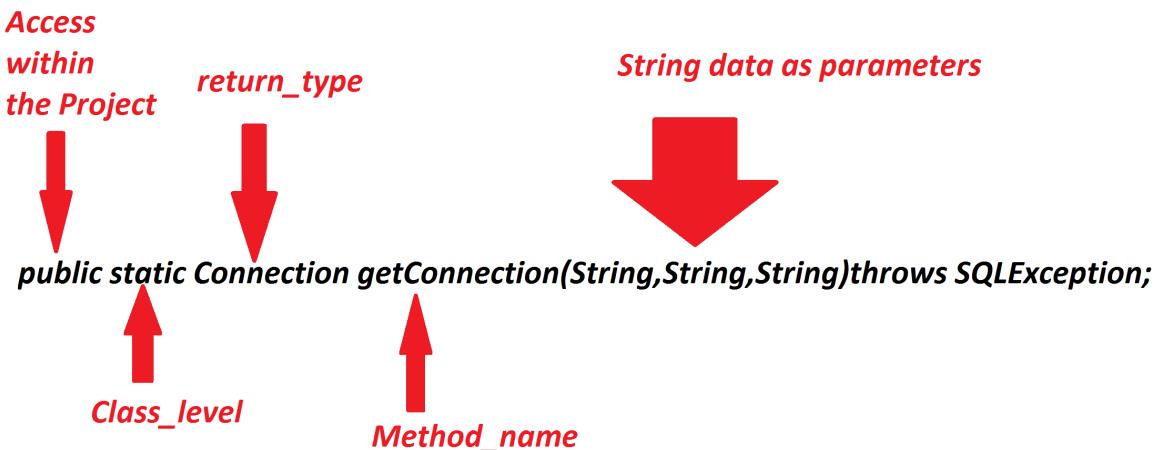
=>we use `getConnection()` method to create the implementation object of 'Connection' interface.

=>This `getConnection()` method is available from 'DriverManager' class

Method signature of `getConnection()`:

```
public static java.sql.Connection getConnection  
(java.lang.String,java.lang.String,java.lang.String)  
throws java.sql.SQLException;
```

diagram:



syntax:

`Connection con = DriverManager.getConnection`

`("DB-URL", "UserName", "PassWord");`

Dt : 20/9/2022

DB-URL => jdbc:oracle:thin:@localhost:1521:xe

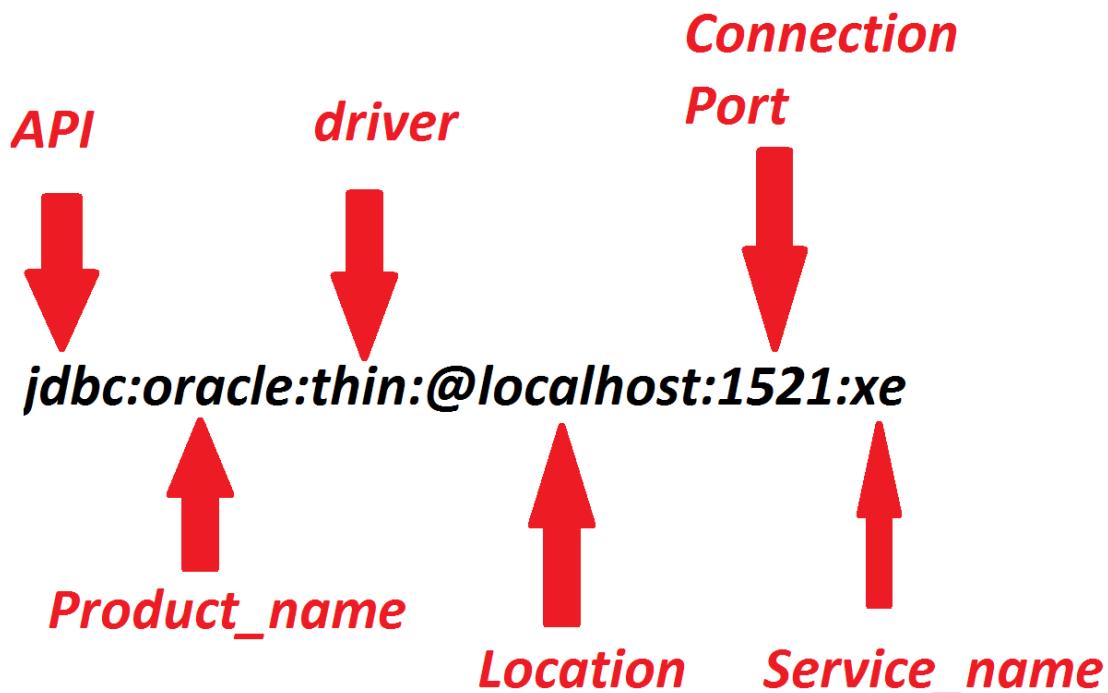
UserName=> system

PassWord=> manager

syntax:

`Connection con = DriverManager.getConnection`

`("jdbc:oracle:thin:@localhost:1521:xe", "system", "manager");`



**imp*

JDBC statements:

=>*JDBC statements specify the type of operations performed on*

DataBase product.

=>*JDBC statements are categorized into three types:*

1.Statement

2.PreparedStatement

3.CallableStatement

1.Statement:

=>'Statement' is an interface from `java.sql` package and which is used to execute normal queries without IN-Parameters.

(Normal Queries : `create,insert,retrieve,update,delete`)

=>we use `createStatement()` method from 'Connection' interface to create implementation object of 'Statement'

syntax:

`Statement stm = con.createStatement();`

=>The following are some important methods of Statement:

(a)`executeQuery()`

(b)`executeUpdate()`

(a)`executeQuery()`:

=>`executeQuery()` method is used to execute select-Queries.

Method Signature:

```
public abstract java.sql.ResultSet executeQuery(java.lang.String)
throws java.sql.SQLException;
```

syntax:

`ResultSet rs = stm.executeQuery("query");`

Ex:

`ResultSet rs = stm.executeQuery("select * from Product48");`

(b)executeUpdate():

=>executeUpdate() method is used to execute Non-Select-Queries like Create,insert,update and delete.

Method Signature:

```
public abstract int executeUpdate(java.lang.String) throws  
java.sql.SQLException;
```

syntax:

```
int k = stm.executeUpdate("query");
```

**imp*

Construct JDBC Application Using IDE Eclipse:

step-1 : Open IDE Eclipse,while opening name the WorkSpace and click 'Launch'.

step-2 : Create JavaProject

step-3 : Add DB-JAR file to Java Project

RightClick on JavaProject->Build path->Configure Build path->

Libraries->select 'classpath' and click 'Add External Jars'->

Browse and select DB-Jar file from User defined folder->Open->

Apply->Apply and Close.

step-4 : Create package

step-5 : Create class to construct JDBC code

Ex : DBCon1.java

```
package test;
import java.sql.*;
public class DBCon1 {
    public static void main(String[] args) {
        try {
            Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe", "system", "manager");
            Statement stm = con.createStatement();
            ResultSet rs = stm.executeQuery
                ("select * from Product48");
            while(rs.next())
            {
                System.out.println(rs.getString(1)+"\t"+
                    rs.getString(2)+"\t"+rs.getFloat(3)+"\t"+
                    rs.getInt(4));
            } //end of loop
            con.close();
        } catch (Exception e) {e.printStackTrace();}
    }
}
```

o/p:

A111 Mouse 234.56 12
A222 KBBB 232.56 12
A333 CDDR 134.56 10

=====

Assignment:

Create DB Table BookDetails48(bcode,bname,bauthor,bprice,bqty)

primary key : bcode

Insert min 5 records into BookDetails48 from SQL-CommandLine

Construct JDBC Application to display BookDetails

=====

Dt : 21/9/2022

Ex-2 :

Construct JDBC Application to read Product details from Console

input and insert into DB-Table(Product48).

DBCon2.java

```
package test;

import java.util.*;
import java.sql.*;

public class DBCon2 {
    public static void main(String[] args) {
        try {
            Scanner s = new Scanner(System.in);
            System.out.println("Enter the ProdCode:");
            String code = s.nextLine();
            System.out.println("Enter the ProdName:");
            String name = s.nextLine();
            System.out.println("Enter the ProdPrice:");
        }
    }
}
```

```

float price = Float.parseFloat(s.nextLine());
System.out.println("Enter the ProdQty:");
int qty = Integer.parseInt(s.nextLine());

Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
Statement stm = con.createStatement();
int k = stm.executeUpdate
        ("insert into Product48 values('"+code
        +"','"+name
        +"','"+price+"','"+qty+"')");
if(k>0) {
    System.out.println
        ("Product details inserted Successfully");
}
con.close();
s.close();
}catch(Exception e) {e.printStackTrace();}
}

```

o/p:

Enter the ProdCode:

A121

Enter the ProdName:

TTT

Enter the ProdPrice:

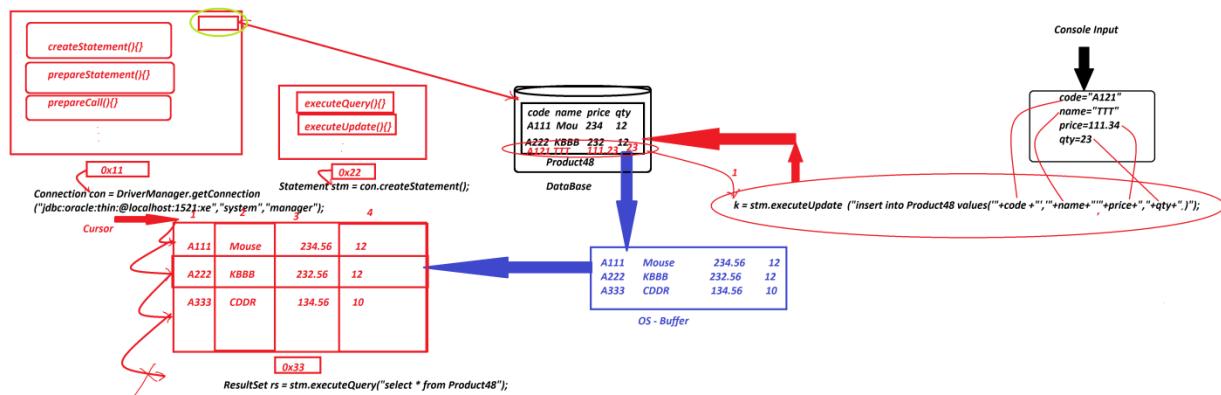
111.23

Enter the ProdQty:

23

Product details inserted Successfully

Diagram:



Assignment:

**Construct JDBC Application to read BookDetails from Console and
insert into DB Table(BookDetails48)**

Ex-3 :

**Construct JDBC Application to display product details based on
ProdCode.**

DBCon3.java

```
package test;

import java.sql.*;
import java.util.*;

public class DBCon3 {

    public static void main(String[] args) {

        try {

            Scanner s = new Scanner(System.in);

            System.out.println("Enter the ProdCode:");

            String pCode = s.nextLine();

            Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

            Statement stm = con.createStatement();

            ResultSet rs = stm.executeQuery
                ("select * from Product48 where code='"+pCode+"');

            if(rs.next()) {

                System.out.println(rs.getString(1)+"\t"
                    +rs.getString(2)+"\t"+rs.getFloat(3)+"\t"
                    +rs.getInt(4));

            }else {

                System.out.println("Invalid prodCode.... ");

            }

            con.close();
        }
    }
}
```

```
s.close();  
}  
}  
}
```

o/p:

Enter the ProdCode:

A222

A222 KBBB 232.56 12

Assignment:

Construct JDBC Application to display Book details based on BookCode.

**imp*

2.PreparedStatement:

*=>PreparedStatement is an interface from java.sql package and
which is used to execute normal queries with IN-Parameters.*

*=>we use prepareStatement() method from 'Connection' interface
to create PreparedStatement object.*

syntax:

PreparedStatement ps = con.prepareStatement("query-structure");

Note:

=>query-structure means format of query without Values.

=>The following are two important methods of PreparedStatement:

(a)executeQuery()

(b)executeUpdate()

(a)executeQuery():

=>executeQuery() method is used to execute select-Queries.

Method Signature:

`public abstract java.sql.ResultSet executeQuery()`

`throws java.sql.SQLException;`

syntax:

`ResultSet rs = stm.executeQuery();`

(b)executeUpdate():

=>executeUpdate() method is used to execute Non-Select-Queries

like Create,insert,update and delete.

Method Signature:

`public abstract int executeUpdate() throws java.sql.SQLException;`

syntax:

`int k = stm.executeUpdate();`

=====

Application:

DBTable : Employee48(id,name,desg,bsal,totsal)

```
create table Employee48(id varchar2(10),name varchar2(15),
desg varchar2(10),bsal number(10),totsal number(10,2),
primary key(id));
```

DBCon4.java

```
package test;

import java.util.*;
import java.sql.*;

public class DBCon4 {

    public static void main(String[] args) {

        try {

            Scanner s = new Scanner(System.in);

            Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

            PreparedStatement ps1 = con.prepareStatement
                ("insert into Employee48 values(?,?,?,?,?)");
            //Compilation process

            PreparedStatement ps2 = con.prepareStatement
                ("select * from Employee48");
            //Compilation process

            while(true) {

                System.out.println("====Choice====");
                System.out.println
```

```

("1.AddEmployee\n2.ViewEmployees\n3.exit");

System.out.println("Enter the Choice:");

switch(Integer.parseInt(s.nextLine())) {

case 1:

    System.out.println("Enter the empId:");

    String id = s.nextLine();

    System.out.println("Enter the empName:");

    String name = s.nextLine();

    System.out.println("Enter the empDesg:");

    String desg = s.nextLine();

    System.out.println("Enter the empBSal:");

    int bSal = Integer.parseInt(s.nextLine());

    float totSal = bSal+(0.93F*bSal)+(0.63F*bSal);

    ps1.setString(1,id);

    ps1.setString(2,name);

    ps1.setString(3,desg);

    ps1.setInt(4,bSal);

    ps1.setFloat(5,totSal);

    int k = ps1.executeUpdate();//Execution process

    if(k>0) {

        System.out.println("Employee Added Successfully");

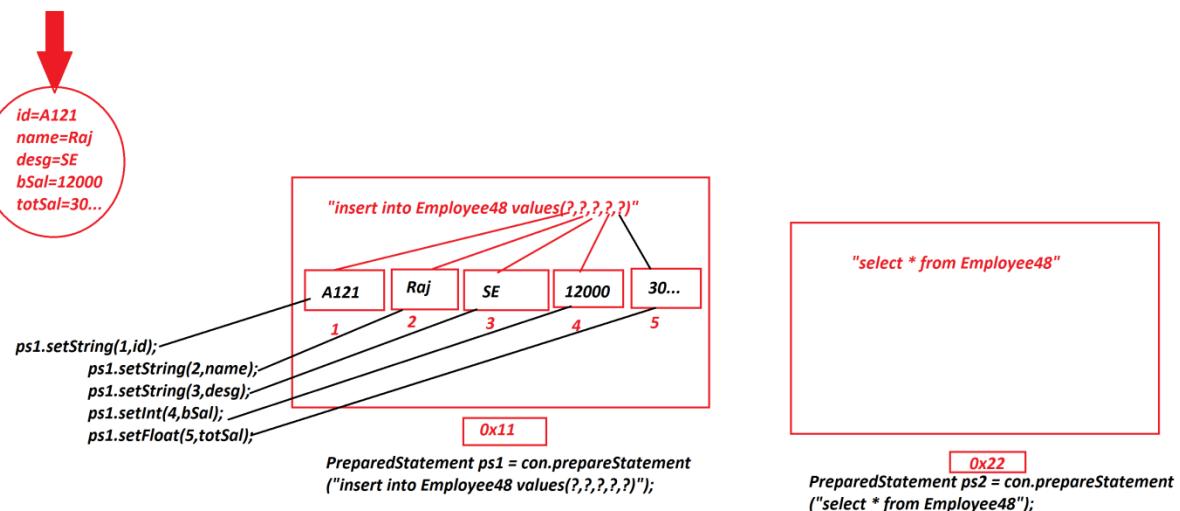
    }

}

```

```
break;  
case 2:  
    ResultSet rs = ps2.executeQuery();  
    while(rs.next()) {  
        System.out.println(rs.getString(1)+"\t"  
            +rs.getString(2)+"\t"  
            +rs.getString(3)+"\t"  
            +rs.getInt(4)+"\t"+rs.getFloat(5));  
    //end of loop  
    break;  
case 3:  
    System.out.println("Operations Stopped...");  
    System.exit(0);  
default:  
    System.out.println("Invalid Choice...");  
    //end of switch  
  //end of while  
}catch(Exception e) {e.printStackTrace();}  
}
```

Console



Dt : 22/9/2022

Ex:

Construct JDBC application to perform Update and delete operations

based employee Id.

DBCon5.java

```
package test;

import java.sql.*;
import java.util.*;

public class DBCon5 {

    public static void main(String[] args) {

        try {
            Scanner s = new Scanner(System.in);

```

```

Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

PreparedStatement ps1=con.prepareStatement
("select * from Employee48 where id=?");

PreparedStatement ps2 = con.prepareStatement
("update Employee48 set bsal=?,totsal=? where id=?");

PreparedStatement ps3 = con.prepareStatement
("delete from Employee48 where id=?");

System.out.println("Enter the emplId:");

String id = s.nextLine();

ps1.setString(1, id);

ResultSet rs = ps1.executeQuery();

if(rs.next()) {

    System.out.println("====Choice====");
    System.out.println
("1.ViewDetails\n2.UpdateDetails\n3.DeleteDetails");
    System.out.println("Enter the Choice:");
    switch(Integer.parseInt(s.nextLine())) {

        case 1:
            System.out.println(rs.getString(1)+"\t"+
rs.getString(2)+"\t"+rs.getString(3)+

"\t"+rs.getInt(4)+"\t"+rs.getFloat(5));
            break;

        case 2:
    }
}

```

```

System.out.println("Old bSal:"+rs.getInt(4));

System.out.println("Enter new bSal:");

int newBSal = Integer.parseInt(s.nextLine());

float newTotSal =

+(0.93F*newBSal)+(0.63F*newBSal);

ps2.setInt(1,newBSal);

ps2.setFloat(2,newTotSal);

ps2.setString(3, id);

int k = ps2.executeUpdate();

if(k>0) {

    System.out.println("Details updated Successfully... ");

}

break;

case 3:

ps3.setString(1,id);

int z = ps3.executeUpdate();

if(z>0) {

    System.out.println("Details deleted Successfully... ");

}

break;

default:

System.out.println("Invalid Choice... ");

}//end of switch

}else {

```

```

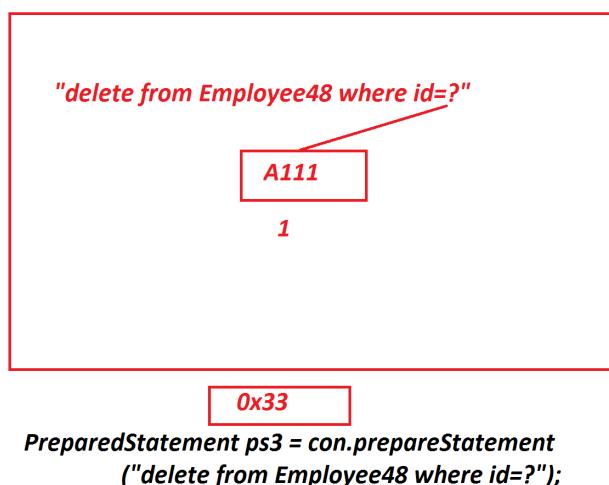
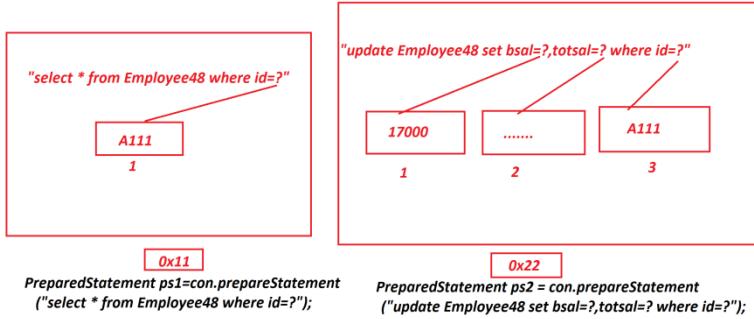
        System.out.println("Invalid empId...");

    }

} catch(Exception e) {e.printStackTrace();}

}

```



Assignment-1:

Construct JDBC application to combine above two applications and perform the following operations on Employee48 table

- 1.AddEmployee**
- 2.ViewAllEmployees**
- 3.ViewEmployeeByEmpId**

- 1.ViewDetails**
- 2.updateDetails**
- 3.DeleteDetails**

Note:

=>repeat the above operations until user exit.

Assignment-2:

DB Table : StudentDetails48

(rollno, name, branch, totmarks, per, result)

primary key : rollno

Construct JDBC Application to perform the following operations on

StudentDetails48 table

- 1.AddStudentDetails**
- 2.ViewAllStudentDetais**
- 3.ViewStudentByRollNo**

Note:

=>repeat the above operations until user exit.

=>Use all Validations

(i)rollNo must be 10 digits

(ii)branch must be in CIVIL or MECK or EEE or ECE or CSE

(iii) rollNo must match the branch

(iv) All Sub marks must be in b/w 0 to 100

(v) Sub marks b/w 0 to 34 result : Fail

Assignment-3:

DB table : UserReg48(uname,pword,fname,lname,city,mid,phno)

primary key : uname and pword

Construct JDBC Application to perform the following operations:

1. UserRegistration

2. UserLogin

=>User login Successfull display the msg as "Welcome fName.."

1. ViewDetails

2. UpdateDetails(city,mid,phno)

faq:

wt is the diff b/w

(i) Statement

(ii) PreparedStatement

(i) Statement :

=>**Statement is used to execute normal queries without IN-Parameters,**

which means when we execute insert-query for multiple times then

the query is compiled multiple times and executes multiple times.

(ii)PreparedStatement:

=>PreparedStatement is used to execute normal queries with IN-parameters, which means when we execute insert-query multiple times then the query is compiled once and executes multiple times.

Note:

=>PreparedStatement will save the execution time and generate HighPerformance of an application when compared to Statement.

Dt : 23/9/2022

*imp

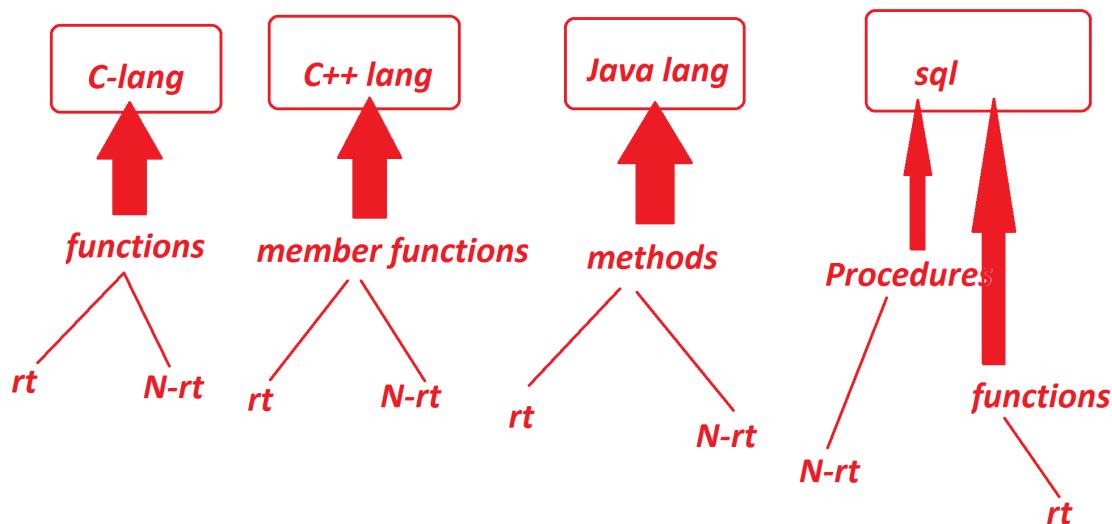
3.CallableStatement:

=>CallableStatement is used to execute procedures and functions from DataBase product.

=>we use prepareCall() method from 'Connection' interface to create the implementation object for CallableStatement interface.

syntax:

```
CallableStatement cs = con.prepareCall("{call proc_name/func_name}");
```



faq:

define Procedure?

=>Procedure is a set-of-queries executed on DataBase product

and which will not return any value after execution.

=>which is also known as Stored procedure.

structure of procedure:

create or replace procedure Proc_name

(para_list) is

begin

query1;

query2;

...

end;

/

define Function?

=>Function is also a set-of-queries executed on DataBase product

and which will return the value after execution.

=>which is also known as Stored Function.

structure of Function:

create or replace function Func_name

(para_list) return data_type as var data_type;

begin

```
query1;  
  
query2;  
  
....  
  
return var;  
  
end;  
  
/  
  
=====
```

*imp

Creating and Execution Procedure:

Step-1 : Create the following DB-Tables

EmpData48(id,name,desg)

EmpAddress48(id,hno,sname,city,pincode)

EmpContact48(id,mid,phno)

EmpSalary48(id,bsal,totsal)

```
create table EmpData48(id varchar2(10),name varchar2(15),  
desg varchar2(10),primary key(id));
```

```
create table EmpAddress48(id varchar2(10),hno varchar2(15),  
sname varchar2(15),city varchar2(15),pincode number(10),  
primary key(id));
```

```
create table EmpContact48(id varchar2(10),mid varchar2(25),
phno number(15),primary key(id));
```

```
create table EmpSalary48(id varchar2(10),bsal number(10),
totsal number(10,2),primary key(id));
```

step-2 : Construct procedure to update DB tables

```
create or replace procedure EmpDetails48
(eid varchar2,ename varchar2,edesg varchar2,ehno varchar2,
esname varchar2,ecity varchar2,epincode number,emid varchar2,
ephno number,ebsal number,etotsal number) is
begin
insert into EmpData48 values(eid,ename,edesg);
insert into EmpAddress48 values(eid,ehno,esname,ecity,epincode);
insert into EmpContact48 values(eid,emid,ephno);
insert into EmpSalary48 values(eid,ebsal,etotsal);
end;
/
```

step-3 : Construct JDBC Application to execute procedure

DBCon5.java

```
package test;
import java.util.*;
```

```

import java.sql.*;
public class DBCon5 {
    public static void main(String[] args) {
        try {
            Scanner s = new Scanner(System.in);
            System.out.println("Enter the EmpId:");
            String id = s.nextLine();
            System.out.println("Enter the EmpName:");
            String name = s.nextLine();
            System.out.println("Enter the EmpDesg:");
            String desg = s.nextLine();
            System.out.println("Enter the EmpHNo:");
            String hNo = s.nextLine();
            System.out.println("Enter the EmpSName:");
            String sName = s.nextLine();
            System.out.println("Enter the EmpCity:");
            String city = s.nextLine();
            System.out.println("Enter the EmpPinCode:");
            int pinCode = Integer.parseInt(s.nextLine());
            System.out.println("Enter the EmpMailId:");
            String mId = s.nextLine();
            System.out.println("Enter the EmpPhNo:");
            long phNo = Long.parseLong(s.nextLine());
            System.out.println("Enter the EmpBSal:");
            int bSal = Integer.parseInt(s.nextLine());
            float totSal = bSal +(0.93F*bSal) +(0.63F*bSal);
            Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
            CallableStatement cs = con.prepareCall
                ("{call
EmpDetails48(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}");
            cs.setString(1,id);
            cs.setString(2,name);
            cs.setString(3,desg);
            cs.setString(4,hNo);
            cs.setString(5,sName);
            cs.setString(6,city);
            cs.setInt(7,pinCode);
            cs.setString(8, mId);
            cs.setLong(9,phNo);
            cs.setInt(10, bSal);
            cs.setFloat(11, totSal);
            cs.execute(); //Procedure executed
            System.out.println("Emp details updated
successfully..");
            s.close();
        }catch(Exception e) {e.printStackTrace();}
    }
}

```

```
    }  
}
```

o/p:

Enter the EmpId:

A121

Enter the EmpName:

Raj

Enter the EmpDesg:

SE

Enter the EmpHNo:

12-34

Enter the EmpSName:

SRN

Enter the EmpCity:

Hyd

Enter the EmpPinCode:

612345

Enter the EmpMailId:

r@gmail.com

Enter the EmpPhNo:

9898981234

Enter the EmpBSal:

17000

Emp details updated successfully..

*SQL> select * from EmpData48;*

<i>ID</i>	<i>NAME</i>	<i>DESG</i>
A121	Raj	SE

*SQL> select * from EmpAddress48;*

<i>ID</i>	<i>HNO</i>	<i>SNAME</i>	<i>CITY</i>	<i>PINCODE</i>
A121	12-34	SRN	Hyd	612345

*SQL> select * from EmpContact48;*

<i>ID</i>	<i>MID</i>	<i>PHNO</i>
A121	r@gmail.com	9898981234

*SQL> select * from EmpSalary48;*

<i>ID</i>	<i>BSAL</i>	<i>TOTSAL</i>
-----------	-------------	---------------

A121 17000 43520

SQL>

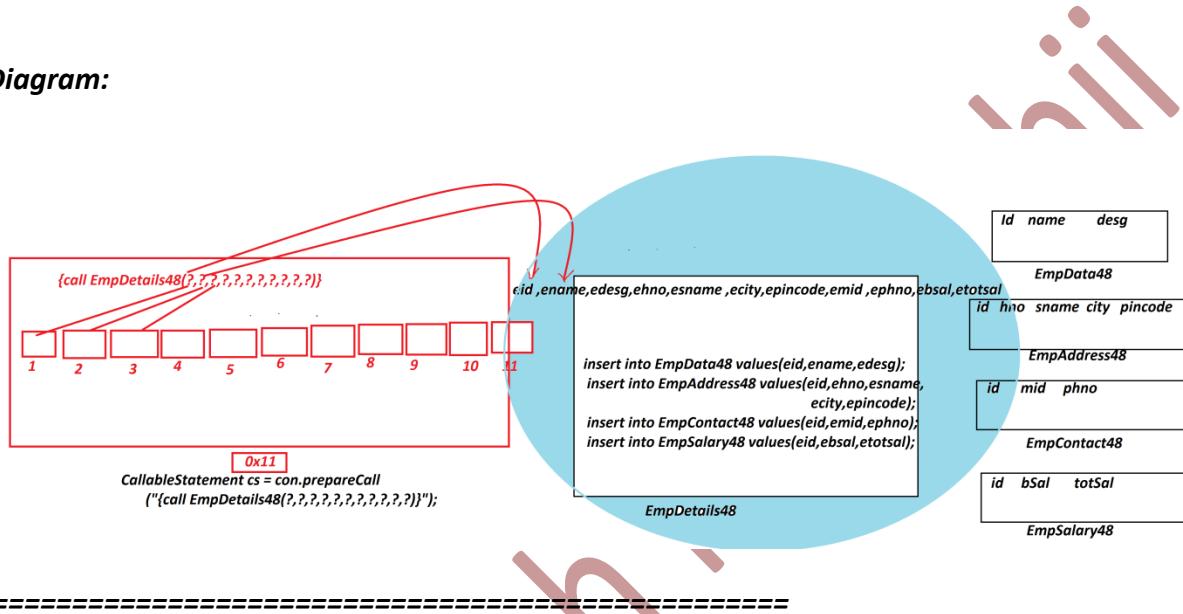
Venkatesh Maiopathii

Dt : 24/9/2022

CallableStatement cs = con.prepareCall

```
{"{call EmpDetails48(?,?,?,?,?,?,?,?,?,?)}"}
```

Diagram:



Types of Stored procedures:

=>Stored Procedures are categorized into two types:

- 1.IN-Parameter Procedures
- 2.OUT-Parameter Procedures

1.IN-Parameter Procedures:

=>The procedures which take the data from JavaPrograms and update DataBase tables are known as In-Parameter Procedures.

Ex:

above application

2.OUT-Parameter Procedures:

=>The procedures which take the data from the DataBase tables and sent to JavaProgram are known as OUT-Parameter Procedures.

Ex:

Construct OUT-parameter procedure to retrieve EmpDetails based on empId.

```
create or replace procedure EmpRetrieve48
(eid varchar2,ename OUT varchar2,edesg OUT varchar2,
ehno OUT varchar2,esname OUT varchar2,ecity OUT varchar2,
epincode OUT number,emid OUT varchar2,
ephno OUT number,ebsal OUT number,etotsal OUT number) is
begin
select name,desg into ename,edesg from EmpData48 where id=eid;
select hno,sname,city,pincode into ehno,esname,ecity,epincode
from EmpAddress48 where id=eid;
select mid,phno into emid,ephno from EmpContact48 where id=eid;
select bsal,totsal into ebsal,etotsal from EmpSalary48
where id=eid;
end;
/
```

Ex : DBCon7.java

```
package test;

import java.util.*;
import java.sql.*;

public class DBCon7 {

    public static void main(String[] args) {

        try {

            Scanner s = new Scanner(System.in);

            System.out.println("Enter the EmpId:");

            String id = s.nextLine();

            Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

            CallableStatement cs = con.prepareCall
                ("'{call EmpRetrieve48(?, ?, ?, ?, ?, ?, ?, ?, ?)}'");

            cs.setString(1,id);

            cs.registerOutParameter(2,Types.VARCHAR);
            cs.registerOutParameter(3,Types.VARCHAR);
            cs.registerOutParameter(4,Types.VARCHAR);
            cs.registerOutParameter(5,Types.VARCHAR);
            cs.registerOutParameter(6,Types.VARCHAR);
            cs.registerOutParameter(7,Types.INTEGER);
            cs.registerOutParameter(8,Types.VARCHAR);
            cs.registerOutParameter(9,Types.BIGINT);
```

```

cs.registerOutParameter(10,Types.INTEGER);
cs.registerOutParameter(11,Types.FLOAT);
cs.execute();//Procedure executed
System.out.println("id:"+id);
System.out.println("Name:"+cs.getString(2));
System.out.println("Desg:"+cs.getString(3));
System.out.println("HNo:"+cs.getString(4));
System.out.println("SName:"+cs.getString(5));
System.out.println("City:"+cs.getString(6));
System.out.println("PinCode:"+cs.getInt(7));
System.out.println("MailId:"+cs.getString(8));
System.out.println("PhNo:"+cs.getLong(9));
System.out.println("BSal:"+cs.getInt(10));
System.out.println("TotSal:"+cs.getFloat(11));
s.close();
}catch(Exception e) {e.printStackTrace();}
}

```

o/p:

Enter the EmpId:

A121

id:A121

Name:Raj

Desg:SE

HNo:12-34

SName:SRN

City:Hyd

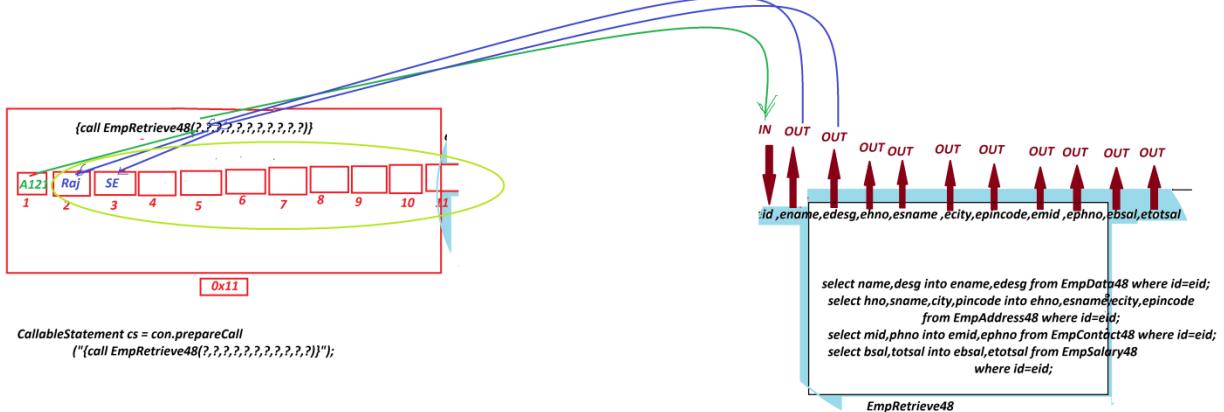
PinCode:612345

MailId:r@gmail.com

PhNo:9898981234

BSal:17000

TotSal:43520.0



dt : 26/9/2022

faq:

define registerOutParameter() method?

=>registerOutParameter() method specify parameter-index and type-of-data registered in para-meter-index-field.

Ex:

```
cs.registerOutParameter(5,Types.VARCHAR);
cs.registerOutParameter(7,Types.INTEGER);
cs.registerOutParameter(9,Types.BIGINT);
cs.registerOutParameter(11,Types.FLOAT);
```

Note:

=>'java.sql.Types' is a class which holds the variable names to specify sql-totypes.

Assignment:

DB Tables :

StuData48(rollno,stuname,branch)
StuAddress48(rollno,hno,sname,city,pincode)
StuContact48(rollno,mailid,phno)

StuResult48

(rollno,sub1,sub2,sub3,sub4,sub5,sub6,totMarks,per,result)

Procedures:

(i)Procedure to insert data to DataBase tables

(ii)Procedure to retrieve data based on rollno

Application:

Construct JDBC Application based on user choice:

1.InsertStudentDetails

2.RetrieveStudentDetails(based on rollNo)

***imp**

Creating and Executing Function:

step-1 : Construct function to retrieve Employee totSal based on emp id.

```
create or replace function RetrieveTotSal48
(eid varchar2) return number as tsal number;
begin
select totsal into tsal from EmpSalary48 where id=eid;
return tsal;
end;
/
```

step-2 : Construct JDBC Application to execute function.

DBCon8.java

```
package test;

import java.util.*;
import java.sql.*;

public class DBCon8 {

    public static void main(String[] args) {

        try {

            Scanner s = new Scanner(System.in);
            System.out.println("Enter the emp-Id:");
            String id = s.nextLine();

            Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

            CallableStatement cs = con.prepareCall
                ("{call ?:=RetrieveTotSal48(?)}");
            cs.setString(2, id);
            cs.registerOutParameter(1,Types.FLOAT);
            cs.execute();
            System.out.println("TotSal : "+cs.getFloat(1));
            con.close();
            s.close();
        }catch(Exception e) {e.printStackTrace();}
    }
}
```

o/p:

Enter the emp-Id:

A121

TotSal : 43520.0

=====

Assignment:

Construct JDBC Application to perform the following specifications

ob DB-Table : BookDetails48

a.read BookCode

b.If bookCode is available then perform the following based on

UserChoice:

1.ViewName

2.ViewAuthor

3.ViewPrice

4.ViewQty

c.If bookcode is not-available display the msg as "Invalid bookCode"

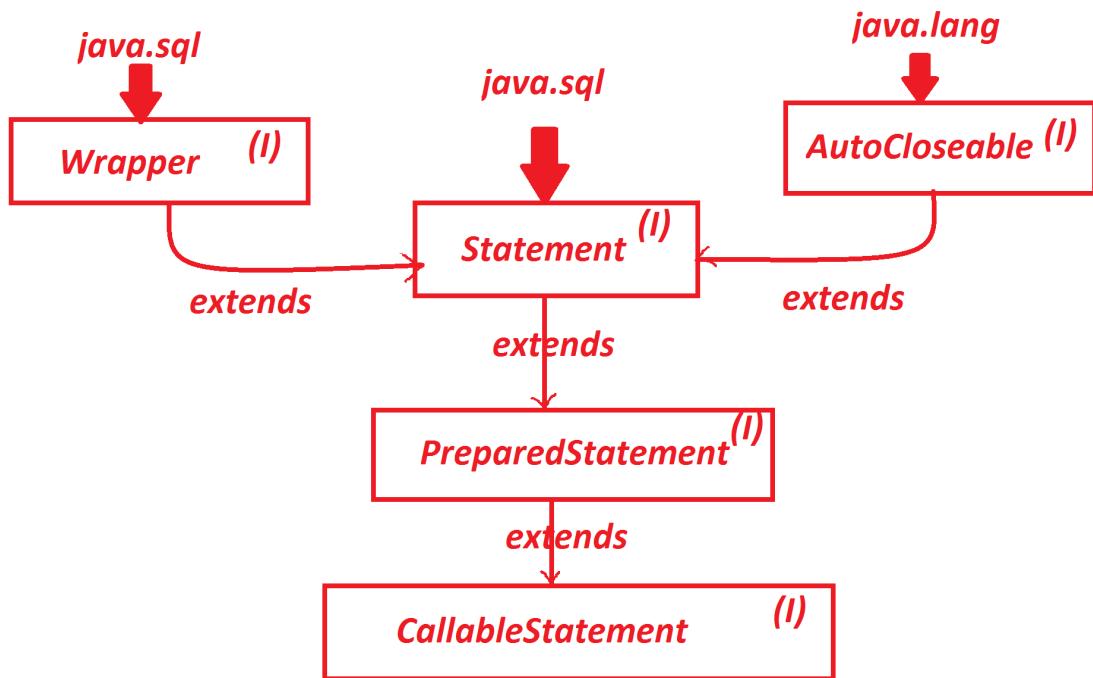
and repeat the choices.

=====

Dt : 27/9/2022

*imp

Hierarchy of JDBC-statements:



*imp

Transaction Management in JDBC:

faq:

define Transaction?

=>*The set-of-statements executed on a resource or resources using*

ACID properties is known as "Transaction"

A - Atomicity

C - Consistency

I - Isolation

D - Durability

A - Atomicity:

=>*The process in which the transaction is completed successfully or Not Completed, is known as Atomicity.*

C - Consistency:

=>*The process in which the selected resource states are same until the transaction is completed, is known as Consistency.*

I - Isolation:

=>*The process in which the multiple users execute independently is known as Isolation.*

D - Durability:

=>*The process in which the transaction state is stored and available when transaction is successfull, is known as Durability.*

faq:

define Transaction Management?

=>The process of controlling the transaction from starting to ending is known as Transaction Management.

=>The following methods are used in Transaction Management:

1.setAutoCommit()

2.getAutoCommit()

3.setSavepoint()

4.getSavepoint()

5.commit()

6.rollback()

DB-Table : Bank48(accno,custname,balance,acctype)

```
create table Bank48(accno number(15),custname varchar2(15),
balance number(10,2),acctype varchar2(15),primary key(accno));
```

```
insert into Bank48 values(6123456,'Raj',12000,'savings');
```

```
insert into Bank48 values(313131,'Ram',500,'savings');
```

ACCNO CUSTNAME

BALANCE ACCTYPE

6123456 Raj 12000 savings

313131 Ram 500 savings

Venkatesh Maiopathiji

Dt : 28/9/2022

Note:

=>JDBC applications will perform auto-commit operations, which commit operation is performed automatically.

Transaction : Transfer amt:3000/- from accNo:6123456 to accNo:313131

SubT1 : subtract 3000/- from accNo:6123456

SubT2 : add 3000/- to accNo:313131

=>If two SubTransactions are Successfull then perform Commit operation

DBCon9.java(MainClass)

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon9 {
    public static void main(String[] args) {
        try {
            Scanner s = new Scanner(System.in);
            Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
            System.out.println("Commit status :
"+con.getAutoCommit());
            con.setAutoCommit(false);
            System.out.println("Commit status :
"+con.getAutoCommit());
            PreparedStatement ps1 = con.prepareStatement
("select * from Bank48 where accno=?");
            PreparedStatement ps2 = con.prepareStatement
("update bank48 set balance=balance+? where
accno=?");
            Savepoint sp = con.setSavepoint();
            System.out.println("Enter homeAccNo:");
        }
    }
}
```

```

long hAccNo = s.nextLong(); //6123456
ps1.setLong(1, hAccNo);
ResultSet rs1 = ps1.executeQuery();
if(rs1.next()) {
    float bal = rs1.getFloat(3); //12000
    System.out.println("Enter beneficiaryAccNo:");
    long bAccNo = s.nextLong(); //313131
    ps1.setLong(1, bAccNo);
    ResultSet rs2 = ps1.executeQuery();
    if(rs2.next()) {
        System.out.println("Enter the amt to be
transferred:");
        float amt = s.nextFloat(); //3000
        if(amt<=bal) {

            ps2.setFloat(1,-amt);
            ps2.setLong(2,hAccNo);
            int i = ps2.executeUpdate(); //Updated
in buffer

            ps2.setFloat(1,amt);
            ps2.setLong(2,bAccNo);
            int j = ps2.executeUpdate(); //Updated
in buffer

            if(i==1 && j==1) {
                con.commit(); //Updated DataBase
                System.out.println("Transaction
Successfully..");
            } else {
                con.rollback(sp);
                System.out.println("Transaction
failed..");
            }
        } else {
            System.out.println("Insufficient
fund..");
        }
    } else {
        System.out.println("Invalid bAccNo..");
    }
} else {
    System.out.println("Invalid homeAccNo..");
}

} catch(Exception e) {e.printStackTrace();}
}

```

}

o/p:

Commit status : true

Commit status : false

Enter homeAccNo:

6123456

Enter benefiecieryAccNo:

313131

Enter the amt to be transferred:

3000

Transaction Successfully..

*SQL> select * from bank48;*

ACCNO	CUSTNAME	BALANCE	ACCTYPE
6123456	Raj	12000	savings
313131	Ram	500	savings

*SQL> select * from bank48;*

ACCNO	CUSTNAME	BALANCE	ACCTYPE
6123456	Raj	9000	savings

313131 Ram

3500 savings

SQL>

=====

Assignment:

Update above application by recording the transaction when the transaction is Successfull.

=>record the traction in the following Db-table:

DB Table : TansLog48(AccNo,bAccNo,amt,time)

=====

Assignment:

Construct JDBC application to display all transaction details based on accNo.

=====

Dt : 29/9/2022

1.setAutoCommit():

=>This setAutoCommit() method is used set the commit operation false.

2.getAutoCommit():

=>This getAutoCommit() method is used know the status of commit operation.

3.setSavepoint():

=>This setSavepoint() method is used to set savepoint to perform rollback process.

4.releaseSavepoint():

=>This releaseSavepoint() method is used remove the savepoint.

5.commit():

=>This commit() method is used to update the database permanently when the transaction is successfull.

6.rollback():

=>This rollback() method is used to get back the original status back when the transaction failed.

faq:

Explain the process of Transaction Management:

step-1 : set auto-commit operation to "false"

**step-2 : Take savepoint to perform rollback operation when
transaction failed.**

step-3 : perform Sub-Transactions(Execute Sub-Transactions)

**step-4 : Check all Sub-Transactions are successfull or not,then
perform commit operation.**

step-5 : release the savepoint

=====

***imp**

Connection Pooling in JDBC:

**=>The process of organizing multiple pre-initialized database
connections among multiple users is known as Connection pooling
process.**

Note:

**=>we use Vector<E> object to hold multiple Pre-Initialized database
connections in Connection Pooling process.**

Ex:

ConnectionPooling.java

package test;

```

import java.util.*;
import java.sql.*;

public class ConnectionPooling {
    public String url,uName,pWord;
    public ConnectionPooling(String url,String uName,String pWord) {
        this.url=url;
        this.uName=uName;
        this.pWord=pWord;
    }
    public Vector<Connection> v = new Vector<Connection>();
    public void createConnection() {
        try {
            while(v.size()<5)
            {
                System.out.println("Pool is Not-full...");
                Connection con=DriverManager.getConnection
                (url,uName,pWord);
                v.add(con);//Adding Connection object to Pool
                System.out.println(con);
            }//end of loop
            if(v.size()==5)
            {
                System.out.println("Pool is full...");
            }
        }
    }
}

```

```

}catch(Exception e) {e.printStackTrace();}

}//end of method

public Connection useConnection()

{
    Connection con = v.firstElement();

    v.removeElementAt(0);

    return con;
}

public void returnConnection(Connection con) {

    v.addElement(con);

    System.out.println("Connection added back to pool....");

}
}

DBCon10.java(MainClass)

package test;

import java.sql.*;
import java.util.*;

public class DBCon10 {

    public static void main(String[] args) {

        try {

            ConnectionPooling cp = new ConnectionPooling

            ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

            cp.createConnection();

            System.out.println("pool size : "+cp.v.size());
        }
    }
}

```

```

System.out.println("====User-1=====");
Connection con = cp.useConnection();
System.out.println(con);
System.out.println("pool size : "+cp.v.size());
System.out.println("====User-2=====");
Connection con2 = cp.useConnection();
System.out.println(con2);
System.out.println("pool size : "+cp.v.size());
System.out.println("====return Connection by user-1=====");
cp.returnConnection(con);
System.out.println("pool size : "+cp.v.size());
System.out.println("====return Connection by user-2=====");
cp.returnConnection(con2);
System.out.println("pool size : "+cp.v.size());
System.out.println("====Display Connection from pool====");
Iterator<Connection> it = cp.v.iterator();
while(it.hasNext()) {
    System.out.println(it.next());
}//end of loop
}catch(Exception e) {e.printStackTrace();}
}

```

o/p:

Pool is Not-full...

oracle.jdbc.driver.T4CConnection@4b952a2d

Pool is Not-full...

oracle.jdbc.driver.T4CConnection@1a38c59b

Pool is Not-full...

oracle.jdbc.driver.T4CConnection@7f77e91b

Pool is Not-full...

oracle.jdbc.driver.T4CConnection@2357d90a

Pool is Not-full...

oracle.jdbc.driver.T4CConnection@6328d34a

Pool is full...

pool size : 5

====User-1=====

oracle.jdbc.driver.T4CConnection@4b952a2d

pool size : 4

====User-2=====

oracle.jdbc.driver.T4CConnection@1a38c59b

pool size : 3

====return Connection by user-1=====

Connection added back to pool....

pool size : 4

====return Connection by user-2=====

Connection added back to pool....

pool size : 5

====Display Connection from pool=====

oracle.jdbc.driver.T4CConnection@7f77e91b

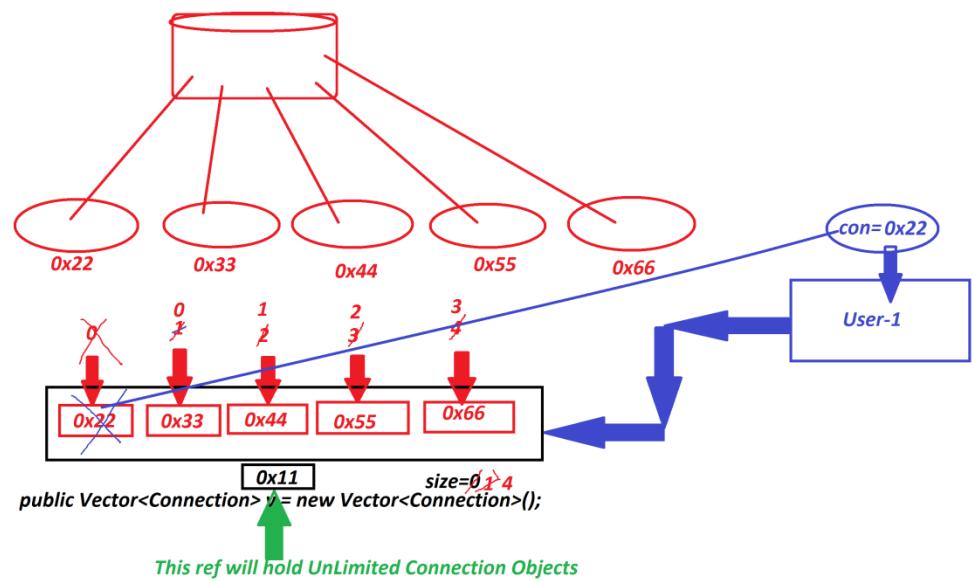
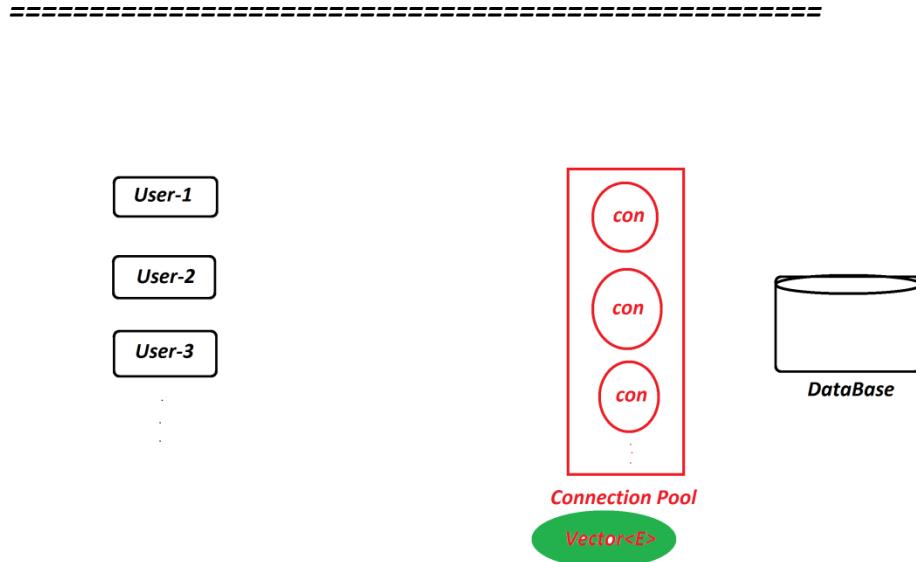
oracle.jdbc.driver.T4CConnection@2357d90a

oracle.jdbc.driver.T4CConnection@6328d34a

oracle.jdbc.driver.T4CConnection@4b952a2d

oracle.jdbc.driver.T4CConnection@1a38c59b

Venkatesh Maiopathiji



Dt : 30/9/2022

faq:

Explain the process of Connection Pooling:

step-1 : Created Connection-pool component,which means create

Vector<E> object.

step-2 : Create Multiple DB-Connections and add to the Vector<E>

Object

step-3 : Take the DB-Connection from pool(Vector<E> object) and

assign to User.

step-4 : Delete the DB-Connection from the pool(Vector<E> object)

step-5 : After using DB-Connection,it must be returned back to the

pool.

*imp

Batch Processing in JDBC:

=>**The process of collecting multiple queries as batch and executing at-a-time is known as Batch Processing**

=>**we use the following methods to perform batch processing:**

(a)addBatch()

(b)executeBatch()

(c)clearBatch()

(a)addBatch():

=>This **addBatch()** method is used to add query to the batch.

Method Signature:

```
public abstract void addBatch(java.lang.String)  
throws java.sql.SQLException;
```

(b)executeBatch():

=>This **executeBatch()** method is used to execute queries from the batch.

Method Signature:

```
public abstract int[] executeBatch() throws java.sql.SQLException;
```

(c)clearBatch():

=>This **clearBatch()** method is used to clear the queries from the batch.

Method Signature:

```
public abstract void clearBatch() throws java.sql.SQLException;
```

Case-1 : Batch processing using 'Statement'.

=>Batch processing using 'Statement',we can update multiple Database tables.

Ex : DBCon11.java

```
package test;  
import java.sql.*;  
public class DBCon11 {  
    public static void main(String[] args) {
```

```

try {
    Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
    Statement stm = con.createStatement();

    stm.addBatch("delete from Product48 where code='A121'");
    stm.addBatch("update Employee48 set
bsal=17000,totsal=45000 where id='A333'");
    stm.addBatch("insert into Bank48
values(232323,'RRR',16000,'savings')");

    int k[] = stm.executeBatch();
    for(int i=0;i<k.length;i++)
    {
        System.out.println("Updated Successfully....");
    }//end of loop

    stm.clearBatch();
    con.close();
} catch(Exception e) {e.printStackTrace();}
}
}

```

Case-2 : Batch processing using 'PreparedStatement'

=>Batch processing using 'PreparedStatement',we can update same
database table.

Ex > DBCon12.java

```

package test;
import java.sql.*;
public class DBCon12 {
    public static void main(String[] args) {
        try {
            Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
            PreparedStatement ps = con.prepareStatement
                ("insert into Bank48 values(?, ?, ?, ?)");
            ps.setLong(1, 345678);

```

```

ps.setString(2, "TYU");
ps.setFloat(3,8000);
ps.setString(4, "savings");
ps.addBatch();

ps.setLong(1, 7678676);
ps.setString(2, "TAL");
ps.setFloat(3,9000);
ps.setString(4, "savings");
ps.addBatch();

int k[] = ps.executeBatch();
for(int i=0;i<k.length;i++)
{
    System.out.println("Updated Successfully... ");
}//end of loop
ps.clearBatch();
con.close();
} catch(Exception e) {e.printStackTrace();}
}
}
=====
```

faq:

Advantage of Batch Processing:

=>*In Batch processing the execution control is transferred to*

DataBase product only once and executes multiple queries at-a-time,

in this process the execution time is saved and generate

HighPerformance of an application.

Note:

=>*Through Batch processing we can execute only Update queries,*

which means we cannot use 'select-queries' in batch processing.

faq:

wt is the diff b/w

(i)Batch processing

(ii)Procedures

=>Through batch processing we can execute only Non-Select queries,

but using Procedures we can execute both select and non-select

queries.

=====

Venkatesh Maipathiji

Dt : 1/10/2022

*imp

Types of ResultSet objects:

=>Based on control over the cursor, the ResultSet objects are

categorized into two types:

1. Non-Scrollable ResultSet object

2. Scrollable ResultSet object

1. Non-Scrollable ResultSet object:

=>In Non-Scrollable ResultSet Objects the cursor is moved only in one direction, which means the cursor moves from top-of-the-table-data to Bottom-of-table-data.

Ex:

above programs related to ResultSet

*imp

2. Scrollable ResultSet object:

=>In Scrollable ResultSet object the cursor can be moved in two directions, which means down the table data and upward the table data.

Syntax for Creating Scrollable ResultSet object:

```
Statement stm = con.createStatement(type,mode);
```

```
PreparedStatement ps = con.prepareStatement("query-S",type,mode);
```

Type:

```
public static final int TYPE_FORWARD_ONLY=1003
```

```
public static final int TYPE_SCROLL_INSENSITIVE=1004
```

```
public static final int TYPE_SCROLL_SENSITIVE=1005
```

Mode:

```
public static final int CONCUR_READ_ONLY=1007
```

```
public static final int CONCUR_UPDATABLE=1008
```

Note:

*'type' specifies the direction of the cursor and 'mode' specifies
the action to be performed(read or update).*

*The following are some Methods used to control cursor on Scrollable
ResultSet object:*

afterLast() => Moves the cursor after the Last row

beforeFirst() => Moves the cursor before the First row

previous() => Moves the Cursor in the BackWard Direction

next() => Moves the cursor in the ForWard Direction

first() => Moves the cursor to the First row

last() => Moves the cursor to the Last row

absolute(int)=>Moves the cursor to the specified row number

relative(int)=>Moves the cursor from the current position

*in forward or backward direction by increment or
decrement.*

Ex : DBCon13.java

```
package test;
import java.sql.*;
public class DBCon13 {
    public static void main(String[] args) {
        try {
            Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
            System.out.println("*****Statement*****");
            Statement stm = con.createStatement
                (ResultSet.TYPE_SCROLL_INSENSITIVE,
                 ResultSet.CONCUR_READ_ONLY);
            ResultSet rs1 = stm.executeQuery("select * from
Product48");
            rs1.afterLast(); //Cursor moved after last row
            while(rs1.previous()) {
                System.out.println(rs1.getString(1)+"\t"+rs1.getString(2)+"
\t"+rs1.getFloat(3)+"\t"+rs1.getInt(4));
            } //end of while
            System.out.println("****PreparedStatement****");
            PreparedStatement ps = con.prepareStatement
                ("select * from Product48",1004,1007);
            ResultSet rs2 = ps.executeQuery();
            System.out.println("====Display row 2====");
            rs2.absolute(2); //Cursor pointing to row number 2
            System.out.println(rs2.getString(1)+"\t"+rs2.getString(2)+"
\t"+rs2.getFloat(3)+"\t"+rs2.getInt(4));
            System.out.println("====Display relative(+1)====");
```

```

        rs2.relative(+1);
        System.out.println(rs2.getString(1)+"\t"+rs2.getString(2)+"
        "\t"+rs2.getFloat(3)+"\t"+rs2.getInt(4));
        con.close();
    }catch(Exception e) {e.printStackTrace();}
}
}

```

o/p:

******Statement******

A333 CDDR 134.56 10

A222 KBBB 232.56 12

A111 Mouse 234.56 12

****PreparedStatement****

====Display row 2=====

A222 KBBB 232.56 12

====Display relative(+1)=====

A333 CDDR 134.56 10

**imp*

define 'RowSet'?

=>RowSet object will encapsulate the rows generated from

ResultSets or any other data sources.

=>RowSet is an interface from javax.sql package and which is

extended from 'java.sql.ResultSet' interface.

=>The following are the interfaces extended from RowSet:

(a)JDBCRowSet

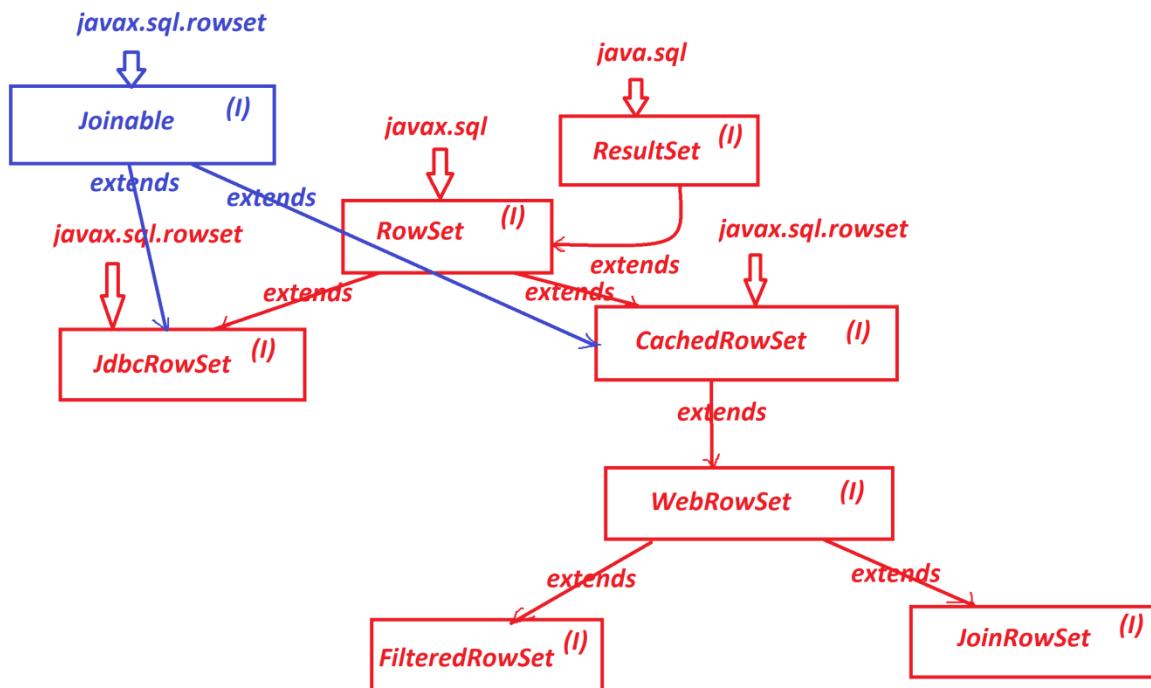
(b)CachedRowSet

=>*WebRowSet*

(i) *FilteredRowSet*

(ii) *JoinRowSet*

Hierarchy of RowSet:



Dt : 10/10/2022

faq:

wt is the diff b/w

(i)JdbcRowSet

(ii)CachedRowSet

(i)JdbcRowSet:

=>JdbcRowSet will hold ResultSet and connection to DataBase is active.

(ii)CachedRowSet:

=>cachedRowSet will hold ResultSet, but connection to DataBase is Dis-Connected automatically.

Note:

=>WebRowSet is used to transfer the data from one layer to another layer in Application architectures.

=>FilteredRowSet will hold the data retrieved based on condition.

=>JoinRowSet will hold the data joined from more than one ResultSet objects.

=====

**imp*

define RowSetFactory?

=>RowSetFactory is an interface from 'javax.sql.rowset' package and

which provide the following methods to create the implementations of

RowSet:

- (a)createJdbcRowSet()*
- (b)createCachedRowSet()*
- (c)createWebRowSet()*
- (d)createFilteredRowSet()*
- (e)createJoinRowSet()*

(a)createJdbcRowSet():

*=>This method is used to create the implementation object of
'JdbcRowSet'.*

Method Signature:

```
public abstract javax.sql.rowset.JdbcRowSet createJdbcRowSet()  
throws java.sql.SQLException;
```

(b)createCachedRowSet():

*=>This method is used to create the implementation object of
'CachedRowSet'.*

Method Signature:

```
public abstract javax.sql.rowset.CachedRowSet createCachedRowSet()  
throws java.sql.SQLException;
```

(c)createWebRowSet():

=>This method is used to create the implementation object of 'WebRowSet'.

Method Signature:

```
public abstract javax.sql.rowset.WebRowSet createWebRowSet()  
throws java.sql.SQLException;
```

(d)createFilteredRowSet():

=>This method is used to create the implementation object of 'FilteredRowSet'.

Method Signature:

```
public abstract javax.sql.rowset.FilteredRowSet createFilteredRowSet()  
throws java.sql.SQLException;
```

(e)createJoinRowSet():

=>This method is used to create the implementation object of 'JoinRowSet'.

Method Signature:

```
public abstract javax.sql.rowset.JoinRowSet createJoinRowSet()  
throws java.sql.SQLException;
```

Note:

=>we use the following methods from 'javax.sql.rowset.RowSetProvider' class to create the implementation object of 'RowSetFactory'

interface.

```
public static javax.sql.rowset.RowSetFactory newFactory()  
throws java.sql.SQLException;  
  
public static javax.sql.rowset.RowSetFactory newFactory  
(java.lang.String, java.lang.ClassLoader)  
throws java.sql.SQLException;
```

Ex : DBCon14.java

```
package test;  
  
import javax.sql.rowset.*;  
  
import java.util.*;  
  
public class DBCon14 {  
  
    public static void main(String[] args) {  
  
        try {  
  
            Scanner s = new Scanner(System.in);  
  
            RowSetFactory rsf = RowSetProvider.newFactory();  
  
            //RowSetFactory Object  
  
            System.out.println("====Choice====");  
  
            System.out.println("1.JdbcRowSet\n2.CachedRowSet");  
  
            System.out.println("Enter the Choice:");  
  
            switch(s.nextInt())  
  
            {
```

case 1:

```
JdbcRowSet jrs = rsf.createJdbcRowSet();
//JdbcRowSet object

jrs.setUrl("jdbc:oracle:thin:@localhost:1521:xe");

jrs.setUsername("system");

jrs.setPassword("manager");

jrs.setCommand("select * from Product48");

jrs.execute();

System.out.println("====JdbcRowSet Product data====");

while(jrs.next())
{
    System.out.println(jrs.getString(1)+"\t"
                    +jrs.getString(2)+"\t"
                    +jrs.getFloat(3)+"\t"
                    +jrs.getInt(4));
}
```

//end of loop

break;

case 2:

```
CachedRowSet crs = rsf.createCachedRowSet();
//CachedRowSet Object

crs.setUrl("jdbc:oracle:thin:@localhost:1521:xe");

crs.setUsername("system");

crs.setPassword("manager");

crs.setCommand("select * from Product48");
```

```

crs.execute();

System.out.println("====CachedRowSet Product data====");

while(crs.next())
{

    System.out.println(crs.getString(1)+"\t"
                       +crs.getString(2)+"\t"
                       +crs.getFloat(3)+"\t"
                       +crs.getInt(4));

}

//end of loop

break;

default:

    System.out.println("InvalidChoice... ");

}//end of switch

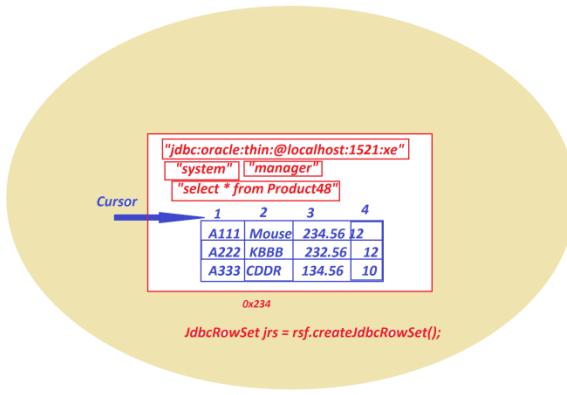
s.close();

}catch(Exception e) {e.printStackTrace();}

}

}

```



faq:

define factory methods?

=>*The methods which hide the Object creation process from end-users*

are known as factory methods.

Dt : 11/10/2022

Summary of Objects generated from CoreJava:

1. User defined Class Object

2. String Objects

3. WrapperClass Objects

4. Array objects

5. Collection<E> objects

6. Map<K,V> objects

7. Enum<E> object

Summary of Objects generated from JDBC:

1. Connection Object

2. Statement Object

3. PreparedStatement Object

4. CallableStatement Object

5. NonScrollable ResultSet Object

6. Scrollable ResultSet Object

7. RowSet Object

(i) JdbcRowSet Object

(ii) CachedRowSet Object

=>*WebRowSet Object*

(i) FilteredRowSet Object

(ii) JoinRowSet Object

8. Metadata Objects

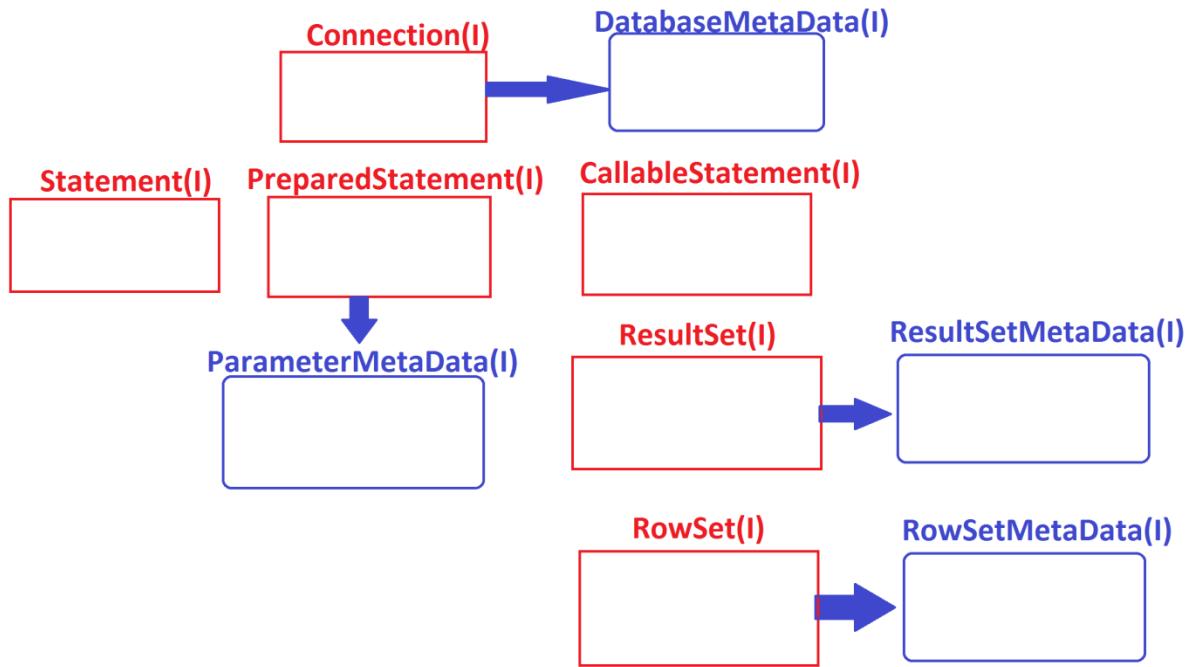
(i) DatabaseMetaData object

(ii) ParameterMetaData Object

(iii) ResultSetMetaData Object

(iv) RowSetMetaData Object

Diagram:



define Metadata in JDBC?

=>Metadata means the data holding information about another data

is known as Metadata.

=>According to Object Oriented programming,Metadata means one object holding information about another object.

=>The following are the metadata components from JDBC:

1. **DatabaseMetaData**

2. **ParameterMetaData**

3. **ResultSetMetaData**

4.RowSetMetaData

1.DatabaseMetaData:

=>*DatabaseMetaData is an interface from java.sql package and which is used to hold information about Connection Object.*

syntax:

```
DatabaseMetaData dmd = con.getMetaData();
```

2.ParameterMetaData:

=>*ParameterMetaData is an interface from java.sql package and which is used to hold information about PreparedStatement object.*

syntax:

```
ParameterMetaData pmd = ps.getParameterMetaData();
```

3.ResultSetMetaData :

=>*ResultSetMetaData is an interface from java.sql package and which is used to hold the information about ResultSet Object.*

syntax:

```
ResultSetMetaData rsmd = rs.getMetaData();
```

4.RowSetMetaData:

=>*RowSetMetaData is also an interface from java.sql package and which is used to hold information about RowSet Object.*

syntax:

```
RowSetMetaData rsmd = (RowSetMetaData)rs.getMetaData();
```

Ex : DBCon15.java

```
package test;

import java.sql.*;

public class DBCon15 {

    public static void main(String[] args) {

        try {

            Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

            DatabaseMetaData dmd = con.getMetaData();

            System.out.println("====DatabaseMetaData====");

            System.out.println("Version : "+dmd.getDatabaseMajorVersion());

            PreparedStatement ps = con.prepareStatement

                ("select name,price from Product48 where code=?");

            ParameterMetaData pmd = ps.getParameterMetaData();

            System.out.println("====ParameterMetaData====");

            System.out.println("Count : "+pmd.getParameterCount());

            ps.setString(1,"A111");

            ResultSet rs = ps.executeQuery();

            ResultSetMetaData rsmd = rs.getMetaData();

            System.out.println("====ResultSetMetaData====");

            System.out.println("Cols Count : "+rsmd.getColumnCount());
```

```
 }catch(Exception e) {e.printStackTrace();}  
 }  
=====
```

List of Jar files of DataBase products:

*Oracle - ojdbc14.jar,ojdbc6.jar, ojdbc7.jar, ojdbc8.jar,
ojdbc10.jar*

Oracle10 - ojdbc14.jar

Oracle11 - ojdbc6.jar

oracle12 - ojdbc7.jar,ojdbc8.jar

*other - UPC.jar(Universal Connection pool)
(ojdbc10.java)*

MySQL - mysql-connector-java-VERSION.jar

SQL Server - sqljdbc41.jar, sqljdbc42.jar

PostgreSQL - postgresql-VERSION.jar

Apache Derby - derby.jar, derbyclient.jar

SQLite - sqlite-jdbc-VERSION.jar

Microsoft Access - ucanaccess-VERSION.jar

```
=====
```

Dt : 12/10/2022

*imp

Servlet Programming:(Unit-2)

faq:

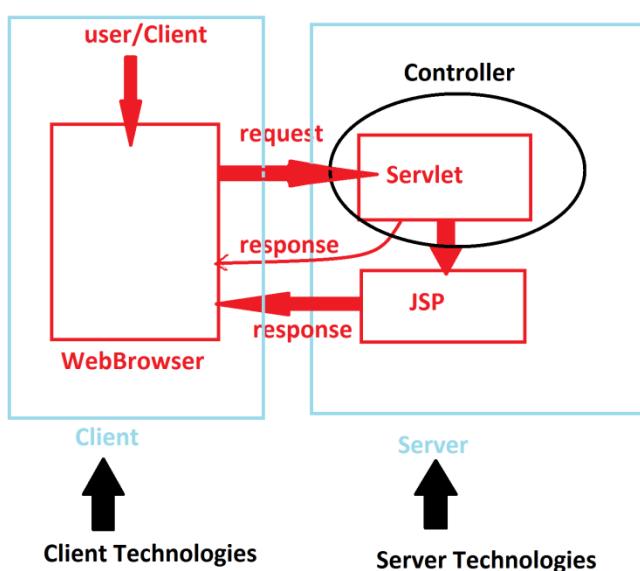
define Servlet program?

=>*The program which is deployed into server for execution is known as Servlet Program or Server Program.*

=>*This Servlet program will accept the request from the users or clients through WebBrowser.*

=>*The Servlet program will provide the response directly or will provide the response in the form of JSP(Java Server Page).*

Diagram:



faq:

define Client?

=>Client means generating request to the Server and waiting for the response.

faq:

define Server?

=>Server means Service Provider, which means accepts the request from the client and provide the response.

faq:

define Application?

=>Set-of-programs collected together to perform defined task is known as Application.

Types of Application:

=>Applications are categorized into the following:

- 1. Stand Alone Applications*
- 2. Web Applications*
- 3. Enterprise Applications*
- 4. Mobile Applications*

1. Stand Alone Applications:

=>The applications which are installed in one computer and performs

actions in the same computer are known as "Stand-Alone Applications"

or DeskTop Applications or Windows Applications

=>According to developer Stand-Alone applications means,

No HTML input

No Server Environment

No DataBase Storage

=>Based on User interaction "Stand-Alone applications" are categorized

into two types:

(i)CUI Applications

(ii)GUI Applications

(i)CUI Applications:

=>In CUI Applications the users interact through Console(CommandPrompt)

(CUI - Console User Interface)

(ii)GUI Applications:

=>In GUI Applications the users interact through GUI-Components.

(GUI - Graphical User Interface)

=>we use the following in GUI-Programming:

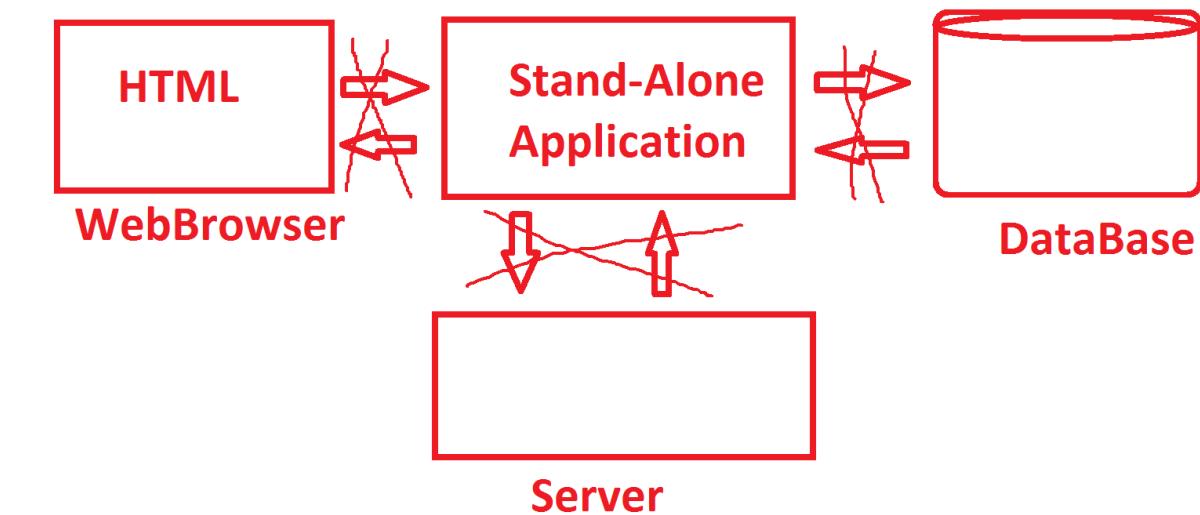
AWT(Abstract Window Toolkit)

Swing

JavaFx(Java8 - Version)

Note:

=>In the process of constructing "Stand-Alone applications", "main()" method is the starting point for execution and the execution environment is "JVM".



2. Web Applications:

=>The applications which are executing in Web environment or Internet environment are known as Web Applications.

Note:

=>In the process of constructing Web Application, "init()" method is the Starting point of execution and the execution environment is "WebContainer".

=>"WebContainer" is available from Servers.

CoreJava:

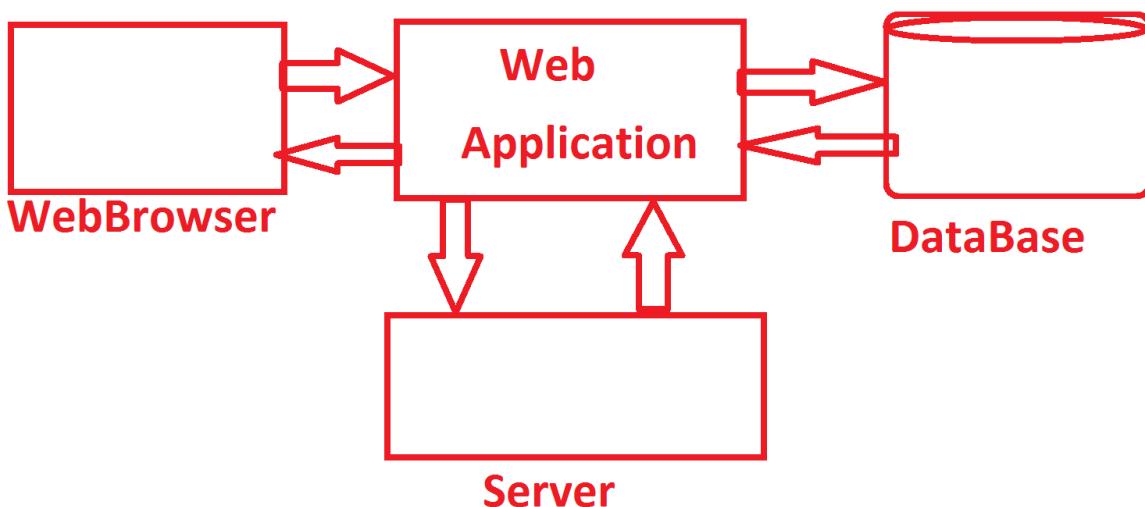
Execution E : JVM

Download : JDK

AdvJava:(WebApp)

Execution E : WebContainer

Download : Server



3. Enterprise Applications:

=>The applications which are executing in distributed environment and depending on the features like "Security", "Load Balancing" and "Clustering" are known as Enterprise Applications.

4. Mobile Applications:

=>The applications which are executing in mobile environment are known as Mobile applications.

Venkatesh Maiopathiji

Dt : 13/10/2022

*imp

Types of Servers:

=>Servers are categorized into the following two types:

(i) Web Servers

(ii) Application Servers

(i) Web Server:

=>Web Server contains only Web container.

=>A web server is good in case of static contents like static html pages.

=>Web server consumes less resources like CPU, Memory etc. as compared to application server.

=>Web Server provides the runtime environment for web applications.

=>Web Server supports HTTP Protocol

=>Apache Web Server.(Tomcat)

(ii) Application Server:

=>Application Server contains both Web Container and EJB Container.(EJB - Enterprise Java Bean)

=>Application server is relevant in case of dynamic contents like bank websites.

=>Application server utilizes more resources

=>*Application server provides the runtime environment for enterprise applications.*

=>*Application Server supports HTTP as well as RPC/RMI protocols.*

(RPC - Remote Procedure call)

(RMI - Remote Method Invocation)

=>*Weblogic, JBoss.*

=====

*imp

Installing Tomcat Server:

step1 : Download Tomcat9.x WebServer

webserver - Tomcat 9.x

(Compatible with JDK1.8 and Above)

vendor - Apache org

default port no - 8080

download - www.apache.org(Open source)

Note:

=>*WebContainer internally has two SubContainers*

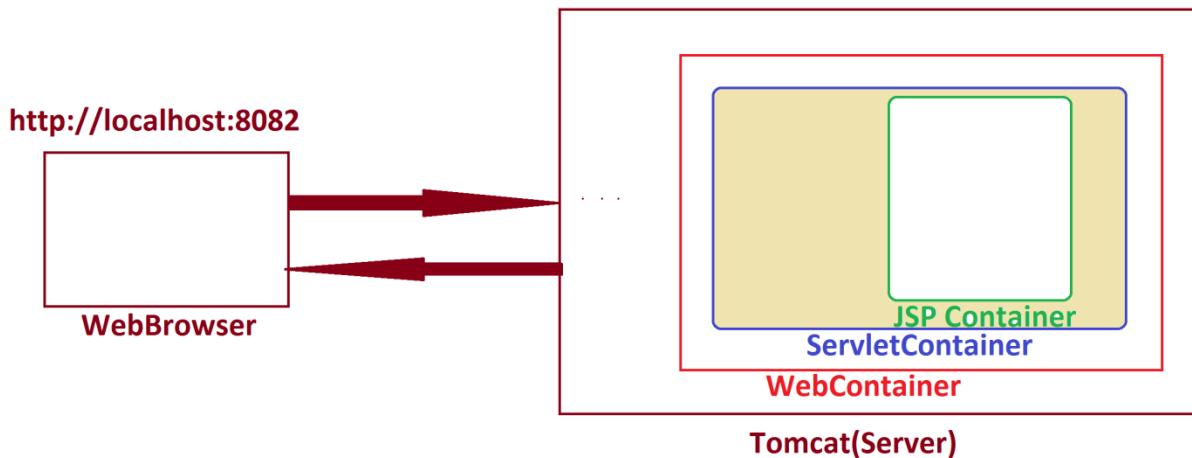
(i)Servlet container

(ii)JSP Container

Servlet container : Catalina

Jsp Container : Jasper

Link : <https://tomcat.apache.org/download-90.cgi>



Step-2 : Install Tomcat Server

=>while installation process,

Select the type of Install : Full (click Next)

Server Shutdown port : 8089

HTTP/1.1 Connector port : 8081 or 8082 or ...

User Name : Venkatesh

Password : nit

(Click Next)

step-3 : Start the Tomcat Server

=>Click on "startup" or "Tomcat9w" file from "bin" folder of Tomcat to

start the Server.

C:\Tomcat 9.0\bin

step-4 : Open WebBrowser and use the following URL in AddressBar to interact with Tomcat Server

<http://localhost:8082>

step-5 : Shutdown the Tomcat Server

=>Click on "shutdown" or "Tomcat9w" file from "bin" folder of Tomcat to stop the Server.

C:\Tomcat 9.0\bin

Dt : 14/10/2022

*imp

Servlet-API?

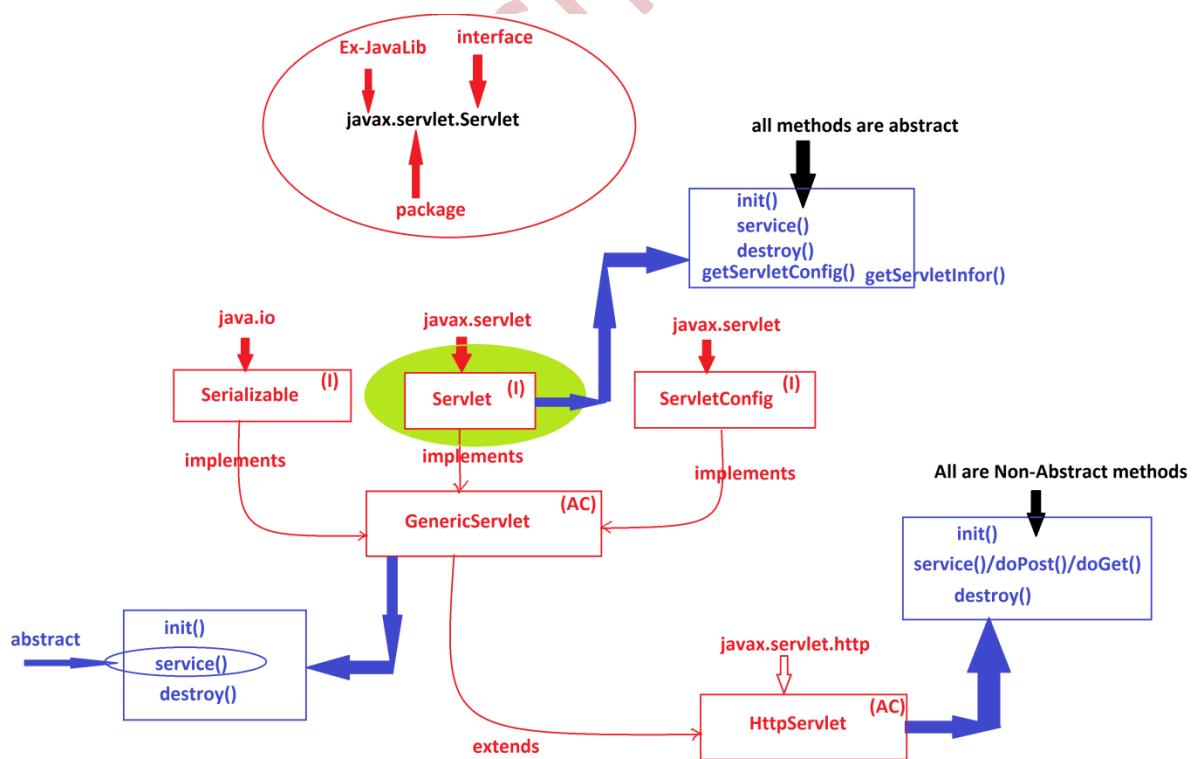
=>"javax.servlet" package is known as Servlet-API.

=>"javax.servlet.Servlet" interface is the root of Servlet-API.

=>The following are some important methods of "Servlet" interface:

- 1.init()
- 2.service()
- 3.destroy()
- 4.getServletConfig()
- 5.getServletInfo()

Hierarchy of Servlet-API:



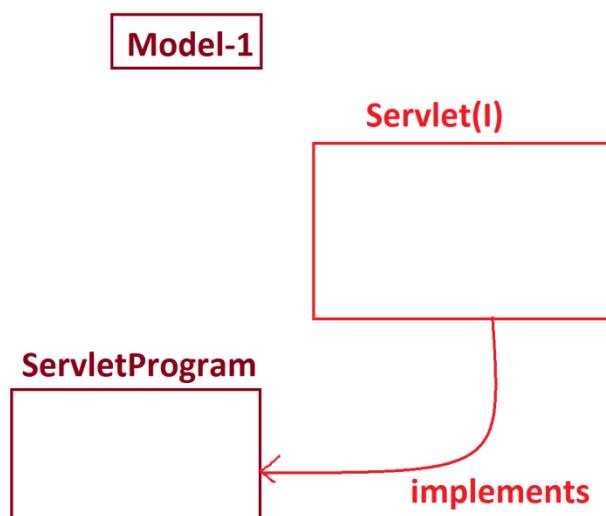
=====
=>*In the process of constructing Servlet-programs we use any one of the following process:*

- 1.using 'Servlet' interface**
- 2.Using 'GenericServlet' AbstractClass**
- 3.Using 'HttpServlet' AbstractClass**

1.using 'Servlet' interface:

=>*In the process of using 'Servlet' interface, the User defined Servlet-Program must be implemented from 'Servlet' interface.*

Diagram:



Coding Rule:

=>*The User defined Servlet-program must construct body for all five*

abstract methods of "Servlet" interface.

Note:

=>WebContainer will instantiate Servlet Program automatically and executes the following Life-cycle methods automatically in the same order:

- (i)init()*
- (ii)service()*
- (iii)destroy()*

(i)init():

=>The process of making the programming components ready which can be used by service() method is known as initialization process.

=>we use init() method to perform initialization process.

(ii)service():

=>After initialization process,service() method will accept the request from the user and provides the response.

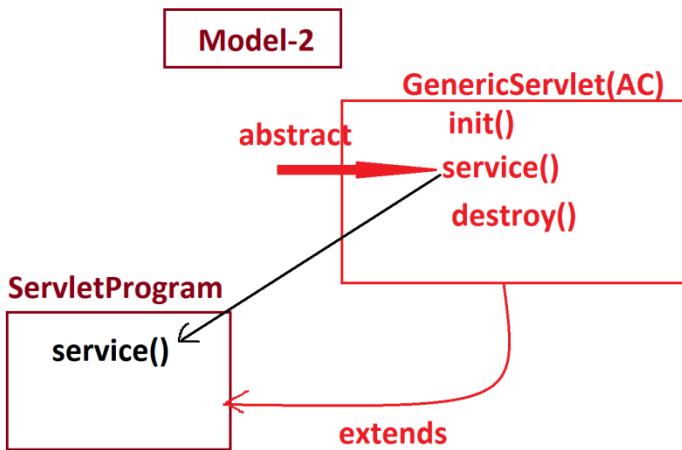
(iii)destroy():

=>destroy() method is used for destroying servlet program

2.Using 'GenericServlet' AbstractClass:

=>In the process of using "GenericServlet",the user defined Servlet-program must be extended from "GenericServlet" abstract class.

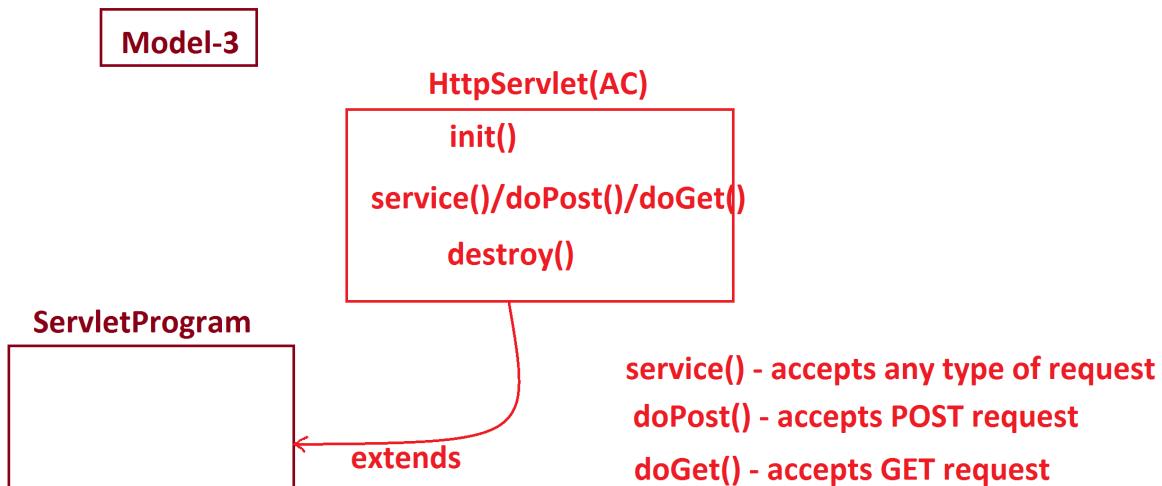
Diagram:



3. Using 'HttpServlet' AbstractClass :

=> In the process of using "HttpServlet", the user defined Servlet-program must be extended from "HttpServlet" abstract class.

Diagram:



Venkatesh Maiopathiji

Dt : 15/10/2022

*imp

Constructing Servlet Application(Web Application) using IDE Eclipse:

step-1 : Open IDE Eclipse,while opening name the WorkSpace and click

"launch"

step-2 : Create Dynamic Web Project

**Click on "File->new->Project->Web->select "Dynamic Web Project" and
click "Next"->name the project and click "finish".**

step-3 : Add "servlet-api.jar" file to Servlet Application

**RightClick on Servlet Application(Web Application)->Build path->
Configure Build path->Libraries->select "classpath" and
click "Add External Jars"->Browse and select "servlet-api.jar" file from
"lib" folder of Tomcat->Open->Appy->Apply and Close.**

step-4 : Add Server(Tomcat) to IDE Eclipse(one time process)

**Click on "servers"->select "click this link to create new Server"->
select the type of Server with version and click "Next"-> Browse
Tomcat Installation Directory(select the folder) and click**

"select folder"->click on "finish"

step-5 : Create HTML file in "webapp" folder

RightClick on "webapp"->new->HTML file->name the file and click "finish"

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="dis" method="post">
UserName:<input type="text" name="uname"><br>
MailId:<input type="text" name="mid"><br>
<input type="submit" value="Display">
</form>
</body>
</html>
```

step-6 : Create package in "src/main/java" from Java Resources

step-7 : create Class(Servlet program) in package

DisplayServlet.java

```
package test;
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/dis")
public class DisplayServlet extends GenericServlet{
    public void init() throws ServletException{
        //NoCode
    }
}
```

```

        }
    public void service(ServletRequest req,ServletResponse res)
throws
    ServletException, IOException{
    String name = req.getParameter("uname");
    String mId = req.getParameter("mid");
    PrintWriter pw = res.getWriter();
    res.setContentType("text/html");
    pw.println("UserName:"+name);
    pw.println("<br>MailId:"+mId);
}
public void destroy() {
    //NoCode
}
}

```

step-8 : Create "web.xml" in "WEB-INF" folder.

**RightClick on WEB-INF->new->Others->XML->XML file->name the file as
"web.xml" and click "finish"**

```

<web-app>
    <welcome-file-list>
        <welcome-file>input.html</welcome-file>
    </welcome-file-list>
</web-app>

```

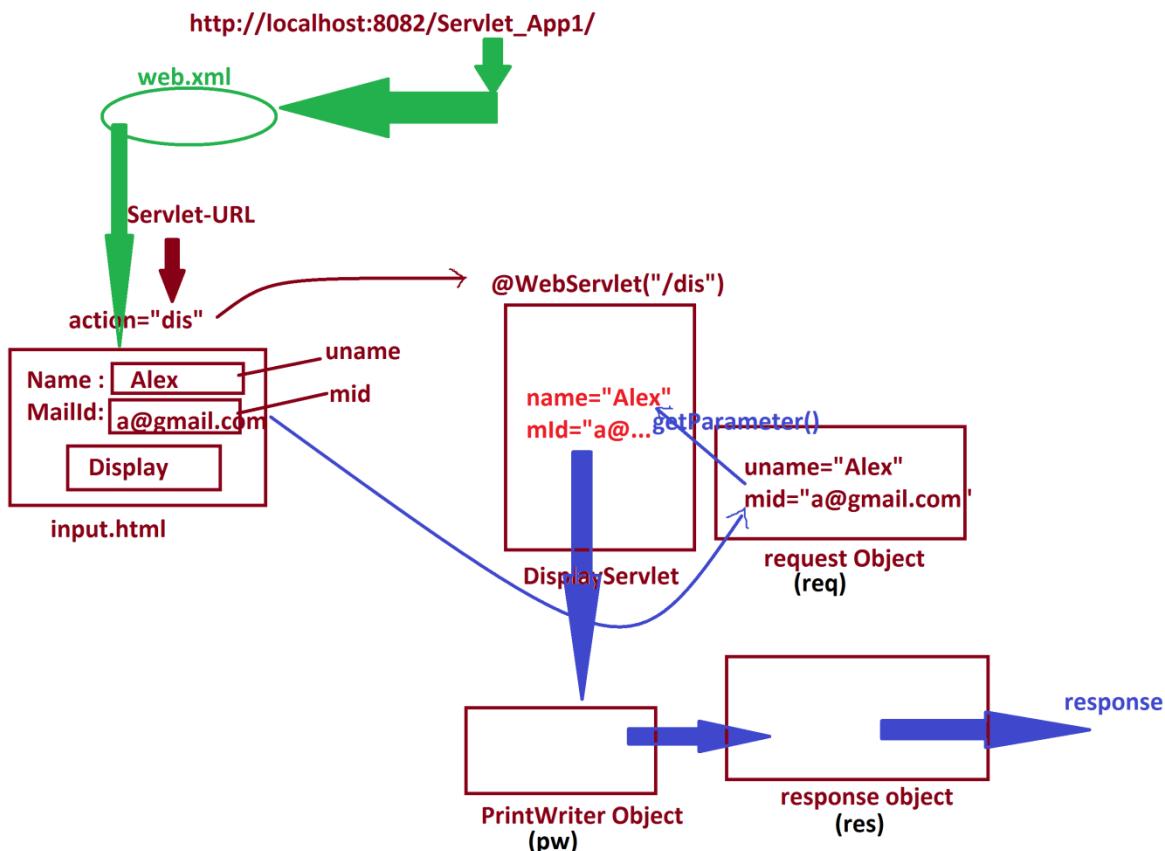
step-9 : Execute Servlet Application(Web Application)

RightClick on Servlet application(Web application)->Run As->

Run on Server->select the Server and click "finish"

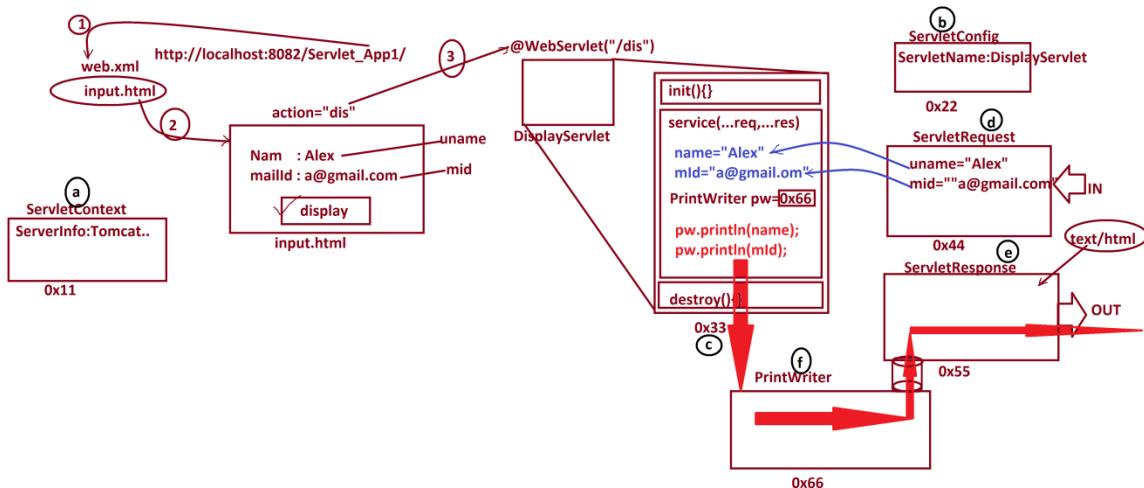
http://localhost:8082/Servlet_App1/

Diagram:



Dt : 17/10/2022

Execution flow of above application:



ServletContext:

=>ServletContext is an interface from "javax.servlet" package and which is instantiated automatically when application deployed into Server.

=>This ServletContext object is loaded with Server_information.

ServletConfig:

=>ServletConfig is an interface from "javax.servlet" package and which is instantiated automatically when Servlet_program loaded for execution.

=>This ServletConfig object is loaded with Servlet_name.

ServletRequest:

=>ServletRequest is an interface from "javax.servlet" package and which is instantiated automatically while service() method execution

=>ServletRequest object will hold the data submitted from HTML form.

ServletResponse:

=>ServletResponse is an interface from "javax.servlet" package and which is also instantiated automatically while service() method execution.

=>ServletResponse object will hold the data which is sent as an response.

define getParameter() method?

=>getParameter() method is used to get the parameters from the ServletRequest object.

syntax:

```
String var = req.getParameter("para_name");
```

define getWriter() method?

=>getWriter() method is used to create the object for "java.io.PrintWriter" class and this PrintWriter object internally linked to ServletResponse object and supports the response.

syntax:

```
PrintWriter pw = res.getWriter();
```

define setContentType() method?

=>*setContentType()* method will specify the type of data sent as response.

syntax:

```
res.setContentType("text/html");
```

```
=====
```

Servlet-App2:

Construct Servlet Application by reading Product details from HTML form and display the same using Servlet program

product.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="dis" method="post">
Product Code:<input type="text" name="code"><br>
Product Name:<input type="text" name="name"><br>
Product Price:<input type="text" name="price"><br>
Product Quantity:<input type="text" name="qty"><br>
<input type="submit" value="Display">
</form>
</body>
</html>
```

DisplayServlet.java

```
package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.*;
```

```

@SuppressWarnings("serial")

@WebServlet("/dis")

public class DisplayServlet extends GenericServlet{

    public void init() throws ServletException{
        //NoCode
    }

    public void service(ServletRequest req,ServletResponse res) throws ServletException,
    IOException{

        String code = req.getParameter("code");
        String name = req.getParameter("name");
        float price = Float.parseFloat(req.getParameter("price"));
        int qty = Integer.parseInt(req.getParameter("qty"));

        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        pw.println("Code:"+code);
        pw.println("<br>Name:"+name);
        pw.println("<br>price:"+price);
        pw.println("<br>Qty:"+qty);

    }
}

```

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>product.html</welcome-file>

```

```
</welcome-file-list>  
</web-app>
```

Assignment:

***Construct Servlet application read and display Student details with
result.***

Dt : 18/10/2022

*imp

RequestDispatcher:

=>*RequestDispatcher is an interface from "javax.servlet" package and which is used to perform Servlet-Communications.*

=>*The following are two important methods of RequestDispatcher:*

(i)forward() method

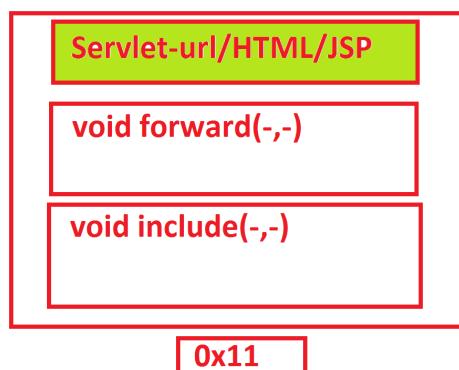
(ii)include() method

=>*we use getRequestDispatcher() method from ServletRequest to create the implementation object of RequestDispatcher.*

syntax:

```
RequestDispatcher rd = req.getRequestDispatcher("Servlet-url/HTML/JSP");
```

Diagram:



```
RequestDispatcher rd = req.getRequestDispatcher("Servlet-url/HTML/JSP");
```

=====

faq:

define Servlet-Communications?

=>The process of establishing communications b/w 'Servlet to Servlet' or 'Servlet to HTML' or 'Servlet to JSP' are known as **Servlet-Communications**.

=>**Servlet-Communications are categorized into two types:**

1.forward communication

2.include communication

1.forward communication:

=>In forward communication process,ServletProgram-1 will take the request and forwards the request to ServletProgram-2,in this process ServletProgram-2 will give the response.

(ServletProgram-2 can be replaced by HTML/JSP)

=>we use **forward()** method from RequestDispatcher to perform forward communication process.

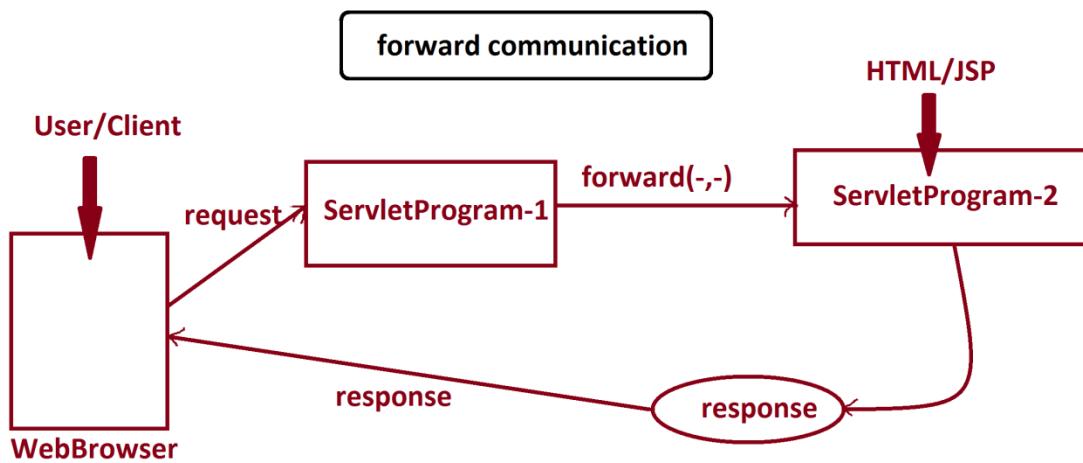
Method Signature:

```
public abstract void forward(javax.servlet.ServletRequest,  
 javax.servlet.ServletResponse) throws  
 javax.servlet.ServletException, java.io.IOException;
```

syntax:

```
rd.forward(req,res);
```

Diagram



2. include communication:

=>In include communication process **ServletProgram-1** will take the request and provide the response, but the response is included with the response of **ServletProgram-2**.

(**ServletProgram-2** can be replaced with HTML/JSP)

=>we use **include()** method from **RequestDispatcher** to perform include communication process.

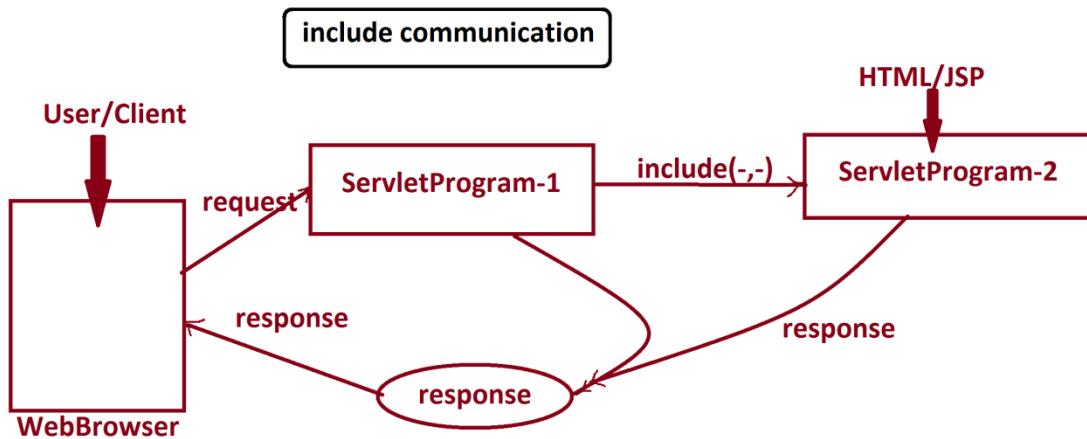
Method Signature:

```
public abstract void include(javax.servlet.ServletRequest,
    javax.servlet.ServletResponse) throws
    javax.servlet.ServletException, java.io.IOException;
```

syntax:

```
rd.include(req,res);
```

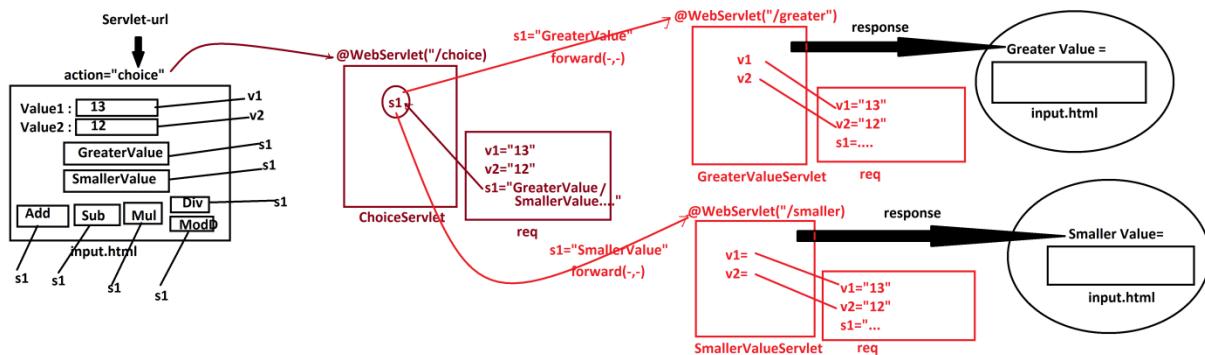
Diagram:



Ex-Application:

Construct Servlet Application to demonstrate "RequestDispatcher".

Layout:



input.html

```

<!DOCTYPE html>
<html>
<head>
  
```

```

<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="choice" method="post">
Enter the Value1:<input type="text" name="v1"><br>
Enter the Value2:<input type="text" name="v2"><br>
<input type="submit" value="GreaterValue" name="s1">
<input type="submit" value="SmallerValue" name="s1">
<input type="submit" value="Add" name="s1">
<input type="submit" value="Sub" name="s1">
<input type="submit" value="Mul" name="s1">
<input type="submit" value="Div" name="s1">
<input type="submit" value="ModDiv" name="s1">
</form>
</body>
</html>

```

ChoiceServlet.java

```

package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/choice")
public class ChoiceServlet extends GenericServlet{
    public void init()throws ServletException{
        //NoCode
    }
    public void service(ServletRequest req,ServletResponse res) throws
    ServletException,IOException{
        String s1 = req.getParameter("s1");
        if(s1.equals("GreaterValue")){

```

```

RequestDispatcher rd = req.getRequestDispatcher("greater");
rd.forward(req, res);

}else {

RequestDispatcher rd = req.getRequestDispatcher("smaller");
rd.forward(req, res);

}

}

public void destroy() {
    //NoCode
}
}

```

GreaterValueServlet.java

```

package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/greater")
public class GreaterValueServlet extends GenericServlet{
    public void init()throws ServletException{
        //NoCode
    }
    public void service(ServletRequest req,ServletResponse res) throws
    ServletException,IOException{

```

```

int v1 = Integer.parseInt(req.getParameter("v1"));

int v2 = Integer.parseInt(req.getParameter("v2"));

int v3 = 0;

if(v1>v2) v3=v1;

else v3=v2;

PrintWriter pw = res.getWriter();

res.setContentType("text/html");

pw.println("GreaterValue:"+v3+"<br>");

RequestDispatcher rd = req.getRequestDispatcher("input.html");

rd.include(req, res);

}

public void destroy() {

//NoCode

}

}

SmallerValueServlet.java

package test;

import java.io.::*;

import javax.servlet.*;

import javax.servlet.annotation..*;

@SuppressWarnings("serial")

@WebServlet("/smaller")

public class SmallerValueServlet extends GenericServlet{

public void init()throws ServletException{

```

```

//NoCode

}

public void service(ServletRequest req,ServletResponse res) throws
ServletException,IOException{

    int v1 = Integer.parseInt(req.getParameter("v1"));

    int v2 = Integer.parseInt(req.getParameter("v2"));

    int v3 = 0;

    if(v1<v2) v3=v1;

    else v3=v2;

    PrintWriter pw = res.getWriter();

    res.setContentType("text/html");

    pw.println("SmallerValue:"+v3+"<br>");

    RequestDispatcher rd = req.getRequestDispatcher("input.html");

    rd.include(req, res);

}

public void destroy() {
    //NoCode
}
}

web.xml

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>input.html</welcome-file>
    </welcome-file-list>
</web-app>
=====
```

Dt : 19/10/2022

Assignment:

Construct Servlet Application using following Layout:

Enter the value :

n

Even Odd PrimeNumner Armstrong fibonacci

Palindrome

input.html

*imp

Java Bean Classes:

=>The classes which are constructed using the following rules are known as "Java Bean Classes" or "Bean Classes"

Rule-1 : The class must be implemented from "java.io.Serializable" interface.

Rule-2 : The variables in class must be "private variables"

Rule-3 : The class must be declared with 0-parameter constructor or

0-argument Constructor.

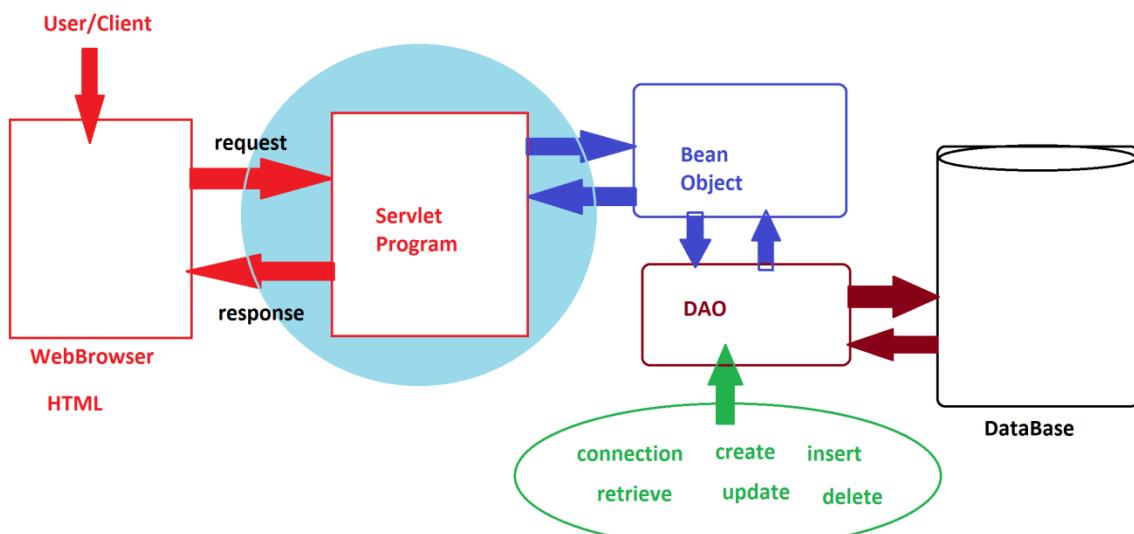
Rule-4 : The class must be declared with "Getter" and "Setter" methods.

Note:

=>These Bean classes will generate Bean objects.

=>These Bean objects will hold data going onto DB-Product and Bean
Objects will hold data coming from DB-Product.

diagram:



faq:

define Setter methods?

=>The methods which are used to set the data to objects are known as

"**Setter methods**".

faq:

define Getter methods?

=>*The methods which are used to get the data from the objects are known as "Getter methods"*

Coding rule of Setter and Getter methods:

=>*Every variable in class will have its own Setter and Getter methods.*

faq:

define DAO?

=>*DAO stands for "Data Access Object" and which is separate layer in MVC(Model View Controller) holding database related logics or code.*

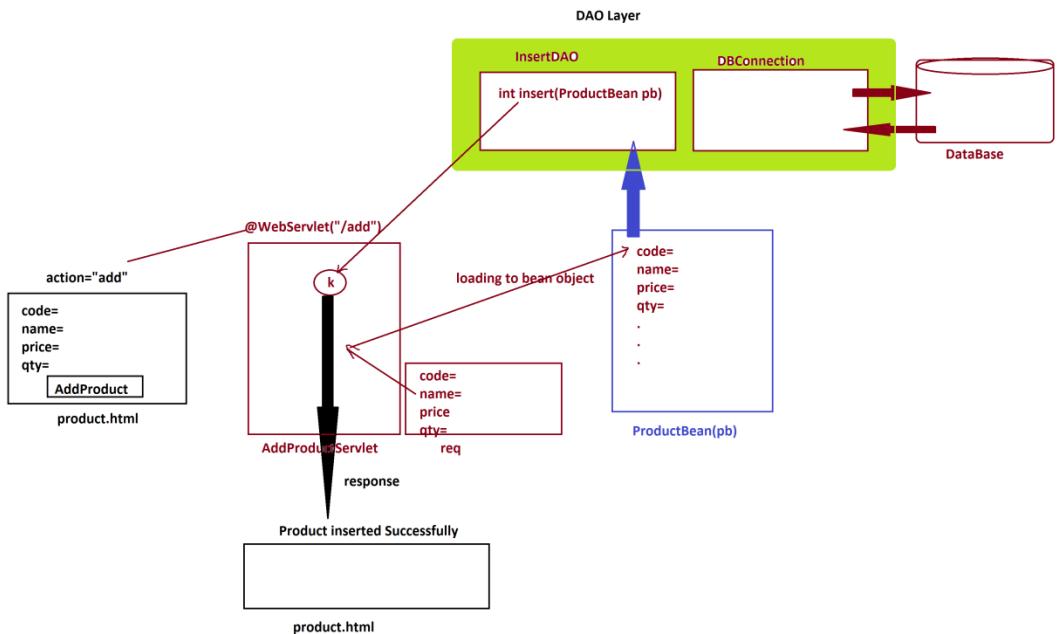
=>*DAO layer will hold persistent logic.*

Note:

=>*In the process of establishing communication b/w Servlet-Program and DataBase product, DB-Jar file must be copied into "lib" folder of "WEB-INF"*

Ex-Application:

Construct Servlet Application to insert product details to DB-table.



product.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="add" method="post">
Product Code:<input type="text" name="code"><br>
Product Name:<input type="text" name="name"><br>
Product Price:<input type="text" name="price"><br>
Product Quantity:<input type="text" name="qty"><br>
<input type="submit" value="AddProduct">
</form>
</body>
</html>
```

ProductBean.java

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class ProductBean implements Serializable{
    private String code, name;
    private float price;
    private int qty;
```

```

public ProductBean() {}
public String getCode() {
    return code;
}
public void setCode(String code) {
    this.code = code;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public float getPrice() {
    return price;
}
public void setPrice(float price) {
    this.price = price;
}
public int getQty() {
    return qty;
}
public void setQty(int qty) {
    this.qty = qty;
}
}

```

DBConnection.java

```

package test;
import java.sql.*;
public class DBConnection {
    private static Connection con=null;
    private DBConnection() {}
    static
    {
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
        }catch(Exception e) {e.printStackTrace();}
    }
    public static Connection getCon() {
        return con;
    }
}

```

```
    }  
}
```

InsertDAO.java

```
package test;  
import java.sql.*;  
public class InsertDAO {  
    public int k=0;  
    public int insert(ProductBean pb) {  
        try {  
            Connection con = DBConnection.getCon(); //Accessing  
the Connection  
            PreparedStatement ps = con.prepareStatement  
                ("insert into Product48 values(?,?,?,?,?)");  
            ps.setString(1,pb.getCode());  
            ps.setString(2,pb.getName());  
            ps.setFloat(3,pb.getPrice());  
            ps.setInt(4, pb.getQty());  
            k = ps.executeUpdate();  
        }catch(Exception e) {e.printStackTrace();}  
        return k;  
    }  
}
```

AddProductServlet.java

```
package test;  
  
import java.io.*;  
  
import javax.servlet.*;  
  
import javax.servlet.annotation.*;  
  
@SuppressWarnings("serial")  
@WebServlet("/add")  
  
public class AddProductServlet extends GenericServlet{  
  
    public ProductBean pb=null;  
  
    public void init() throws ServletException{
```

```

pb = new ProductBean(); //Creating bean object

}

public void service(ServletRequest req, ServletResponse res) throws ServletException,
IOException{

    pb.setCode(req.getParameter("code"));

    pb.setName(req.getParameter("name"));

    pb.setPrice(Float.parseFloat(req.getParameter("price")));

    pb.setQty(Integer.parseInt(req.getParameter("qty")));

    int k = new InsertDAO().insert(pb);

    PrintWriter pw = res.getWriter();

    res.setContentType("text/html");

    if(k>0) {

        pw.println("Product Added Successfully...<br>");

        RequestDispatcher rd = req.getRequestDispatcher("product.html");

        rd.include(req, res);

    }

}

}

```

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>product.html</welcome-file>

```

```
</welcome-file-list>  
</web-app>
```

Venkatesh Maiopathiji

Dt : 20/10/2022

Assignment:

Construct Servlet Application to add book details to DB-Table(Book48)

*imp

"attribute" in Servlet Programming:

=>"attribute" is a variable which can be added to *ServletContext object,*

ServletRequest object and HttpSession object.

=>The following are the methods related to "attribute" in Servlet
programming:

- 1.**setAttribute()**
- 2.**getAttribute()**
- 3.**removeAttribute()**

1.setAttribute():

=>we use *setAttribute()* method to add "attribute" to objects.

Method Signature:

public abstract void setAttribute(java.lang.String,java.lang.Object);

2.getAttribute():

=>we use *getAttribute()* method to get the "attribute" from the objects.

Method Signature:

public abstract java.lang.Object getAttribute(java.lang.String);

3.removeAttribute():

=>we use removeAttribute() method to delete the "attribute" from the objects.

Method Signature:

public abstract void removeAttribute(java.lang.String);

=====

*imp

Scope of "attribute" in Servlet programming:

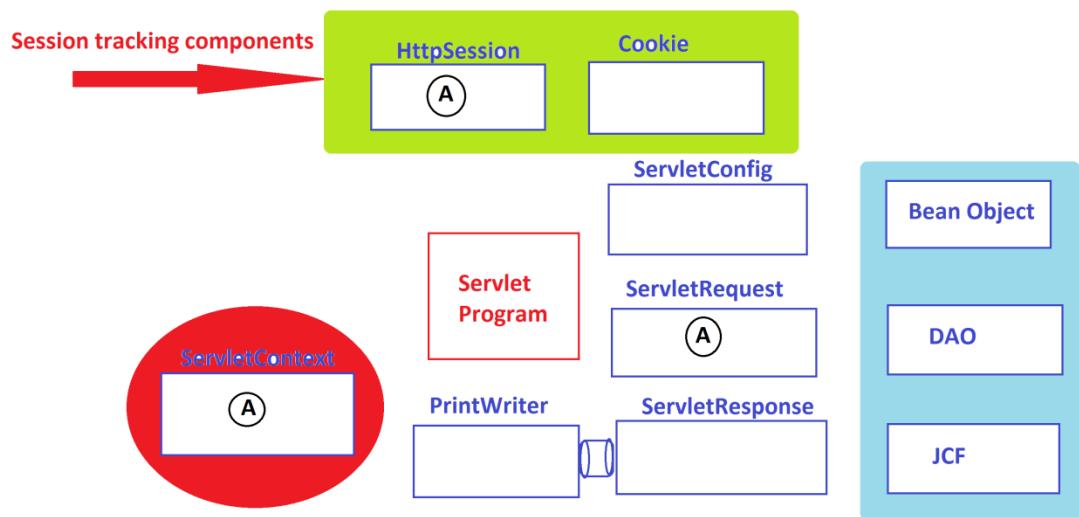
(i)"attribute" in ServletContext object can be used by all the Servlet programs of WebApplication.

(ii)"attribute" in ServletRequest object is available to next Servlet program in forward communication process.

(iii)"attribute" in HttpSession object is available from user-login to user-logout.

=====

Diagram representing objects in Servlet programming:



=====

Venkatesh Mai'

Dt : 21/10/2022

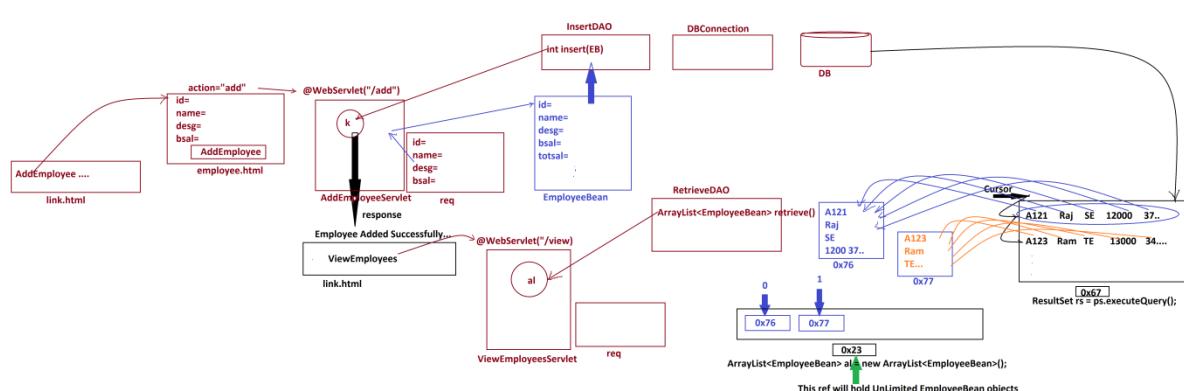
Ex-Application:

Construct Servlet Application related to Employee details.

DB-Table : Employee48

**create table Employee48(id varchar2(10),name varchar2(15),
desg varchar2(10),bsal number(10),totsal number(10,2),primary key(id));**

Layout:



link.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<a href="employee.html">AddEmployee</a>
<a href="view">ViewEmployees</a>
<a href="update">UpdateEmployee</a>
```

```
<a href="delete">DeleteEmployee</a>
</body>
</html>
```

employee.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="add" method="post">
Employee id:<input type="text" name="id"><br>
Employee name:<input type="text" name="name"><br>
Designation:<input type="text" name="desg"><br>
Basic Salary:<input type="text" name="bsal"><br>
<input type="submit" value="AddEmployee">
</form>
</body>
</html>
```

EmployeeBean.java

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class EmployeeBean implements Serializable{
    private String id, name, desg;
    private int bSal;
    private float totSal;
    public EmployeeBean() {}
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDesg() {
        return desg;
    }
}
```

```

}

public void setDesg(String desg) {
    this.desg = desg;
}

public int getbSal() {
    return bSal;
}

public void setbSal(int bSal) {
    this.bSal = bSal;
}

public float getTotSal() {
    return totSal;
}

public void setTotSal(float totSal) {
    this.totSal = totSal;
}

}

```

DBConnection.java

```

package test;
import java.sql.*;
public class DBConnection {
    private static Connection con=null;
    private DBConnection() {}
    static
    {
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe", "system", "manager");
        }catch(Exception e) {e.printStackTrace();}
    }
    public static Connection getCon() {
        return con;
    }
}

```

InsertDAO.java

```

package test;
import java.sql.*;
public class InsertDAO {

```

```

public int k=0;
public int insert(EmployeeBean eb) {
    try {
        Connection con = DBConnection.getCon(); //Accessing
the Connection
        PreparedStatement ps = con.prepareStatement
            ("insert into Employee48
values(?, ?, ?, ?, ?)");
        ps.setString(1, eb.getId());
        ps.setString(2, eb.getName());
        ps.setString(3, eb.getDesg());
        ps.setInt(4, eb.getbSal());
        ps.setFloat(5, eb.getTotSal());
        k = ps.executeUpdate();
    } catch (Exception e) {e.printStackTrace();}
    return k;
}
}

```

AddEmployeeServlet.java

```

package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/add")
public class AddEmployeeServlet extends GenericServlet{

    public EmployeeBean eb=null;
    public InsertDAO ob=null;

    public void init() throws ServletException{
        eb = new EmployeeBean();
        ob = new InsertDAO();
    }
}

```

```

public void service(ServletRequest req,ServletResponse res) throws
ServletException,IOException{
    eb.setId(req.getParameter("id"));
    eb.setName(req.getParameter("name"));
    eb.setDesg(req.getParameter("desg"));
    eb.setbSal(Integer.parseInt(req.getParameter("bsal")));
    eb.setTotSal(eb.getTotSal()+(0.93F*eb.getbSal())+(0.63F*eb.getbSal()));
    int k = ob.insert(eb);
    PrintWriter pw = res.getWriter();
    res.setContentType("text/html");
    if(k>0) {
        pw.println("Employee Added Successfully...<br>");
        RequestDispatcher rd = req.getRequestDispatcher("link.html");
        rd.include(req, res);
    }
}
public void destroy() {
    ob=null;
    eb=null;
}

```

RetrieveDAO.java

package test;

import java.sql.;*

```

import java.util.*;

public class RetrieveDAO {

    public ArrayList<EmployeeBean> al = new ArrayList<EmployeeBean>();

    public ArrayList<EmployeeBean> retrieve()

    {

        try {

            Connection con = DBConnection.getCon();

            PreparedStatement ps = con.prepareStatement

                ("select * from Employee48");

            ResultSet rs = ps.executeQuery();

            while(rs.next())

            {

                EmployeeBean eb = new EmployeeBean();//Bean object

                eb.setId(rs.getString(1));

                eb.setName(rs.getString(2));

                eb.setDesg(rs.getString(3));

                eb.setbSal(rs.getInt(4));

                eb.setTotSal(rs.getFloat(5));

                al.add(eb);//Adding bean to ArrayList

            }//end of loop

        }catch(Exception e) {e.printStackTrace();}

        return al;

    }

}

```

ViewEmployeesServlet.java

```
package test;

import java.io.*;

import javax.servlet.*;
import javax.servlet.annotation.*;
import java.util.*;

@SuppressWarnings("serial")

@WebServlet("/view")

public class ViewEmployeesServlet extends GenericServlet{

    public RetrieveDAO ob = null;

    public void init()throws ServletException{
        ob = new RetrieveDAO();
    }

    public void service(ServletRequest req,ServletResponse res)throws
ServletException,IOException{
        ArrayList<EmployeeBean> al = ob.retrieve();

        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        if(al.size()==0) {
            pw.println("Employees not Available...<br>");
        }else {

            Splitterator<EmployeeBean> sp = al.splitterator();
            sp.forEachRemaining((k)->
{

```

```

EmployeeBean eb = (EmployeeBean)k;
pw.println(eb.getId()+" &nbsp"+eb.getName()+" &nbsp"+
eb.getDesg()+" &nbsp"+eb.getbSal()+" &nbsp"+
eb.getTotSal()+"<br>");
});

}

RequestDispatcher rd = req.getRequestDispatcher("link.html");
rd.include(req, res);

}

public void destroy() {
    ob=null;
}

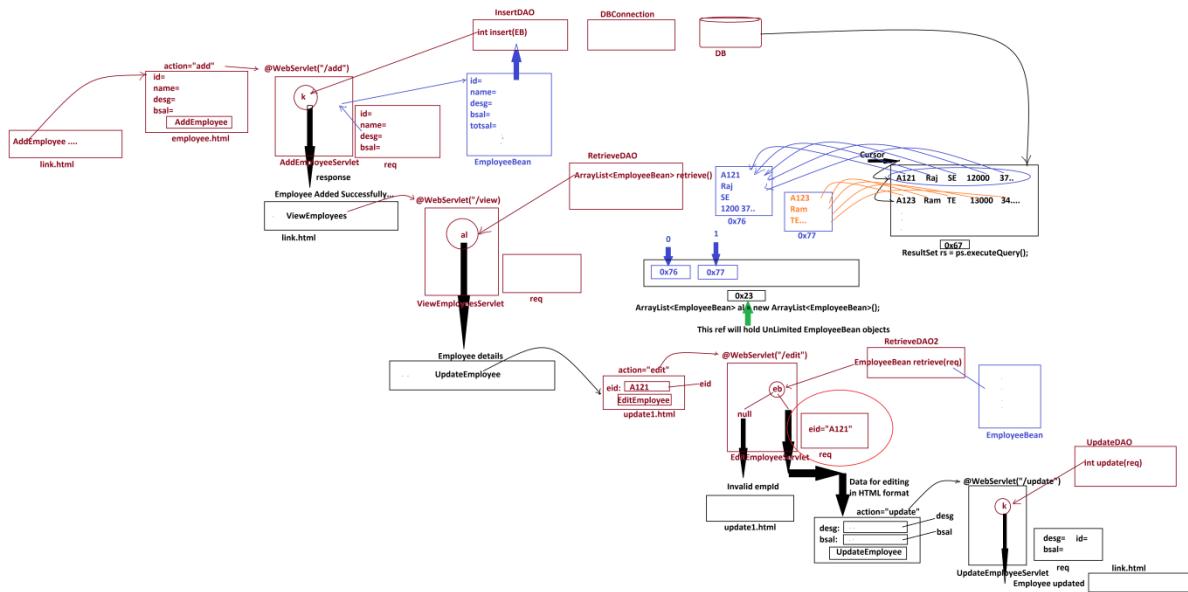
}

web.xml

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>link.html</welcome-file>
    </welcome-file-list>
</web-app>

```

Dt : 22/10/2022



update1.html

```

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="edit" method="post">
Enter EmployeeId:<input type="text" name="eid"><br>
<input type="submit" value="EditEmployee">
</form>
</body>
</html>
  
```

RetrieveDAO2.java

```

package test;

import java.sql.*;
import javax.servlet.*;

public class RetrieveDAO2 {
    public EmployeeBean eb=null;
  
```

```

public EmployeeBean retrieve(ServletRequest req) {
    try {
        Connection con = DBConnection.getCon();
        PreparedStatement ps = con.prepareStatement
            ("select * from Employee48 where id=?");
        ps.setString(1, req.getParameter("eid"));
        ResultSet rs = ps.executeQuery();
        if(rs.next()) {
            eb = new EmployeeBean(); //Bean object
            eb.setId(rs.getString(1));
            eb.setName(rs.getString(2));
            eb.setDesg(rs.getString(3));
            eb.setbSal(rs.getInt(4));
            eb.setTotSal(rs.getFloat(5));
        }
    } catch(Exception e) {e.printStackTrace();}
    return eb;
}
}

```

EditEmployeeServlet.java

```

package test;
import java.io.*;
import javax.servlet.*;

```

```
import javax.servlet.annotation.*;  
  
@SuppressWarnings("serial")  
  
@WebServlet("/edit")  
  
public class EditEmployeeServlet extends GenericServlet{  
  
    public RetrieveDAO2 ob=null;  
  
    public void init()throws ServletException{  
  
        ob = new RetrieveDAO2();  
  
    }  
  
    public void service(ServletRequest req,ServletResponse res)throws  
ServletException,IOException{  
  
        EmployeeBean eb = ob.retrieve(req);  
  
        PrintWriter pw = res.getWriter();  
  
        res.setContentType("text/html");  
  
        if(eb==null) {  
  
            pw.println("Invalid employee id...<br>");  
  
            RequestDispatcher rd = req.getRequestDispatcher("update1.html");  
  
            rd.include(req, res);  
  
        }else {  
  
            pw.println("<form action='update' method='post'>");  
  
            pw.println("<input type='hidden' name='id' value='"+eb.getId()+"'>");  
  
            pw.println("Designation:<input type='text' name='desg'  
value='"+eb.getDesg()+"'><br>");  
  
            pw.println("BasicSalary:<input type='text' name='bsal'  
value='"+eb.getbSal()+"'><br>");  
  
            pw.println("<input type='submit' value='UpdateEmployee'>");  
        }  
    }  
}
```

```

    pw.println("</form>");
}

}

public void destroy() {
    ob=null;
}

}

UpdateDAO.java

package test;

import java.sql.*;
import javax.servlet.*;

public class UpdateDAO {

    public int k=0;

    public int update(ServletRequest req) {
        try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                ("update Employee48 set desg=?,bsal=?,totalsal=? where id=?");
            ps.setString(1,req.getParameter("desg"));
            int bS = Integer.parseInt(req.getParameter("bsal"));
            ps.setInt(2,bS);
            float totS = bS+(0.93F*bS)+(0.63F*bS);
            ps.setFloat(3,totS);
            ps.setString(4,req.getParameter("id"));
        }
    }
}

```

```

        k = ps.executeUpdate();

    }catch(Exception e) {e.printStackTrace();}

    return k;

}

}

UpdateEmployeeServlet.java

package test;

import java.io.*;

import javax.servlet.*;
import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/update")

public class UpdateEmployeeServlet extends GenericServlet{

    public UpdateDAO ob = null;

    public void init()throws ServletException{
        ob = new UpdateDAO();
    }

    public void service(ServletRequest req,ServletResponse res)throws
    ServletException,IOException{

        int k = ob.update(req);

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

        if(k>0) {

            pw.println("Employee Updated Successfully...<br>");
        }
    }
}

```

```
RequestDispatcher rd = req.getRequestDispatcher("link.html");
rd.include(req, res);
}

}

public void destroy() {
    ob=null;
}
=====
```

Assignment-1:

Construct Product Application

AddProduct

ViewProducts

UpdateProduct

DeleteProduct

Assignment-2:

Construct Book Application

AddBook

ViewBooks

UpdateBook

DeleteBook

Dt : 25/10/2022

faq:

define request?

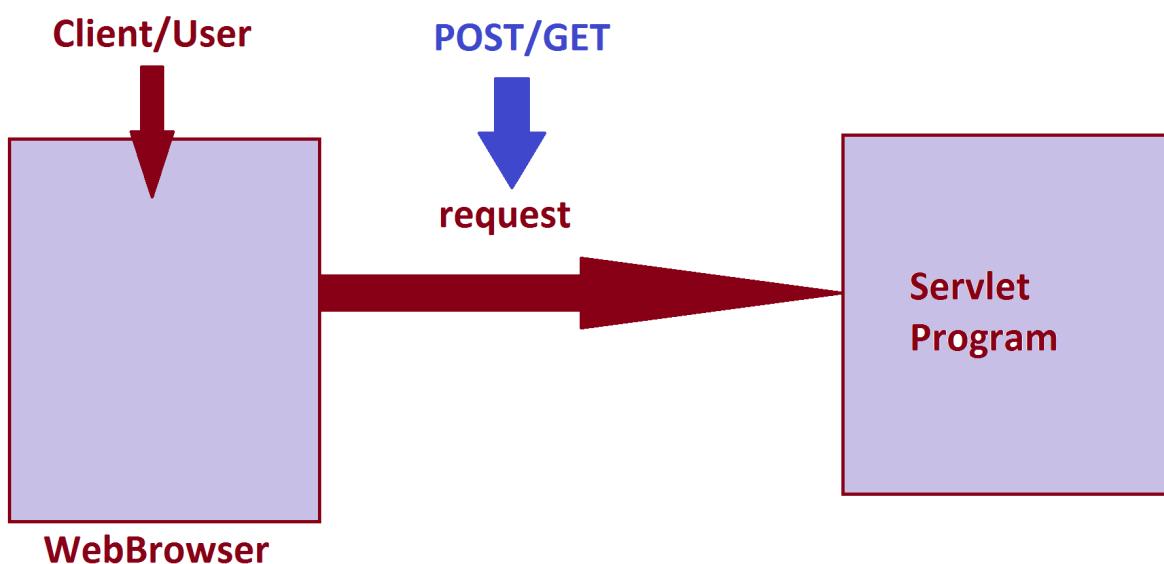
=>The query which is generated from User/Client to the application is known as request.

=>The request is categorized into two types:

1. POST request

2. GET request

Diagram:



1. POST request:

=>The request in which we send the data to Server-program is known as

POST-request.

=>Through POST request we can send UnLimited data.

=>Through POST request we can send any type of data,which means Text,

Audio, Video, Image and Animation data.

=>*The data which is sent through POST request is encapsulated into*

body part of HTTP and which is secured.(Post request is secured)

=>*we raise POST request by declaring method="POST" in <form> tag*

syntax:

```
<form action="servlet-url" method="POST">  
...  
</form>
```

=>*we use doPost() method from "javax.servlet.http.HttpServlet" to*

accept "POST" request.

Method signature of doPost() method:

```
protected void doPost(javax.servlet.http.HttpServletRequest,  
javax.servlet.http.HttpServletResponse)  
throws javax.servlet.ServletException,java.io.IOException;
```

2.GET request:

=>*The request which is used to get the data from Server is known as*

GET-request.

=>*Through GET request we can send Limited data upto 2kb or 4kb or 8kb.*

=>*Through GET request we can send only Text data.*

=>*The data which is sent through GET request is added to query String*

and displayed in address-bar and which is not-secured.

(GET request is not-secured)

=>we use the following four ways to raise GET request:

(i)by declaring method="GET" in <form> tag

syntax:

```
<form action="servlet-url" method="GET">  
....  
</form>
```

(ii)Submitting <form> tag without "method" attribute

syntax:

```
<form action="servlet-url">  
....  
</form>
```

(iii)The request raised through Hyperlinks(href) is GET request

(iv)Get request is raised when we use servlet-url-pattern in
address-bar

=>we use doGet() method from "javax.servlet.http.HttpServlet" to
accept "GET" request.

Method signature of doPost() method:

```
protected void doGet(javax.servlet.http.HttpServletRequest,  
javax.servlet.http.HttpServletResponse)  
throws javax.servlet.ServletException,java.io.IOException;
```

Dt : 26/10/2022

*imp

define Session?

=>The time-interval b/w user-login to user-logout is known as "Session".

define Session Tracking process?

=>The process of checking the state-of-user in session is known as "Session Tracking process".

=>This Session Tracking process in Servlet Programming can be done in four ways:

1. **Cookie Session Tracking process**
2. **HttpSession Session Tracking process**
3. **URL re-write Session Tracking process**
4. **Hidden Form fields Session Tracking process**

*imp

1.Cookie Session Tracking process:

faq:

define Cookie?

=>The piece of information which is persisted b/w multiple client requests is known as "cookie".(persisted means stored and available)

=>cookies are categorized into two types:

(i) **Persistent cookies**

(ii) **Non-Persistent cookies**

(i) Persistent cookies:

=>The Cookies which are stored permanently in WebBrowser until user logout is known as Persistent cookie,which means Cookie is not destroyed when the WebBrowser is closed.

(ii) Non-Persistent cookies:

=>The Cookies which are automatically destroyed when the WebBrowser is closed is known as Non-Persistent cookies.

Note:

=>To "Construct Cookie Session Tracking process",we use "javax.servlet.http.Cookie" class.

=>The following are some important methods of "Cookie" class:

public javax.servlet.http.Cookie(java.lang.String, java.lang.String);

public void setMaxAge(int);

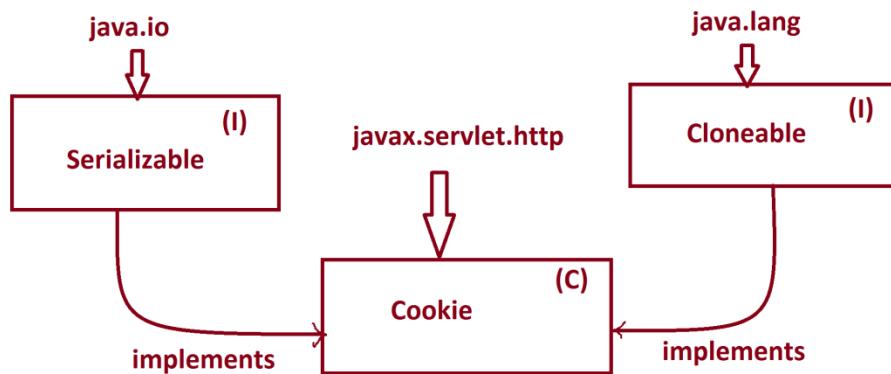
public int getMaxAge();

public void setValue(java.lang.String);

public java.lang.String getValue();

public java.lang.String getName();

Hierarchy of "Cookie":



`Serializable` : supports Serialization process, which convert Object into Stream.

`Cloneable` : supports Cloning process, which means taking backup of object.
(Creating duplicate copy)

Cookie objects are Serializable and Cloneable Objects

=====

=>we use the following steps to construct "Cookie Session Tracking

Process:

step-1 : when User Login-Successfull, then create cookie object

syntax:

```
Cookie ck = new Cookie("name","value");
```

step-2 : Add cookie to the response

syntax:

```
res.addCookie(ck);
```

Note:

=>cookie created in server but stored in WebBrowser

step-3 : Get cookies from the request(request Object)

syntax:

```
Cookie c[] = req.getCookie();
```

step-4 : destroy the Cookie

syntax:

```
c[index].setMaxAge(0);
```

DB-Table : Admin48(uname,pword, fname, lname, addr, mid, phno)

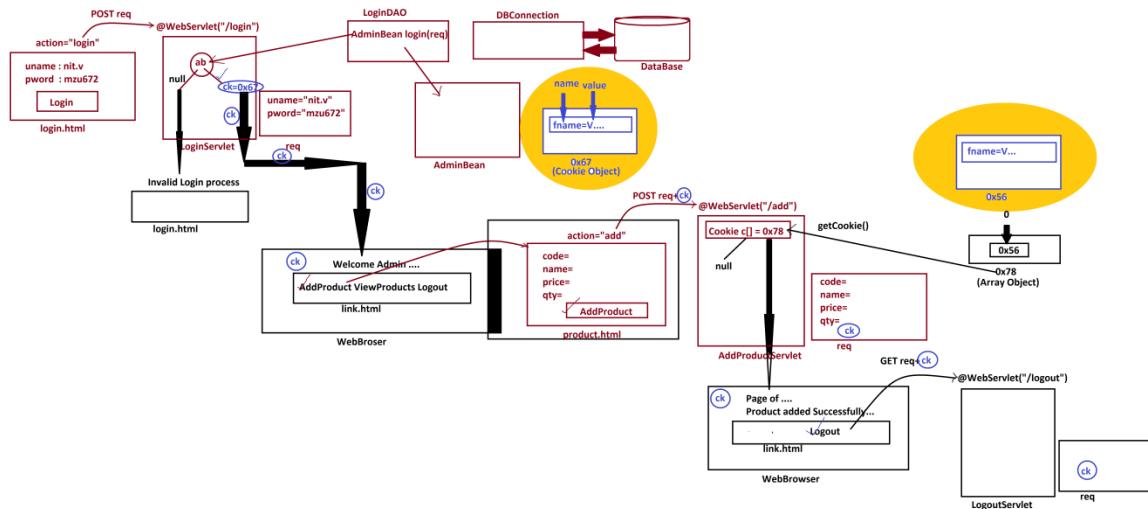
```
create table Admin48(uname varchar2(15), pword varchar2(15),
fname varchar2(15), lname varchar2(15), addr varchar2(15),
mid varchar2(25), phno number(15), primary key(uname, pword));
```

```
insert into Admin48 values('nit.v', 'mzu672', 'Venkat', 'M', 'SRN',
'v@gmail.com', 9898981234);
```

Dt : 27/10/2022

Application : Demonstrating "Cookie" in Session Tracking process

Layout:



`login.html`

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="login" method="post">
UserName:<input type="text" name="uname"><br>
Password:<input type="password" name="pword"><br>
<input type="submit" value="Login">
</form>
</body>
</html>
```

`AdminBean.java`

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class AdminBean implements Serializable{
    private String uName,pWord,fName,lName,addr,mId;
```

```
private long phNo;
public AdminBean() {}
public String getuName() {
    return uName;
}
public void setuName(String uName) {
    this.uName = uName;
}
public String getpWord() {
    return pWord;
}
public void setpWord(String pWord) {
    this.pWord = pWord;
}
public String getfName() {
    return fName;
}
public void setfName(String fName) {
    this.fName = fName;
}
public String getlName() {
    return lName;
}
public void setlName(String lName) {
    this.lName = lName;
}
public String getAddr() {
    return addr;
}
public void setAddr(String addr) {
    this.addr = addr;
}
public String getmId() {
    return mId;
}
public void setmId(String mId) {
    this.mId = mId;
}
public long getPhNo() {
    return phNo;
}
public void setPhNo(long phNo) {
    this.phNo = phNo;
}

}
```

DBConnection.java

```
package test;
import java.sql.*;
public class DBConnection {
    private static Connection con=null;
    private DBConnection() {}
    static
    {
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
        }catch(Exception e) {e.printStackTrace();}
    }
    public static Connection getCon() {
        return con;
    }
}
```

LoginDAO.java

```
package test;

import java.sql.*;
import javax.servlet.http.*;
public class LoginDAO {
    public AdminBean ab=null;
    public AdminBean login(HttpServletRequest req) {
        try{
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                ("select * from Admin48 where uname=? and pword=?");
            ps.setString(1,req.getParameter("uname"));
            ps.setString(2,req.getParameter("pword"));
        }
```

```
ResultSet rs = ps.executeQuery();

if(rs.next()) {

    ab = new AdminBean();

    ab.setuName(rs.getString(1));

    ab.setpWord(rs.getString(2));

    ab.setfName(rs.getString(3));

    ab.setlName(rs.getString(4));

    ab.setAddr(rs.getString(5));

    ab.setmId(rs.getString(6));

    ab.setPhNo(rs.getLong(7));

}

}catch(Exception e) {e.printStackTrace();}

return ab;

}

}

link.html
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<a href="product.html">AddProduct</a>
<a href="view">ViewProducts</a>
<a href="logout">Logout</a>
</body>
</html>
```

LoginServlet.java

```
package test;
```

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/login")
public class LoginServlet extends HttpServlet{
protected void doPost(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException{
AdminBean ab = new LoginDAO().login(req);
PrintWriter pw = res.getWriter();
res.setContentType("text/html");
if(ab==null) {
pw.println("Invalid Login process...<br>");
RequestDispatcher rd = req.getRequestDispatcher("login.html");
rd.include(req, res);
} else {
Cookie ck = new Cookie("fname",ab.getfName());//Creating Cookie
res.addCookie(ck);//Adding Cookie to response
pw.println("Welcome Admin "+ab.getfName()+"<br>");
RequestDispatcher rd = req.getRequestDispatcher("link.html");
rd.include(req, res);
}
}

```

```
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>login.html</welcome-file>
    </welcome-file-list>
</web-app>
```

product.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <form action="add" method="post">
        Product Code:<input type="text" name="code"><br>
        Product Name:<input type="text" name="name"><br>
        Product Price:<input type="text" name="price"><br>
        Product Quantity:<input type="text" name="qty"><br>
        <input type="submit" value="AddProduct">
    </form>
</body>
</html>
```

ProductBean.java

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class ProductBean implements Serializable{
    private String code,name;
    private float price;
    private int qty;
    public ProductBean() {}
    public String getCode() {
        return code;
    }
    public void setCode(String code) {
        this.code = code;
    }
    public String getName() {
```

```

        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public float getPrice() {
        return price;
    }
    public void setPrice(float price) {
        this.price = price;
    }
    public int getQty() {
        return qty;
    }
    public void setQty(int qty) {
        this.qty = qty;
    }
}

}

```

InsertDAO.java

```

package test;
import java.sql.*;
public class InsertDAO {
    public int k=0;
    public int insert(ProductBean pb) {
        try {
            Connection con = DBConnection.getCon(); //Accessing
the Connection
            PreparedStatement ps = con.prepareStatement
                ("insert into Product48 values(?, ?, ?, ?, ?)");
            ps.setString(1, pb.getCode());
            ps.setString(2, pb.getName());
            ps.setFloat(3, pb.getPrice());
            ps.setInt(4, pb.getQty());
            k = ps.executeUpdate();
        } catch (Exception e) {e.printStackTrace();}
        return k;
    }
}

```

AddProductServlet.java

```
package test;
```

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/add")
public class AddProductServlet extends HttpServlet{
protected void doPost(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException{
PrintWriter pw = res.getWriter();
res.setContentType("text/html");
Cookie c[] = req.getCookies(); //Getting cookies from request object
if(c==null) {
pw.println("Session Expired...perform Login process...<br>");
RequestDispatcher rd = req.getRequestDispatcher("login.html");
rd.include(req, res);
} else {
String fName = c[0].getValue();
ProductBean pb = new ProductBean();
pb.setCode(req.getParameter("code"));
pb.setName(req.getParameter("name"));
pb.setPrice(Float.parseFloat(req.getParameter("price")));
pb.setQty(Integer.parseInt(req.getParameter("qty")));
int k = new InsertDAO().insert(pb);
}
}

```

```

pw.println("Page of "+fName+"<br>");

if(k>0) {

    pw.println("Product Added Successfully...<br>");

}

RequestDispatcher rd = req.getRequestDispatcher("link.html");

rd.include(req, res);

}

}


```

LogoutServlet.java

```

package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/logout")

public class LogoutServlet extends HttpServlet{

protected void doGet(HttpServletRequest req,HttpServletResponse res)throws

ServletException,IOException{

PrintWriter pw = res.getWriter();

res.setContentType("text/html");

Cookie c[] = req.getCookies();

if(c==null) {


```

```
pw.println("Session expired...Perform login process...<br>");

}else {

    c[0].setMaxAge(0);

    pw.println("Admin Loggedout Successfully..<br>");

}

RequestDispatcher rd = req.getRequestDispatcher("login.html");

rd.include(req, res);

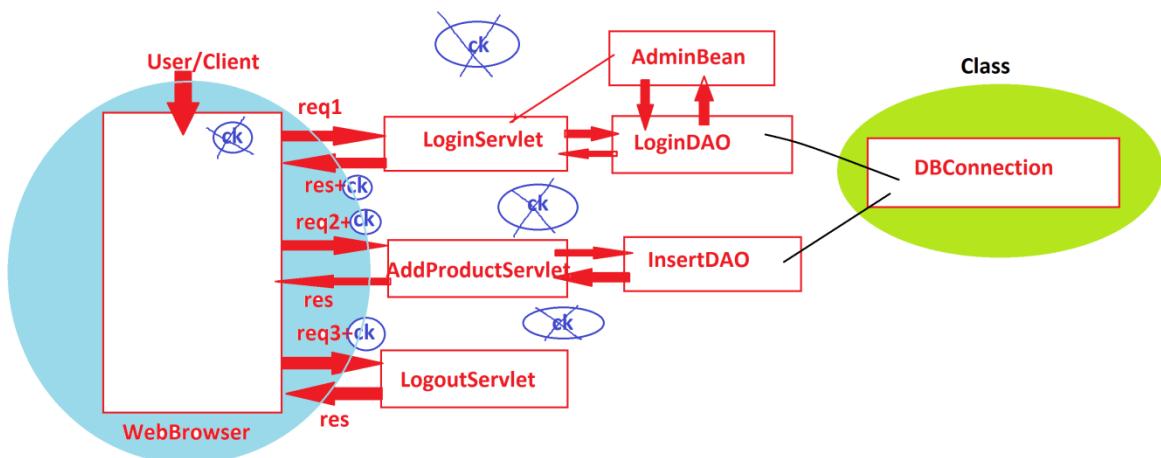
}

}

=====
```

Dt : 28/10/2022

Diagram:



Dis-Advantage of "Cookie" Session Tracking process:

=>"Cookie" Session Tracking process is Browser dependent, which means when we make any changes related to Cookie in WebBrowser then Cookie related application will not work.

*imp

2. "HttpSession" Session Tracking process:

=> HttpSession is an interface from javax.servlet.http package and which is used in Session Tracking process and known as "HttpSession in Session Tracking process"

=>The following are some important methods from HttpSession:

```
public abstract javax.servlet.ServletContext getServletContext();  
  
public abstract void setAttribute(java.lang.String, java.lang.Object);  
public abstract java.lang.Object getAttribute(java.lang.String);  
public abstract void removeAttribute(java.lang.String);  
  
public abstract void putValue(java.lang.String, java.lang.Object);  
public abstract java.lang.Object getValue(java.lang.String);  
public abstract void removeValue(java.lang.String);  
  
public abstract void invalidate();
```

=>we use getSession() method from HttpServletRequest to create the implementation object for "HttpSession" interface.

syntax:

```
HttpSession hs = req.getSession();  
HttpSession hs = req.getSession(false);
```

Application : UserApplication(UserApp)

DB Table : User48(uname,pword, fname, lname, addr, mid, phno)

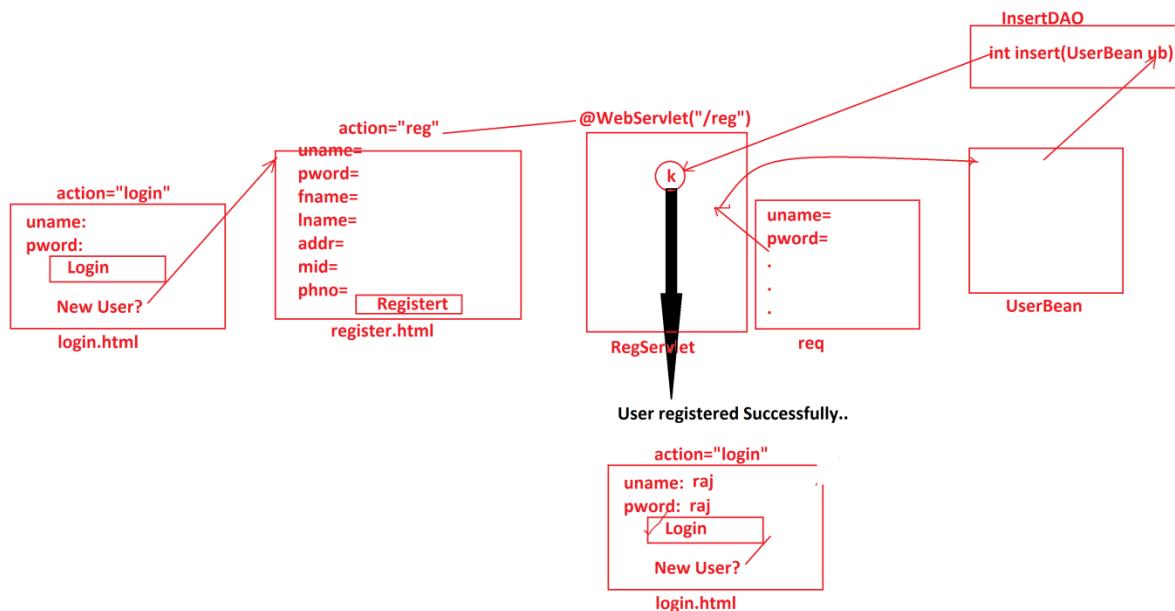
create table User48(uname varchar2(15), pword varchar2(15),

```

fname varchar2(15),lname varchar2(15),addr varchar2(15),
mid varchar2(25),phno number(15),primary key(uname,pword));

```

Layout:



login.html

```

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="login" method="post">
    UserName:<input type="text" name="uname"><br>
    Password:<input type="password" name="pword"><br>
    <input type="submit" value="Login">
    <a href="register.html">NewUser?</a>
</form>
</body>
</html>

```

register.html

```

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="reg" method="post">
UserName:<input type="text" name="uname"><br>
Password:<input type="password" name="pword"><br>
FirstName:<input type="text" name="fname"><br>
LastName:<input type="text" name="lname"><br>
Address:<input type="text" name="addr"><br>
MailId:<input type="text" name="mid"><br>
PhoneNo:<input type="text" name="phno"><br>
<input type="submit" value="Register">
</form>
</body>
</html>

```

UserBean.java

```

package test;
import java.io.*;
@SuppressWarnings("serial")
public class UserBean implements Serializable{
    private String uName,pWord,fName,lName,addr,mId;
    private long phNo;
    public UserBean() {}
    public String getuName() {
        return uName;
    }
    public void setuName(String uName) {
        this.uName = uName;
    }
    public String getpWord() {
        return pWord;
    }
    public void setpWord(String pWord) {
        this.pWord = pWord;
    }
    public String getfName() {
        return fName;
    }
    public void setfName(String fName) {
        this.fName = fName;
    }
}

```

```

public String getName() {
    return lName;
}
public void setlName(String lName) {
    this.lName = lName;
}
public String getAddr() {
    return addr;
}
public void setAddr(String addr) {
    this.addr = addr;
}
public String getmId() {
    return mId;
}
public void setmId(String mId) {
    this.mId = mId;
}
public long getPhNo() {
    return phNo;
}
public void setPhNo(long phNo) {
    this.phNo = phNo;
}

}

```

DBConnection.java

```

package test;
import java.sql.*;
public class DBConnection {
    private static Connection con=null;
    private DBConnection() {}
    static
    {
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
        }catch(Exception e) {e.printStackTrace();}
    }
    public static Connection getCon() {
        return con;
    }
}

```

```
}
```

InsertDAO.java

```
package test;
import java.sql.*;
public class InsertDAO {
    public int k=0;
    public int insert(UserBean ub) {
        try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                ("insert into User48
values(?,?,?,?,?,?)");
            ps.setString(1,ub.getuName());
            ps.setString(2,ub.getpWord());
            ps.setString(3,ub.getfName());
            ps.setString(4,ub.getlName());
            ps.setString(5,ub.getAddr());
            ps.setString(6,ub.getmId());
            ps.setLong(7,ub.getPhNo());
            k = ps.executeUpdate();
        }catch(Exception e) {e.printStackTrace();}
        return k;
    }
}
```

RegServlet.java

```
package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

@SuppressWarnings("serial")
@WebServlet("/reg")

public class RegServlet extends HttpServlet{
```

```

protected void doPost(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException{
    PrintWriter pw = res.getWriter();
    res.setContentType("text/html");
    UserBean ub = new UserBean();
    ub.setuName(req.getParameter("uname"));
    ub.setpWord(req.getParameter("pword"));
    ub.setfName(req.getParameter("fname"));
    ub.setlName(req.getParameter("lname"));
    ub.setAddr(req.getParameter("addr"));
    ub.setmId(req.getParameter("mid"));
    ub.setPhNo(Long.parseLong(req.getParameter("phno")));
    int k = new InsertDAO().insert(ub);
    if(k>0) {
        pw.println("User registered Successfully...<br>");
        RequestDispatcher rd = req.getRequestDispatcher("login.html");
        rd.include(req, res);
    }
}
}

```

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>login.html</welcome-file>
    </welcome-file-list>
</web-app>

```

Dt : 1/11/2022

link.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<a href="view">ViewProfile</a>
<a href="">EditUpdateProfile</a>
<a href="logout">Logout</a>
</body>
</html>
```

LoginDAO.java

```
package test;

import java.sql.*;
import javax.servlet.http.*;

public class LoginDAO {

    public UserBean ub = null;

    public UserBean login(HttpServletRequest req) {
        try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                ("select * from User48 where uname=? and pword=?");
            ps.setString(1,req.getParameter("uname"));
            ps.setString(2,req.getParameter("pword"));
            ResultSet rs = ps.executeQuery();
            if(rs.next()) {
                ub = new UserBean(); //Bean object
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

```

        ub.setuName(rs.getString(1));
        ub.setpWord(rs.getString(2));
        ub.setfName(rs.getString(3));
        ub.setlName(rs.getString(4));
        ub.setAddr(rs.getString(5));
        ub.setmId(rs.getString(6));
        ub.setPhNo(rs.getLong(7));
    }

} catch(Exception e) {e.printStackTrace();}
return ub;
}

}

UserLoginServlet.java

package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/login")

public class UserLoginServlet extends HttpServlet{
protected void doPost(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException{

```

```

PrintWriter pw = res.getWriter();
res.setContentType("text/html");
UserBean ub = new LoginDAO().login(req);
if(ub==null) {
    pw.println("Invalid Login process...<br>");
    RequestDispatcher rd = req.getRequestDispatcher("login.html");
    rd.include(req, res);
} else {
    HttpSession hs = req.getSession(); //Creating new Session
    hs.setAttribute("ubean", ub); //Adding bean to Session
    pw.println("Welcome User "+ub.getfName()+"<br>");
    RequestDispatcher rd = req.getRequestDispatcher("link.html");
    rd.include(req, res);
}
}

ViewProfileServlet.java
package test;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/view")

```

```

public class ViewProfileServlet extends HttpServlet{
    protected void doGet(HttpServletRequest req,HttpServletResponse res) throws
    ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        HttpSession hs = req.getSession(false); //Accessing existing Session
        if(hs==null) {
            pw.println("Session Expired....perform Login process...<br>");
            RequestDispatcher rd = req.getRequestDispatcher("login.html");
            rd.include(req, res);
        }else {
            UserBean ub = (UserBean)hs.getAttribute("ubean");
            pw.println("Page of "+ub.getfName()+"<br>");
            RequestDispatcher rd = req.getRequestDispatcher("link.html");
            rd.include(req, res);
            pw.println("<br>"+ub.getfName()+"&nbsp&nbsp"+ub.getlName()+"&nbsp&nbsp"+
            "ub.getAddr()+"&nbsp&nbsp"+ub.getId()+"&nbsp&nbsp"+ub.getPhNo());
        }
    }
}

```

LogoutServlet.java

```

package test;
import java.io.*;

```

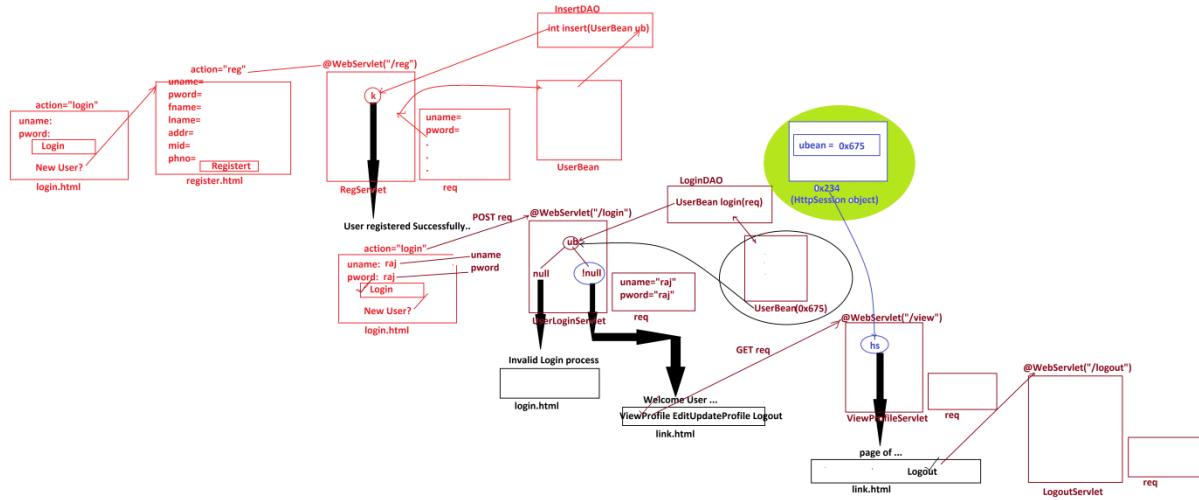
```

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

@SuppressWarnings("serial")
@WebServlet("/logout")

public class LogoutServlet extends HttpServlet{
    protected void doGet(HttpServletRequest req,HttpServletResponse res) throws
    ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        HttpSession hs = req.getSession(false); //Accessing existing Session
        if(hs==null) {
            pw.println("Session expired...perform Login process...<br>");
        }else {
            hs.removeAttribute("ubean");
            hs.invalidate(); //Session destroyed
            pw.println("User Logged out Successfully...<br>");
        }
        RequestDispatcher rd = req.getRequestDispatcher("login.html");
        rd.include(req, res);
    }
}

```



Assignment:

Update above application EditUpdate profile.

=>`addr,mailId,phNo`

*imp

Process of Constructing HttpSession in Session Tracking process:

step-1 : when login process is successfull then create Session object.

step-2 : Add tracking information to Session Object

step-3 : we permit user to perform actions in the application when Session is available.

step-4 : Invalidate the Session when user logs out.

faq:

wt is the diff b/w

(i)getSession()

(ii)getSession(false)

(iii)getSession(true)

=>getSession() will create new Session when Session not available.

=>getSession(false) will access existing session if available,

if not available it will not create new session.

=>getSession(true) is same like getSession()

faq:

wt is the diff b/w

(i)Cookie

(ii)HttpSession

(i)Cookie:

=>Cookie is a class and which used in Session Tracking process.

=>Cookie is Browser dependent or Client dependent.

(ii)HttpSession:

=>HttpSession is an interface and which is used in Session Tracking process.

=>*HttpSession is Server dependent.*

Venkatesh Maiopathiji

Dt : 2/11/2022

3. URL re-write Session Tracking process:

=> The process of adding parameter-values to Servlet-url-pattern and passing the information from one Servlet-program to another Servlet-program is known as "URL re-writing process".

syntax:

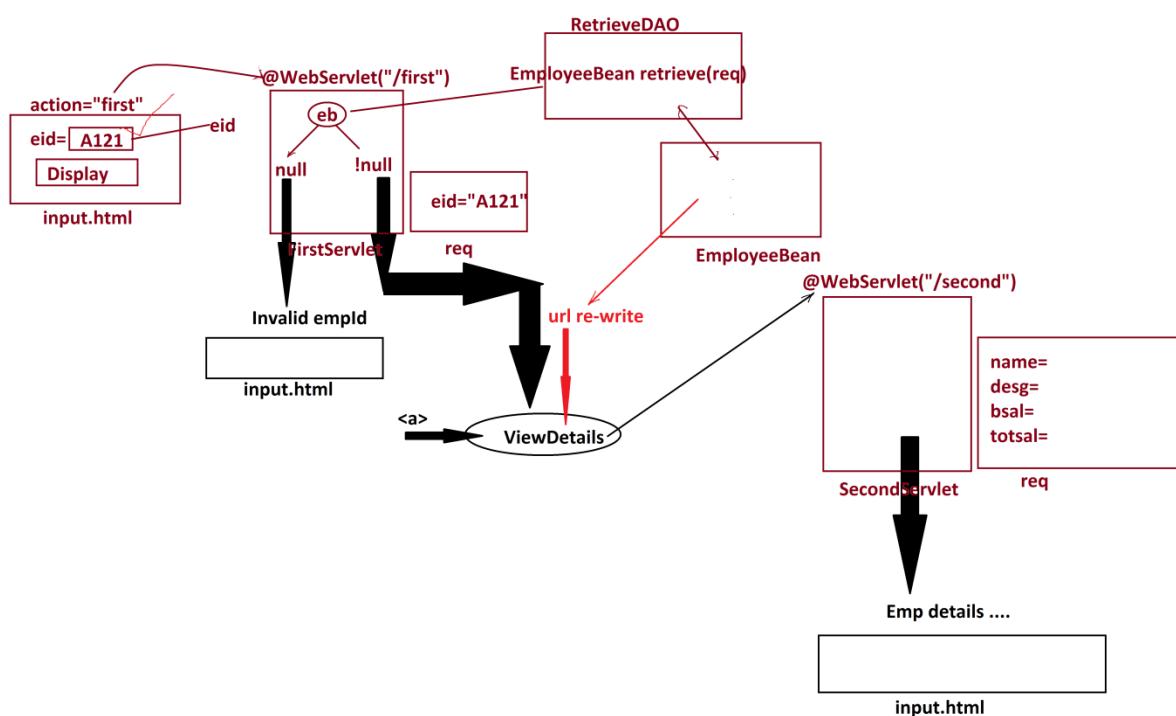
Servlet-url-pattern?para1=value¶2=value¶3=value&...

=> "?" symbol is separator b/w Servlet-url-pattern and parameters.

=> "&" symbol is separator b/w parameters

Ex-Application:

Layout:



input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <form action="first" method="post">
        Enter Employee Id : <input type="text" name="eid"><br>
        <input type="submit" value="Display">
    </form>
</body>
</html>
```

EmployeeBean.java

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class EmployeeBean implements Serializable{
    private String id,name,desg;
    private int bSal;
    private float totSal;
    public EmployeeBean() {}
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDesg() {
        return desg;
    }
    public void setDesg(String desg) {
        this.desg = desg;
    }
    public int getbSal() {
        return bSal;
    }
}
```

```

public void setbSal(int bSal) {
    this.bSal = bSal;
}
public float getTotSal() {
    return totSal;
}
public void setTotSal(float totSal) {
    this.totSal = totSal;
}

}

```

DBConnection.java

```

package test;
import java.sql.*;
public class DBConnection {
    private static Connection con=null;
    private DBConnection() {}
    static
    {
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe", "system", "manager");
        }catch(Exception e) {e.printStackTrace();}
    }
    public static Connection getCon() {
        return con;
    }
}

```

RetrieveDAO.java

```

package test;
import java.sql.*;
import javax.servlet.http.*;
public class RetrieveDAO {
    public EmployeeBean eb=null;

```

```

public EmployeeBean retrieve(HttpServletRequest req)
{
    try {
        Connection con = DBConnection.getCon();
        PreparedStatement ps = con.prepareStatement
            ("select * from Employee48 where id=?");
        ps.setString(1,req.getParameter("eid"));
        ResultSet rs = ps.executeQuery();
        if(rs.next())
        {
            eb = new EmployeeBean(); //Bean object
            eb.setId(rs.getString(1));
            eb.setName(rs.getString(2));
            eb.setDesg(rs.getString(3));
            eb.setbSal(rs.getInt(4));
            eb.setTotSal(rs.getFloat(5));
        }
        //end of loop
    }catch(Exception e) {e.printStackTrace();}
    return eb;
}

```

FirstServlet.java

```

package test;

import java.io.*;

```

```

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

@SuppressWarnings("serial")
@WebServlet("/first")

public class FirstServlet extends HttpServlet{

protected void doPost(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException{

PrintWriter pw = res.getWriter();
res.setContentType("text/html");
EmployeeBean eb = new RetrieveDAO().retrieve(req);
if(eb==null) {
pw.println("Invalid employee id...<br>");
RequestDispatcher rd = req.getRequestDispatcher("input.html");
rd.include(req, res);
} else {
pw.println("<a href='second?name="+eb.getName()
+"&desg="+eb.getDesg()+"&bsal="+eb.getbSal()
+"&totsal="+eb.getTotSal()+"'>ViewEmployeeDetails</a>");
}
}
}

```

SecondServlet.java

```

package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/second")

public class SecondServlet extends HttpServlet{

    protected void doGet(HttpServletRequest req,HttpServletResponse res) throws
    ServletException,IOException{

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

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

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

        int bSal = Integer.parseInt(req.getParameter("bsal"));

        float totSal = Float.parseFloat(req.getParameter("totsal"));

        pw.println(name+" &nbsp;"+desg+" &nbsp;"+bSal+" &nbsp;"+totSal+"  
");

        RequestDispatcher rd = req.getRequestDispatcher("input.html");

        rd.include(req, res);
    }
}

```

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>

```

```
<welcome-file-list>
    <welcome-file>input.html</welcome-file>
</welcome-file-list>
</web-app>
```

=====

http://localhost:8082/Servlet_App8_URL_ReWrite/second?

name=Raj

&desg=SE

&bsal=12000

&totsal=30720.0

https://www.google.com/search?

q=gmail

&rlz=1C1YUH_enIN1024IN1024

&oq=

&aqs=chrome.0.35i39i362l2j46i39i199i362i465j35i39i362l5.3531427585j0j7

&sourceid=chrome

&ie=UTF-8

https://www.google.com/search?

q=new+gmail+login

&rlz=1C1YUH_enIN1024IN1024

&sxsrf=ALiCzsq_4BSAnIVGs3PU6zty9qhBq5MmjQ%3A1667388085567

&ei=tVJiY9ygluKS8QP4wozYDA

&oq=gmail

&gs_lcp=Cxnd3Mtd2I6LXNlcnAQARgBMgoIBBHENEELADMgoIBBHENEELADMgoIBBHENEELADMgoIBBHENEELADMgoIBBHENEELADMgoIBBHENEELADMgoIBBHENEELADMgoIBBHENEELADMgoIBBHENEELADMgoIBBHENEELADMgoIBBHENEELADMgoIBBHENEELADMgcIABCwAxBDMgcIABCwAxBDSgQIQRgASgQIRhgAUABYAGCGCmgBcAF4AIABAigBAJIBAJgBAMgBCsABAQ

&client=gws-wiz-serp

4. Hidden Form fields Session Tracking process:

=>The process of declaring `<input type="hidden">` in `<form>` tag is known as **Hidden Form field**.

=>Using Hidden form fields we can send information from one Servlet program to another Servlet program.

syntax:

```
<form action="" method="POST">  
    <input type="hidden" name="name" value="value">  
    .  
</form>
```

=>The data which is available in hidden form fields are not displayed to the end-users.

Ex-application:

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
```

```

<title>Insert title here</title>
</head>
<body>
    <form action="first" method="post">
        Enter Employee Id : <input type="text" name="eid"><br>
        <input type="submit" value="Display">
    </form>
</body>
</html>

```

EmployeeBean.java

```

package test;
import java.io.*;
@SuppressWarnings("serial")
public class EmployeeBean implements Serializable{
    private String id, name, desg;
    private int bSal;
    private float totSal;
    public EmployeeBean() {}
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDesg() {
        return desg;
    }
    public void setDesg(String desg) {
        this.desg = desg;
    }
    public int getbSal() {
        return bSal;
    }
    public void setbSal(int bSal) {
        this.bSal = bSal;
    }
    public float getTotSal() {
        return totSal;
    }
}

```

```
    public void setTotSal(float totSal) {
        this.totSal = totSal;
    }
}
```

DBConnection.java

```
package test;
import java.sql.*;
public class DBConnection {
    private static Connection con=null;
    private DBConnection() {}
    static
    {
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
        }catch(Exception e) {e.printStackTrace();}
    }
    public static Connection getCon() {
        return con;
    }
}
```

RetrieveDAO.java

```
package test;
import java.sql.*;
import javax.servlet.http.*;
public class RetrieveDAO {
    public EmployeeBean eb=null;
    public EmployeeBean retrieve(HttpServletRequest req)
    {
        try {

```

```

Connection con = DBConnection.getCon();

PreparedStatement ps = con.prepareStatement
        ("select * from Employee48 where id=?");

ps.setString(1,req.getParameter("eid"));

ResultSet rs = ps.executeQuery();

if(rs.next())
{

    eb = new EmployeeBean(); //Bean object

    eb.setId(rs.getString(1));

    eb.setName(rs.getString(2));

    eb.setDesg(rs.getString(3));

    eb.setbSal(rs.getInt(4));

    eb.setTotSal(rs.getFloat(5));

}

//end of loop

}catch(Exception e) {e.printStackTrace();}

return eb;
}
}

FirstServlet.java

package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.*;

```

```

@SuppressWarnings("serial")

@WebServlet("/first")

public class FirstServlet extends HttpServlet{

protected void doPost(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException{

PrintWriter pw = res.getWriter();

res.setContentType("text/html");

EmployeeBean eb = new RetrieveDAO().retrieve(req);

if(eb==null){

pw.println("Invalid employee id...<br>");

RequestDispatcher rd = req.getRequestDispatcher("input.html");

rd.include(req, res);

} else {

pw.println("<form action='second'>");

pw.println("<input type='hidden' name='name' value='"+eb.getName()+"'>");

pw.println("<input type='hidden' name='desg' value='"+eb.getDesg()+"'>");

pw.println("<input type='hidden' name='bsal' value='"+eb.getbSal()+"'>");

pw.println("<input type='hidden' name='totsal' value='"+eb.getTotSal()+"'>");

pw.println("<input type='submit' value='ViewEmployeeDetails'>");

pw.println("</form>");

}

}
}

```

SecondServlet.java

```
package test;

import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/second")

public class SecondServlet extends HttpServlet{

    protected void doGet(HttpServletRequest req,HttpServletResponse res) throws
    ServletException,IOException{

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

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

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

        int bSal = Integer.parseInt(req.getParameter("bsal"));

        float totSal = Float.parseFloat(req.getParameter("totsal"));

        pw.println(name+"  "+desg+"  "+bSal+"  "+totSal+"<br>");

        RequestDispatcher rd = req.getRequestDispatcher("input.html");

        rd.include(req, res);

    }

}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>input.html</welcome-file>
    </welcome-file-list>
</web-app>
```

Note:

=>"URL re-write process" and "Hidden Form fields" are Sub-techniques used in Session tracking process to transfer the information b/w Servlet Programs.

Assignment:

Transfer book details from Servlet program to another Servlet program based on bookCode.

Dt : 3/11/2022

Summary of an Objects Created in Servlet Programming:

1.ServletContext

2.ServletConfig

3.ServletRequest/HttpServletRequest

4.ServletResponse/HttpServletResponse

5.PrintWriter(Class)

6.Cookie(Class)

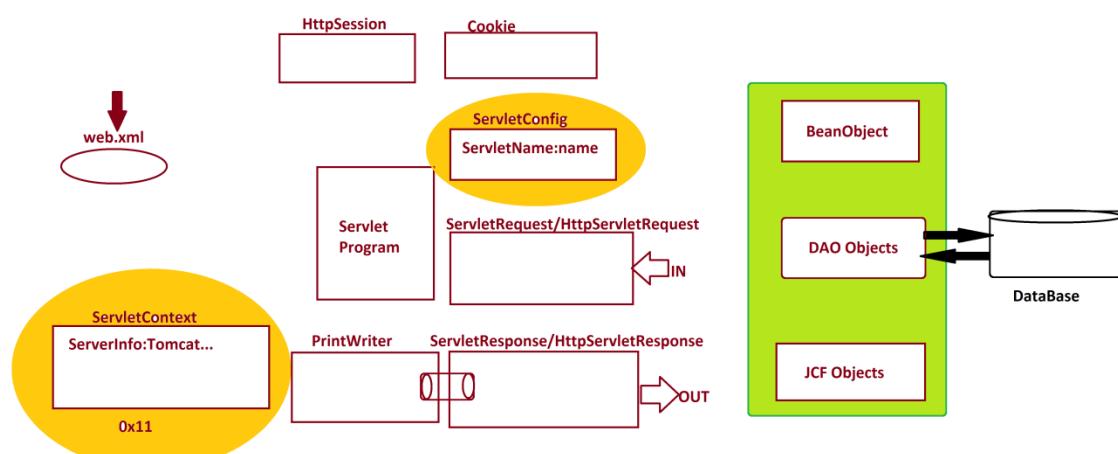
7.HttpSession

8.Bean Objects

9.DAO Objects

10.JCF Objects

Diagram:



***imp**

ServletContext:

=>*ServletContext is an interface from javax.servlet package and which is instantiated automatically when the WebApp is deployed into Server.*

=>*This ServletContext is recorded with Server-information*

=>*we use getServletContext() method to access the reference of ServletContext object.*

syntax:

ServletContext sct = this.getServletContext();

=>*we use <context-param> tag in web.xml to initialize parameters in ServletContext object.*

**imp*

ServletConfig:

=>*ServletConfig is an interface from javax.servlet package and which is instantiated automatically when Servlet program loaded for execution.*

=>*This ServletConfig object is recorded with Servlet-name.*

=>*we use getServletConfig() method to access the reference of ServletConfig Object.*

syntax:

ServletConfig sc = this.getServletConfig();

=>*we use <init-param> tag in <servlet> tag to initialize parameters in ServletConfig object.*

structure of web.xml

```
<web-app>

<context-param></context-param>

<servlet>

<servlet-name> name </servlet-name>

<servlet-class> class </servlet-class>

<init-param></init-param>

</servlet>

<servlet-mapping>

<servlet-name> name </servlet-name>

<url-pattern> url </url-pattern>

</servlet-mapping>

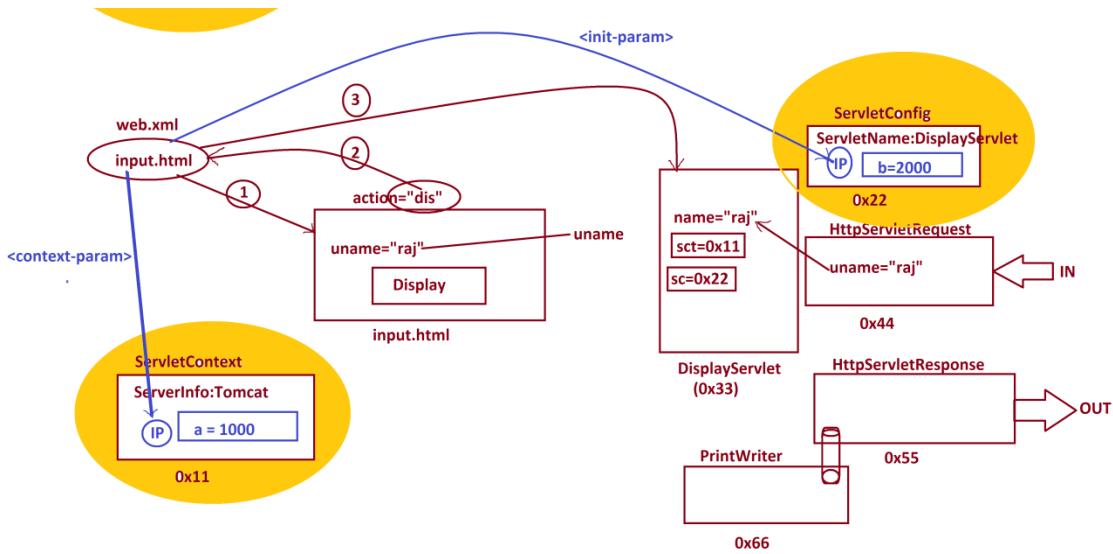
<welcome-file-list>

<welcome-file> file </welcome-file>

</welcome-file-list>

</web-app>
```

Layout:



`input.html`

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="dis" method="post">
UserName:<input type="text" name="uname"><br>
<input type="submit" value="Display">
</form>
</body>
</html>
```

`web.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
<context-param>
<param-name>a</param-name>
<param-value>1000</param-value>
</context-param>

<servlet>
<servlet-name>DisplayServlet</servlet-name>
<servlet-class>test.DisplayServlet</servlet-class>
<init-param>
<param-name>b</param-name>
```

```

        <param-value>2000</param-value>
    </init-param>
</servlet>
<servlet-mapping>
    <servlet-name>DisplayServlet</servlet-name>
    <url-pattern>/dis</url-pattern>
</servlet-mapping>

<welcome-file-list>
    <welcome-file>input.html</welcome-file>
</welcome-file-list>
</web-app>

```

Display.java

```

package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

@SuppressWarnings("serial")

public class DisplayServlet extends HttpServlet{
    protected void doPost(HttpServletRequest req,HttpServletResponse res) throws
    ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        String name = req.getParameter("uname");
        pw.println("Welcome User "+name+"<br>");
        ServletContext sct = this.getServletContext();
        //Accessing reference of Context Object
        pw.println("****ServletContext****");
        pw.println("<br>ServerInfo:"+sct.getServerInfo());
        int a = Integer.parseInt(sct.getInitParameter("a"));
    }
}

```

```
pw.println("<br>ContextValue:"+a);  
  
ServletConfig sc = this.getServletConfig();  
  
//Accessing reference of Config Object  
  
pw.println("<br>****ServletConfig****");  
  
pw.println("<br>ServletName:"+sc.getServletName());  
  
int b = Integer.parseInt(sc.getInitParameter("b"));  
  
pw.println("<br>ConfigValue:"+b);  
  
}  
  
}  
  
=====
```

Dt : 4/11/2022

faq:

wt is the diff b/w

(i)getParameter()

(ii)getInitParameter()

(i)getParameter():

*=>getParameter() method is used to read parameter-values from
ServletRequest object.*

(ii)getInitParameter():

*=>getInitParameter() method is used to read initialized parameter-values
from ServletContext and ServletConfig Objects.*

Note:

*=>Only one ServletContext object is created for WebApplication and the
information in ServletContext object can be used by all the Servlet
programs of WebApplication.*

*=>Every Servlet program will have its own ServletConfig object and
the information in ServletConfig object can be used by only one Servlet
program.*

**imp*

define Servlet-life cycle?

=>Servlet-life cycle demonstrates diff states of Servlet-program from Starting to ending.

=>The followings are the stages in Servlet-life cycle:

1. Loading process
2. Instantiation process
3. Initialization process
4. Request handling process
5. destroying process

1. Loading process:

=>The process of identifying the servlet-program based on url-pattern available in 'web.xml or annotation' and loading onto WebContainer for execution is known as "Loading process".

2. Instantiation process:

=>when the Servlet-program loaded on to WebContainer, it is automatically instantiated known as Instantiation process. (Object creation process)

=>After Instantiation process the following three Life-cycle methods are executed automatically:

GenericServlet:

```
init()  
service()  
destroy()
```

HttpServlet :

init()

service() / doPost() / doGet()

destroy()

3. Initialization process:

=> The process of making the programming components ready available for *service()* or *doPost()* or *doGet()* methods using *init()* method is known as *Initialization process*.

Note:

=> Through initialization process we create Bean objects, DAO objects or any SubClass objects.

4. Request handling process:

=> The process of accepting the request from User or Client using *service()* method or *doPost()* method or *doGet()* method and providing the response is known as "Request handling process".

5. Destroying process:

=> The process of destroying the opened resources using *destroy()* method is known as *destroying process*.

=====

faq:

define Life-cycle methods?

=> The methods which are executed automatically in the same order are

known as *Life-cycle methods*.

Note:

=>In the process of executing multiple requests from multiple users,

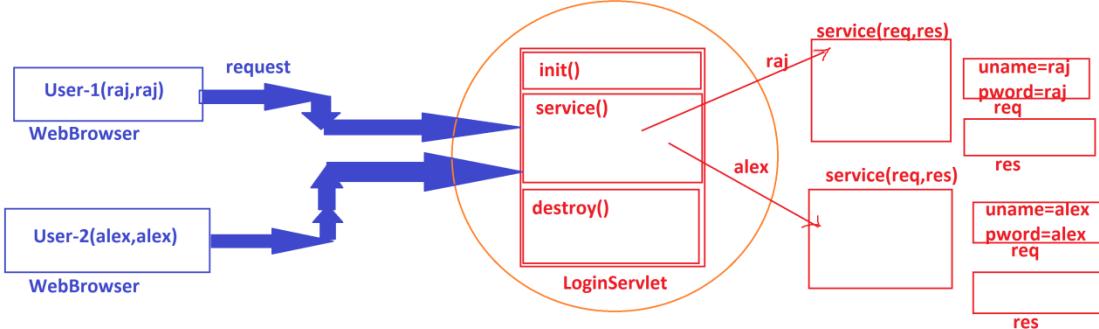
(i) Loading process performed only once

(ii) Initialization process performed only once

(iii) service() method executed multiple times to handle multiple requests

(iv) Destroying process performed only once.

Diagram:



Dt : 5/11/2022

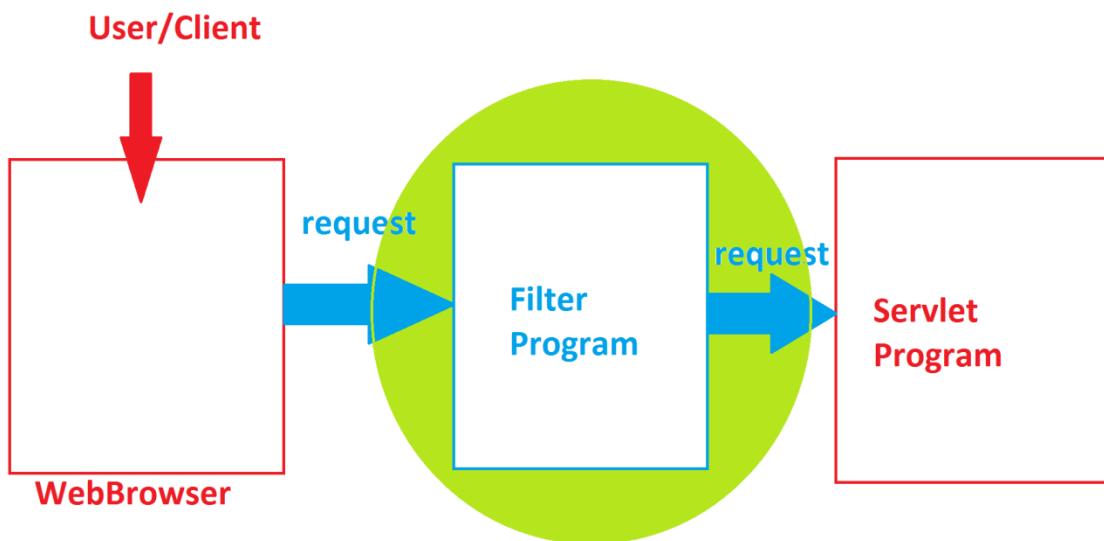
*imp

Filters in Servlet Programming:

=>**Filter is a pre-processing component which is executed before**

Servlet program, which means Filter is a Security-Gateway for Servlet program.

Diagram:



Advantage of Filters:

- (i) we can perform Authentication and Authorization process
- (ii) we can perform Data Conversion process
- (iii) we can perform Data Compression process
- (iv) we can write security-code in filters

=>The use the following components in constructing Filter programs:

1.Filter

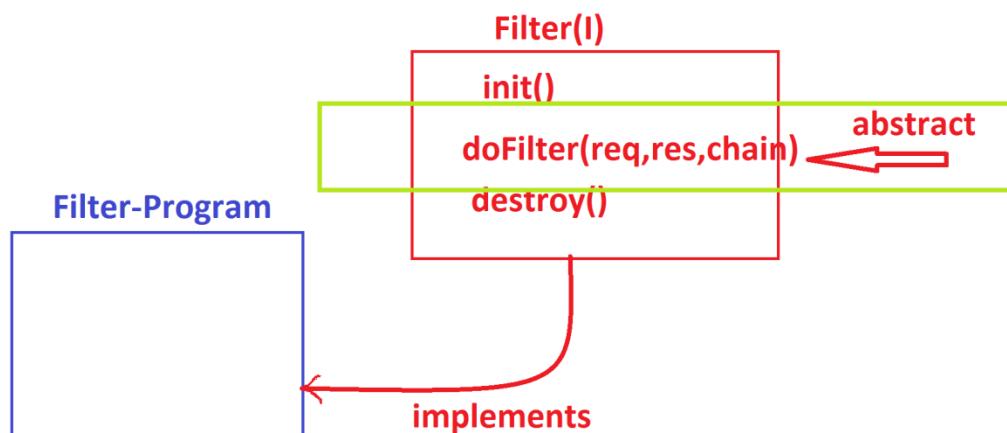
2.FilterChain

3.FilterConfig

1.Filter:

=>Filter is an interface from javax.servlet package and which is implemented,in the process of constructing filter-programs.

Diagram:



=>The following are the Life-Cycle methods Filter program:

```
public void init(javax.servlet.FilterConfig)
```

```
throws javax.servlet.ServletException;  
public abstract void doFilter(javax.servlet.ServletRequest,  
javax.servlet.ServletResponse, javax.servlet.FilterChain)  
throws java.io.IOException, javax.servlet.ServletException;  
public void destroy();
```

2.FilterChain:

- =>*FilterChain is an interface from javax.servlet package and which is instantiated automatically while doFilter() method execution.*
- =>*FilterChain object is used to establish link to the Servlet program*
- =>*The following is one important method of FilterChain:*

```
public abstract void doFilter(javax.servlet.ServletRequest,  
javax.servlet.ServletResponse) throws java.io.IOException,  
javax.servlet.ServletException;
```

3.FilterConfig:

- =>*FilterConfig is an interface from javax.servlet package and which is instantiated automatically when the filter program is loaded onto WebContainer for execution.*
 - =>*FilterConfig object is loaded with Filter-name.*
 - =>*we access FilterConfig object using init() method*
-

Note:

=>we can use **<init-param>** tag in **<filter>** tag to initialize parameters in **FilterConfig** object.

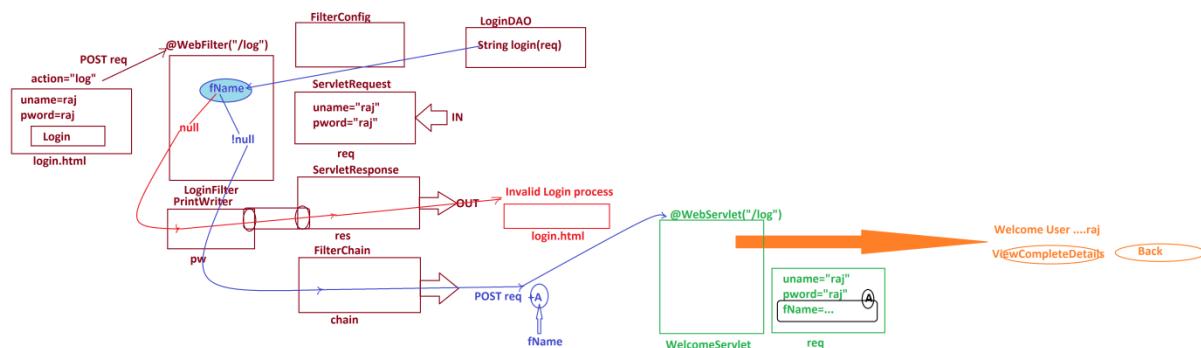
syntax:

```
<web-app>
.
.
<filter>
    <filter-name> name </filter-name>
    <filter-class> class </filter-class>
    <init-param>
        <param-name> name </param-name>
        <param-value> value </param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name> name </filter-name>
    <url-pattern> url </url-pattern>
</filter-mapping>
.
.
</web-app>
```

Dt : 7/11/2022

EX-Application : Demonstrationg Login process using Filter.

Layout:



DBConnection.java

```

package test;
import java.sql.*;
public class DBConnection {
    private static Connection con=null;
    private DBConnection() {}
    static
    {
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
        }catch(Exception e) {e.printStackTrace();}
    }
    public static Connection getCon() {
        return con;
    }
}

```

LoginDAO.java

```

package test;
import java.sql.*;

```

```

import javax.servlet.*;
public class LoginDAO {
    public String fName = null;
    public String login(ServletRequest req) {
        try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                ("select * from User48 where uname=? and pword=?");
            ps.setString(1,req.getParameter("uname"));
            ps.setString(2,req.getParameter("pword"));
            ResultSet rs = ps.executeQuery();
            if(rs.next()) {
                fName = rs.getString(3);
            }
        }catch(Exception e) {e.printStackTrace();}
        return fName;
    }
}
login.html
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="log" method="post">

```

```

UserName:<input type="text" name="uname"><br>
Password:<input type="password" name="pword"><br>
<input type="submit" value="Login">
</form>
</body>
</html>

```

LoginFilter.java

```

package test;

import java.io.*;

import javax.servlet.*;
import javax.servlet.annotation.*;

@WebFilter("/log")
public class LoginFilter implements Filter{

    public LoginDAO ob = null;

    public void init(FilterConfig fc) throws ServletException{
        ob = new LoginDAO();
    }

    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
    throws IOException, ServletException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        String fName = ob.login(req);
        if(fName==null) {
            pw.println("Invalid Login process...<br>");
            RequestDispatcher rd = req.getRequestDispatcher("login.html");
            rd.include(req, res);
        }else {
    }
}

```

```

        req.setAttribute("fname", fName); //Adding attribute to request object
        chain.doFilter(req, res); //linking Servlet_program with same url_pattern
    }
}

public void destroy(){
    //NoCode
};

}

WelcomeServlet.java

package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

@SuppressWarnings("serial")
@WebServlet("/log")

public class WelcomeServlet extends HttpServlet{
    protected void doPost(HttpServletRequest req, HttpServletResponse res) throws
    ServletException, IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        String fName = (String)req.getAttribute("fname");
        pw.println("Welcome User "+fName);
    }
}

```

}

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>login.html</welcome-file>
    </welcome-file-list>
</web-app>
```

Assignment:

Update above application to display the complete details of Login-user.

Note:

=>*Servlet program must not interact with DAO*

Note:

=>*The FilterProgram and ServletProgram will be executed with same*

url-pattern.

=>*FilterProgram will have highest priority in execution than Servlet*

program.

Ex-Application : Demonstrating FilterConfig

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
```

```
<form action="dis" method="post">
UserName:<input type="text" name="uname"><br>
<input type="submit" value="Display">
</form>
</body>
</html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <filter>
        <filter-name>DisplayFilter</filter-name>
        <filter-class>test.DisplayFilter</filter-class>
        <init-param>
            <param-name>a</param-name>
            <param-value>1000</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>DisplayFilter</filter-name>
        <url-pattern>/dis</url-pattern>
    </filter-mapping>
    <welcome-file-list>
        <welcome-file>input.html</welcome-file>
    </welcome-file-list>
</web-app>
```

DisplayFilter.java

```
package test;

import java.io.*;
import javax.servlet.*;

public class DisplayFilter implements Filter{
    public FilterConfig fc = null;

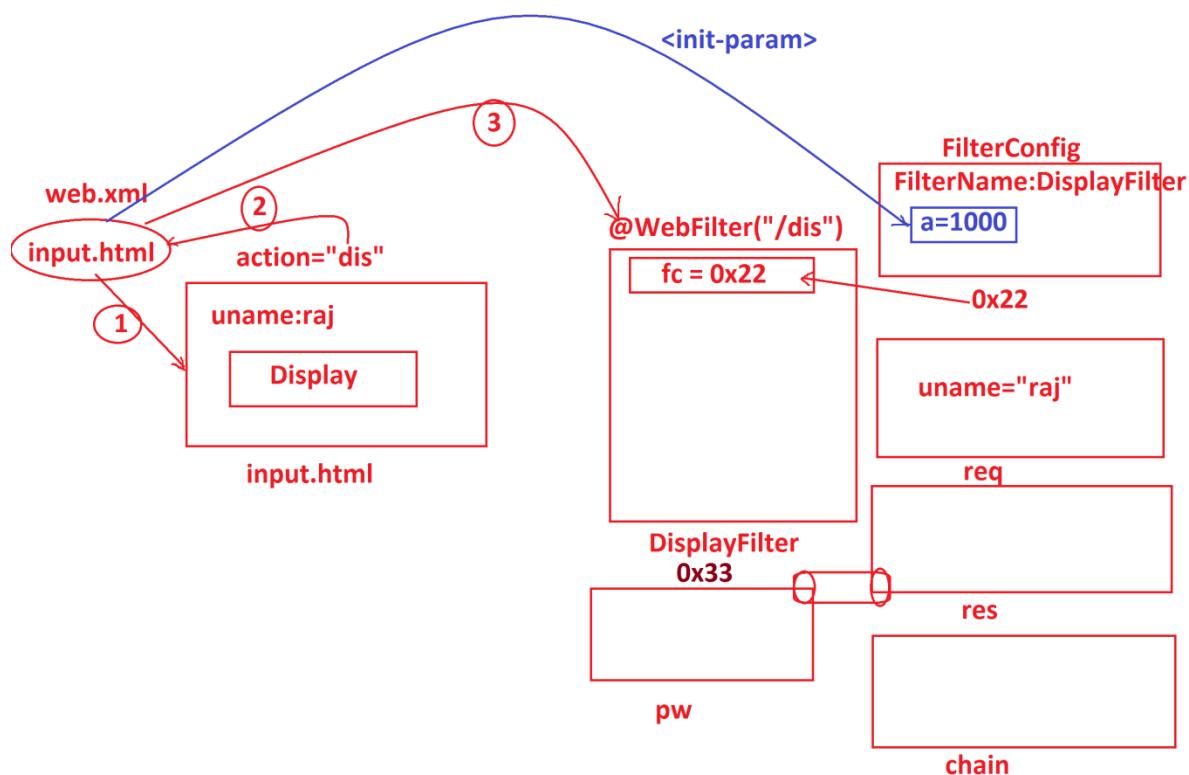
    public void init(FilterConfig fc) throws ServletException{
        this.fc=fc;
    }

    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
```

```
throws IOException,ServletException{  
  
PrintWriter pw = res.getWriter();  
  
res.setContentType("text/html");  
  
String uName = req.getParameter("uname");  
  
pw.println("Welcome User "+uName+"<br>");  
  
pw.println("*****FilterConfig*****");  
  
pw.println("<br>FilterName:"+fc.getFilterName());  
  
pw.println("<br>ConfigValue:"+fc.getInitParameter("a"));  
  
}  
  
public void destroy() {  
    //NoCode  
}  
}  
=====
```

Dt : 8/11/2022

Layout:



*imp

Listeners in Servlet Programming:

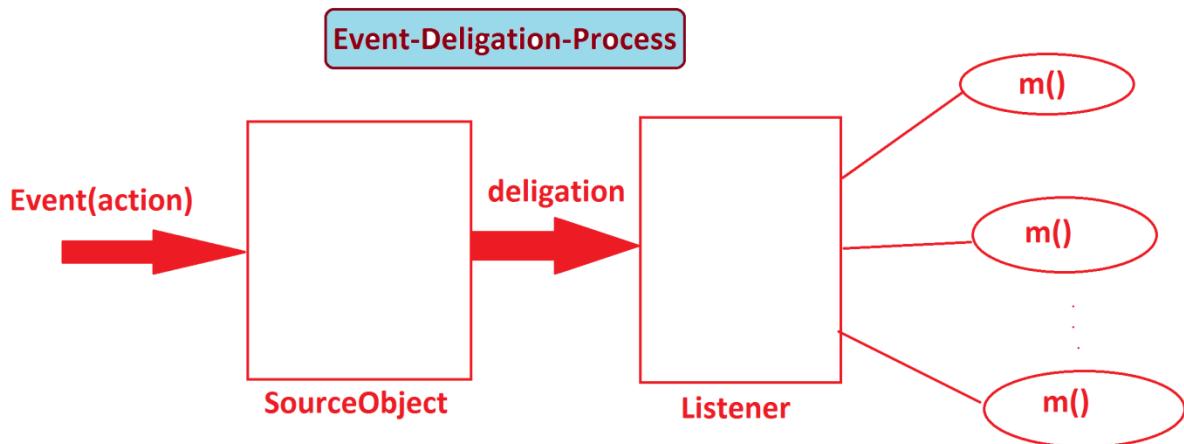
=>The component which is executed background to Servlet Objects is known as Listener.

Note:

=>When event is performed on Source-object, it is delegated to listener and the listener will execute one method to handle the event.

=>This process is known as "Event-Deligation-Process"

Diagram:



=>The following are the Source objects of Servlet programming where we can add Listeners:

- (a)ServletContext
- (b)ServletRequest
- (b)HttpSession

(a)ServletContext:

=>we use "javax.servlet.ServletContextListener" to handle the events performed on ServletContext object.

=>The following are the methods from "ServletContextListener":

```
public void contextInitialized(javax.servlet.ServletContextEvent);
```

```
public void contextDestroyed(javax.servlet.ServletContextEvent);
```

=>"ServletContextEvent" class will accept the events performed on ServletContext

object.

=>*The following are some important methods of ServletContextEvent:*

```
public javax.servlet.ServletContext getServletContext();
```

Ex : ContextListener.java

```
package test;

import javax.servlet.*;
import javax.servlet.annotation.*;

@WebListener

public class ContextListener implements ServletContextListener{

    public void contextInitialized(ServletContextEvent sce) {
        System.out.println("ServletContext initialized....");
        ServletContext sct = sce.getServletContext();
        System.out.println("Application deployed to "+sct.getServerInfo());
    }

    public void contextDestroyed(ServletContextEvent sce){
        System.out.println("ServletContext destroyed.... ");
        ServletContext sct = sce.getServletContext();
        System.out.println("Application destroyed from "+sct.getServerInfo());
    }
}
```

(b)ServletRequest:

=>we use "javax.servlet.ServletRequestListener" to handle the events performed on ServletRequest object.

=>The following are some important methods of ServletRequestListener:

public void requestInitialized(javax.servlet.ServletRequestEvent);

public void requestDestroyed(javax.servlet.ServletRequestEvent);

=>ServletRequestEvent class will hold the events performed on ServletRequest Object.

=>The following are some important methods of ServletRequestEvent:

public javax.servlet.ServletRequest getServletRequest();

public javax.servlet.ServletContext getServletContext();

Ex : RequestListener.java

```
package test;  
import javax.servlet.*;  
import javax.servlet.annotation.*;  
@WebListener  
public class RequestListener implements ServletRequestListener{  
    public void requestInitialized(ServletRequestEvent sre) {  
        System.out.println("ServletRequest Initialized....");  
    }  
    public void requestDestroyed(ServletRequestEvent sre) {  
    }  
}
```

```
        System.out.println("ServletRequest destroyed... ");
    }
}
```

(b) HttpSession:

=>we use "javax.servlet.http.HttpSessionListener" to handle the events performed on HttpSession object.

=>The following are some important methods of HttpSessionListener:

```
public void sessionCreated(javax.servlet.http.HttpSessionEvent);
```

```
public void sessionDestroyed(javax.servlet.http.HttpSessionEvent);
```

=>HttpSessionEvent class will hold the events performed of HttpSession object.

=>The following are some important methods of HttpSessionEvent:

```
public javax.servlet.http.HttpSession getSession();
```

Ex : SessionListener.java

```
package test;

import javax.servlet.http.*;
import javax.servlet.annotation.*;

@WebListener
public class SessionListener implements HttpSessionListener{

    public void sessionCreated(HttpSessionEvent hse) {
        System.out.println("Session Created....");
    }

    public void sessionDestroyed(HttpSessionEvent hse) {
```

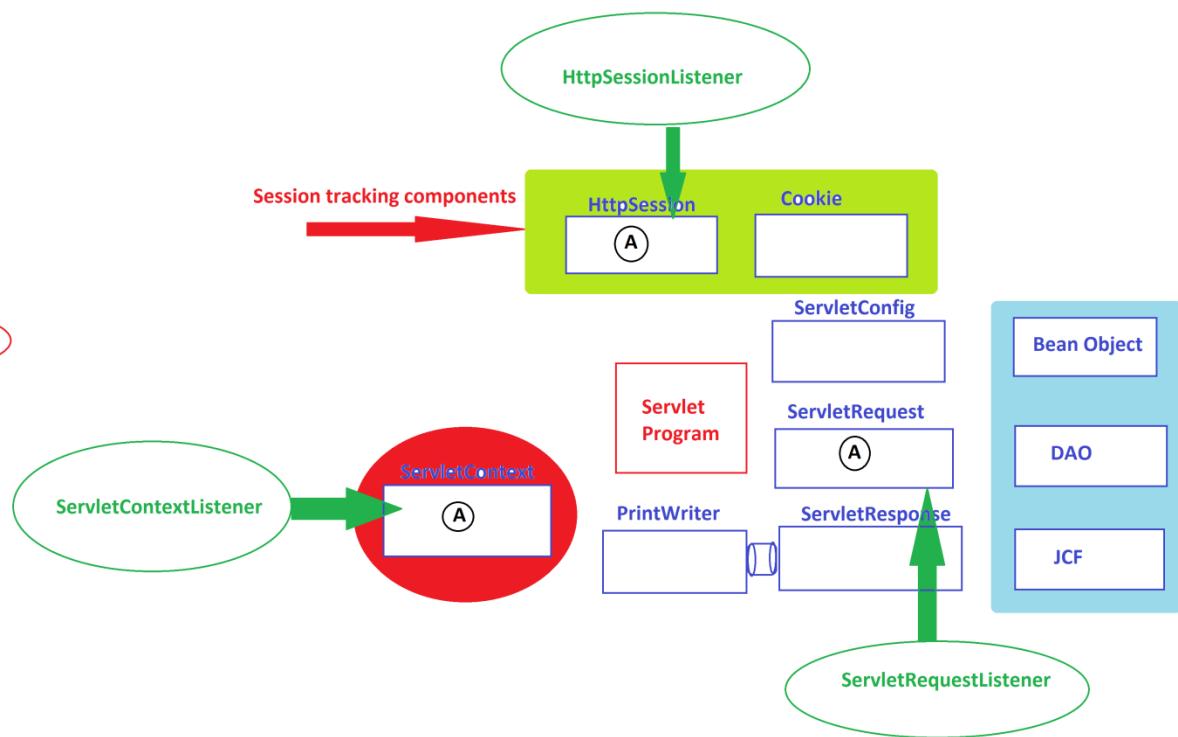
```

        System.out.println("Session Destroyed... ");
    }

}

=====

```



Venkatesh

Dt : 9/11/2022

faq:

wt is the diff b/w

(i)Application Listener

(ii)Request Listener

(iii)Session Listener

(i)Application Listener:

=>*The Listener which is added to the ServletContext object is known as*

Application Listener.

(ii)Request Listener:

=>*The Listener which is added to the ServletRequest is known as Request*

Listener.

(iii)Session Listener :

=>*The Listener which is added to the HttpSession is known as Session Listener.*

=====

*imp

Annotations in Servlet Programming:

define Annotation?

=>*The tag based information which is added to the programming components like*

Interface, class, methods and variable is known as Annotation.

=>we use "@" symbol to represent Annotation.

=>Annotations will give information to compiler at compilation stage or will give information to execution controls while execution process.

Ex:

`@SuppressWarnings`

`@Override`

`@SuppressWarnings:`

=>"`@SuppressWarnings`" annotation is used to give information to compiler to close the raised warnings.

`@Override:`

=>"`@Override`" annotation is used to give information to compiler to check the method is Overriding method or not.

=>The following are some important annotations used in Servlet programming:

(i)`@WebServlet`

(ii)`@WebFilter`

(iii)`@Listener`

(i)`@WebServlet:`

=>`@WebServlet` annotation will hold servlet-url-pattern and used to identify the Servlet program for loading onto WebContainer.

(ii)@WebFilter:

=>@WebFilter annotation will hold filter-url-pattern and used to identify the filter program for loading onto WebContainer.

(iii)@Listener:

=>@Listener annotation is used to identify listener for execution.

faq:

define Servlet Collaboration?

=>The process of establishing communication b/w Servlet programs is known as **Servlet Collaboration**.

=>we use the following to perform Servlet Collaboration:

- 1.RequestDispatcher**
- 2.sendRedirect()**

1.RequestDispatcher:

=>RequestDispatcher is used to establish communication b/w servlet-programs of WebApplication.

***imp**

2.sendRedirect():

=>sendRedirect() method is from javax.servlet.http.HttpServletResponse and which is used to establish communication b/w two Servlet programs available from

different WebApplications.

Ex:

WebApplication : TestApp1

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="first" method="post">
UserName:<input type="text" name="uname"><br>
MailId:<input type="text" name="mid"><br>
<input type="submit" value="Display">
</form>
</body>
</html>
```

FirstServlet.java

```
package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/first")

public class FirstServlet extends HttpServlet{
protected void doPost(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException{
String uName = req.getParameter("uname");
```

```
String mId = req.getParameter("mid");
res.sendRedirect("http://localhost:8082/TestApp2/second?name="+uName
+"&id="+mId);
}
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>input.html</welcome-file>
    </welcome-file-list>
</web-app>
```

WebApplication : TestApp2

SecondServlet.java

```
package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.*;
import javax.servlet.http.*;

@SuppressWarnings("serial")
@WebServlet("/second")

public class SecondServlet extends HttpServlet{
    protected void doGet(HttpServletRequest req,HttpServletResponse res) throws
    ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
    }
}
```

```
String uName = req.getParameter("name");
String mId = req.getParameter("id");
pw.println("====Display from SecondServlet=====");
pw.println("<br>UserName:"+uName);
pw.println("<br>MailId:"+mId);
}
}
```

<http://localhost:8082/TestApp2/second?name=raj&id=raj@gmail.com>

Dt : 10/11/2022

*imp

JSP Programming:(Unit-3)

=>JSP Stands for 'Java Server Page' and which is response from
WebApplication.

=>JSP is tag based programming language and which is more easy when
compared to Servlet programming.

=>Programs in JSP are saved with (.jsp) as an extention.

=>JSP programs are combination of both HTML code and Java Code.

=>JSP provides the following tags to write JavaCode part of JSP
programs:

1.Scripting tags

=>Scriptlet tag

=>Expression tag

=>Declarative tag

2.Directive tags

=>page

=>include

=>taglib

3.Action tags

=>jsp:include

=>jsp:forward

=>jsp:param

=>jsp:useBean

=>*jsp:setProperty*

=>*jsp:getProperty*

1. Scripting tags:

=>*Scripting tags are used to write JavaCode part of JSP programs.*

=>*Scripting tags are categorized into the following:*

(a) Scriptlet tag

(b) Expression tag

(c) Declarative tag

(a) Scriptlet tag:

=> Scriptlet tag is used to write normal JavaCode part of JSP programs

syntax:

<% ---JavaCode--- %>

(b) Expression tag:

=> Expression tag is used to assign the value to variable or which is used to display the data to the WebBrowser.

syntax:

<%= expression %>

(c) Declarative tag:

=>*Declarative tag is used to declare variables and methods in JSP programs.*

syntax:

`<%! variables;methods %>`

2. Directive tags:

=>*The tags which are used to specify the directions in translation process are known as Directive Tags.*

The following are the types of Directive tags:

(a) page

(b) include

(c) taglib

(a) page:

=>*'page' directive tag specifies the translator to add the related attribute to current JSP page.*

syntax:

```
<%@ page attribute="value" %>
```

exp:

```
<%@ page import="java.util.*"%>
```

List of attributes:

1. *import*
 2. *contentType*
 3. *extends*
 4. *info*
 5. *buffer*
 6. *language*
 7. *isELIgnored*
 8. *isThreadSafe*
 9. *autoFlush*
 10. *session*
 11. *pageEncoding*
 12. *errorPage*
 13. *isErrorPage*
-

(b) *include:*

=> 'include' directive tag specifies the file to be included to
current JSP page.

syntax:

`<%@ include file="file-name"%>`

Exp:

`<%@ include file="input.html" %>`

(c)taglib:

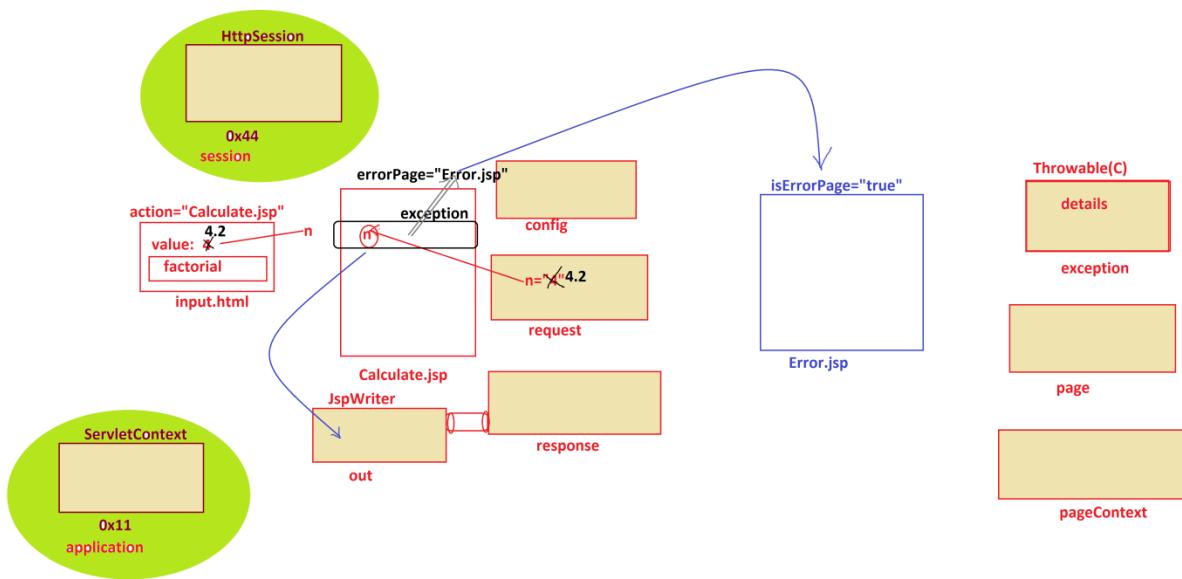
=>'taglib' directive tag specifies to add specified url to current JSP page and which is used part of EL(Expression Lang) and JSTL (JSP Standard Tag Lib).

syntax:

`<%@ taglib url="urloftaglib" prefix="prefixoftaglib"%>`

Ex : Application

Layout:



input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="Calculate.jsp" method="post">
Enter the value:<input type="text" name="n"><br>
<input type="submit" value="Factorial">
</form>
</body>
</html>
```

Calculate.jsp

```
<%@ page language="java"
contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"
errorPage="Error.jsp"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
```

```

<%!
int fact;
int factorial(int n)
{
    fact=1;
    for(int i=n;i>=1;i--)
    {
        fact=fact*i;
    }
    return fact;
}
%>
<%
int n = Integer.parseInt(request.getParameter("n"));
int r = factorial(n);
out.println("Factorail="+r+"<br>");
%>
<%@ include file="input.html" %>
</body>
</html>

```

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>input.html</welcome-file>
    </welcome-file-list>
</web-app>

```

Error.jsp

```

<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    isErrorPage="true"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
out.println("Enter only Integer values....<br>"); 
%>
<%= exception %>

```

```
<br>
<%@ include file="input.html"%>
</body>
</html>
```

=>The following are the implicit objects of JSP:

application - javax.servlet.ServletContext
config - javax.servlet.ServletConfig
request - javax.servlet.http.HttpServletRequest
response - javax.servlet.http.HttpServletResponse
out - javax.servlet.jsp.JspWriter
session - javax.servlet.http.HttpSession
exception - java.lang.Throwable
page - java.lang.Object
pageContext - javax.servlet.jsp.PageContext

3.Action tags:

=>Action Tags are used to include some basic actions like inserting some other page resources ,forwarding the request to another page, creating or locating the JavaBean instances and,setting and retrieving the bean properties in JSP pages.

Note:

=>These action tags are used in Execution process at runtime.

=>The following are some Important action tags available in JSP:

1.<jsp:include>

2.<jsp:forward>

3.<jsp:param>

4.<jsp:useBean>

5.<jsp:setProperty>

6.<jsp:getProperty>

=====

1.<jsp:include> :

=>*This action tag allows to include a static or dynamic resource such as HTML or JSP specified by an URL to be included in the current JSP while processing request.*

=>*If the resource is static then its content is included in the JSP page.*

=>*If the resource is dynamic then its result is included in the JSP page.*

syntax:

```
<jsp:include attributes>
<---Zero or more jsp:param tags--->
</jsp:include>
```

attributes of include tag:

**imp*

page : Takes a relative URL, which locates the resource
to be included in the JSP page.

```
<jsp:include page="/Header.html"/>  
<jsp:include page="<%="mypath%>"/>
```

flush : Takes true or false, which indicates whether or not the buffer
needs to be flushed before including resource.

2. **<jsp:forward>**:

=> This action tag forwards a JSP request to another resource and
which can be either static or dynamic.

=> If the resource is dynamic then we can use a **jsp:param** tag to
pass name and value of the parameter to the resource.

syntax:

```
<jsp:forward attributes>  
  <-- Zero or more jsp:param tags-->  
</jsp:forward>
```

Exp:

```
<jsp:forward page="/Header.html"/>  
<jsp:forward page="<%="mypath%>"/>
```

3.<jsp:param>:

This action tag is used to hold the parameter with value and which is to be forwarded to the next resource.

syntax:

```
<jsp:param name="paramName" value="paramValue"/>
```

```
=====
```

4.<jsp:useBean>:

=>This tag is used to instantiate a JavaBean, or locate an existing bean instance and assign it to a variable name(id).

syntax:

```
<jsp:useBean attributes>
  <!-optional body content--->
</jsp:useBean>
```

Attributes of <jsp:useBean> tag:

- (a)id
- (b)scope
- (c)class
- (d)beanName
- (e)type

(a)id:

=>which represents the variable name assigned to id attribute of
<jsp:useBean> tag and which holds the reference of JavaBean instance.

(b)scope:

=>which specifies the scope in which the bean instance has to be
created or located.

scope can be the following:

- (i)page scope : within the JSP page,until the page sends response.
- (ii)request scope : JSP page processing the same request until a JSP sends response.
- (iii)session scope - Used with in the Session.
- (iv)application scope - Used within entire web application.

*imp

(c)class :

=>The class attribute takes the qualified class name to create a
bean instance.

(d)beanName :

=>The beanName attribute takes a qualified class name.

(e)type :

=>The "type" attribute takes a qualified className or

interfaceName, which can be the classname given in the class or beanName attribute or its super type.

5.<jsp:setProperty>:

=>This action tag sets the value of a property in a bean, using the bean's setter methods.

Types of attributes:

(a)name

(b)property

(c)value

(d)param

(a)name:

=>The name attribute takes the name of already existing bean as a reference variable to invoke the setter method.

(b)property:

=>which specifies the property name that has to be set, and specifies the setter method that has to be invoked.

(c)value:

=>The value attribute takes the value that has to be set to the specified bean property.

(d)param:

=>which specify the name of the request parameter whose value to be assigned to bean property.

6.<jsp:getProperty>:

=>This action tag gets the value of a property in a bean by using the bean's getter method and writes the value to the current JspWriter.

Types of attributes:

(a)name

(b)property

(a)name:

=>The name attribute takes the reference variable name on which we want to invoke the getter method.

(b)property:

=>which gets the value of a bean property and invokes the getter method of the bean property.

The following are some rare used Actions tags:

7.<jsp:plugin>:

The <jsp:plugin> action tag provide easy support for including a java applet in the client Web browser, using a built-in or downloaded java plug-in.

Syntax:

```
<jsp:plugin attributes>
<!-optionally one jsp:params or jsp:fallback tag-
</jsp:plugin>
```

8. <jsp:fallBack >

The <jsp:fallback> action tag allows us to specify a text message to be displayed if the required plug-in cannot run and this action tag must be used as a child tag with the <jsp:plugin> action tag.

Syntax:

```
<jsp:fallback>
Test message that has to be displayed if the plugin cannot be started
</jsp:fallback>
```

9. <jsp:params >

The <jsp:params> action tag sends the parameters that we want to pass to an applet.

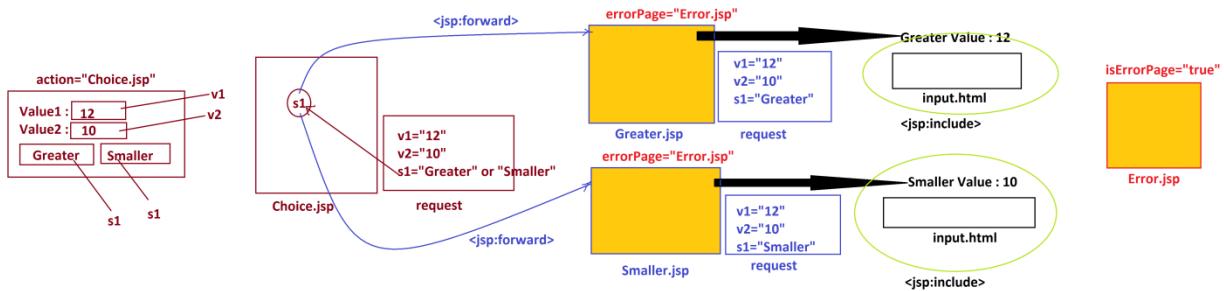
Syntax:

```
<jsp:params>  
  <!-one or more jsp:param tags---  
</jsp:params>
```

Dt : 11/11/2022

Ex-Application:(Demonstrating <jsp:include> <jsp:forward> <jsp:param>)

Layout:



input.html

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    isErrorPage="Error.jsp"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%!
int SmallerValue(int x,int y)
{
    if(x<y) return x;
    else return y;
}
%>
<%
int v1 = Integer.parseInt(request.getParameter("v1"));
int v2 = Integer.parseInt(request.getParameter("v2"));
int v3 = SmallerValue(v1,v2);
String p = request.getParameter("p");
out.println(p+"<br>");
out.println("SmallerValue:"+v3+"<br>");
%>
```

```
<jsp:include page="input.html"/>
</body>
</html>
```

Choice.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
    String s1 = request.getParameter("s1");
    String msg="Arithmetic";
    if(s1.equals("Greater")){
        %
        <jsp:forward page="Greater.jsp">
            <jsp:param value="<%msg%>" name="p"/>
        </jsp:forward>
        %
    }else{
        %
        <jsp:forward page="Smaller.jsp">
            <jsp:param value="<%msg%>" name="p"/>
        </jsp:forward>
        %
    }
%
</body>
</html>
```

Greater.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    errorPage="Error.jsp"%>
<!DOCTYPE html>
<html>
```

```

<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%!
int GreaterValue(int x,int y)
{
    if(x>y) return x;
    else return y;
}
%>
<%
int v1 = Integer.parseInt(request.getParameter("v1"));
int v2 = Integer.parseInt(request.getParameter("v2"));
String p = request.getParameter("p");
int v3 = GreaterValue(v1,v2);
out.println(p+"<br>");
out.println("GreaterValue:"+v3+"<br>");
%>
<jsp:include page="input.html" flush="true"/>
</body>
</html>

```

Smaller.jsp

```

<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    errorPage="Error.jsp"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%!
int SmallerValue(int x,int y)
{
    if(x<y) return x;
    else return y;
}
%>

```

```

}
%>
<%
int v1 = Integer.parseInt(request.getParameter("v1"));
int v2 = Integer.parseInt(request.getParameter("v2"));
int v3 = SmallerValue(v1,v2);
String p = request.getParameter("p");
out.println(p+"<br>");
out.println("SmallerValue:"+v3+"<br>");
%>
<jsp:include page="input.html"/>
</body>
</html>

```

Error.jsp

```

<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    isErrorPage="true"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
out.println("Enter Only Integer Values...<br>"); 
%>
<%= exception %>
<br>
<jsp:include page="input.html"/>
</body>
</html>

```

web.xml

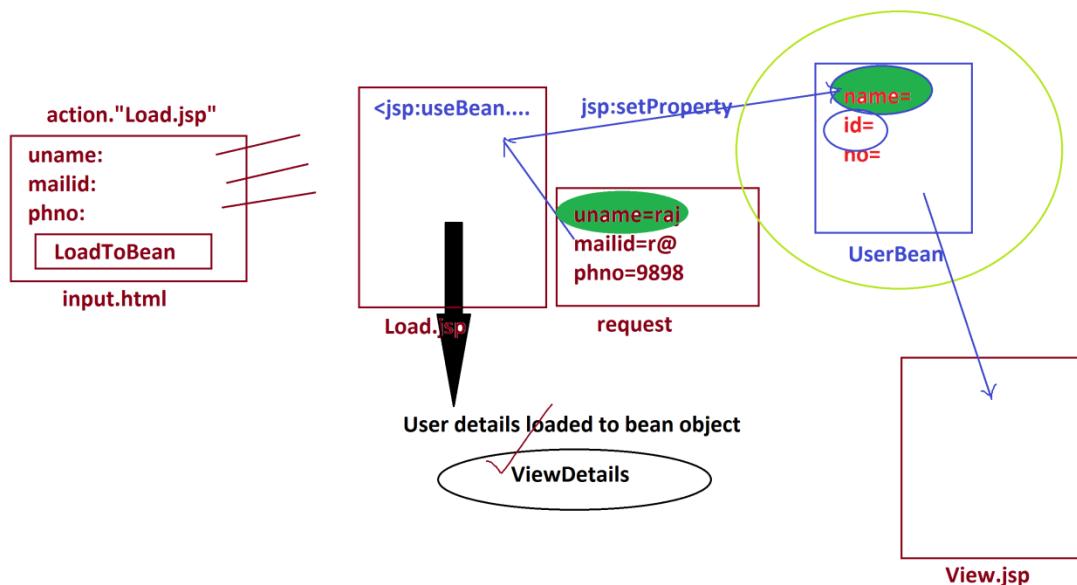
```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>input.html</welcome-file>
    </welcome-file-list>
</web-app>

```

Ex-Application : (Demonstrating <jsp:useBean> <jsp:setProperty> <jsp:getProperty>)

Layout:



input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="Load.jsp" method="post">
UserName:<input type="text" name="uname"><br>
MailId:<input type="text" name="mid"><br>
PhoneNo:<input type="text" name="phno"><br>
<input type="submit" value="LoadToBean">
</form>
</body>
</html>
```

Load.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.UserBean"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<jsp:useBean id="ob" class="test.UserBean" scope="session"/>
<jsp:setProperty property="name" param="uname" name="ob"/>
<jsp:setProperty property="id" param="mid" name="ob"/>
<jsp:setProperty property="no" param="phno" name="ob"/>
<%
out.println("User details loaded to bean object...<br>");
%>
<a href="View.jsp">ViewUserDetails</a>
</body>
</html>
```

View.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.UserBean"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<jsp:useBean id="ob" type="test.UserBean" scope="session"/>
UserName:<jsp:getProperty property="name" name="ob"/><br>
MailId:<jsp:getProperty property="id" name="ob"/><br>
PhoneNO:<jsp:getProperty property="no" name="ob"/><br>
</body>
</html>
```

UserBean.jsp

```
package test;
```

```
import java.io.*;
@SuppressWarnings("serial")
public class UserBean implements Serializable{
    private String name,id;
    private long no;
    public UserBean() {}
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public long getNo() {
        return no;
    }
    public void setNo(long no) {
        this.no = no;
    }
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>input.html</welcome-file>
    </welcome-file-list>
</web-app>
```

Dt : 12/11/2022

*imp

Web Architecture Models:(Web Application Architectures)

=>*Two types of development models are used in Java,for Web applications, and these models are classified based on the different approaches used to develop Web applications.*

These models are:

- 1. Model-1 Architecture**
- 2. Model-2 Architecture(MVC)**

1. Model-1 Architecture:

The Model-1 architecture was the first development model used to develop Web applications and this model uses JSP to design applications and, which is responsible for all the activities and functionalities provided by the application.

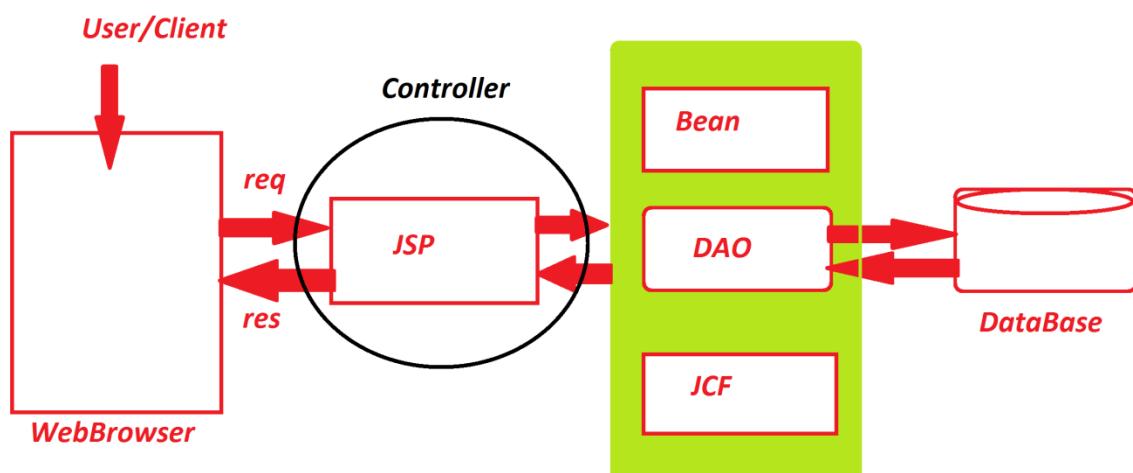
Limitations of the Model-1 Architecture:

(i). Applications are inflexible and difficult to maintain.

A single change in one page may cause changes in other pages, leading to unpredictable results.

(ii).Involves the developer at both the page development and the business logic implementation stages.

(iii).Increases the complexity of a program with the increase in the size of the JSP page.



2. Model-2 Architecture:

=>The draw backs in the Model-1 architecture led to the introduction of a new model called Model-2.

=>The Model-2 architecture was targeted at overcoming the drawbacks of Model-1 and helping developers to design more powerful Web applications and this Model-2 architecture is based on the MVC design model.

=>MVC Stands for Model View Controller.

Model: Represents enterprise data and business rules that specify how data is accessed and updated, and which is generally implemented by using JavaBeans.

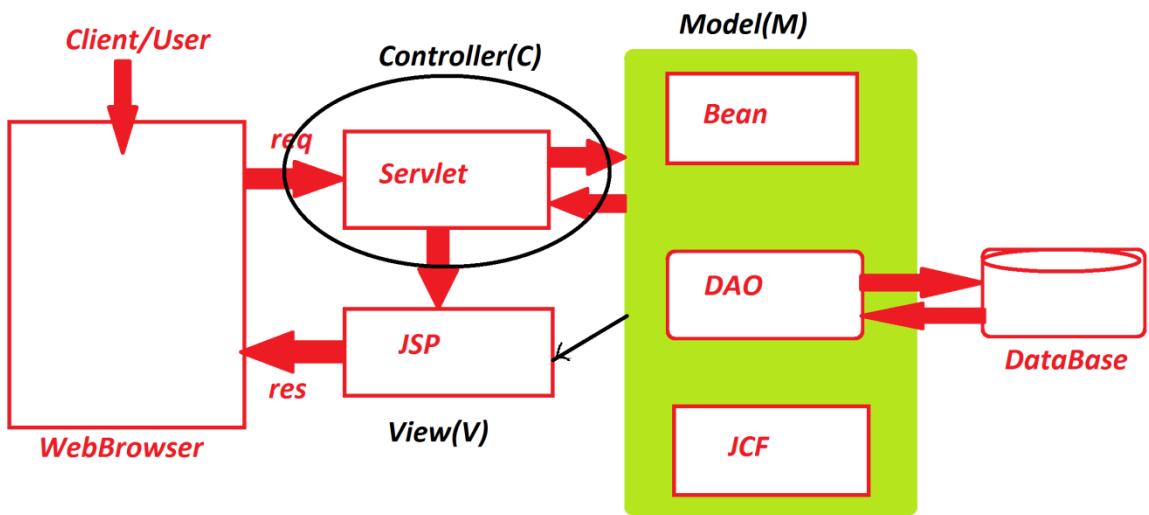
View: Shows the contents of a Model. The View component accesses enterprise data through the Model component and specifies how that data should be presented and this View Component is designed by JSP.

Controller: Receives HTTP requests. The Controller component receives requests from a client, determines the business logic to be performed, and delegates the responsibility for producing the next phase of the user interface to an appropriate view component. The Controller has complete control over each view, implying that any change in the Model component is immediately reflected in all the Views of an application. The Controller component is implemented by servlets.

Advantages of Model-2 Architecture:

(i) Allows use of reusable software components to design the Business logic. Therefore, these components can be used in the business logic of other applications.

(ii) Offers great flexibility to the presentation logic, which can be modified without effecting the business logic.



Dt : 14/11/2022

ProjectName : OnlineProductStore

DB-Tables : Product48, Admin48, User48

home.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="choice" method="post">
<input type="submit" name="s1" value="UserLogin">
<input type="submit" name="s1" value="AdminLogin">
</form>
</body>
</html>
```

UserLogin.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="user" method="post">
UserName:<input type="text" name="uname"><br>
PassWord:<input type="text" name="pword"><br>
<input type="submit" value="Login">
<a href="register.html">NewUser</a>
</form>
</body>
</html>
```

AdminLogin.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="admin" method="post">
UserName:<input type="text" name="uname"><br>
PassWord:<input type="text" name="pword"><br>
<input type="submit" value="Login">
</form>
</body>
</html>
```

```
</form>
</body>
</html>
```

ChoiceServlet.java

```
package test;

import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/choice")

public class ChoiceServlet extends HttpServlet{

    protected void doPost(HttpServletRequest req,HttpServletResponse res) throws
    ServletException,IOException{

        String s1 = req.getParameter("s1");

        this.getServletContext().setAttribute("s1", s1);

        if(s1.equals("UserLogin")) {

            req.getRequestDispatcher("UserLogin.html").forward(req, res);

        }else {

            req.getRequestDispatcher("AdminLogin.html").forward(req, res);

        }

    }

}
```

UserDetails.java

```

package test;
import java.io.*;
@SuppressWarnings("serial")
public class UserDetails implements Serializable{
    private String uName,pWord,fName,lName,addr,mId;
    private long phNo;
    public UserDetails() {}
    public String getuName() {
        return uName;
    }
    public void setuName(String uName) {
        this.uName = uName;
    }
    public String getpWord() {
        return pWord;
    }
    public void setpWord(String pWord) {
        this.pWord = pWord;
    }
    public String getfName() {
        return fName;
    }
    public void setfName(String fName) {
        this.fName = fName;
    }
    public String getlName() {
        return lName;
    }
    public void setlName(String lName) {
        this.lName = lName;
    }
    public String getAddr() {
        return addr;
    }
    public void setAddr(String addr) {
        this.addr = addr;
    }
    public String getmId() {
        return mId;
    }
    public void setmId(String mId) {
        this.mId = mId;
    }
    public long getPhNo() {
        return phNo;
    }
    public void setPhNo(long phNo) {
}

```

```

        this.phNo = phNo;
    }

}


```

DBConnection.java

```

package test;
import java.sql.*;
public class DBConnection {
    private static Connection con=null;
    private DBConnection() {}
    static
    {
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
        }catch(Exception e) {e.printStackTrace();}
    }
    public static Connection getCon() {
        return con;
    }
}

```

LoginDAO.java

```

package test;

import java.sql.*;
import javax.servlet.http.*;

public class LoginDAO {

    public UserDetails ud = null;

    public UserDetails login(HttpServletRequest req) {
        try {
            Connection con = DBConnection.getCon();
            String s1 = (String)req.getServletContext().getAttribute("s1");

```

```

if(s1.equals("UserLogin")) {

    PreparedStatement ps = con.prepareStatement
        ("select * from User48 where uname=? and pword=?");

    ps.setString(1, req.getParameter("uname"));

    ps.setString(2, req.getParameter("pword"));

    ResultSet rs = ps.executeQuery();

    if(rs.next()) {

        ud = new UserDetails();

        ud.setuName(rs.getString(1));

        ud.setpWord(rs.getString(2));

        ud.setfName(rs.getString(3));

        ud.setlName(rs.getString(4));

        ud.setAddr(rs.getString(5));

        ud.setmId(rs.getString(6));

        ud.setPhNo(rs.getLong(7));

    }

} else {

    PreparedStatement ps = con.prepareStatement
        ("select * from Admin48 where uname=? and
pword=?");

    ps.setString(1, req.getParameter("uname"));

    ps.setString(2, req.getParameter("pword"));

    ResultSet rs = ps.executeQuery();

    if(rs.next()) {
}

```

```

        ud = new UserDetails();
        ud.setuName(rs.getString(1));
        ud.setpWord(rs.getString(2));
        ud.setfName(rs.getString(3));
        ud.setlName(rs.getString(4));
        ud.setAddr(rs.getString(5));
        ud.setmId(rs.getString(6));
        ud.setPhNo(rs.getLong(7));
    }

}

} catch(Exception e) {e.printStackTrace();}

return ud;
}

}

UserLoginServlet.java

package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/user")

public class UserLoginServlet extends HttpServlet{
protected void doPost(HttpServletRequest req,HttpServletResponse res) throws

```

```

ServletException, IOException {
    UserDetails ud = new LoginDAO().login(req);
    if(ud==null) {
        req.setAttribute("msg", "Invalid Login process...<br>");
        req.getRequestDispatcher("LoginFail.jsp").forward(req, res);
    } else {
        HttpSession hs = req.getSession();
        hs.setAttribute("ud", ud);
        req.getRequestDispatcher("LoginSuccess.jsp").forward(req, res);
    }
}
}

```

LoginFail.jsp

```

<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String msg = (String) request.getParameter("msg");
out.println(msg);
%>
<jsp:include page="home.html"/>
</body>
</html>

```

LoginSuccess.jsp

```

<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"

```

```

pageEncoding="ISO-8859-1"
import="test.UserDetails"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
UserDetails ud = (UserDetails) session.getAttribute("ud");
String s1 = (String) application.getAttribute("s1");
if(s1.equals("UserLogin")){
    out.println("Welcome User "+ud.getfName()+"<br>");
    %>
    <a href="view">ViewProducts</a>
    <a href="logout">Logout</a>
    <%
} else{
    out.println("Welcome Admin "+ud.getfName()+"<br>");
    %>
    <a href="view">ViewProducts</a>
    <a href="product.html">AddProduct</a>
    <a href="">Edit-UpdateProduct</a>
    <a href="">DeleteProduct</a>
    <a href="logout">Logout</a>
    <%
}
%>

</body>
</html>

```

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>home.html</welcome-file>
    </welcome-file-list>
</web-app>
=====
```

Dt : 15/11/2022

LogoutServlet.java

```

package test;

import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/logout")

public class LogoutServlet extends HttpServlet{

protected void doGet(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException{

HttpSession hs = req.getSession(false);

if(hs==null) {

req.setAttribute("msg","Session Expired...perform Login process...<br>");

} else {

req.getServletContext().removeAttribute("s1");

hs.invalidate();

req.setAttribute("msg","Logged out Successfully...<br>");

}

req.getRequestDispatcher("Logout.jsp").forward(req, res);

}

}

```

Logout.jsp

```

<%@ page language="java"
contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>

```

```

<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String msg = (String)request.getAttribute("msg");
out.println(msg);
%>
<jsp:include page="home.html"/>
</body>
</html>

```

ProductBean.java

```

package test;
import java.io.*;
@SuppressWarnings("serial")
public class ProductBean implements Serializable {
    private String code, name;
    private float price;
    private int qty;
    public ProductBean() {}
    public String getCode() {
        return code;
    }
    public void setCode(String code) {
        this.code = code;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public float getPrice() {
        return price;
    }
    public void setPrice(float price) {
        this.price = price;
    }
    public int getQty() {
        return qty;
    }
    public void setQty(int qty) {
        this.qty = qty;
    }
}

```

```
 }  
 }
```

RetrieveProductsDAO.java

```
package test;  
  
import java.util.*;  
  
import java.sql.*;  
  
public class RetrieveProductsDAO {  
  
    public ArrayList<ProductBean> al = new ArrayList<ProductBean>();  
  
    public ArrayList<ProductBean> retrieve()  
    {  
        try {  
            Connection con = DBConnection.getCon();  
            PreparedStatement ps = con.prepareStatement("select * from Product48");  
            ResultSet rs = ps.executeQuery();  
            while(rs.next()) {  
                ProductBean pb = new ProductBean();  
                pb.setCode(rs.getString(1));  
                pb.setName(rs.getString(2));  
                pb.setPrice(rs.getFloat(3));  
                pb.setQty(rs.getInt(4));  
                al.add(pb); //Adding to ArrayList object  
            } //end of loop  
        } catch(Exception e) {e.printStackTrace();}  
        return al;
```

```

    }

}

ViewProductsServlet.java

package test;

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/view")

public class ViewProductsServlet extends HttpServlet{

protected void doGet(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException{

    HttpSession hs = req.getSession(false);

    if(hs==null) {

        req.setAttribute("msg","Session expired...Perform Login process...<br>");

        req.getRequestDispatcher("Logout.jsp").forward(req, res);

    }else {

        ArrayList<ProductBean> al = new RetrieveProductsDAO().retrieve();

        req.setAttribute("al", al);

        req.getRequestDispatcher("ViewProducts.jsp").forward(req, res);

    }

}

```

}

ViewProducts.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
import="java.util.* , test.ProductBean, test.UserDetails"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
ArrayList<ProductBean> al
=(ArrayList<ProductBean>)request.getAttribute("al");
UserDetails ud = (UserDetails)session.getAttribute("ud");
out.println("page of "+ud.getfName()+"<br>");
if(al.size()==0){
    out.println("Products Not available....");
}else{
    Iterator<ProductBean> it = al.iterator();
    while(it.hasNext()){
        ProductBean pb = (ProductBean)it.next();

        out.println(pb.getCode()+"  "+pb.getName()+" &ampnbsp"+
pb.getPrice()+" &ampnbsp"+pb.getQty()+"<br>");
    }
}
%>
<%
String s1 = (String)application.getAttribute("s1");
if(s1.equals("UserLogin")){
    %>
    <a href="logout">Logout</a>
    <%
}else{
    %>
    <a href="view">ViewProducts</a>
    <a href="product.html">AddProduct</a>
    <a href="">Edit-UpdateProduct</a>
    <a href="">DeleteProduct</a>
    <a href="logout">Logout</a>
<%
```

```
}

%>

</body>
</html>
```

AdminLoginServlet.java

```
package test;

import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/admin")

public class AdminLoginServlet extends HttpServlet{

protected void doPost(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException{

UserDetails ud = new LoginDAO().login(req);

if(ud==null) {

req.setAttribute("msg","Invalid Login process...<br>");

req.getRequestDispatcher("LoginFail.jsp").forward(req, res);

} else {

HttpSession hs = req.getSession();

hs.setAttribute("ud", ud);

req.getRequestDispatcher("LoginSuccess.jsp").forward(req, res);

}

}
```

```
}
```

```
product.html
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="add" method="post">
Product Code:<input type="text" name="code"><br>
Product Name:<input type="text" name="name"><br>
Product Price:<input type="text" name="price"><br>
Product Quantity:<input type="text" name="qty"><br>
<input type="submit" value="AddProduct">
</form>
</body>
</html>
```

```
AddProductServlet.java
```

```
package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

@SuppressWarnings("serial")
@WebServlet("/add")

public class AddProductServlet extends HttpServlet{

protected void doPost(HttpServletRequest req,HttpServletResponse res) throws
ServletException,
IOException{
HttpSession hs = req.getSession(false);
```

```

if(hs==null) {

    req.setAttribute("msg", "Session expired...Perform login process...<br>");

    req.getRequestDispatcher("Logout.jsp").forward(req, res);

} else {

    ProductBean pb = new ProductBean();

    pb.setCode(req.getParameter("code"));

    pb.setName(req.getParameter("name"));

    pb.setPrice(Float.parseFloat(req.getParameter("price")));

    pb.setQty(Integer.parseInt(req.getParameter("qty")));

    int k = new InsertProductDAO().insert(pb);

    if(k>0) {

        req.setAttribute("msg", "Product Added Successfully...<br>");

        req.getRequestDispatcher("AddProduct.jsp").forward(req, res);

    }

}

}

```

InsertProductDAO.java

```

package test;
import java.sql.*;
public class InsertProductDAO {
    public int k=0;
    public int insert(ProductBean pb) {
        try {
            Connection con = DBConnection.getCon(); //Accessing
the Connection

```

```

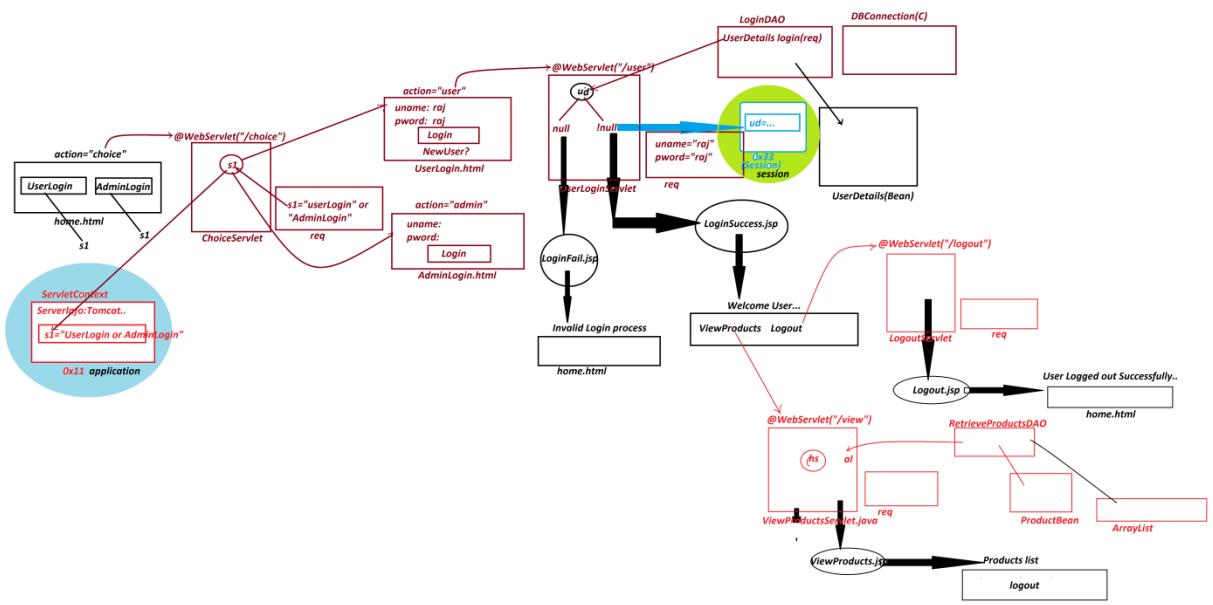
        PreparedStatement ps = con.prepareStatement
            ("insert into Product48 values(?, ?, ?, ?)");
        ps.setString(1, pb.getCode());
        ps.setString(2, pb.getName());
        ps.setFloat(3, pb.getPrice());
        ps.setInt(4, pb.getQty());
        k = ps.executeUpdate();
    } catch (Exception e) {e.printStackTrace();}
    return k;
}
}

```

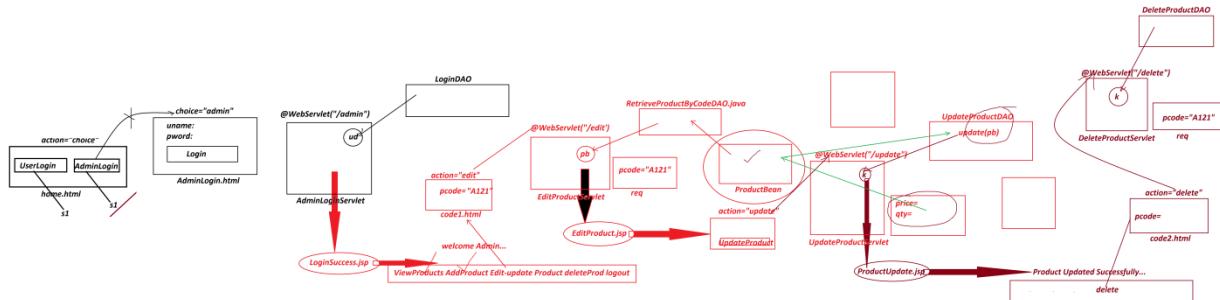
AddProduct.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-
8859-1"
    pageEncoding="ISO-8859-1"
    import="test.UserDetails"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String msg = (String) request.getAttribute("msg");
UserDetails ud = (UserDetails) session.getAttribute("ud");
out.println("page of "+ud.getfName()+"<br>");
out.println(msg);
%>
<a href="view">ViewProducts</a>
<a href="product.html">AddProduct</a>
<a href="">Edit-UpdateProduct</a>
<a href="">DeleteProduct</a>
<a href="logout">Logout</a>
</body>
</html>
=====
```



Dt : 16/11/2022



code1.html

```

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="edit" method="post">
  ProductCode:<input type="text" name="pcode"><br>
  <input type="submit" value="EditProduct">
</form>
</body>
</html>
  
```

RetrieveProductByCode.java

```

package test;

import java.sql.*;
import javax.servlet.http.*;

public class RetrieveProductByCode {

    public ProductBean pb = null;

    public ProductBean retrieve(HttpServletRequest req) {
        try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
  
```

```

("select * from Product48 where code=?");

ps.setString(1,req.getParameter("PCODE"));

ResultSet rs = ps.executeQuery();

if(rs.next()) {

    pb = new ProductBean();

    pb.setCode(rs.getString(1));

    pb.setName(rs.getString(2));

    pb.setPrice(rs.getFloat(3));

    pb.setQty(rs.getInt(4));

}

}catch(Exception e){e.printStackTrace();}

return pb;
}

}

EditProductServlet.java

package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/edit")

public class EditProductServlet extends HttpServlet{

protected void doPost(HttpServletRequest req,HttpServletResponse res) throws

```

```

ServletException, IOException {
    HttpSession hs = req.getSession(false);
    if(hs==null) {
        req.setAttribute("msg","Session Expired...Perform login process...<br>");
        req.getRequestDispatcher("Logout.jsp").forward(req, res);
    } else {
        ProductBean pb = new RetrieveProductByCode().retrieve(req);
        hs.setAttribute("pb",pb);
        req.getRequestDispatcher("EditProduct.jsp").forward(req, res);
    }
}
}

```

EditProduct.jsp

```

<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.ProductBean"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
ProductBean pb = (ProductBean) session.getAttribute("pb");
%>
<form action="update" method="post">
    ProductPrice:<input type="text" name="price" value="<%=
pb.getPrice() %>"><br>
    ProductQty:<input type="text" name="qty" value="<%=
pb.getQty() %>"><br>
    <input type="submit" value="UpdateProduct">
</form>

```

```
</body>
</html>
```

UpdateProductDAO.java

```
package test;
import java.sql.*;
public class UpdateProductDAO {
    public int k=0;
    public int update(ProductBean pb) {
        try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                ("update Product48 set price=? , qty=qty+?
where code=?");
            ps.setFloat(1, pb.getPrice());
            ps.setInt(2, pb.getQty());
            ps.setString(3, pb.getCode());
            k = ps.executeUpdate();
        } catch (Exception e) {e.printStackTrace();}
        return k;
    }
}
```

UpdateProductServlet.java

```
package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/update")

public class UpdateProductServlet extends HttpServlet{
    protected void doPost(HttpServletRequest req,HttpServletResponse res) throws
    ServletException,IOException{
        HttpSession hs = req.getSession(false);
```

```

if(hs==null) {

    req.setAttribute("msg","Session expired...<br>");

    req.getRequestDispatcher("Logout.jsp").forward(req, res);

}else {

    ProductBean pb = (ProductBean)hs.getAttribute("pb");

    pb.setPrice(Float.parseFloat(req.getParameter("price")));

    pb.setQty(Integer.parseInt(req.getParameter("qty")));

    int k = new UpdateProductDAO().update(pb);

    if(k>0) {

        req.setAttribute("msg","Product Updated Successfully...");

        hs.removeAttribute("pb");

        req.getRequestDispatcher("ProductUpdate.jsp").forward(req, res);

    }

}

}

```

ProductUpdate.jsp

```

<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.UserDetails"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
UserDetails ud = (UserDetails)session.getAttribute("ud");
out.println("Page of "+ud.getfName()+"<br>");
```

```

String msg =(String) request.getAttribute("msg");
out.println(msg);
%>
    <a href="view">ViewProducts</a>
    <a href="product.html">AddProduct</a>
    <a href="code1.html ">Edit-UpdateProduct</a>
    <a href="code2.html ">DeleteProduct</a>
    <a href="logout">Logout</a>
</body>
</html>
=====
```

Dt : 17/11/2022

code2.html

```

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <form action="delete" method="post">
        Enter the code:<input type="text" name="PCODE"><br>
        <input type="submit" value="DeleteProduct">
    </form>
</body>
</html>
```

DeleteProductDAO.java

```

package test;

import java.sql.*;
import javax.servlet.http.*;

public class DeleteProductDAO {

    public int k=0;

    public int delete(HttpServletRequest req) {
        try {
            Connection con = DBConnection.getCon();
```

```


PreparedStatement ps = con.prepareStatement
        ("delete from Product48 where code=?");

ps.setString(1,req.getParameter("PCODE"));

k = ps.executeUpdate();

}catch(Exception e) {e.printStackTrace();}

return k;
}
}


```

DeleteProductServlet.java

```


package test;

import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/delete")

public class DeleteProductServlet extends HttpServlet{

protected void doPost(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException{

HttpSession hs = req.getSession(false);

if(hs==null) {

req.setAttribute("msg","Session expired...<br>");

req.getRequestDispatcher("Logout.jsp").forward(req, res);

}else {
}
}
}


```

```

        req.setAttribute("msg", "Product deleted Successfully..<br>");
        req.getRequestDispatcher("DeleteProduct.jsp").forward(req, res);
    }
}

}

```

DeleteProduct.jsp

```

<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.UserDetails"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
UserDetails ud = (UserDetails)session.getAttribute("ud");
out.println("Page of "+ud.getfName()+"<br>");
String msg = (String)request.getAttribute("msg");
out.println(msg);
%>
<a href="view">ViewProducts</a>
<a href="product.html">AddProduct</a>
<a href="code1.html">Edit-UpdateProduct</a>
<a href="code2.html">DeleteProduct</a>
<a href="logout">Logout</a>
</body>
</html>
=====
```

Note:

=>Update above application with User registration process.

=>Update above application with HTML/CSS/JS(Optional)

=>ORM(Object Relation Mapping) tools

=>**Hibernate with JPA**

=>**Spring Data or Spring ORM**

=>**Spring(SpringCore, Spring MVC)**

=>**Spring JDBC(Optional)**

Note:

=>**IDEs are use to generate fast sources codes.**

faq:

wt it the diff b/w

(i)JAR

(ii)WAR

(iii)EAR

(i)JAR:

=>**JAR means 'Java Archeive' and which is compressed format of class files.**

(ii)WAR:

=>**WAR means 'Web Archeive' and which is compressed format of HTML files, XML files, UI-Files, Servlets, JSP and Jars.**

=>**WebApplications are converted into WAR files.**

(iii)EAR:

=>**EAR means 'Enterprise Archeive' and which is compressed format**

of Enterprise Java Beans, WARs and JARs.

=>Enterprise Applications are converted into EAR files.

**imp*

Generating WAR file and executing from Tomcat Server:

step-1 : Generating WAR file using IDE Eclipse

RightClick on WebApplication->Export->WAR file->Browse the destination folder to store WAR file->name the file and click 'save'->click 'finish'

step-2 : start the Tomcat server

step-3 : Open WebBrowser and access the Server(Tomcat)

step-4 : Deploy the WAR file into Tomcat

Click on Manager App->perform login process->

From 'WAR file to deploy' click 'choose file'->

browse and select the file->click 'deploy'

step-5 : Execute the application as follows

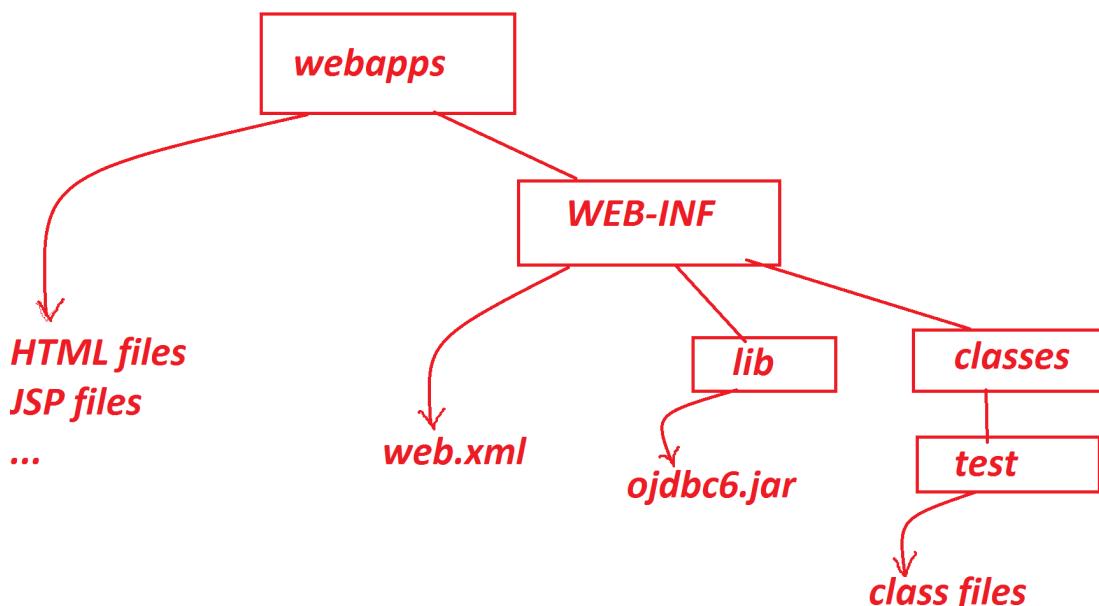
http://localhost:8082/JSP_MVC_App/

Note:

=>The Web-App war file is deployed into "webapps" folder of "Tomcat"

=>The following is the deployment directory structure of Tomcat:

Diagram:



Dt : 14/10/2022

Expression Language(EL) in JSP:

=>This EL simplifies the accessibility of data stored in java Bean component and other objects like request,session,application,etc..

Note:

It is newly added feature in JSP technology.

syntax of EL:

`$(expression)`

The following are the implicit objects of EL:

- 1.pageScope**
- 2.requestScope**
- 3.sessionScope**
- 4.applicationScope**
- 5.param**
- 6.paramValues**
- 7.header**
- 8.headerValues**
- 9.cookie**
- 10.initParam**
- 11.pageContext**

1.pageScope : It maps the given attribute name with value set

in the page scope

2.requestScope : It maps the given attribute name with value set

in the request scope

3.sessionScope : It maps the given attribute name with value set

in the session scope

4.applicationScope : It maps the given attribute name with value set

in the application scope

5.param : It maps the request parameter to the single value

6.paramValues :

It maps the request parameters to the array of

values

7.header : It maps the request header name to the single value

8.headerValues : It maps the request header names to the array of

values

9.cookie : It maps the cookie name to the cookie value

10.initParam : It maps the Initialization parameter

11.pageContext : It provides access to many objects request,session,...

=====

Summary of Objects Generated:

CoreJava Objects:

1.UserDefined Class objects

2.WrapperClass objects

(i)Byte Object

(ii)Short Object

(iii)Integer object

(iv)Long Object

(v)Float Object

(vi)Double Object

(vii)Character object

(viii)Boolean Object

3.String Objects

(i)String Object

(ii)StringBuffer object

(iii)StringBuilder object

4.Array Objects

(i)User defined Class Array

(ii)String Array

(iii)WrapperClass Array

(iv)Object Array(Dis-Similer objects)

(v)Jagged Array(Array holding Array Objects)

5.Collection<E> Objects

(a)Set<E>

(i)HashSet<E> object

(ii)LinkedHashSet<E> Object

(iii)TreeSet<E> Object

(b)List<E>

(i)ArrayList<E> Object

(ii)Vector<E> Object

|->Stack<E> Object

(iii)LinkedList<E> Object

(c) Queue<E>

(i) PriorityQueue<E> Object

| -> Deque<E>

(ii) ArrayDeque<E> Object

(iii) LinkedList<E> Object

6. Map<K,V> Objects

(i) HashMap<K,V> Object

(ii) LinkedHashMap<K,V> Object

(iii) TreeMap<K,V> Object

(iv) Hashtable<K,V> Object

7. Enum<E> objects

Utility Classes:

(i) java.util.Scanner

(ii) java.util.StringTokenizer

(iii) java.util.StringJoiner(Java8 version)

JDBC Objects:

1. Connection Object

2. Statement Object

3. PreparedStatement Object

4. CallableStatement object

5. Scrollable ResultSet Object

6. Non-Scrollable ResultSet Object

7.DatabaseMetaData object

8.ParameterMetaData object

9.ResultSetMetaData object

10.RowSet Object

11.RowSetMetaData

12.Connection pooling object(Vector<E> object)

=====

Servlet Objects:

1.ServletContext object

2.ServletConfig Object

3.ServletRequest object/HttpServletRequest object

4.ServletResponse object/HttpServletResponse object

5.PrintWriter object

6.HttpSession object

7.Cookie object

8.DAO Layer object

9.JavaBean object

10.JCF Object

JSP objects:

1.application

2.config

3.request

4.response

5.out

6.session

7.exception

8.page

9.pageContext

JSP EL objects:

1.pageScope

2.sessionScope

3.requestScope

4.applicationScope

5.param

6.paramValues

7.cookie

8.pageContext

9.header

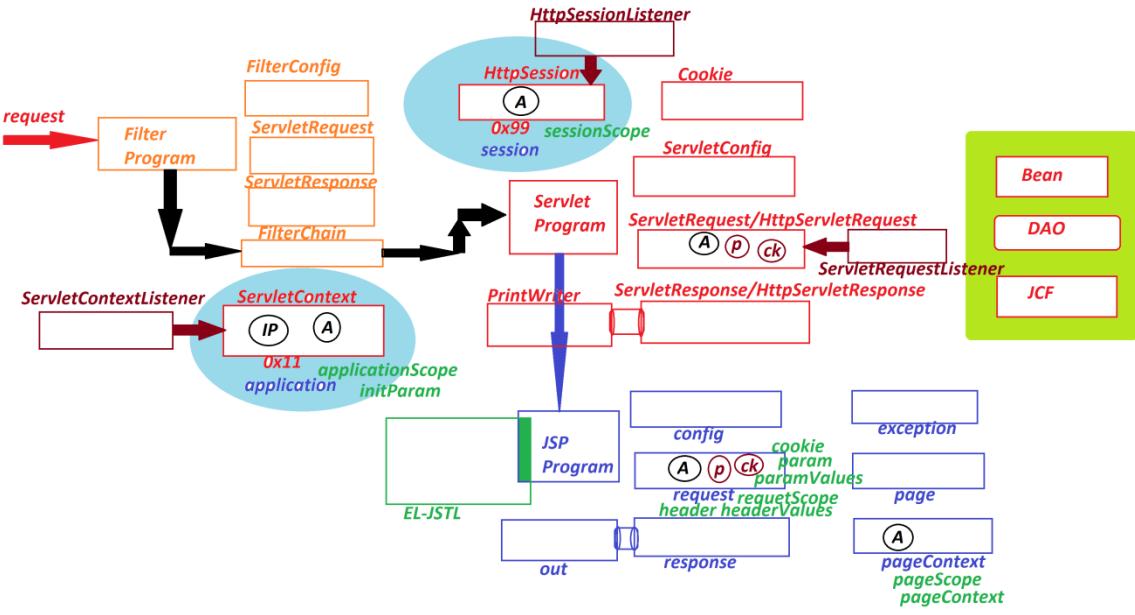
10.headerValues

11.initParam

Note:

=>In realtime JSP-EL replaced by SpEL.

Objects Layout diagram:



JSTL(JSP Standard Tag Lib):

=>This JSTL represents a set of tags to simplify the JSP development.

Advantages of JSTL:

- (i)Fast Development
- (ii)Code Reusability
- (iii)No need to use scriptlet tag

The following are the JSTL tags:

(1)Core Tags

(2)Function Tags

(3)Formatting Tags

(4)XML Tags

(5)SQL Tags

Note:

=>we use "taglib" directive tag to declare JSTL Tags.

=>*To Execute JSTL Tags we use the following steps:*

(i) *Download the following Jar files to execute JSTL tags*

javax.servlet.jsp.jstl-1.2.1.jar

javax.servlet.jsp.jstl-api-1.2.1.jar

(ii) *These jars must be available within 'lib' folder of*

'WEB-INF' in deployment directory

(1)Core Tags:

=>*These JSTL core tags provides variable support,URL management, flow control etc. (Basic code writing)*

syntax:

`<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>`

The following are the List of JSTL Core Tags:

c:out ->It displays the result of an expression, similar to

<%=...%> tag.

c:import->It Retrieves relative or an absolute URL.

c:set-> It sets the value to variable.

c:remove->It is used for removing the variable .

c:catch-> It is used for Catching any Throwabe exception that occurs in the body.

c:if-> It is an conditional tag

c:choose, c:when, c:otherwise->

It is the simple conditional tag that includes its body content if the evaluated condition is true.

*imp

c:forEach-> It is the basic iteration tag.

c:forTokens-> It iterates over tokens which is separated by the supplied delimeters.

c:param-> It adds a parameter in a containing 'import' tag's URL.

c:redirect-> It redirects the browser to a new URL and supports the context-relative URLs.

c:url-> It creates a URL with optional query parameters.

Ex:

input.html

```
<form method="post" action="CoreTags.jsp">
```

*Enter the String:<input type="text" name="str">
*

<input type="submit" value="Display">

```
</form>
```

web.xml

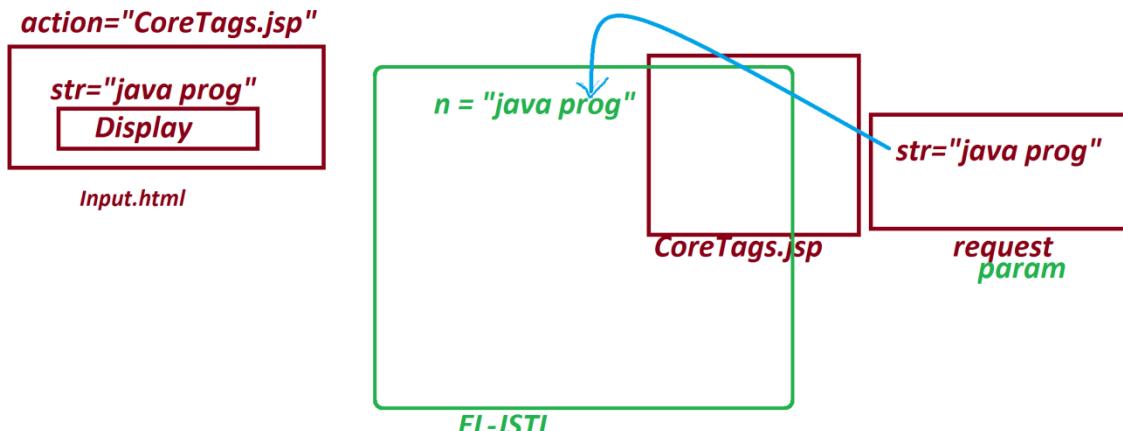
```
<web-app>
  <welcome-file-list>
    <welcome-file>input.html</welcome-file>
  </welcome-file-list>
</web-app>
```

CoreTags.jsp

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="n" value="${param.str}" />
<c:forEach var="j" begin="1" end="5">
  <c:out value="${n}" /><p>
</c:forEach>

<c:forTokens items="${param.str}" delims=" " var="name">
  <c:out value="${name}" /><p>
</c:forTokens>
```

Diagram:



(2) Function Tags:

=> These JSTL function tags provides a number of standard functions, most of these functions are common string manipulation functions.

syntax:

```
<%@taglib uri= "http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
```

List of Some Function tags:

fn:contains() : It is used to test if an input string containing the specified substring or not.

fn:containsIgnoreCase(): It is used to test if an input string contains the specified substring as a case insensitive way.

fn:endsWith() : It is used to test if an input string ends with the specified suffix.

fn:indexOf(): It returns an index within a string of first occurrence of a specified substring.

fn:trim(): It removes the blank spaces from both the ends of a string.

fn:startsWith(): It is used for checking whether the given string is started with a particular string value or not.

fn:split(): It splits the string into an array of substrings.

fn:toLowerCase(): It converts all the characters of a string to lower case.

fn:toUpperCase(): It converts all the characters of a string to uppercase.

fn:substring(): It returns the subset of a string according to the given start and end position.

fn:length(): It returns the number of characters inside a string, or the number of items in a collection.

fn:replace(): It replaces all the occurrence of a string with another string sequence.

Ex:

input.html

```
<form method="post" action="FunctionTag.jsp">  
    Enter the String:<input type="text" name="str"><br>  
    <input type="submit" value="Submit">  
</form>
```

web.xml

```
<web-app>  
  <welcome-file-list>  
    <welcome-file>input.html</welcome-file>  
  </welcome-file-list>  
</web-app>
```

FunctionTags.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"  
        prefix="c" %>  
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions"  
        prefix="fn" %>  
<c:set var="s1" value="${param.str}" />  
<c:choose>  
  <c:when test="${fn:contains(s1, 'java')}">  
    <p>String Founded</p>  
  </c:when>  
  <c:otherwise>  
    <p>String Not Founded</p>  
  </c:otherwise>  
</c:choose>  
  
<c:if test="${fn:containsIgnoreCase(s1, 'JAVA')}">
```

```
<p>String Founded</p>  
</c:if>
```

(3)Formatting Tags:

=>These formatting tags provide support for message formatting, number formating and date formatting etc.

=>These formatting tags are also used for internationalized web sites to display and format text,time,date and numbers.

syntax:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
```

List of some Formatting tags:

fmt:parseNumber : It is used to Parse the string representation of a currency, percentage or number.

fmt:formatNumber : It is used to format the numerical value with specific format or precision.

fmt:parseDate : It parses the string representation of a time and date.

fmt:setTimeZone : It stores the time zone inside a time zone configuration variable.

*imp

fmt:formatDate : It formats the time and date using the supplied pattern and styles.

Ex:

FormatTags.jsp

```
<%@ taglib prefix="c"  
    uri="http://java.sun.com/jsp/jstl/core"%>  
  
<%@ taglib prefix="fmt"  
    uri="http://java.sun.com/jsp/jstl/fmt"%>  
  
<html>  
  
<head>  
  
<title>fmt:formatDate</title>  
  
</head>  
  
<body>  
  
<h2>Different Formats of the Date</h2>  
  
<c:set var="Date" value="<%=new java.util.Date()%>" />  
  
<p>  
  
Formatted Time :  
  
<fmt:formatDate type="time" value="${Date}" />  
  
</p>  
  
<p>  
  
Formatted Date :  
  
<fmt:formatDate type="date" value="${Date}" />  
  
</p>  
  
<p>  
  
Formatted Date and Time :
```

```
<fmt:formatDate type="both" value="${Date}" />  
</p>
```

```
<p>
```

Formatted Date and Time in short style :

```
<fmt:formatDate type="both" dateStyle="short" timeStyle="short"  
value="${Date}" />  
</p>
```

```
<p>
```

Formatted Date and Time in medium style :

```
<fmt:formatDate type="both" dateStyle="medium" timeStyle="medium"  
value="${Date}" />  
</p>
```

```
<p>
```

Formatted Date and Time in long style :

```
<fmt:formatDate type="both" dateStyle="long" timeStyle="long"  
value="${Date}" />  
</p>
```

```
</body>
```

```
</html>
```

web.xml

```
<web-app>  
<welcome-file-list>  
<welcome-file>FormatTag1.jsp</welcome-file>
```

```
</welcome-file-list>  
</web-app>
```

(4)XML Tags:

=>The JSTL XML tags are used for providing a JSP-centric way of manipulating and creating XML documents.

syntax:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>
```

(5)SQL Tags:

=>The SQL tag library allows the tags to interact with RDBMS (Relational Databases) such as Microsoft SQL Server, MySQL, or Oracle.

syntax:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
```

Note:

=>In realtime we must not have JSP Centric XML and DB Connections, because of this reason XML tags and SQL Tags are less used when compared to other tags.

define pageContext?

=>pageContext is an implicit object generated for javax.servlet.jsp.PageContext class and which is extended from javax.servlet.jsp.JspContext class.
=>using pageContext object we can access remaining implicit objects

of JSP.

The following are some important methods:

```
public abstract javax.servlet.jsp.JspWriter getOut();  
public abstract javax.servlet.http.HttpSession getSession();  
public abstract javax.servlet.ServletRequest getRequest();  
public abstract javax.servlet.ServletResponse getResponse();  
public abstract java.lang.Object getPage();  
public abstract java.lang.Exception getException();  
public abstract javax.servlet.ServletConfig getServletConfig();  
public abstract javax.servlet.ServletContext  
getServletContext();
```

Note:

=>we can also add "attribute" to the "pageContext" object.
=>'pageScope' implicit object of EL-JSTL will access the attribute
of "pageContext" object of JSP.
=>"pageContext" implicit object of EL-JSTL will access the
remaining objects of EL-JSTL.

=====

define "page"?
=>"page" is an implicit object of java.lang.Object
class, this "page" object is used when we want the methods of

java.lang.Object class.

Note:

=>*page implicit object is used part of JSP, when we want to perform*

Cloning process or Thread communication process.

faq:

wt is the dif b/w

(i) static Web Project

(ii) Dynamic Web Project

(iii) Web Fragments Project

(i) Static Web Project is written entirely using HTML.

Each web page is a separate document and there are no databases or external files depending on HTML pages.

(ii) Dynamic Web Project depends on more complex code like PHP, ASP,

JSP, Servlet java file etc., and this project is for server side coding.

(iii) Essentially, Web Fragments (introduced in Servlet 3.x) allows you to create a reusable portion (or fragment) of the deployment descriptor, web.xml.

=>we use Web Fragments project to work with Web Fragments.

Venkatesh Maiopathi