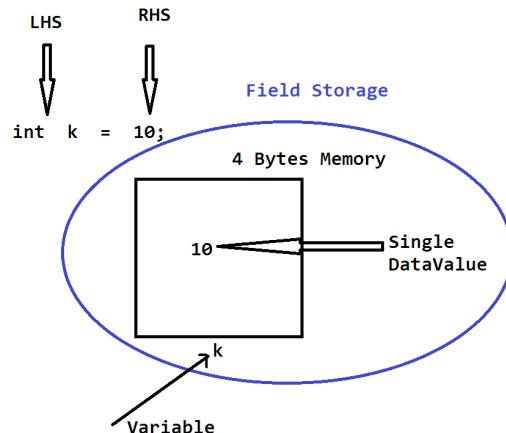
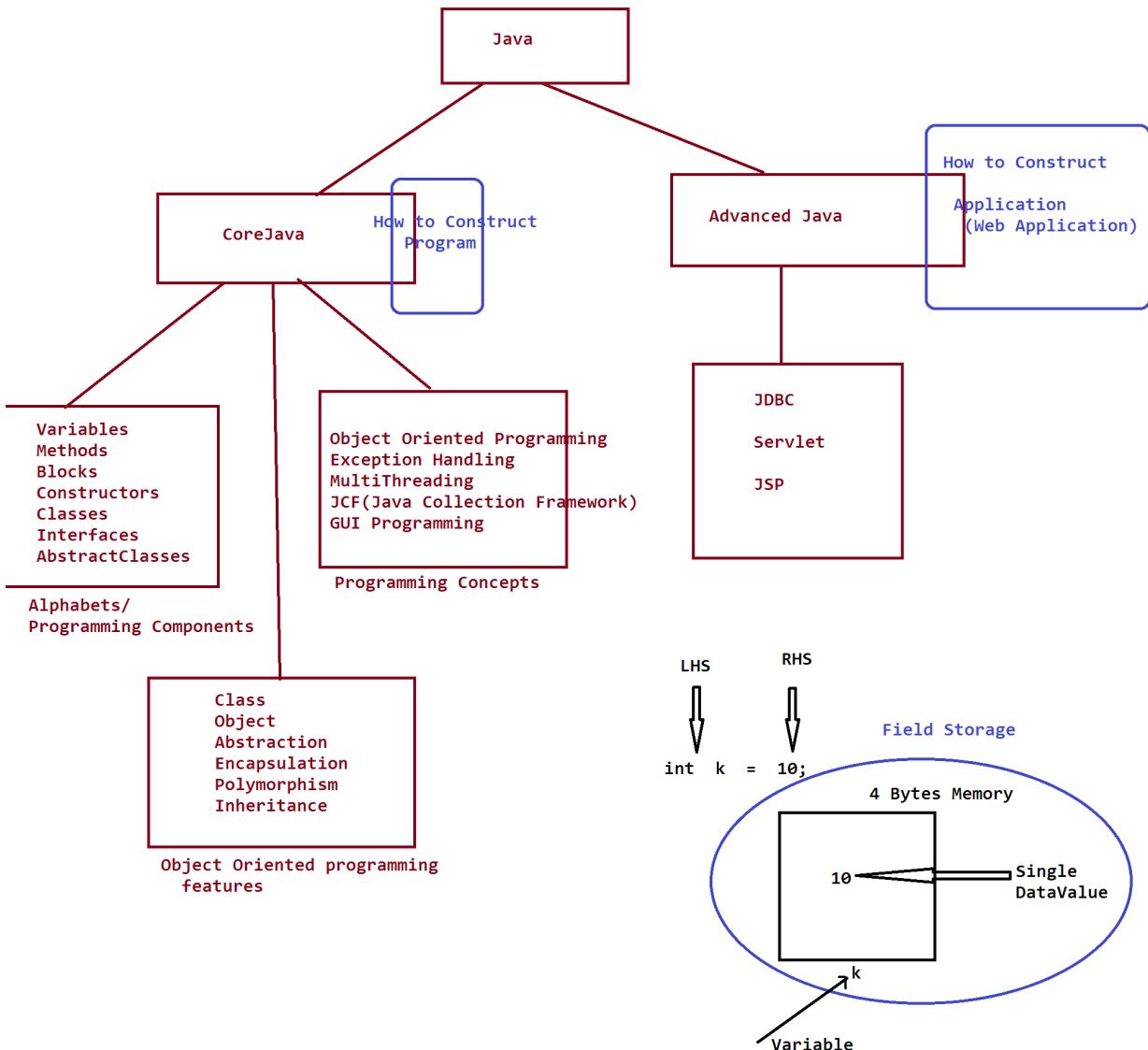


Dt : 18/11/2020(Adv Java)



define Application?

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



define Web Application?

=>The application which is running in Web Environment or Internet

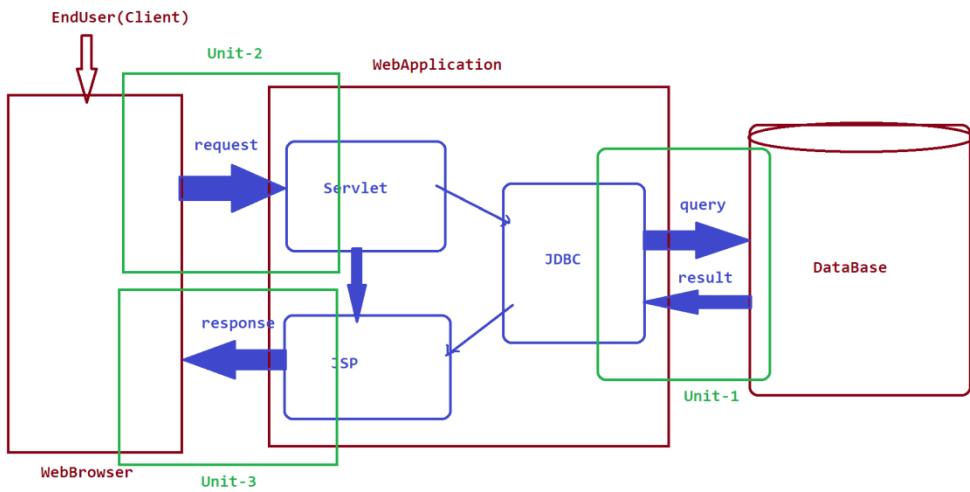
Environment in the process of interacting with EndUser(Client) through
WebBrowser is known as Web Application.

=>we use the following technologies to construct WebApplication:

1.JDBC

2.Servlet

3.JSP



=====

==

Dt : 19/11/2020

1.JDBC(Unit-1)

=>JDBC Stands for 'Java DataBase Connection' and which is used to interact with DataBase product.

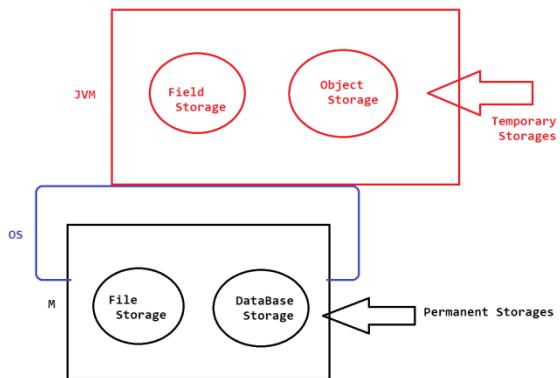
(using JDBC we can control DB Product)

Note:

=>DB Product comes under Storage category.

=>The following are the storages realated to Java Application:

- (a)Field Storage**
- (b)Object Storage**
- (c)File Storage**
- (d)DataBase Storage**



(a) Field Storage:

=>The memory which is generated to hold single data value is known as **Field Storage**.

Note:

=>The primitive datatypes which are declared part of application will generate **Field Storages**.

List of Primitive DataTypes:

byte

short

int

long

float

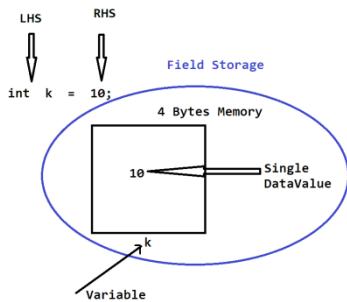
double

char

boolean

Exp:

```
int k = 10;
```



Dt : 20/11/2020

(b)Object Storage:

=>The memory which is generated to hold group members is known as Object Storage.

Note:

=>The NonPrimitive datatypes which are declared part of application will generate Object Storages.

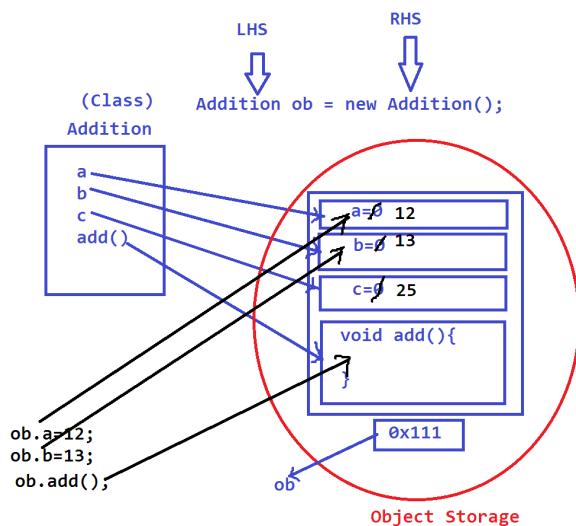
List of NonPrimitive DataTypes:

Class	Interface	Array	Enum
-------	-----------	-------	------

Exp:

```
class Addition
{
    int a,b,c;
    void add()
    {
        c = a+b;
        System.out.print(c);
    }
}
```

Addition ob = new Addition();



Note:

=>The Field Storages and the Object Storages which are generated part of JVM are temporary Storages,because these storages will be destroyed automatically when the JVM shutdowns.

=>when we want to have permanent storages for JavaAppl then we use File Storage or DB Storage.

Dt : 23/11/2020(Monday)

*imp

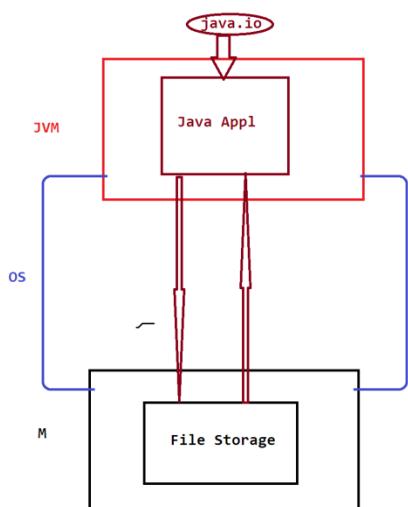
(c)File Storage

=>The Smallest permanent storage of Computer System which is organized by the Operating System is known as File Storage.

Note:

=>when we want to establish communication b/w JavaAppl and FileStorage
then the JavaAppl must be constructed using classes and Interfaces
available from "java.io" package.

Diagram:



faq:

define API?

=>API stands for 'Application Programming Interface' and which provides the related classes and Interfaces used for constructing applications.

Note:

=>API is also known as package.

List of some packages:(APIs)

java.lang - Language package

java.util - Utility package

java.io - IO Stream and Files package

java.net - Networking package

java.sql - DataBase Connection package

javax.servlet - Servlet programming package

javax.servlet.jsp - Jsp programming package

Disadvantage of FileStorage:

- 1. Data Redundancy**
- 2. Data Inconsistency**
- 3. Difficulty in Accessing Data**
- 4. Limited Data Sharing**
- 5. Integrity Problems**
- 6. Atomicity Problems**
- 7. Concurrent Access Anomalies**
- 8. Security Problems**

Dt : 24/11/2020

1. Data Redundancy:

It is possible that the same information may be duplicated in different files and this leads to data redundancy results in memory wastage.

2. Data Inconsistency:

Because of data redundancy,it is possible that data may not be in consistent state.

3. Difficulty in Accessing Data:

Accessing data is not convenient and efficient in file processing system.

4. Limited Data Sharing:

Data are scattered in various files and also different files may have different formats and these files may be stored in different folders may be of different departments.

So, due to this data isolation, it is difficult to share data among different applications.

5. Integrity Problems:

Data integrity means that the data contained in the database is both correct and consistent, for this purpose the data stored in database must satisfy correct and constraints. (which is not possible in FileStorage)

6. Atomicity Problems:

**Any operation on database must be atomic.
this means, it must happen in its entirely or not at all.
(which is not possible in File Stoarge)**

7. Concurrent Access Anomalies:

**Multiple users are allowed to access data simultaneously. this is for the sake of better performance and faster response.
(which is not possible in File Storage)**

8. Security Problems:

**Database should be accessible to users in limited way.
Each user should be allowed to access data concerning his requirements only. (which is not possible in FileStorage)**

=====

Note:

=>The DisAdvantages of File Storage can be overcomed using DB Storage.

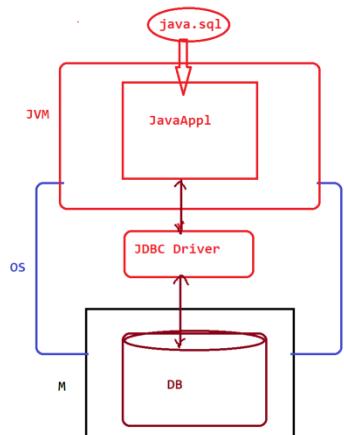
*imp

(d) DataBase Storage:

=>The Largest permanent Storage which is installed into Computer System from externally is known as DB Storage.

Note:

=>In the process of establishing communication b/w JavaAppl and DB Storage, the JavaAppl must be constructed using classes and Interfaces available from 'java.sql' package and the Appl must use "JDBC Driver".



Steps to Install Oracle DataBase:

Step 1

First, we have to download the software from the Oracle website.
Choose "Accept License Agreement" and proceed further more to download.

Oracle Database 11g Release 2

Standard Edition Standard Edition One, and Enterprise Edition

7/13: Patch Set 11.2.0.4 for Linux and Solaris is now available on support.oracle.com. Note: it is a full installation (you do not need to download 11.2.0.1 first). See the [README](#) for more info (login to My Oracle Support required).

(11.2.0.4.0)

 [OpenVMS](#)

(11.2.0.2.0)

 [zLinux64](#)

(11.2.0.1.0)

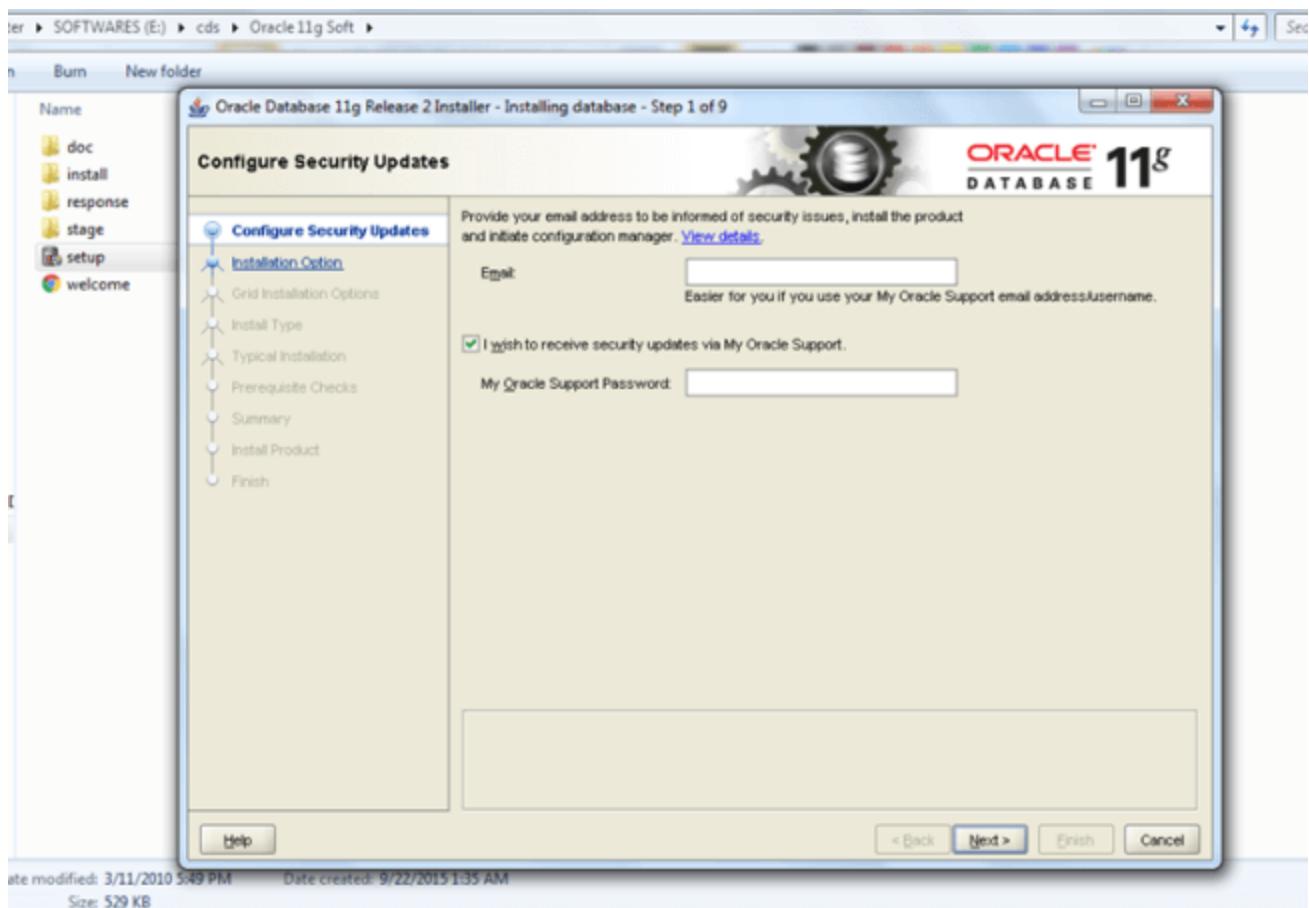
 [Microsoft Windows \(32-bit\)](#)
 [Microsoft Windows \(64\)](#)
 [Linux x86](#)
 [Linux x86-64](#)
 [Solaris \(SPARC\) \(E4-bit\)](#)
 [Solaris \(x86-64\)](#)
 [HP-UX Itanium](#)

Choose your
Platform

[File 1](#) [File 2](#) (23B) [See All](#)
[File 1](#) [File 2](#) (23B) [See All](#)

Step 2

When the download completes, unzip the downloaded file and you will get a setup.exe file. Now, right click on setup.exe and "Run as Administrator". An installer window will pop up.

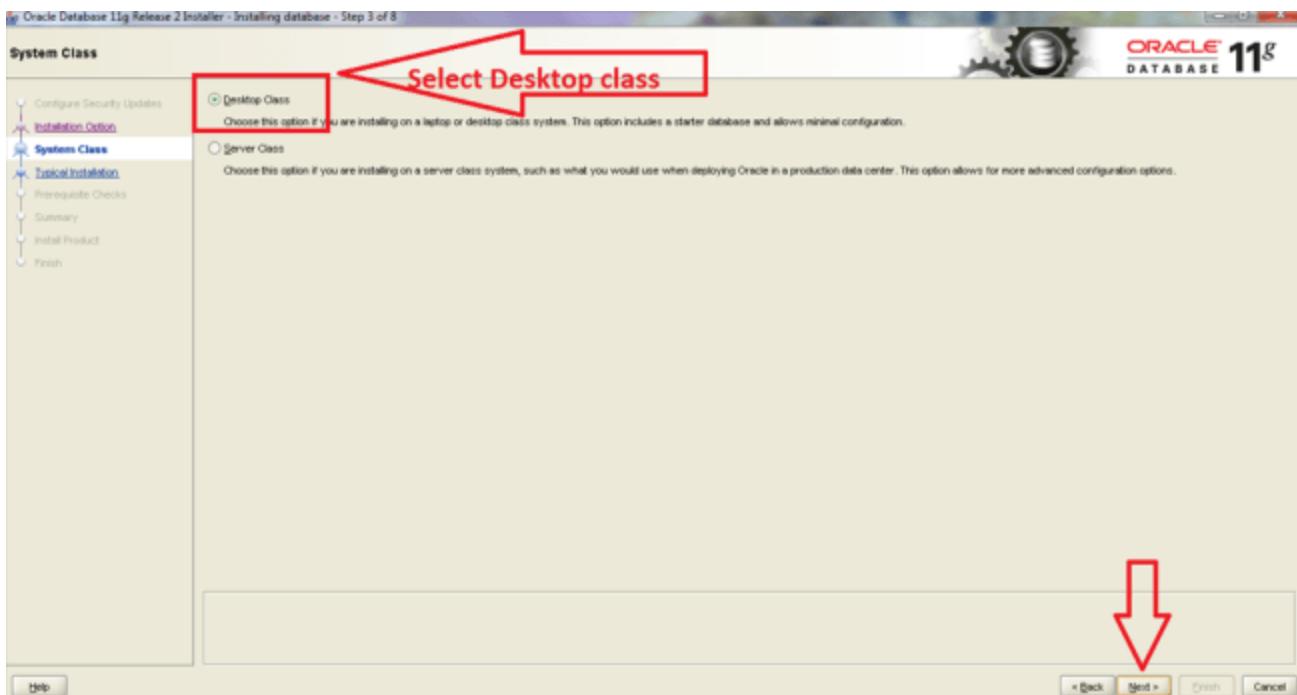


Step 3

You can skip the "configure security updates" option. Now, click on Next.

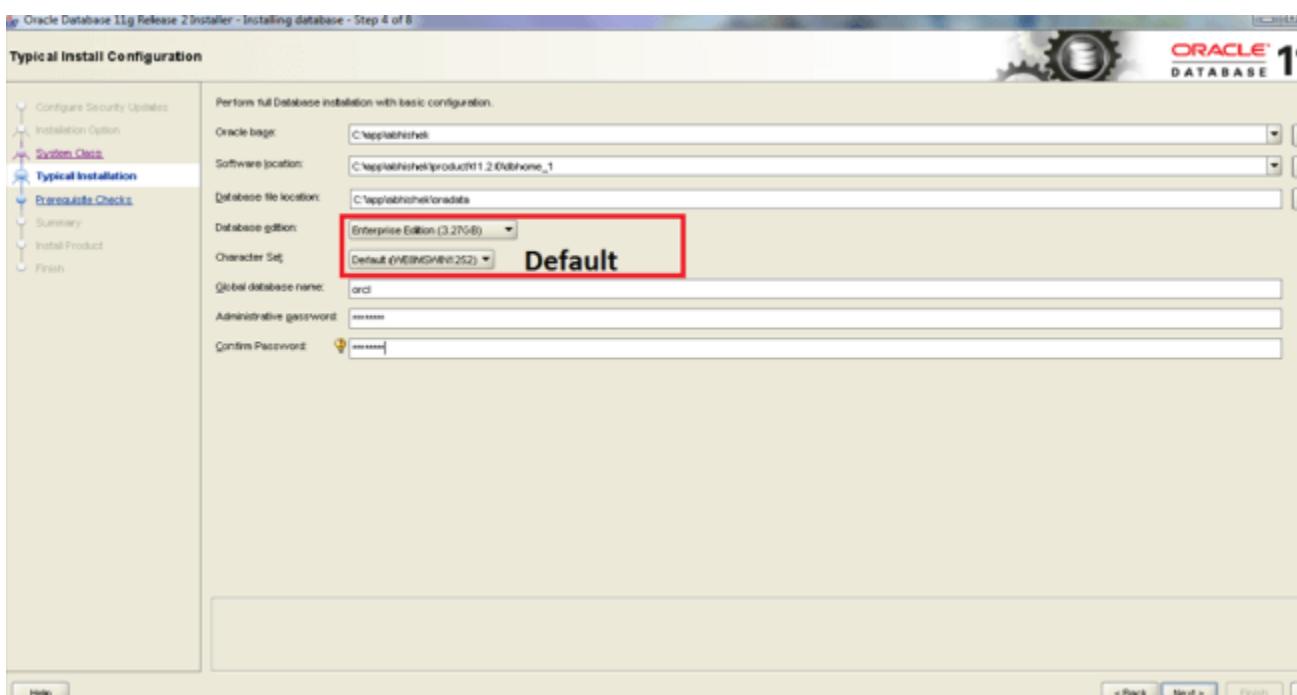
Step 4

Now, choose your System Class(I prefer desktop class for beginners) and click on Next.



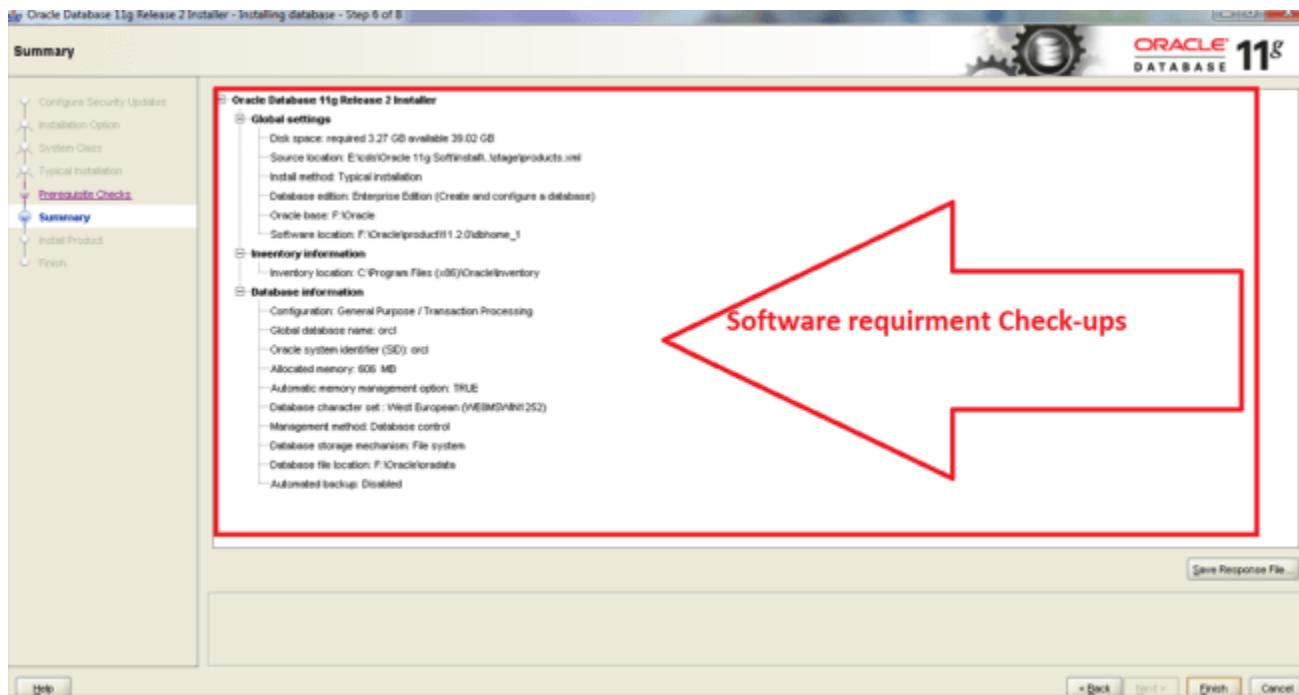
Step 5

Choose your path wherever you want to install your product. If you want to change the drive, then replace only Character 'C:\' to 'D:\' and all the options should be left the same as usual. Just choose your Oracle Standard Password and proceed. The global database name should be 'orcl'.



Step 6

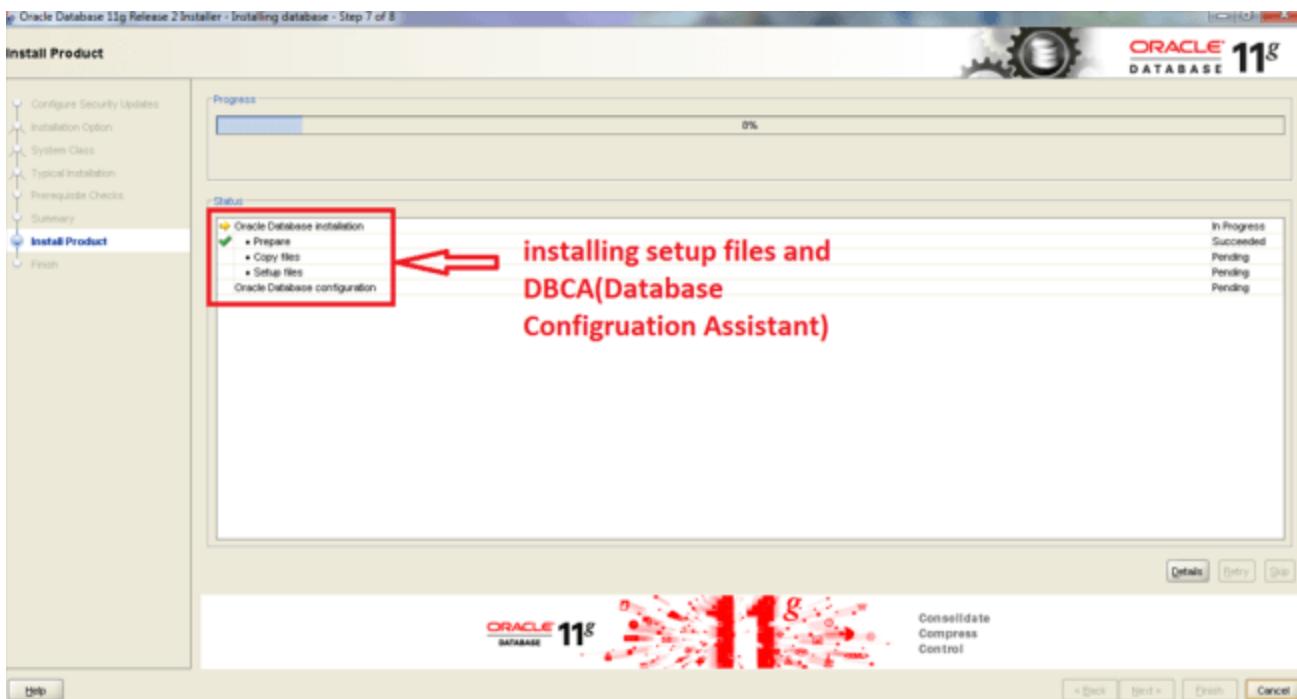
Now, Oracle checks up the software requirements and dependencies and it will give the installation summary. If everything is fine, then proceed further more.



Step 7

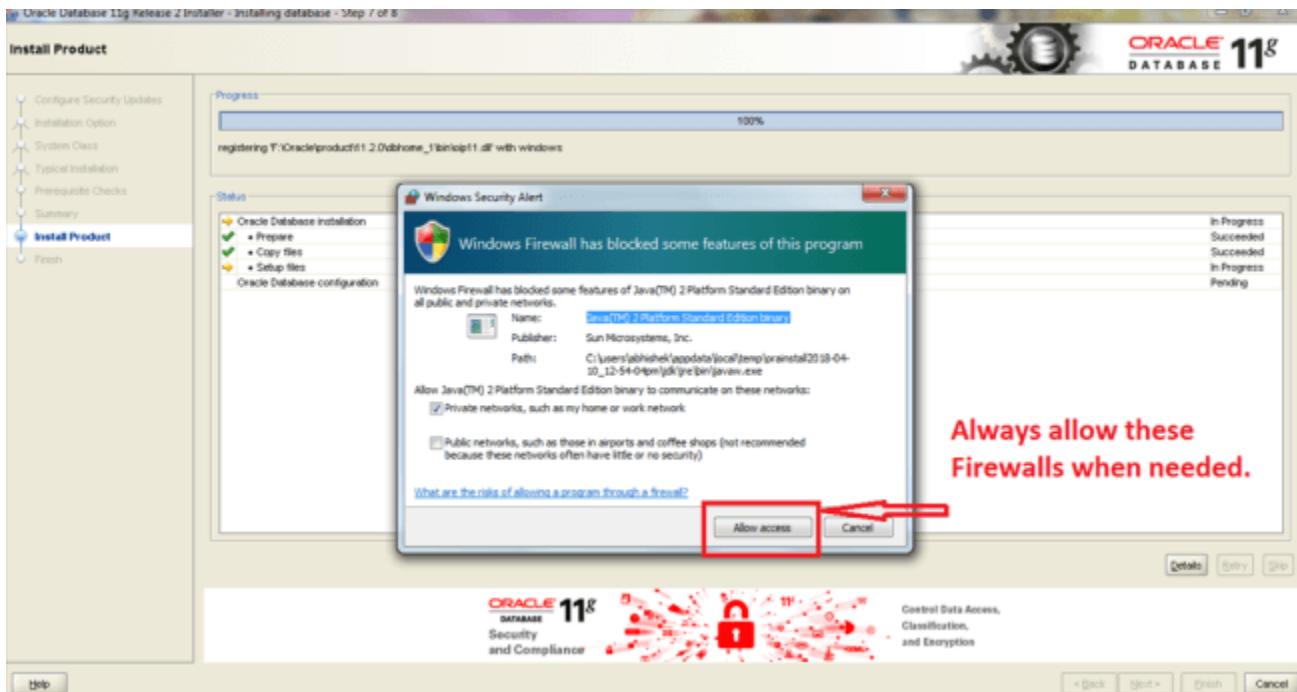
Now oracle will install the following products in your system.

1. Setup files
2. Database configuration assistant



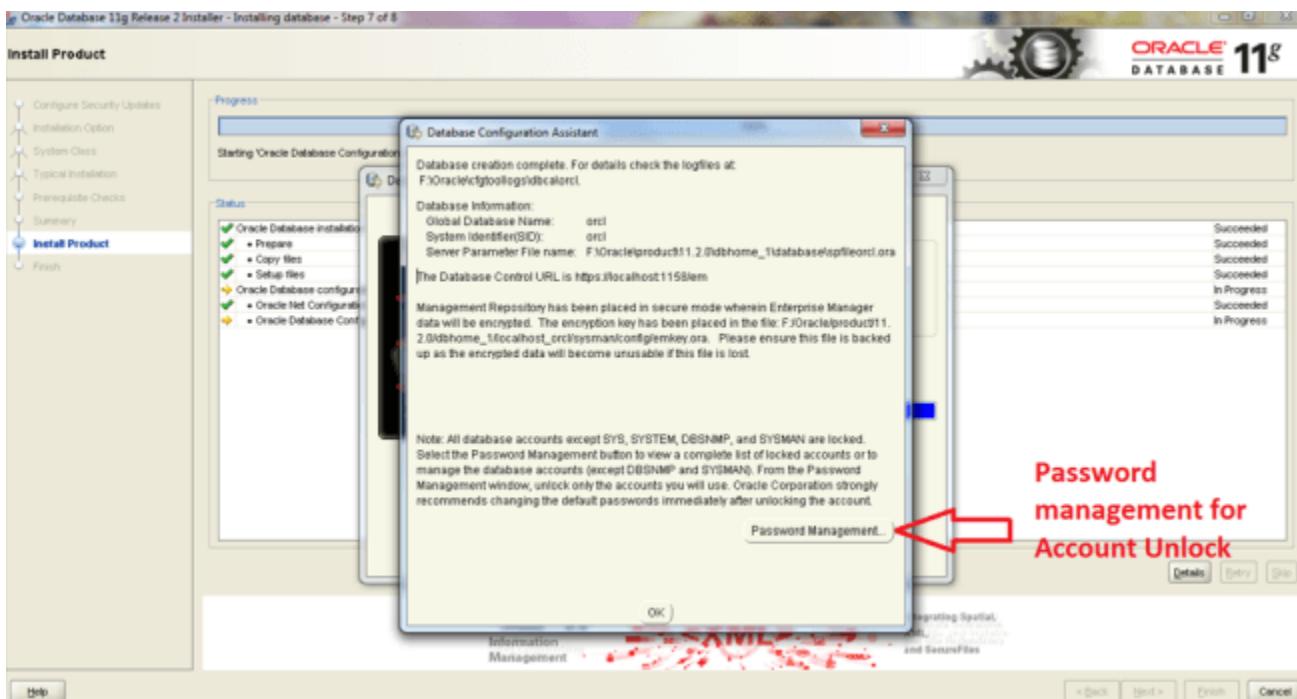
Step 8

Always allow these firewalls when it pop ups in your system.

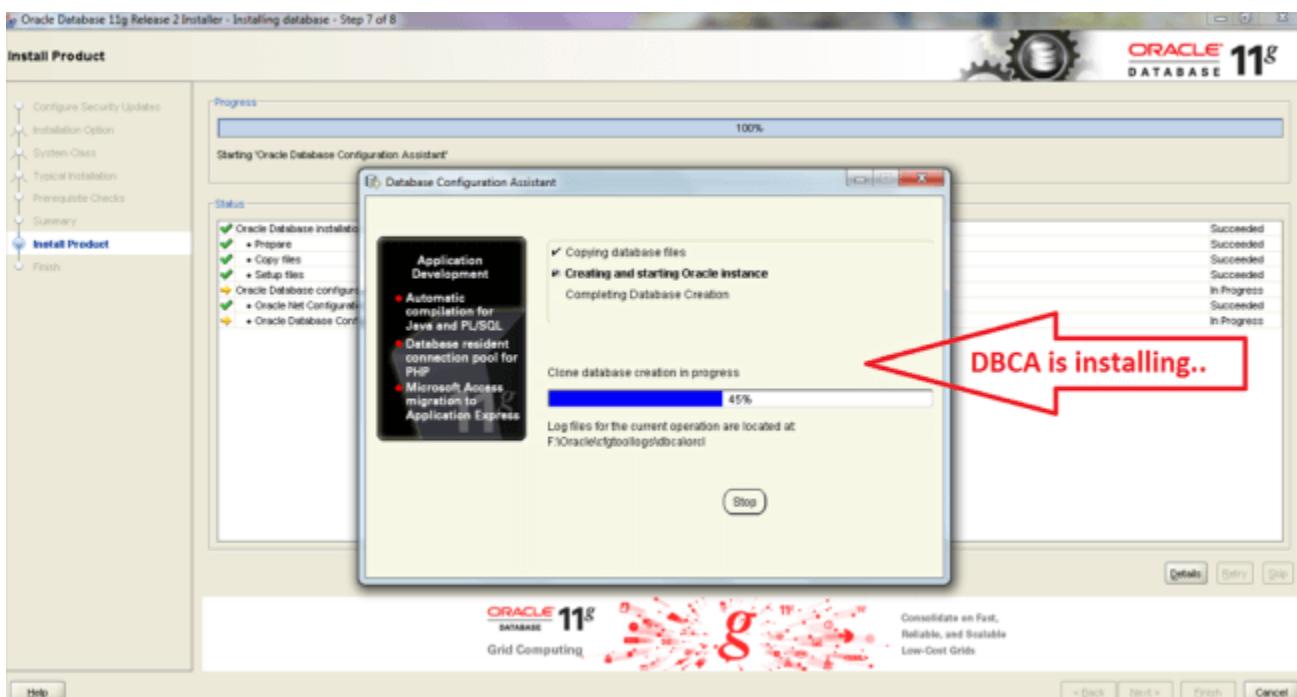


Step 9

Now you can see DBCA(Database configuration assistant) is Installing.



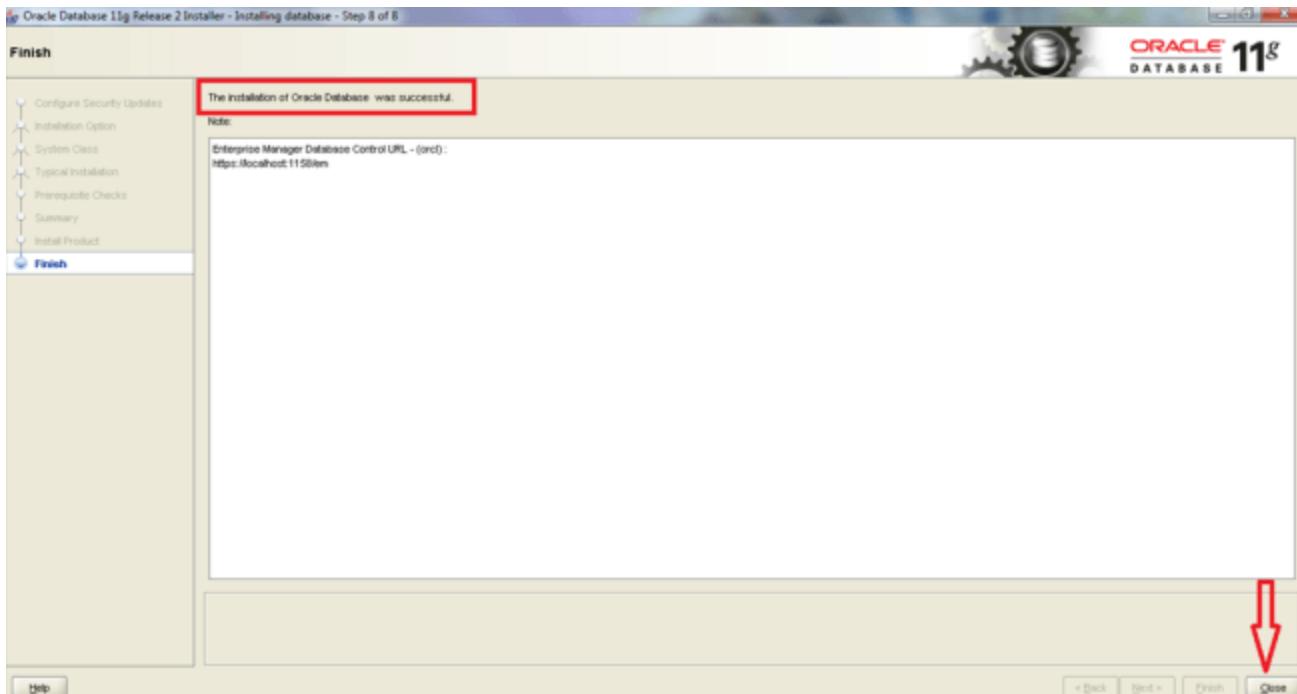
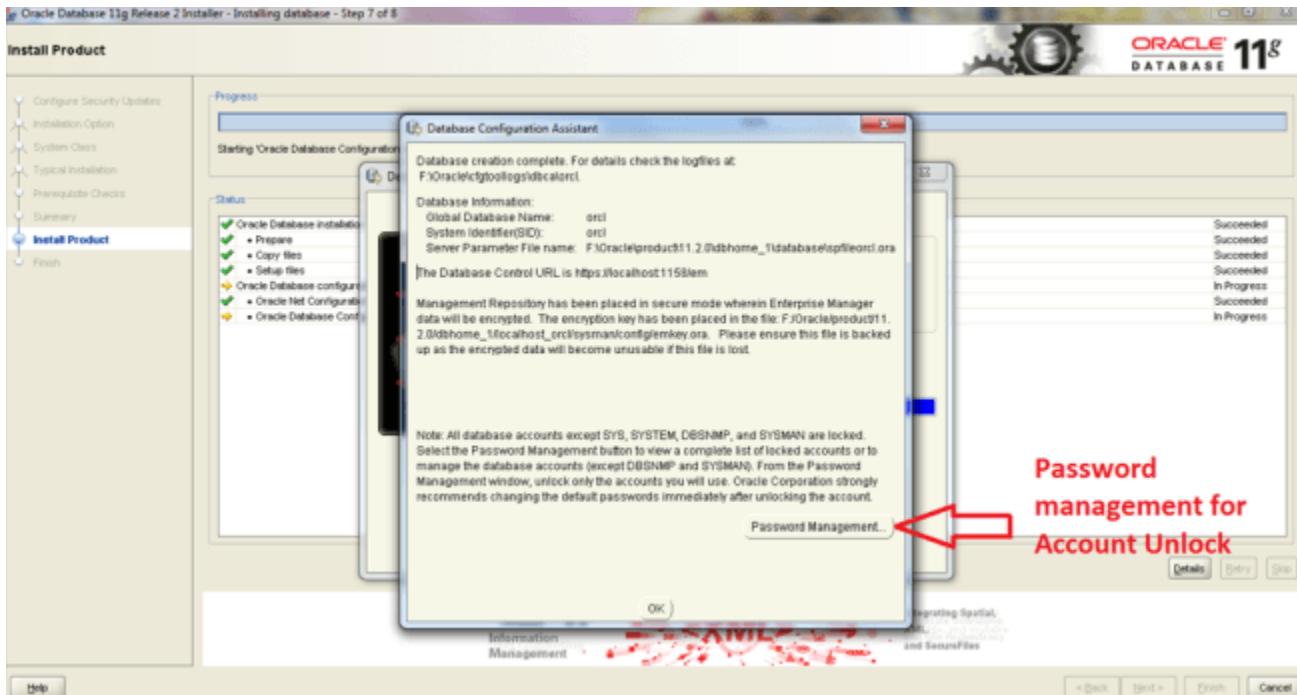
DBCA is a tool who help to create and manage Oracle Databases with GUI.



Step 10

When DBCA installation will complete then a pop up window will open.

With the help of "Password management" tool you can alter the default passwords for the particular database accounts

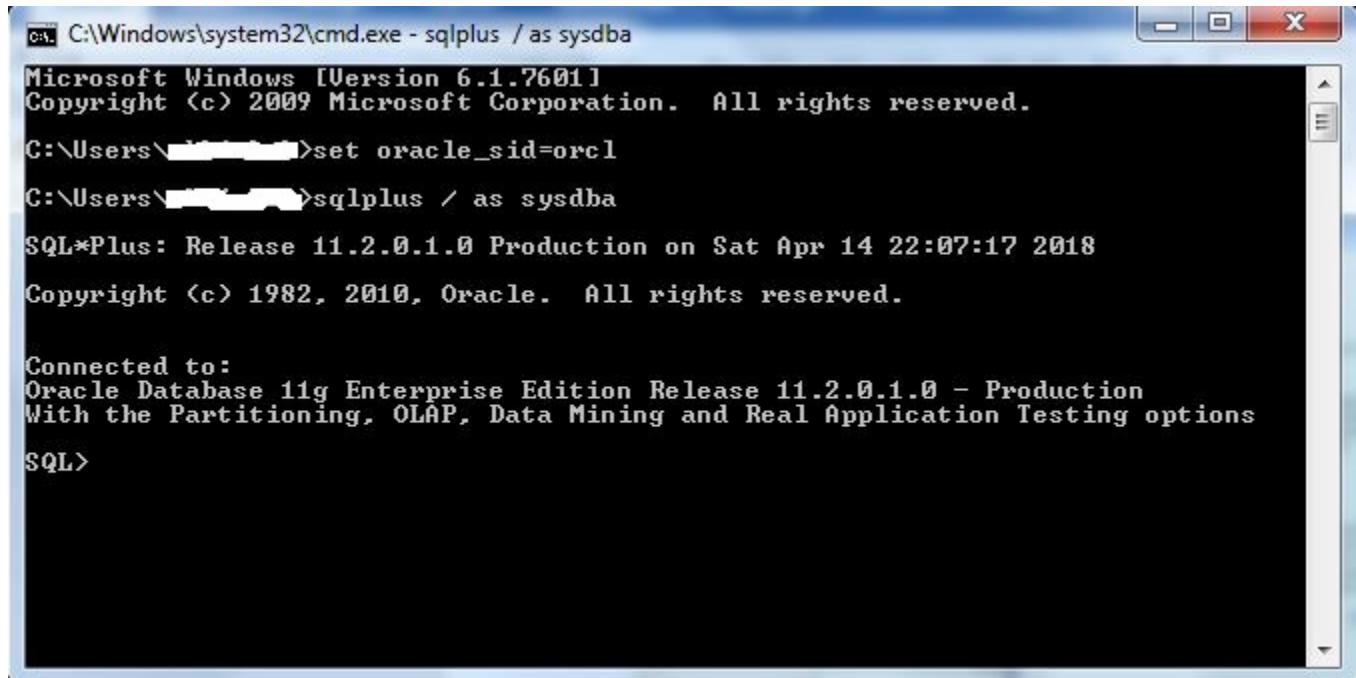


Step 11

Now test your installation done or not.

Open cmd with Run As administrator.

1. `set oracle_sid=orcl`
2. `Sqlplus / as sysdba`



The screenshot shows a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe - sqlplus / as sysdba". The window displays the following text:

```
Microsoft Windows [Version 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. All rights reserved.

C:\Users\████████>set oracle_sid=orcl
C:\Users\████████>sqlplus / as sysdba

SQL*Plus: Release 11.2.0.1.0 Production on Sat Apr 14 22:07:17 2018

Copyright <c> 1982, 2010, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>
```

After this command, you will see Oracle Database 11g Enterprise is connected. Now, you are connected to the Oracle Database 11g Enterprise Edition.

Dt : 25/11/2020

define Driver?

=>The small s/w program used by Operating System(OS) to control the Components of Computer System is known as "Driver program".

Exp:

Audio driver

Video driver

Printer driver

N/W driver

.

.

.

define JDBC Driver?

=>The driver program which is used to communicate with DB Storage is known as "JDBC Driver".

(Java DataBase Connection Driver)

=>The following are the types of JDBC Drivers:

1.JDBC-ODBC bridge driver(Type1)

2.Native-API driver(Type2)

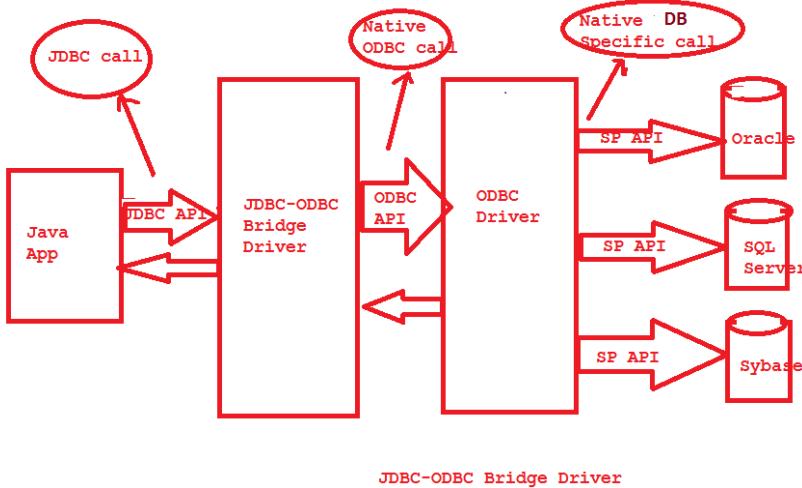
3.Network Protocol driver(Type3)

4.Thin driver(Type4)

1.JDBC-ODBC bridge driver(Type1):

=>JDBC-ODBC bridge driver will take the support of 'ODBC driver' to communicate with DB Storage.

=>In this process JDBC-ODBC bridge driver will convert 'JDBC call' to 'Native ODBC call',and ODBC driver will convert 'Native ODBC call' to DB Specific call for connection.



DisAdvantage:

=>In Type1 driver internally having more number of conversions and consumes more execution, and degrades the performance of an application ,because of this reason which is not preferable in realtime.

Note:

=>From Java8 version onwards the Type1 driver is not available.

faq:

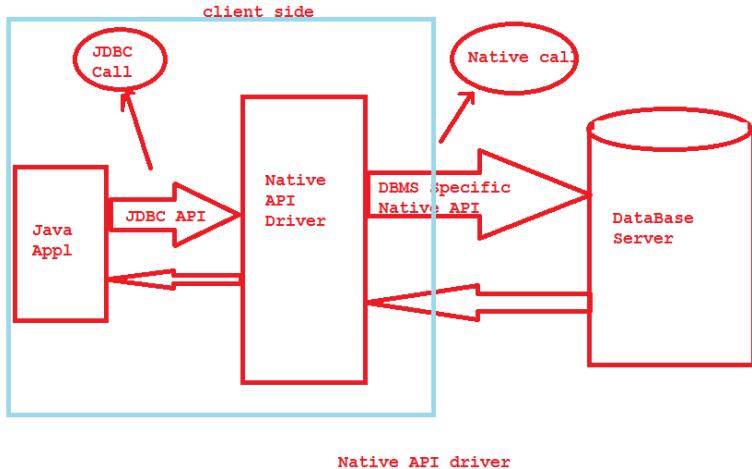
define ODBC driver?

=>ODBC stands for "Open DataBase Connection" and which is used to convert Native ODBC calls to DB Specific call.

=>ODBC driver is Platform dependent driver.

2.Native-API driver(Type2):

=>Native-API driver will convert 'JDBC call' to 'DB Specific call' for connection.(which will not take support of ODBC driver)



Advantage:

=>Type2 driver is having less number of conversions when compared to Type1 and which is high performance when compared to type1.

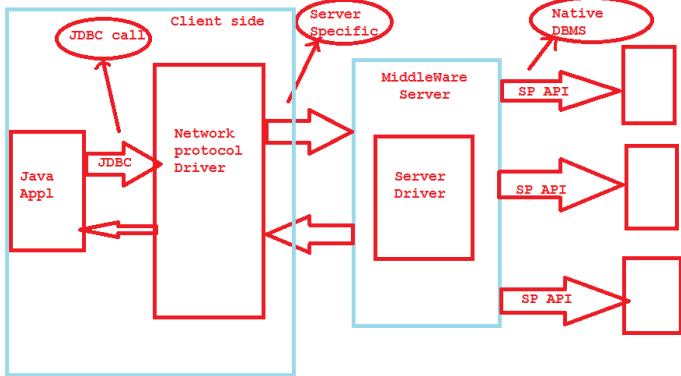
DisAdvantage:

=>In Type2 driver the Client Computer must be installed with DB Specific API and in this process the application will become DB dependent and which is not preferable in realtime.

Dt : 26/11/2020

3.Network Protocol driver(Type3):

=>Network protocol driver will convert JDBC call to Server Call and this Server will raise DB Specific call for connection.



Advantage:

=>Type3 is more advantageous than type1 and type2,becuase there is no ODBC driver and there is no DB Specific API in Client computer.

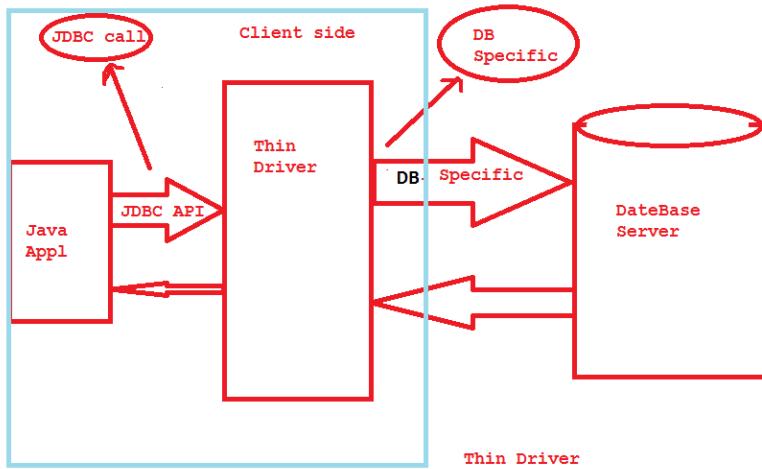
DisAdvantage:

=>Type3 driver consumes more execution time when compared to Type1 and Type2,because the network components are involved in execution process.

*imp

4.Thin driver(Type4):

=>Thin driver will convert JDBC call to DB Specific call for communication.Which is pure Java driver.



Advantage:

=>which is more efficient and HighPerformance when compared to Type1,Type2 and Type3.

=>This "Thin" driver will not use ODBC driver,No need to install DB Specific Lib in Client Computer and it will not take the support of MiddleWare Server.

Dt : 27/11/2020

Installing DB product and setting the Environment ready for executing

JDBC Applications:

step1 : Download DB Product and Install the product.

step2 : Perform Login process to DB Product

User-name : system

password : manager

step3 : Create table with name "Emp29"

(eid,ename,edesg,esal)

```
create table Emp29(eid varchar2(10),ename varchar2(15),
edesg varchar2(10),esal number(10));
```

step4 : Insert min 5 records into "Emp29"

```
insert into Emp29 values('A121','Raj','SE',30000);
```

step5 : Copy DB Jar file into 'ext' folder of JDK

DB Jar file available at:

G:\app\home\product\11.2.0\dbhome_1\jdbc\lib

Oracle10 - ojdbc14.jar

Oracle11 - ojdbc6.jar

Oracle12 - ojdbc7.jar/ojdbc8.jar

'ext' folder at :

C:\Program Files\Java\jdk1.8.0_251\jre\lib\ext

Note:

=>If 'ext' folder not available then create one user defined folder
in any drive and copy the DB Jar file.

step6 : Find the DB product PortNo and Service_name

=>we can find portNo and serviceName in "tnsnames.ora" file available
in "Admin" folder of "Network" folder.

G:\app\home\product\11.2.0\dbhome_3\NETWORK\ADMIN

Port No : 1522

Service Name : orcl

=====

==

***imp**

Constructing JDBC Application using IDE Eclipse:

step1 :Open IDE Eclipse,while opening name the WorkSpace and click 'ok'

step2 : create Java Project

**Click on 'File'->new->Project->Java->select 'Java Project' and click
'next'->name the project and click 'finish'.**

step3 : Add DB Jar to the Java project.

Right Click on Java Project->Build path->Configure Build path->
Libraries->Add External Jars->Browse and select DB Jar file from 'ext'
folder->Open->Apply->ok

step4 : Create package

Right click on 'src'->new->package, name the package and click finish.

step5 : create class

Right click on package->new->class, name the class and click finish.

**step6 : Type the following Program to display the records from the
table "Emp29".**

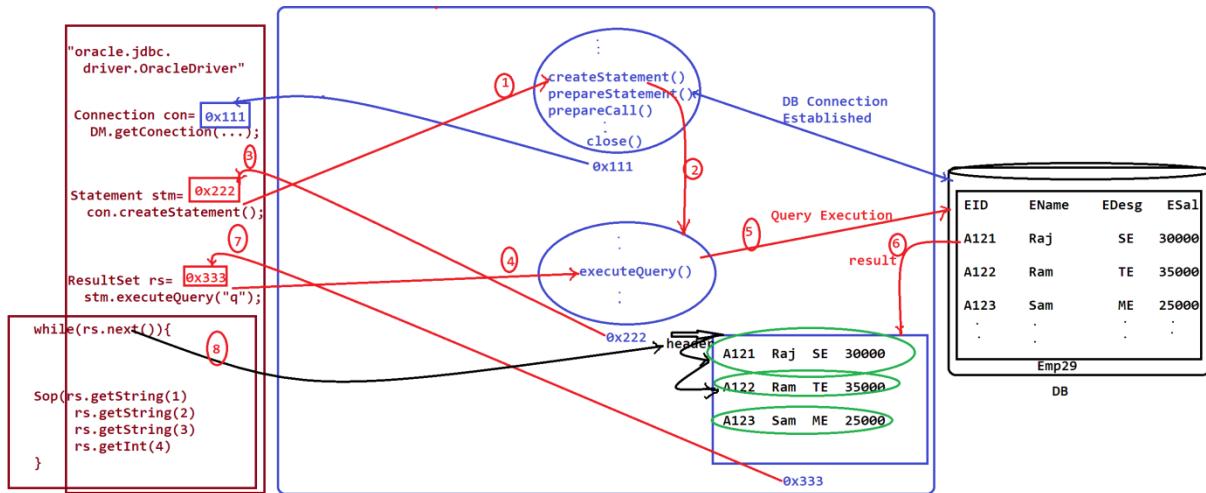
```
package test;
import java.sql.*;
public class DBCon1 {
    public static void main(String[] args) {
        try{
Class.forName("oracle.jdbc.driver.OracleDriver");//step1
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1522:orcl","system","ma
nager");//step2
Statement stm = con.createStatement();//step3
ResultSet rs = stm.executeQuery("select * from
Emp29");//step4
while(rs.next()){
System.out.println(rs.getString(1)+"\t"+rs.getString(
2)+
```

```
    "\t"+rs.getString(3)+"\t"+rs.getInt(4));
}//end of loop
con.close(); //step5
}//end of try
catch(ClassNotFoundException cnfe){
System.out.println("Check the driver...");
}
catch(SQLException se){
System.out.println("Check the connection...");
}
}
```

step7 : Execute the program

Dt : 30/11/2020

Execution flow of above program:(DBCon1.java)



step1 : call `createStatement()` method for execution.

step2 : `createStatement()` method will create the implementation object of 'java.sql.Statement' interface and this object will hold one important method '`executeQuery()`'.

step3 : The object reference is copied on the reference variable.

(`stm=0x222`)

step4 : call `executeQuery()` method for execution.

step5 : This `executeQuery()` method will take query as parameter and executes on DB product.

step6 : This `executeQuery()` method also creates implementation object of 'java.util.ResultSet' interface and which is loaded with query result. In this process one header is pointing before the first row.

step7 : The object reference is copied onto reference variable.

(rs=0x333)

**step8 : we use next() method to move the header record-by-record,in
this proces if record is available means 'true' if nor 'false'.**

=====

faq:

wt is the execution behaviour of getConnection() method?

=>getConnection() method will create the implementation object of
'java.util.Connection' interface and this object will hold DB
Connection details.

=>The following important methods are available from 'Connection'
Object.

- (i)createStatement()
- (ii)prepareStatement()
- (iii)prepareCall()

Assignment(Solutustion):

Create DB Table : Product29

(pcode,pname,pprice,pqty)

Insert records : min 5 records

write a program to display the records

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

```
public class DBCon2 {
    public static void main(String[] args) {
        try{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1522:orcl","system","ma
nager");
Statement stm = con.createStatement();
ResultSet rs = stm.executeQuery("select * from
Product29");
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(ClassNotFoundException cnfe){
System.out.println("Check the Driver class...");

        }catch(SQLException se){
System.out.println("Checl connection process...");

        }
    }
}
```

Dt : 1/12/2020

=>Types of Statements in JDBC:

=>Statements in JDBC 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 on DB Product.
(Queries without IN and OUT parameters).

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

Method Signature:

```
public abstract java.sql.Statement createStatement() throws  
        java.sql.SQLException;
```

syntax:

```
Statement stm = con.createStatement();
```

Exp:

above programs

DBCon1.java

DBCon2.java

*imp

2.PreparedStatement:

=>'PreparedStatement' is an interface from 'java.sql' package and which is used to execute Queries with IN parameters.

(Parameterized queries)

=>we use 'prepareStatement' method from 'java.sql.Connection' interface to create implementation object of 'PreparedStatement'

Method Signature:

```
public abstract java.sql.PreparedStatement prepareStatement  
        (java.lang.String) throws java.sql.SQLException;
```

syntax:

```
PreparedStatement ps = con.prepareStatement("query");
```

Exp program:

DB Table : Book29

```
create table Book29(bcode varchar2(10), bname varchar2(15),  
bauthor varchar2(15), bprice number(10,2), bqty number(10));
```

Write JDBC Appl to read data from Console Input(Keyboard) and insert the data to DB Table.

```
package test;
```

```
import java.util.*;
```

```
import java.sql.*;

public class DBCon3 {

    public static void main(String[] args) {

try{

Scanner s = new Scanner(System.in);

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

String bCode = s.nextLine();

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

String bName = s.nextLine();

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

String bAuthor = s.nextLine();

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

float bPrice = s.nextFloat();

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

int bQty = s.nextInt();

```

```
Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");

PreparedStatement ps = con.prepareStatement

("insert into Book29 values(?,?,?,?,?)");

ps.setString(1,bCode);

ps.setString(2,bName);

ps.setString(3,bAuthor);

ps.setFloat(4,bPrice);

```

```
ps.setInt(5,bQty);

int k = ps.executeUpdate();

if(k>0)

{

System.out.println("Book data inserted Successfully...");

}

con.close();

s.close();

}catch(ClassNotFoundException cnfe){

    System.out.println("Check the JDBC Driver...");

}

catch(SQLException se){

System.out.println("Check the connection or sql statements...");

}

}

=====
```

define primary key?

=>The Unique field among DB Table record is known as Primary Key.

Note:

=>we use the following syntax to add primary key constraint

alter table table_name add primary key(col_name);

Exp:

```
alter table Book29 add primary key(bcode);
```

Note:

=>we use the following syntax to remove primary key constraint

```
alter table table_name drop primary key;
```

Exp:

```
alter table Book29 drop primary key;
```

Assignment1:

DB Table : UserReg29

(uname,pword, fname, lname, addr, phno, mid)

Primary Keys : uname and pword

write JDBC Appl to read data from Console Input and insert into DB Table.

Assignment2:

write JDBC Appl to read data from console input and insert into Emp29.

Assignment3:

**write JDBC Appl to read data from Console input and insert into
Product29.**

Dt : 2/12/2020

Assignment1:(Solution)

```
create table UserReg29(uname varchar2(15),pword varchar2(15),
fname varchar2(15),lname varchar2(15),addr varchar2(15),
phno number(15),mid varchar2(25));
```

Exp program:

```
package test;

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

public class DBCon4 {

    public static void main(String[] args) {

try{
Scanner s = new Scanner(System.in);
System.out.println("Enter the uName:");
String uName = s.nextLine();
System.out.println("Enter the pWord:");
String pWord = s.nextLine();
System.out.println("Enter the fName:");
String fName = s.nextLine();
System.out.println("Enter the lName:");
String lName = s.nextLine();
System.out.println("Enter the addr:");
String addr = s.nextLine();
System.out.println("Enter the phNo:");
String phNo = s.nextLine();

}
}
}
```

```
long phNo = Long.parseLong(s.nextLine());  
  
System.out.println("Enter the mailId:");  
  
String mId = s.nextLine();  
  
  
Class.forName("oracle.jdbc.driver.OracleDriver");  
  
Connection con = DriverManager.getConnection  
("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");  
  
PreparedStatement ps = con.prepareStatement  
("insert into UserReg29 values(?,?,?,?,?,?)");  
  
ps.setString(1,uName);  
  
ps.setString(2,pWord);  
  
ps.setString(3,fName);  
  
ps.setString(4,lName);  
  
ps.setString(5,addr);  
  
ps.setLong(6,phNo);  
  
ps.setString(7,mId);  
  
  
int k = ps.executeUpdate();  
  
if(k>0){  
  
System.out.println("User registered Successfully...");  
  
}  
  
con.close();  
  
s.close();  
  
}catch(ClassNotFoundException cnfe){  
  
System.out.println("Check the driver...");
```

```
}

catch(SQLException sqe){

    System.out.println("Check the connection or SQL Statements..");

}

-----
```

Assignment2(Solution)

```
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 empld:");

String eId = s.nextLine();

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

String eName = s.nextLine();

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

String eDesg = s.nextLine();

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

int eSal = Integer.parseInt(s.nextLine());
```

```
Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");

PreparedStatement ps = con.prepareStatement
("insert into Emp29 values(?,?,?,?,?)");

ps.setString(1,eld);
ps.setString(2,eName);
ps.setString(3,eDesg);
ps.setInt(4,eSal);

int k = ps.executeUpdate();

if(k>0){

System.out.println("EmpData inserted Successfully...");

}

con.close();

s.close();

}catch(ClassNotFoundException cnfe){

System.out.println("Check the driver...");

}catch(SQLException se){

System.out.println("Check the connection or SQL Statements..");

}

}

-----
```

Assignment3:(Solution)

```
package test;

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

public class DBCon6 {

    public static void main(String[] args) {

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

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

String pCode = s.nextLine();

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

String pName = s.nextLine();

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

float pPrice = Float.parseFloat(s.nextLine());

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

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

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");

PreparedStatement ps = con.prepareStatement

("insert into Product29 values(?,?,?,?,?)");

ps.setString(1,pCode);

ps.setString(2,pName);

ps.setFloat(3,pPrice);
```

```
ps.setInt(4,pQty);

int k = ps.executeUpdate();

if(k>0){

System.out.println("Product data inserted Successfully...");

}

con.close();

s.close();

}catch(ClassNotFoundException cnfe){

System.out.println("Check the driver...");

}

}catch(SQLException se){

System.out.println("Check the connection or SQL Statements..");

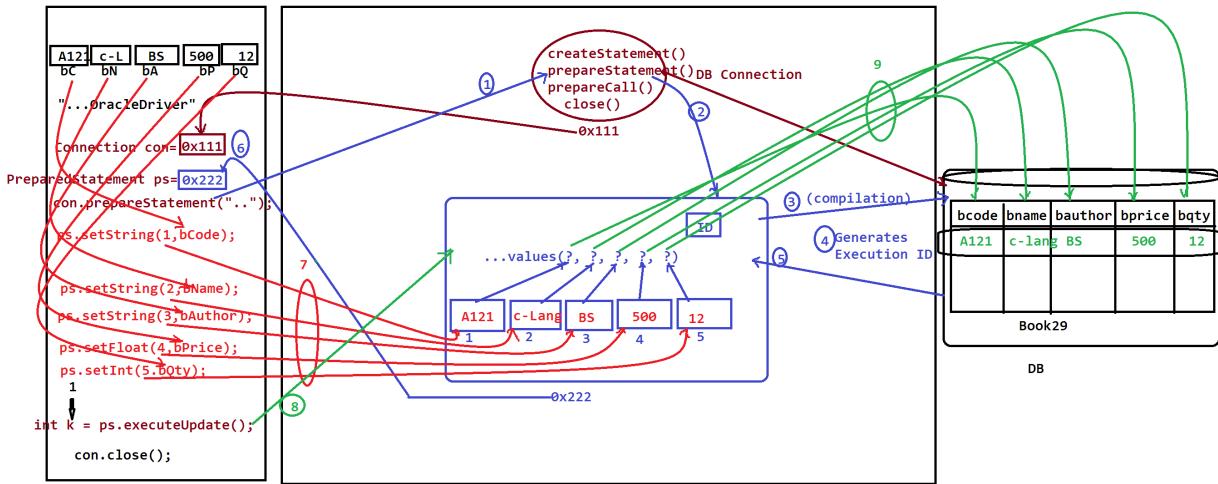
}

}

-----
```

Dt : 3/12/2020

Execution flow of above program:(DBCon3.java)



step1 : call prepareStatement() method for execution.

step2 : This prepareStatement() method will create implementation

object of 'java.sql.PreparedStatement' interface.

**step3 : In this process the prepareStatement() method also compiles
the query-structure on DB Product.**

step4 : If the compilation of query-structure is successfull,then

'generates Execution id'

**step5 : This 'Execution Id' is stored in PreparedStatement object and
also generates Fields which are equal to the number of Index
parameters.**

**step6 : The reference of PreparedStatement object is loaded on to
reference variable(ps).**

step7 : Using setter method we set the data-values to the fields.

step8 : call executeUpdate() method for execution.

**step9 : This executeUpdate() method will execute query on DB product
with values.**

**step10: This executeUpdate() method will return int value which is
equal to the number of rows updated.**

Dt : 4/12/2020

Exp program:

Write JDBC App to display BookDetails based on bCode?

```
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 bookCode:");
String bCode = s.nextLine();

Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");

PreparedStatement ps = con.prepareStatement
("select * from Book29 where bcode=?");

ps.setString(1,bCode);

ResultSet rs = ps.executeQuery();
if(rs.next()){

System.out.println(rs.getString(1)+"\t"+rs.getString(2)+"\t"+
rs.getString(3)+"\t"+rs.getFloat(4)+"\t"+rs.getInt(5));

}//end of if

else{
```

```

System.out.println("Invalid bookCode...");
}

con.close();

s.close();

}catch(ClassNotFoundException cnfe){

    System.out.println("Check the Driver class...");

}catch(SQLException se){

}

```

```

System.out.println("Check the Connection or SQL Statements...");

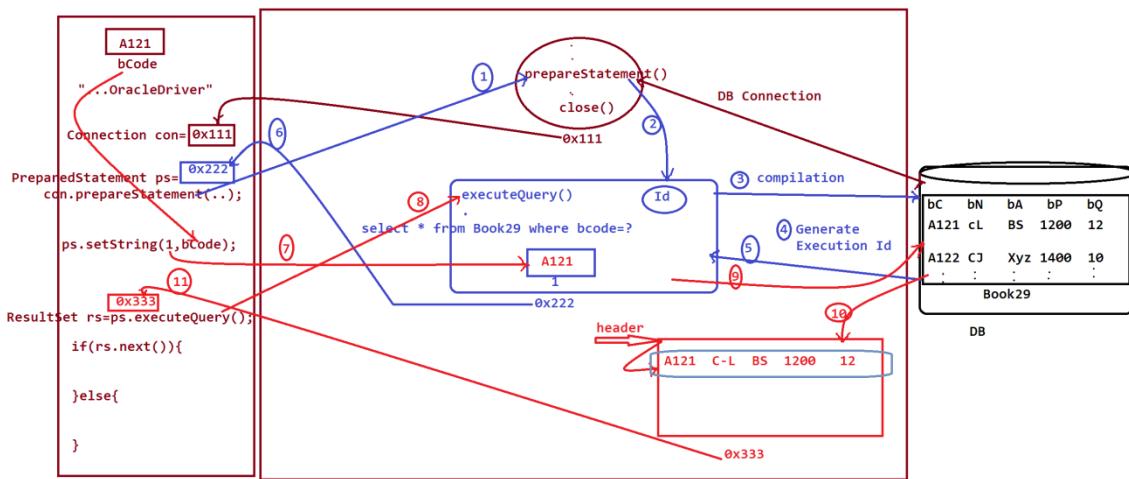
}

}

}

```

Execution flow of above program:



Assignment1:

JDBC App to display Employee data based on emplId?

Assignment2:

JDBC App to display Product details based on pCode?

Assignment3:

JDBC App to display User details based on uName and pWord?

Assignment4:

DB Table : Student29

(rNo,sname,branch,totMarks,per,result)

JDBC App to insert the data to DB table Student29.

Read(I/P):

rollNo

StuName

Branch

six sub marks

Calculate

totMarks

per

result

JDBC App to display Student details based on rNo?

=====

Dt : 5/12/2020

SQL Queries:

Create

Insert

Retrive(select)

Update

Delete

Ex program:

wap to updtae qty based on the bCode?

```
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 bookCode:");  
            String bCode = s.nextLine();  
  
            Class.forName("oracle.jdbc.driver.OracleDriver");  
            Connection con = DriverManager.getConnection  
                ("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");  
            PreparedStatement ps1 = con.prepareStatement  
                ("select * from Book29 where bcode=?");  
            ps1.setString(1,bCode);  
            ResultSet rs = ps1.executeQuery();
```

```
if(rs.next()){

PreparedStatement ps2 = con.prepareStatement
("update Book29 set bqty=bqty+? where bcode=?");

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

int qty = s.nextInt();

ps2.setInt(1,qty);

ps2.setString(2,bCode);

int k = ps2.executeUpdate();

if(k>0){

System.out.println("Book qty updated for bCode:"+bCode);

}

}else{

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

}

con.close();

s.close();

}catch(ClassNotFoundException cnfe){

System.out.println("Check the Driver Class...");

}catch(SQLException se){

System.out.println("Check the connection or SQLStatements...");

}

}
```

Exp program:

wap to delete book details based on bcode?

```
package test;

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

public class DBCon9 {

    public static void main(String[] args) {

try{

Scanner s = new Scanner(System.in);

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

String bCode = s.nextLine();

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");

PreparedStatement ps1 = con.prepareStatement

("select * from Book29 where bcode=?");

ps1.setString(1,bCode);

ResultSet rs = ps1.executeQuery();

if(rs.next()){

PreparedStatement ps2 = con.prepareStatement

("delete from Book29 where bcode=?");

ps2.setString(1,bCode);

int k = ps2.executeUpdate();

if(k>0){

System.out.println("Book Details deleted for bCode:"+bCode);
}
}
}
}
}
```

```
        }

    }else{
        System.out.println("Invalid bookCode...");
    }

    con.close();
    s.close();

}catch(ClassNotFoundException cnfe){
    System.out.println("Check the Driver Connection..");

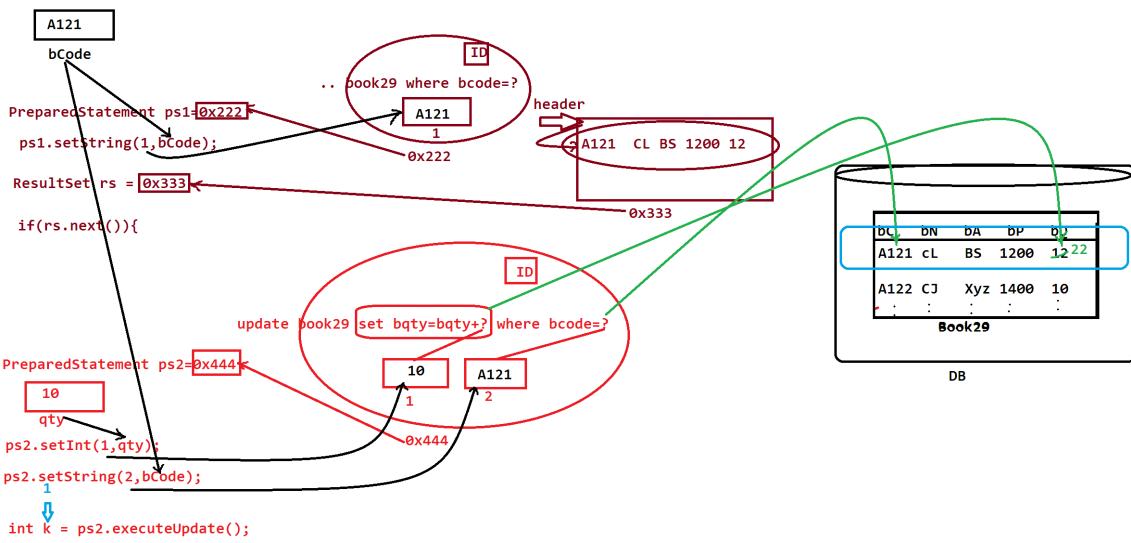
}catch(SQLException se){
    System.out.println("Check the connection or SQL Statements...");
}

}

=====
```

Dt : 7/12/2020

Execution flow of above program:(DBCon8.java)



Note:

=>In JDBC programs 'commit' operation is performed automatically after query execution Completed.

faq:

wt is the diff b/w

- (i)executeQuery()
- (ii)executeUpdate()

(i)executeQuery():

=>executeQuery() method is used to retrieve data from DB Table.

Exp:

select queries

Method Signature of executeQuery() from PreparedStatement:

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

(ii)executeUpdate():

**=>executeUpdate() method is used when we want to make changes in
the DB Table**

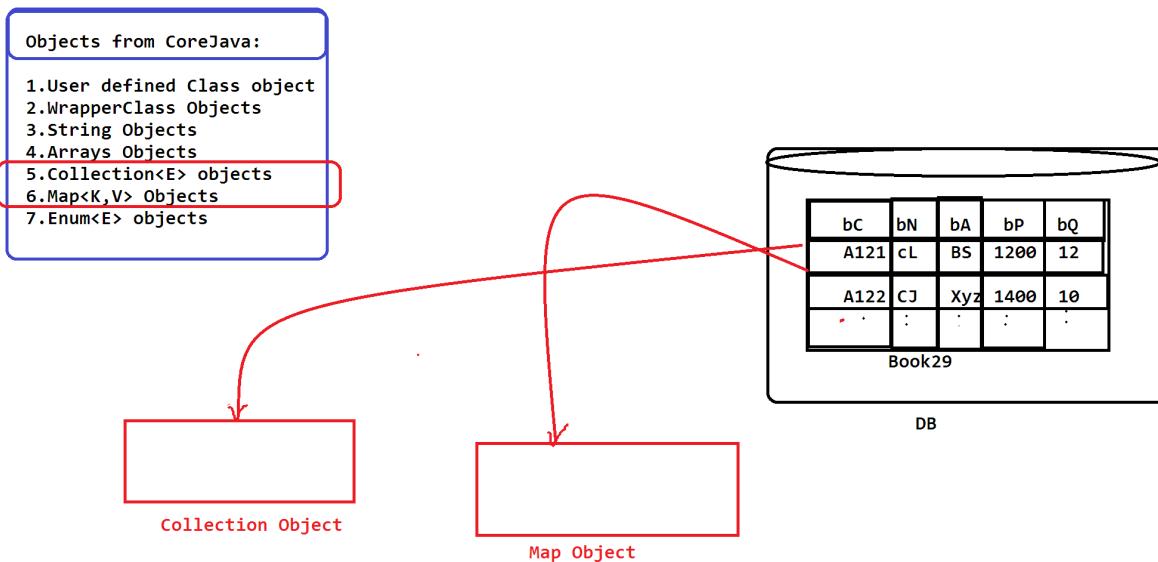
Exp:

Create,insert,update and delete queries

Method Signature of executeUpdate() from PreparedStatement:

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

DT : 8/12/2020



Collection<E> object holding DB Table data:

Colection<E>

=>Collection<E> is an interface from 'java.util' package and which is extended to the following SunInterfaces:

(a)Set<E>

(b)List<E>

(c)Queue<E>

(a)Set<E>

=>Set<E> organizes elements without index values and cannot hold duplicate elements.

=>The following are implementations of Set<E>:

(i)HashSet<E>

=>HashSet<E> organizes elements without any order.

(ii)LinkedHashSet<E>

=>**LinkedHasjSet<E>** organizes elements with insertion order.

(iii)TreeSet<E>

=>**TreeSet<E>** organizes elements in Ascending order.

(b)List<E>

=>**List<E>** organizes elements based on index values and can hold
duplicate elements.

=>The following are the implementations of **List<E>**:

(i)ArrayList<E>

=>**ArrayList<E>** organizes elements in Sequence and
NonSynchronized class

(ii)LinkedList<E>

=>**LinkedList<E>** organizes elements in NonSequence and
NonSynchronized class

(iii)Vector<E>

=>**Vector<E>** organizes elements in Sequence and synchronized
calss

Note:

=>**Stack<E>** organizes elements based on algorithm

First-In-Last-out and which is childClass of Vector<E>

(c)Queue<E>

=>Queue<E> organizes elements based on the algorithm

First-In-First-Out

=====

Note:

=>In the process of holding DB Table data using Collection<E> object,
we declare User defined class having the variables equal to the
cols of DB Table.

BookData.java

```
package test;
public class BookData {
    public String bCode,bName,bAuthor;
    public float bPrice;
    public int bQty;
    public String getbCode() {
        return bCode;
    }
    public void setbCode(String bCode) {
        this.bCode = bCode;
    }
    public String getbName() {
        return bName;
    }
    public void setbName(String bName) {
        this.bName = bName;
    }
    public String getbAuthor() {
        return bAuthor;
    }
    public void setbAuthor(String bAuthor) {
        this.bAuthor = bAuthor;
    }
}
```

```
public float getbPrice() {  
    return bPrice;  
}  
public void setbPrice(float bPrice) {  
    this.bPrice = bPrice;  
}  
public int getbQty() {  
    return bQty;  
}  
public void setbQty(int bQty) {  
    this.bQty = bQty;  
}  
}
```

define 'Getter methods'?

=>The methods which are used to get the data from the objects.

define 'Setter methods'?

=>The methods which are used to set the data in to the objects.

Note:

=>To Generate Setters and Getters methods, Right Click in Editor->

Source->Generate Getters and Setters->Click on 'Select All' and

click ok.

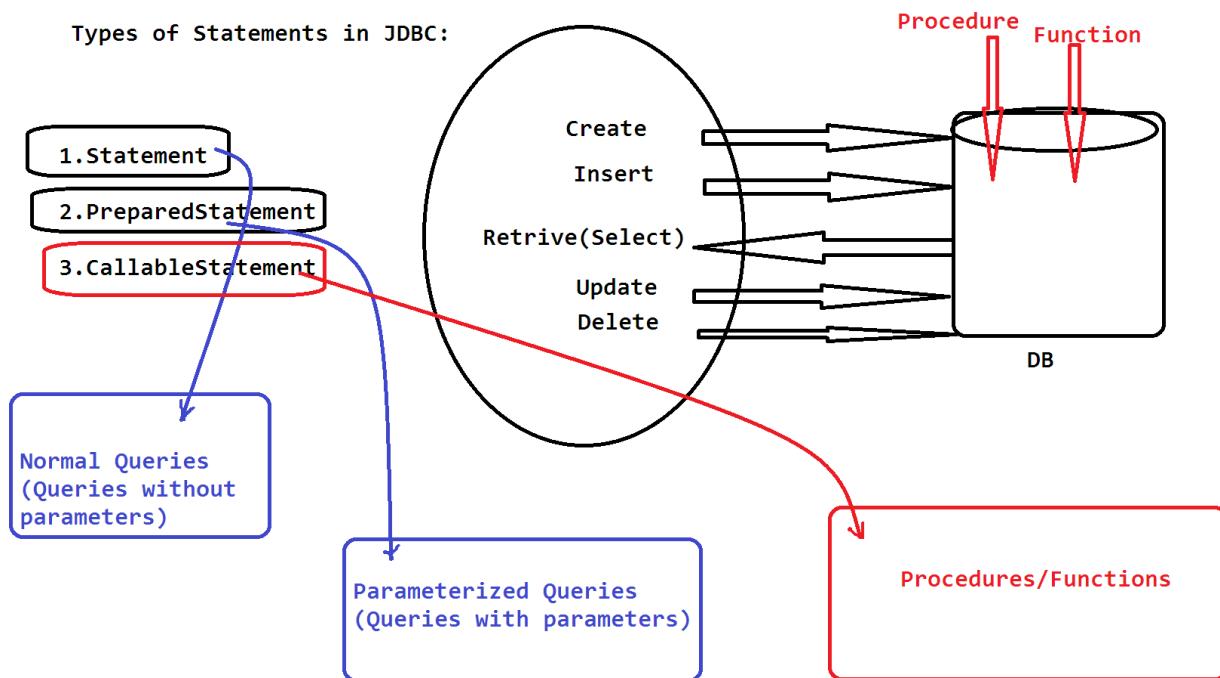
DBCon10.java(MainClass)

```
package maccess;
```

```
import java.util.*;
import java.sql.*;
import test.BookData;
public class DBCon10 {
    public static void main(String[] args) {
try{
ArrayList<BookData> al = new ArrayList<BookData>();
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");
PreparedStatement ps = con.prepareStatement
("select * from Book29");
ResultSet rs = ps.executeQuery();
while(rs.next()){
BookData bd = new BookData();
bd.setbCode(rs.getString(1));
bd.setbName(rs.getString(2));
bd.setbAuthor(rs.getString(3));
bd.setbPrice(rs.getFloat(4));
bd.setbQty(rs.getInt(5));
al.add(bd);
}
//end of loop
System.out.println("==Display BookDetails===");
al.forEach((k)->
```

```
{  
    BookData bd = (BookData)k;  
    System.out.println(bd.getbCode()+"\t"+bd.getbName()+"\t"+  
        bd.getbAuthor()+"\t"+bd.getbPrice()+"\t"+bd.getbQty());  
};  
}catch(ClassNotFoundException cnfe){  
    System.out.println("Check the Driver class...");  
}catch(SQLException se){  
    System.out.println("Check the Connection or SQL Statements...");  
}  
}  
}
```

Dt : 14/12/2020



3.CallableStatement:

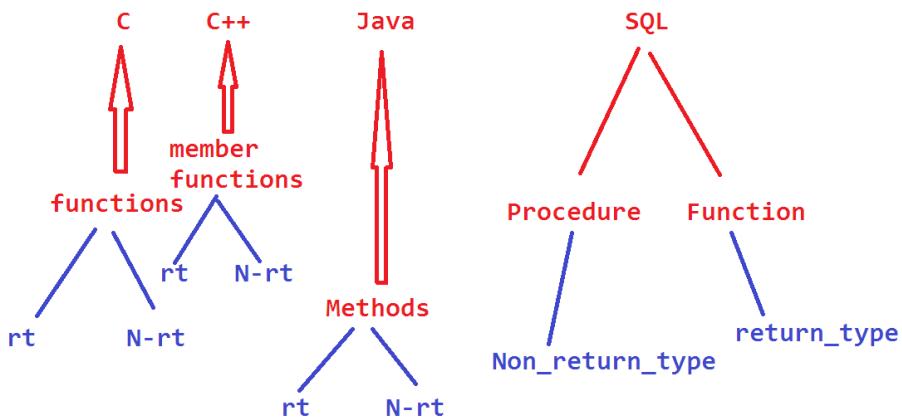
=>'CallableStatement' is an interface from `java.sql` package and which is used to execute Procedures and Functions from the DB Product.
=>we use `prepareCall()` method from `java.util.Connection` interface to create implementation object of 'CallableStatement'.

Method Signature:

```
public abstract java.sql.CallableStatement prepareCall  
(java.lang.String) throws java.sql.SQLException;
```

syntax:

```
CallableStatement cs = con.prepareCall("proc/func");
```



define Procedure?

=>The set-of-queries executed on DB Storage and which will not return any value is known as Procedure.

Structure of Procedure:

```
create or replace procedure proc_name
```

```
(para_list) is
```

```
begin
```

```
...
```

```
...
```

```
end;
```

```
/
```

define Function?

=>The set-of-queries executed on DB Storage and which will return the value is known as Function.

Structure of Function:

```
create or replace function Func_name  
(para_list) return data_type as var data_type;  
begin  
...  
return var;  
end;  
/  
-----
```

Example:

Creating and Executing Procedure:

Step1 : Create the following DB Tables from SQL CommandLine:

```
create table Bank29(AccNo number(15) NOT NULL,  
CustName varchar2(15),bal number(10,2),  
AccType varchar2(15),primary key(AccNo));
```

```
create table CustDetails29(AccNo number(15) NOT NULL,  
addr varchar2(15),phno number(15),primary key(AccNo));
```

Step2 : Construct Procedure from SQL CommandLine to insert the data to DB table.

```

create or replace procedure CreateAccount29
(a number,b varchar2,c number,d varchar2,e varchar2,f number) is
begin
insert into Bank29 values(a,b,c,d);
insert into CustDetails29 values(a,e,f);
end;
/

```

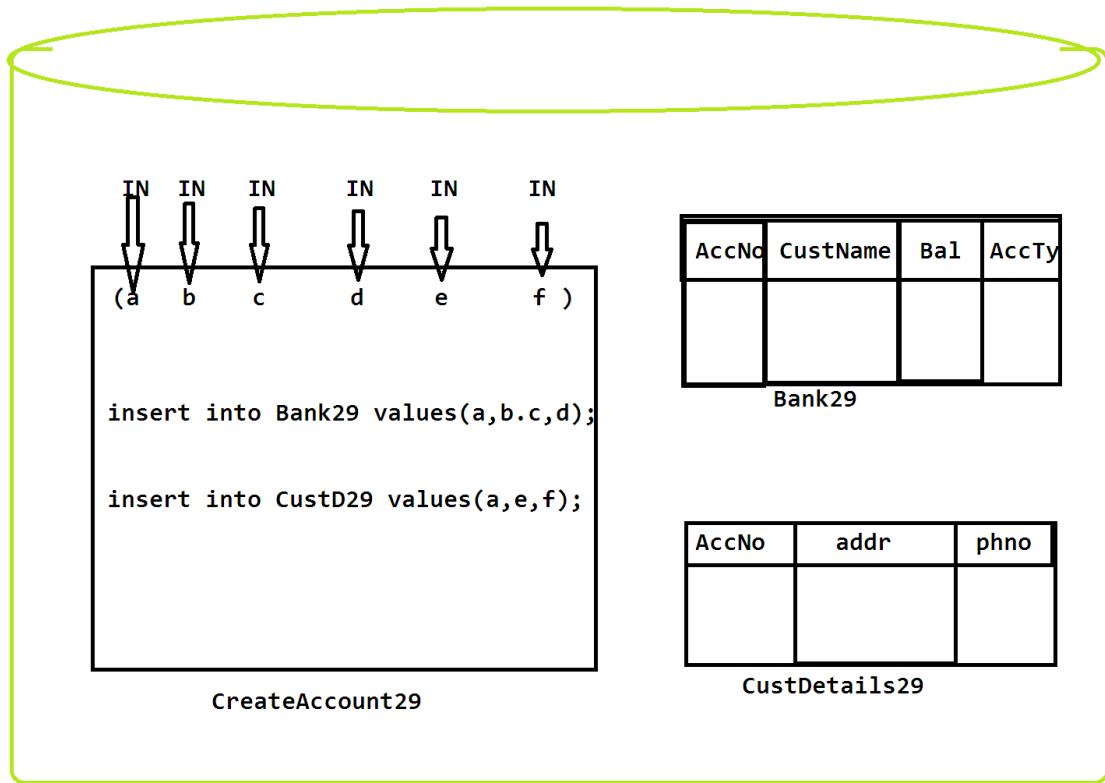
Step3 : Construct JDBC Application to execute Procedure to insert data to DB tables.

```

package test;
import java.util.*;
import java.sql.*;
public class ProceApp1 {
    public static void main(String[] args) {
try{
Scanner s = new Scanner(System.in);
System.out.println("Enter the accNo:");
long accNo = Long.parseLong(s.nextLine());
System.out.println("Enter the CustName:");
String custName = s.nextLine();
System.out.println("Enter the balance:");
float bal = Float.parseFloat(s.nextLine());
System.out.println("Enter the accType:");
String accType = s.nextLine();
System.out.println("Enter the address:");
String addr = s.nextLine();
System.out.println("Enter the phNo:");
long phNo = Long.parseLong(s.nextLine());

```

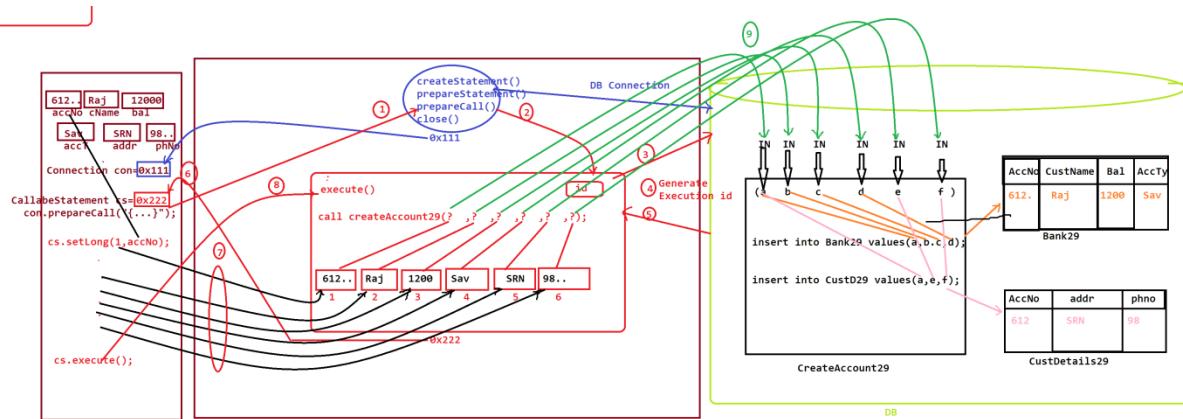
```
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1522:orcl","system","ma
nager");
CallableStatement cs = con.prepareCall
("{call CreateAccount29(?,?,?,?,?,?)}");
cs.setLong(1,accNo);
cs.setString(2,custName);
cs.setFloat(3,bal);
cs.setString(4,accType);
cs.setString(5,addr);
cs.setLong(6,phNo);
cs.execute();
System.out.println("Procedure executed
Successfully...");
con.close();
s.close();
}catch(ClassNotFoundException cnfe){
    System.out.println("Check the driver...");
}catch(SQLException se){
System.out.println("Check the Connection or SQL
Statements...");
}
}
```



DB

Dt : 15/12/2020

Execution flow of above program:



step1 : Call `prepareCall()` method for execution.

step2 : This `prepareCall()` method will create the implementation object of `CallableStatement`.

step3 : In this process the procedure call structure is compiled on the DB Product.

step4 : If the compilation is successfull the execution id is generated.

step5 : This execution id is stored within the `CallableStatement` object,in this process data-fields are generated equal to the number of parameter-indexes.

step6 : The Object reference is loaded onto reference variable(`cs`)

step7 : Using 'Setter methods' we set the data to the data-fields.

step8 : we call `execute()` method for execution.

step9 : This `execute()` method will execute the procedure with values

and update the DB Tables. This execute method returns boolean result.

syntax of viewing the procedure from SQL CommandLine:

`select text from all_source where name='Proc_name';`

Assignment1:

DBTables:

UserDetails29

`(uname,pword, fname, lname, age)`

UserAddress29

`(uname,pword,addr,phno,mid)`

create procedure to update two tables

Assignment2:

DBTables:

EmpDetails29(eId,eName,eDesg,bSal)

EmpAddress29(eId,sName,city,pincode)

create procedure to update two table

=>In JDBC, procedures are categorized into two type:

(i)IN Parameter procedures

(ii)OUT Parameter procedures

(i)IN Parameter procedures:

=>The procedures which take the data from JavaProgram and update DB Tables is known as In Parameter Procedures.

Exp:

above programs.

(ii)OUT Parameter procedures:

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

Exp program:

step1 : Procedure to retrive CustData based on accNo.

```
create or replace procedure RetrieveAccount29
(a number,b OUT varchar2,c OUT number,d OUT varchar2,e OUT varchar2,f OUT
number) is
begin
select custname,bal,acctype into b,c,d from bank29 where accno=a;
select addr,phno into e,f from custdetails29 where accno=a;
end;
/
```

step2 : JDBC Application to execute retrieve procedure.

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

```
import java.sql.*;

public class ProcApp2 {

    public static void main(String[] args) {

try{

Scanner s = new Scanner(System.in);

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

long accNo = Long.parseLong(s.nextLine());

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");

PreparedStatement ps = con.prepareStatement

("select * from Bank29 where accno=?");

ps.setLong(1,accNo);

ResultSet rs = ps.executeQuery();

if(rs.next()){

CallableStatement cs = con.prepareCall

("{call RetrieveAccount29(?,?,?,?,?,?)}");

cs.setLong(1,accNo);

cs.registerOutParameter(2,Types.VARCHAR);

cs.registerOutParameter(3, Types.FLOAT);

cs.registerOutParameter(4,Types.VARCHAR);

cs.registerOutParameter(5,Types.VARCHAR);

cs.registerOutParameter(6, Types.BIGINT);

cs.execute();

System.out.println("AccNo:"+accNo);

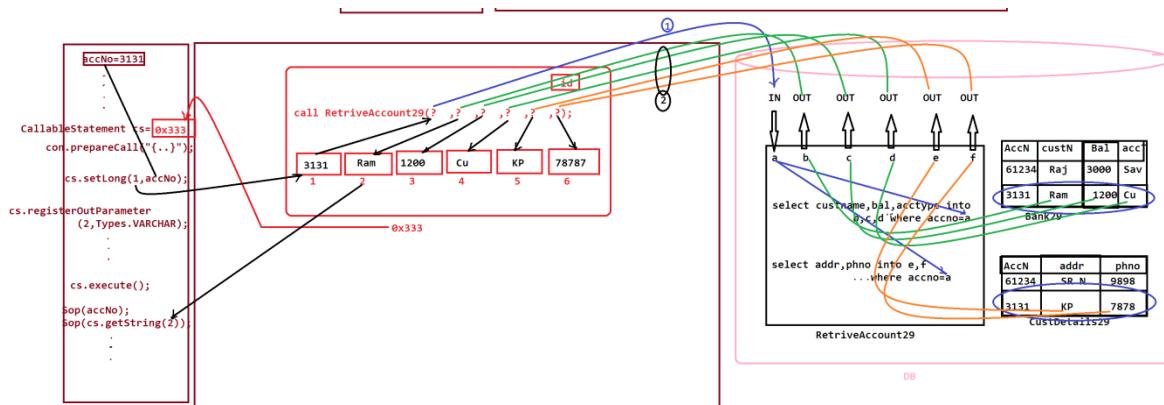
}
```

```
System.out.println("CustName:"+cs.getString(2));
System.out.println("Balance:"+cs.getFloat(3));
System.out.println("AccType:"+cs.getString(4));
System.out.println("Address:"+cs.getString(5));
System.out.println("PhoneNo:"+cs.getLong(6));
}else{
System.out.println("InValid AccNo...");
}
con.close();
s.close();
}catch(ClassNotFoundException cnfe){
    System.out.println("Check the driver...");
}catch(SQLException se){
    System.out.println("Check the connection or SQL Statements...");
}
}

-----
```

Dt : 16/12/2020

Execution flow of above program:



faq:

define registerOutParameter() method?

=>registerOutParameter() method will specify the type of data to be recorded within the data-fields.

=>This method is available from 'java.sql.CallableStatement' interface.

Method Signature:

```
public void registerOutParameter(int,java.sql.SQLType) throws  
java.sql.SQLException;
```

faq:

define 'Types' in JDBC?

=>'Types' is a class from java.sql package and which holds SQLTypes like.

VARCHAR

FLOAT

INT

...

Assignment1:

create procedure to retrieve UserData based on uname and pword.

Assignmnet2:

create procedure to retrieve EmpData based on eid.

Example program to demonstrate Funtion:

step1 : Create Function to retrieve balance based on accNo

```
create or replace function getBalance29  
(a number) return number as amt number;  
begin  
select bal into amt from bank29 where accno=a;  
return amt;  
end;  
/
```

step2:write JDBCAppI to execute the function

syntax:

```
CallableStatement cs = con.prepareCall
```

```
("'{call ?:=getBalance29(?)}');
```

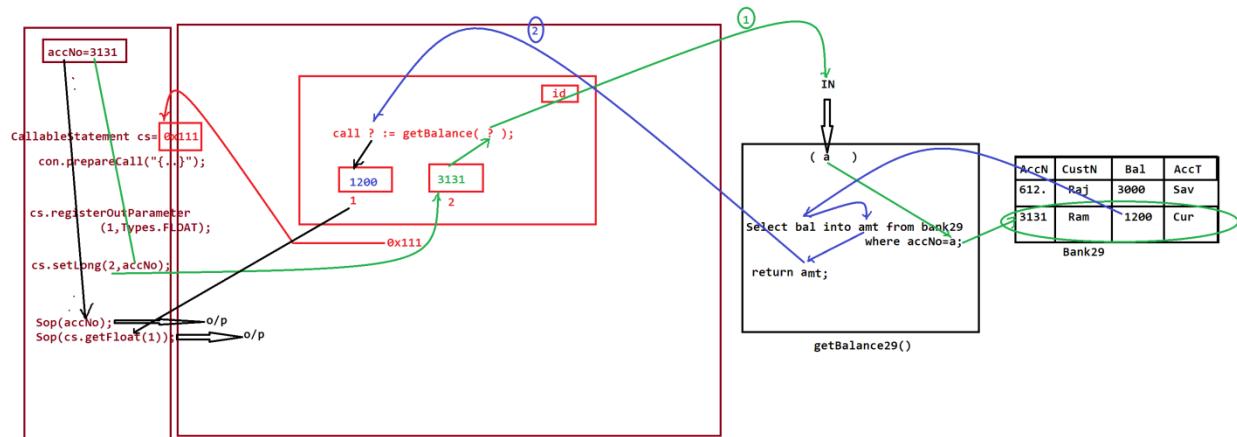
```
package test;
import java.util.*;
import java.sql.*;
public class FuncApp1 {
    public static void main(String[] args) {
try{
Scanner s = new Scanner(System.in);
System.out.println("Enter the accNo:");
long accNo = Long.parseLong(s.nextLine());
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1522:orcl","system","ma
nager");
PreparedStatement ps = con.prepareStatement
("select * from bank29 where accno=?");
ps.setLong(1,accNo);
ResultSet rs = ps.executeQuery();
if(rs.next()){
CallableStatement cs = con.prepareCall
("'{call ?:=getBalance29(?)}'");
cs.registerOutParameter(1,Types.FLOAT);
cs.setLong(2,accNo);
cs.execute();
System.out.println("AccNo:"+accNo);
System.out.println("Balance:"+cs.getFloat(1));
}else{
System.out.println("Invalid accNo...");
```

```
con.close();
s.close();
}catch(ClassNotFoundException cnfe){
    System.out.println("Check the driver...");
}catch(SQLException se){
System.out.println("Check the connection or
SQLStatements..");
}
}

}
```

Dt 17/12/2020

Execution flow of above program:



faq:

wt is the advantage of Procedures and Functions?

=>when we use Procedures or functions,then execution control is transferred to DB Product only once and executes all the queries of Procedure or Function.In this process the execution time is saved and generates HighPerformance.

Note:

=>Functions are used when we want to return single value,if not we use Procedures.

=>In realtime Functions are less used when compared to Procedures.

=====

Assignment1:

DBTables:

StuDetails29(rNo,stuName,branch,phno,mid)

StuAddress29(rNo,sName,city,state,pinCode)

StuResult29(rNo,totMarks,per,result)

Read from JavaApp:

rNo,stuName,branch,phNo,mId,sName,city,state,

pinCode and 6 sub marks.

Note:

Calculate totMarks,per and grade(result)

per>=70 && per<=100 : Distinction

per>=60 && per<70 : First Class

per>=50 && per<60 : Second Class

per>=35 && per<50 : Third Class

else : Fail

Note:

|->UpDate the above 3 tables using procedure.

Assignment2:

Procedure to retrieve Stu data(details) based on rNo.

MetaData in JDBC:

define MetaData?

=>The data which is holding information about other data
is known as MetaData.

Note:

=>According to JDBC,MetaData means Object holding information about
another object.

In JDBC,MetaData components are categorized into the following:

- (i)DatabaseMetaData
- (ii)ParameterMetaData
- (iii)ResultSetMetaData

(i)DatabaseMetaData:

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

syntax:

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

(ii)ParameterMetaData:

=>ParameterMetaData is also an interface from java.sql package and

which is used to hold information about java.sql.PreparedStatement object.

syntax:

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

(iii)ResultSetMetaData:

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

syntax:

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

Exp program:

```
package test;
import java.sql.*;
public class MetaData {
    public static void main(String[] args) {
        try{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1522:orcl","system","ma
nager");
DatabaseMetaData dmd = con.getMetaData();
System.out.println("DriverName:"+dmd.getDriverName())
;
PreparedStatement ps1 = con.prepareStatement
("insert into book29 values(?,?,?,?,?,?)");
ps1.setString(1,"A1100");
```

```

ps1.setString(2, "UUU");
ps1.setString(3, "TTT");
ps1.setFloat(4, 123.45F);
ps1.setInt(5, 23);
int k = ps1.executeUpdate();
ParameterMetaData pmd = ps1.getParameterMetaData();
System.out.println("Number of
para:" + pmd.getParameterCount());
PreparedStatement ps2 = con.prepareStatement
("select * from Book29");
ResultSet rs = ps2.executeQuery();
ResultSetMetaData rsmd = rs.getMetaData();
System.out.println("Column
Count:" + rsmd.getColumnCount());
con.close();
}catch(ClassNotFoundException cnfe){
cnfe.printStackTrace();
}catch(SQLException se){
se.printStackTrace();
}
}
}

```

DT : 18/12/2020

=>Based on control over the header(Pointer) the ResultSet objects are categorized into two types:

(1)Non-scrollable ResultSet Object

(2)Scrollable ResultSet Object

(1)Non-scrollable ResultSet

=>In Non-Scrollable ResultSet objects the header(pointer) is moved only in one direction.(top of the table data to bottom of the Table data)

=>By default ResultSet objects are Non-Scrollable objects.

Exp:

above programs

***imp**

(2)Scrollable ResultSet Object:

=>In Scrollable ResultSet objects the pointer can be moved in both directions.

Syntax for Creating Scrollable ResultSet object:

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

PreparedStatement ps = con.prepareStatement("query",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 pointer and 'mode' specifies the action to be performed(read or update).

The following are some Methods used to control Pointer 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.

Exp program:

```
package test;
import java.sql.*;
public class ScrollApp {
    public static void main(String[] args) {
try{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1522:orcl","system","ma
nager");
Statement stm = con.createStatement(1004,1007);
ResultSet rs1 = stm.executeQuery("select * from
Product29");
```

```

System.out.println("====Display the records in
reverse===");
rs1.afterLast();
while(rs1.previous()){

System.out.println(rs1.getString(1)+"\t"+rs1.getString(2)+"\t"+
rs1.getFloat(3)+"\t"+rs1.getInt(4));
}//end of loop
PreparedStatement ps = con.prepareStatement
("select * from Product29",1004,1007);
ResultSet rs2 = ps.executeQuery();
System.out.println("====Display 3rd row===");
rs2.absolute(3);
System.out.println(rs2.getString(1)+"\t"+rs2.getString(2)+"\t"+
rs2.getFloat(3)+"\t"+rs2.getInt(4));
rs2.relative(1);
System.out.println(rs2.getString(1)+"\t"+rs2.getString(2)+"\t"+
rs2.getFloat(3)+"\t"+rs2.getInt(4));
con.close();
}catch(ClassNotFoundException cnfe){
    cnfe.printStackTrace();
}catch(SQLException se){
    se.printStackTrace();
}
}

```

Batch Processing in JDBC:

=>The process of collecting multiple queries as a batch and executing together on DB Storage, is known as Batch Processing in JDBC.

The following are the Steps used in BatchProcessing:

- 1.Loading Driver**
- 2.creating connection**
- 3.preparing statement**
- 4.add query to the batch**
- 5.execute batch**
- 6.close the connection**

The following are the Methods used in BatchProcessing:

(1)addBatch(String SQLStatement)

=>Collecting the Query under batch.

(2)clearBatch()

=>Remove all the Queries from the batch.

(3)int[] executeBatch()

=>which executes the batch on database

Dt : 19/12/2020

Case-1: Batch Process using "Statement":

=>In Batch Processing Using "Statement",we can execute multiple queries as a batch and can update multiple DB Tables.

Exp program:

```
package test;
import java.sql.*;
public class BatchProc1 {
    public static void main(String[] args) {
try{
Class.forName("oracle.jdbc.driver.OracleDriver");
```

```

Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1522:orcl","system","ma
nager");
Statement stm = con.createStatement();
stm.addBatch("insert into Book29
values('A333','zzz','ddd',1300,12)");
stm.addBatch("insert into Product29
values('A126','GG',234,10)");
int k[] = stm.executeBatch();
for(int i=0;i<k.length;i++){
System.out.println("data inserted...");
}//end of loop
con.close();
}catch(Exception e){e.printStackTrace();}
}

```

=====

Case-2 : Batch Processing using "PreparedStatement":

=>In Batch Processing using "PreparedStatement", we can update single DB Table with multiple records(rows).

Ex program:

```

package test;
import java.sql.*;
public class BatchProc1 {
    public static void main(String[] args) {
try{

```

```

Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1522:orcl","system","ma
nager");
Statement stm = con.createStatement();
stm.addBatch("insert into Book29
values('A333','zzz','ddd',1300,12)");
stm.addBatch("insert into Product29
values('A126','GG',234,10)");
int k[] = stm.executeBatch();
for(int i=0;i<k.length;i++){
System.out.println("data inserted...");
}//end of loop
con.close();
}catch(Exception e){e.printStackTrace();}
}

```

=====
faq:

wt is the advantage of Batch Processing?

=>when we use batch Processing,the execution control is transferred from JavaProgram to DB Storage only once, and all the queries are executed at-a-time on DB Storage,in this process execution time is saved and the appl generates high performance.

Limitation of Batch Processing:

|=>Using Batch processing we can perform only Batch update operations ,because of this reason it can also be known as "Batch Update processing".

faq:

wt is the diff b/w

(i)Batch Processing

(ii)Procedures

(i)Using Batch processing we can perform only update operations.

(ii)Using procedures we can perform both Update and retrive operations.

=====

***imp**

Transaction Management in JDBC:

define Transaction?

Transaction is a 'set of statements' executed on a resource or resources,by applying ACID properties.

A-Atomicity

C-Consistency

I-Isolation

D-Durability:

A-Atomicity

=>The process of Commiting the Transaction after completing all the "SubStatements execution",is known as Automicity.

C-Consistency

=>The resources which are selected for Transaction will be same until the transaction is completed is known as Consistency.

I-Isolation

=>The process of running the Transaction independently is known as Isolation.

D-Durability:

=>Once the transaction is committed, the details of Transaction is stored and available for the user, is known as Durability.

Ex Transaction:

Ticket Booking in BookMyShow(Transaction)

(i)Login

(ii)availability

(iii)Selection

(iv)payment

(i)Net/D/C

(ii)Bank selection

(iii)Authentication

(iv)Bal Availability

(v)payment process

(v)Update the database

(vi)Commit the transaction

(vii)Logout

define Transaction management?

=>The process of controlling the transaction from Starting to ending(commit),is known as Transaction management.

=>The following methods are used in Transaction Management process:

(i) **setAutoCommit()**

(ii) **setSavePoint()**

(iii) **commit()**

(iv) **rollback()**

Note:

=>All these methods are available from Connection object(con).

=>To perform Transaction Management process in JDBC we must set "autocommit" false(off),because the javaApi performs commit process automatically.

Method Signatures:

public abstract void setAutoCommit(boolean)

throws java.sql.SQLException;

public abstract java.sql.Savepoint setSavepoint()

throws java.sql.SQLException;

public abstract void commit() throws java.sql.SQLException;

public abstract void rollback() throws java.sql.SQLException;

Dt : 21/12/2020

Exp program:

DB Table : LogTab29

```
create table LogTab29(accNo number(15),log BLOB,primary key(accNo));
```

TransLog.java

```
package test;

import java.util.Date;

import java.io.*;

@SuppressWarnings("serial")

public final class TransLog implements Serializable{

    private final long hAccNo,bAccNo;

    private final float amt;

    public final Date d;

    public TransLog(long hAccNo,long bAccNo,float amt,Date d){

        this.hAccNo=hAccNo;

        this.bAccNo=bAccNo;

        this.amt=amt;

        this.d=d;

    }

    public final long gethAccNo() {

        return hAccNo;

    }

    public final long getbAccNo() {

        return bAccNo;

    }

    public final float getAmt() {

        return amt;

    }

}
```

```
}
```

```
public final Date getD() {
```

```
    return d;
```

```
}
```

Transaction.java

```
package test;
```

```
import java.util.*;
```

```
import java.sql.*;
```

```
import java.io.*;
```

```
public class Transaction {
```

```
    public static void main(String[] args) {
```

```
        try{
```

```
            Scanner s = new Scanner(System.in);
```

```
            Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
            Connection con = DriverManager.getConnection
```

```
("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");
```

```
            PreparedStatement ps1 = con.prepareStatement
```

```
("select * from Bank29 where accno=?");
```

```
            PreparedStatement ps2 = con.prepareStatement
```

```
("update Bank29 set bal=bal+? where accno=?");
```

```
            con.setAutoCommit(false);//step1
```

```
            Savepoint s1 = con.setSavepoint();//step2
```

```
            System.out.println("Enter the accNo:(Home)");
```

```
            long accNo1 = Long.parseLong(s.nextLine());
```

```
            ps1.setLong(1,accNo1);
```

```
ResultSet rs1 = ps1.executeQuery();

if(rs1.next()){

float bal = rs1.getFloat(3);

System.out.println("Enter bAccNo:(Benificiery)");

long bAccNo = Long.parseLong(s.nextLine());

ps1.setLong(1,bAccNo);

ResultSet rs2 = ps1.executeQuery();

if(rs2.next()){

System.out.println("Enter the amt to be transferred...:");

float amt = Float.parseFloat(s.nextLine());

if(amt<=bal){

ps2.setFloat(1,-amt);

ps2.setLong(2,accNo1);

int i = ps2.executeUpdate();

ps2.setFloat(1,amt);

ps2.setLong(2,bAccNo);

int j = ps2.executeUpdate();

if(i==1 && j==1){

System.out.println("Transaction Successfull...");

con.commit();//step3

TransLog ob = new TransLog(accNo1,bAccNo,amt,new java.util.Date());

PreparedStatement ps3 = con.prepareStatement

("insert into LogTab29 values(?,?)");
```

```
File f = new File("E:\\OnlineData\\Images\\Obj.text");

FileOutputStream fos = new FileOutputStream(f);

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(ob);//Serialization

oos.close();

FileInputStream fis = new FileInputStream(f);

ps3.setLong(1,accNo1);

ps3.setBinaryStream(2,fis,(int)f.length());

int z = ps3.executeUpdate();

f.deleteOnExit();

}else{

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

con.rollback(s1);//step4

}

}else{

System.out.println("Insufficient fund...");

}

}else{

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

}

}else{

System.out.println("Invalid Home AccNo...");

}

con.close();

s.close();
```

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

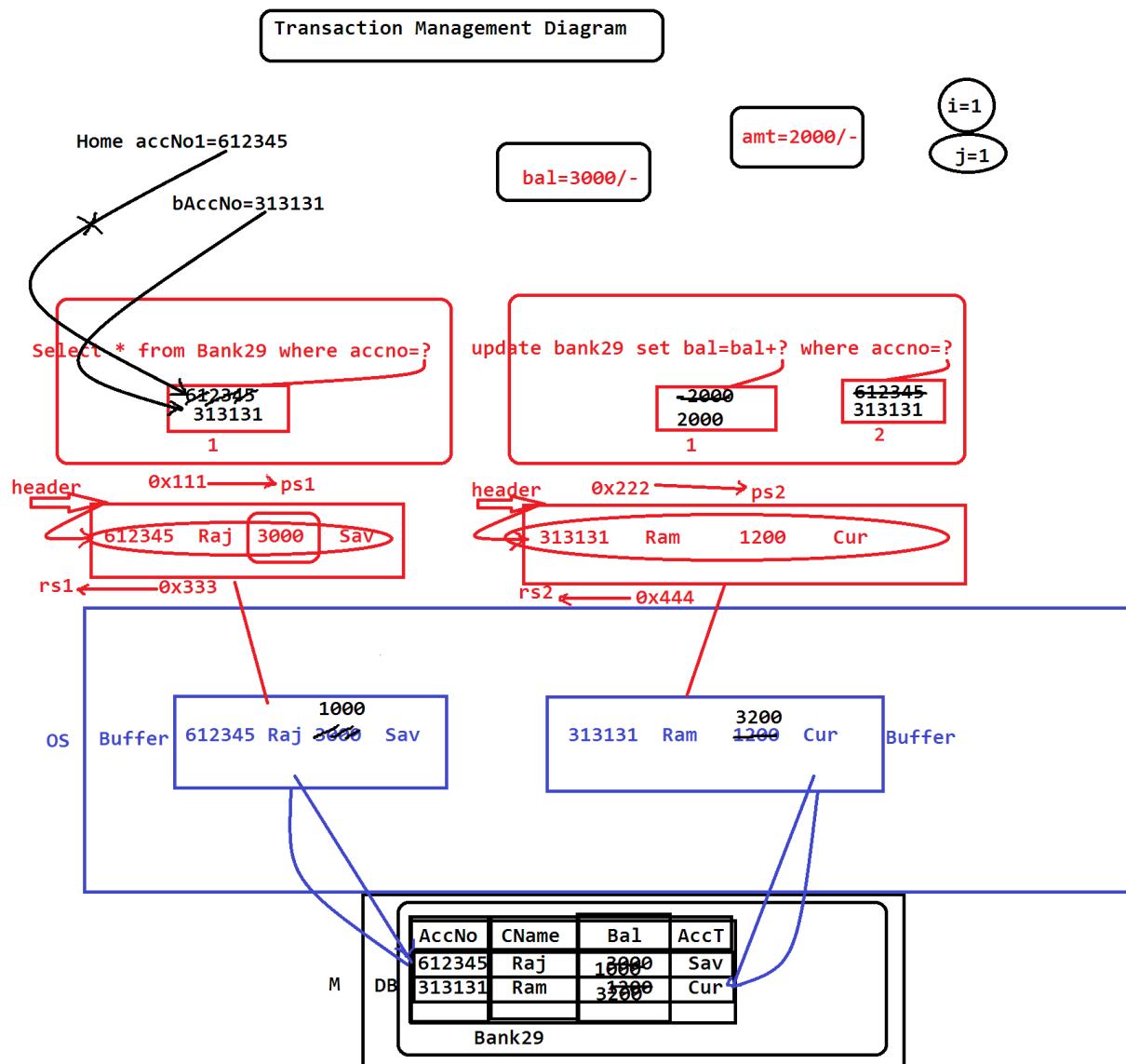
LogDisplay.java

```
package test;  
import java.util.*;  
import java.io.*;  
import java.sql.*;  
public class LogDisplay {  
    public static void main(String[] args) {  
try{  
Scanner s = new Scanner(System.in);  
System.out.println("Enter the hAccNo:");  
long accNo = Long.parseLong(s.nextLine());  
Class.forName("oracle.jdbc.driver.OracleDriver");  
Connection con = DriverManager.getConnection  
("jdbc:oracle:thin:@localhost:1522:orcl","system","ma  
nager");  
PreparedStatement ps = con.prepareStatement  
("select * from LogTab29 where accno=?");  
ps.setLong(1,accNo);  
ResultSet rs = ps.executeQuery();  
if(rs.next()){  
Blob b = rs.getBlob(2);  
byte by[] = b.getBytes(1,(int)b.length());  
File f = new  
File("E:\\\\OnlineData\\\\Images\\\\obj2.txt");  
FileOutputStream fos = new FileOutputStream(f);  
fos.write(by);  
fos.close();  
FileInputStream fis = new FileInputStream(f);  
ObjectInputStream ois = new ObjectInputStream(fis);
```

```
TransLog ob =  
(TransLog)ois.readObject(); //DeSerialization  
System.out.println("HAccNO:"+ob.gethAccNo());  
System.out.println("BAccNO:"+ob.getbAccNo());  
System.out.println("Amt Transferred:"+ob.getAmt());  
System.out.println("Date&Time:"+ob.d);  
ois.close();  
f.deleteOnExit();  
}  
}else{  
System.out.println("Invalid accNo...");  
}  
con.close();  
s.close();  
}  
catch(Exception e){e.printStackTrace();}  
}  
  
-----
```

Dt : 22/12/2020

Execution flow of above program:



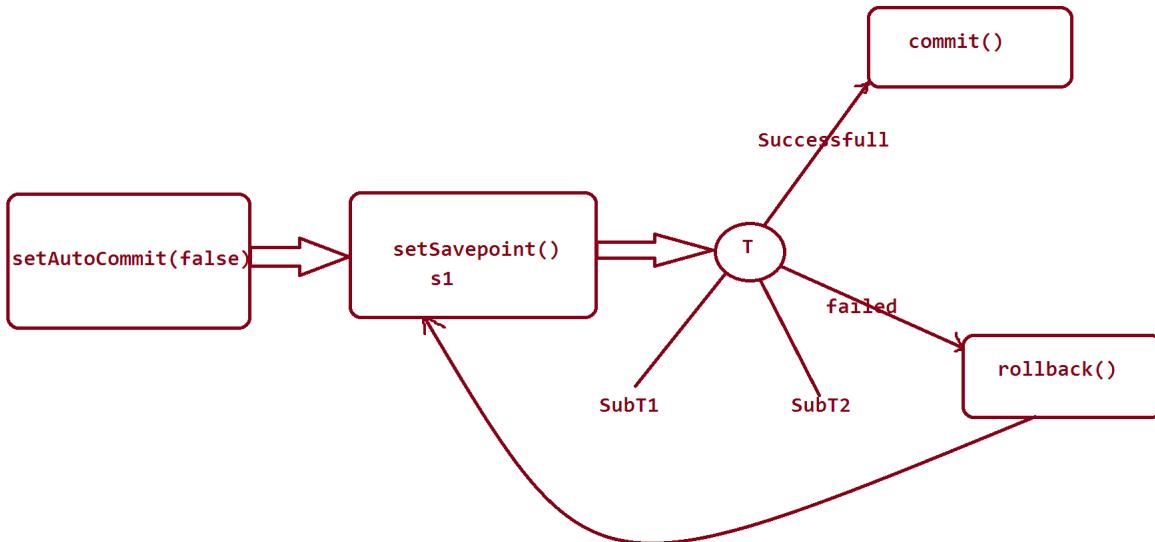
=>From the above program

Transaction T : Transfer the amt(3000/-) from the a/c:612345 to the
a/c:313131

SubTransactions:

SubT1 : Subtracting the amt(3000/-) from the a/c: 612345

SubT2 : Adding the amt(3000/-) to the a/c:313131



*imp

Connection Pooling in JDBC:

=>The process of organizing multiple pre-Initialized DataBase

Connections among multiple users is known as

'Connection Pooling process'

Note:

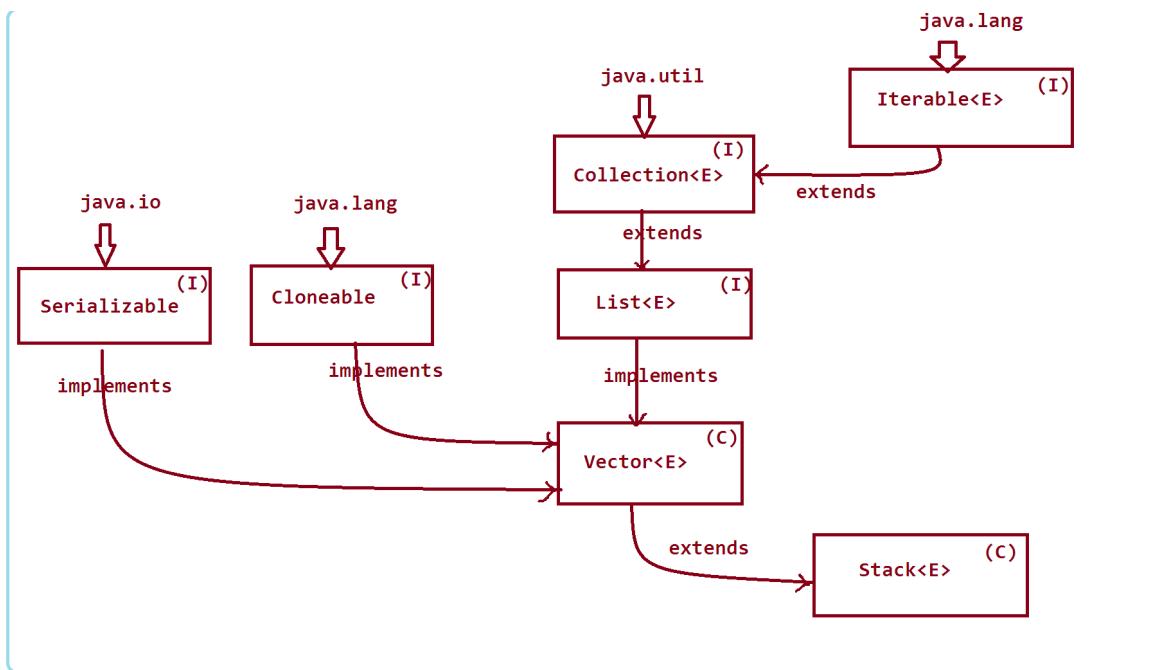
=>In this Connection Pooling Concept we use DB Connection from the pool instead of creating new connection.In this process some execution time is saved and generates HighPerformance.

Note:

=>In Connection Pooling we use `java.util.Vector<E>` object to hold

multiple DB Connections.

Hierarchy of Vector<E>:



Dt : 23/12/2020

Example program:

Pooling.java

```
package test;

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

public class Pooling {

    public String url,uName,pWord;

    public Pooling(String url,String uName,String pWord){

        this.url=url;
        this.uName=uName;
        this.pWord=pWord;
    }
}
```

```
public Vector<Connection> pool = new Vector<Connection>();  
  
public void createConnection(){  
    try{  
        while(pool.size()<5){  
            System.out.println("pool is Not Full...");  
            Class.forName("oracle.jdbc.driver.OracleDriver");  
            Connection con = DriverManager.getConnection(url,uName,pWord);  
            pool.addElement(con);  
            System.out.println(con);  
        }//end of loop  
        if(pool.size()==5){  
            System.out.println("Pool is full...");  
        }  
    }catch(Exception e){e.printStackTrace();}  
}  
  
public synchronized Connection useConnection(){  
    Connection con =(Connection)pool.elementAt(0);  
    pool.removeElementAt(0);  
    return con;  
}  
  
public synchronized void addConnection(Connection con){  
    pool.addElement(con);
```

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

CMainClass.java

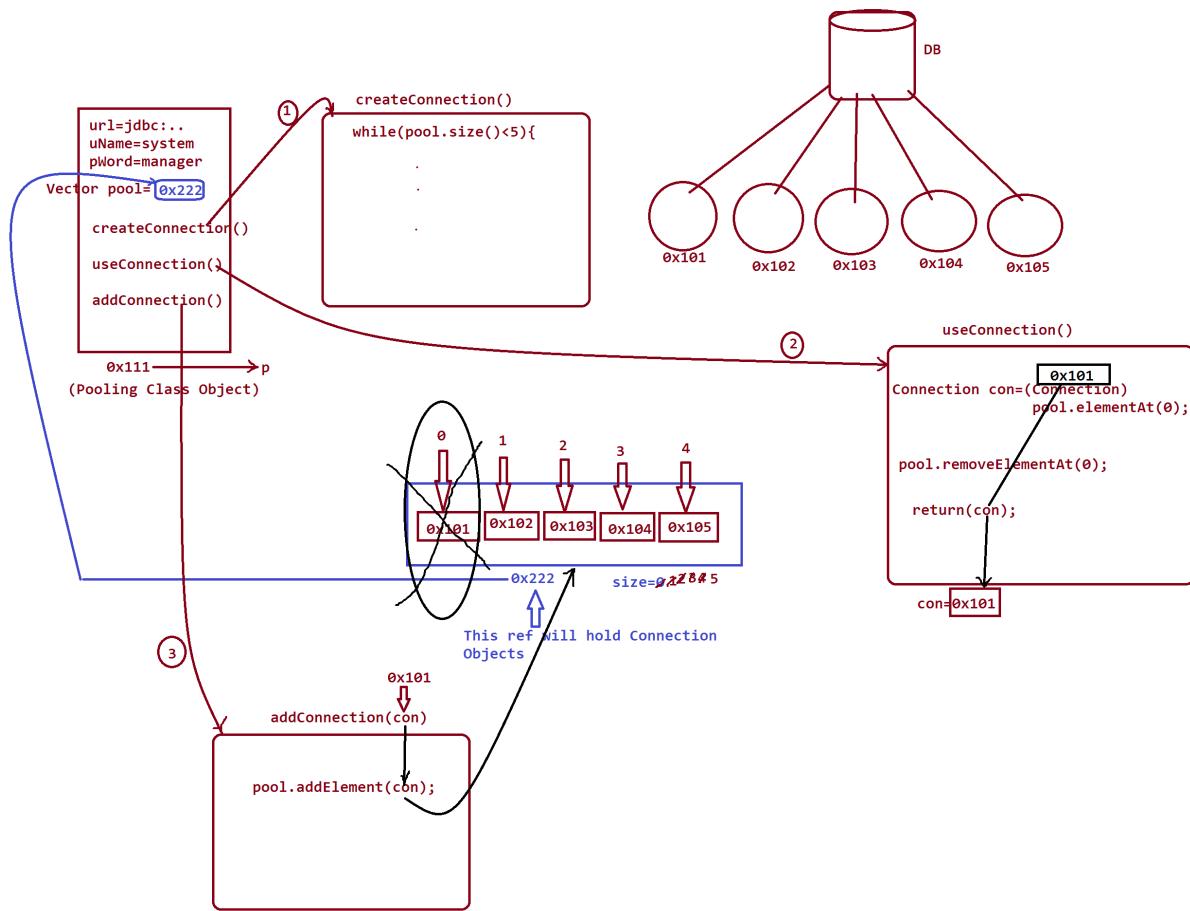
```
package test;  
import java.sql.*;  
public class CMainClass {  
    public static void main(String[] args) {  
        try{  
            Pooling p = new  
            Pooling("jdbc:oracle:thin:@localhost:1522:orcl",  
                    "system","manager");  
            p.createConnection();  
            System.out.println("pool Size="+p.pool.size());  
            Connection con = p.useConnection();  
            System.out.println("Using : "+con);  
            System.out.println("pool Size="+p.pool.size());  
            PreparedStatement ps= con.prepareStatement  
                ("select * from product29");  
            ResultSet rs = ps.executeQuery();  
            while(rs.next()){  
  
                System.out.println(rs.getString(1)+"\t"+rs.getString(2)+  
                    "\t"+rs.getFloat(3)+"\t"+rs.getInt(4));  
            }//end of while  
            p.addConnection(con);  
            System.out.println("pool Size="+p.pool.size());  
        }catch(Exception e){e.printStackTrace();}  
    }  
}
```

Execution flow of above program:

ClassFiles:

Pooling.class

CMainClass.class



Note:

=>Vector<E> class is known as **Synchronized class**.

define Synchronized class?

=>The class which is declared with synchronized methods is known as

Synchronized class.

define Synchronized methods?

=>The methods which are declared with synchronized keyword are known as synchronized methods.

wt is the advantage of synchronized methods?

=>These synchronized methods will be under synchronized locks and the methods are used by one user at a time.

*imp

define RowSet?

=>RowSet is an interface available from "javax.sql" package and which is also used to interact with the DB storage.

=>This RowSet is extended from "java.sql.ResultSet" interface.

=>This RowSet object is more flexible when compared to ResultSet object, and which is by default Scrollable object.

The following are the extended interfaces of RowSet:

(a)JdbcRowSet

(b)CachedRowSet

(c)WebRowSet

(d)JoinRowSet

(e)FilteredRowSet

Note:

=>To interact with DB storage we take the support of JdbcRowSet and which is available from "javax.sql.rowset" package.

Exp program:

```
package test;

import java.sql.*;
import javax.sql.rowset.*;
public class DRowSet {
    public static void main(String[] args) {
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            JdbcRowSet jrs = RowSetProvider.newFactory().createJdbcRowSet();
            jrs.setUrl("jdbc:oracle:thin:@localhost:1522:orcl");
            jrs.setUsername("system");
            jrs.setPassword("manager");
            jrs.setCommand("select * from Product29");
            jrs.execute();

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

            jrs.afterLast();
        }
    }
}
```

```

while(jrs.previous()){

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

}

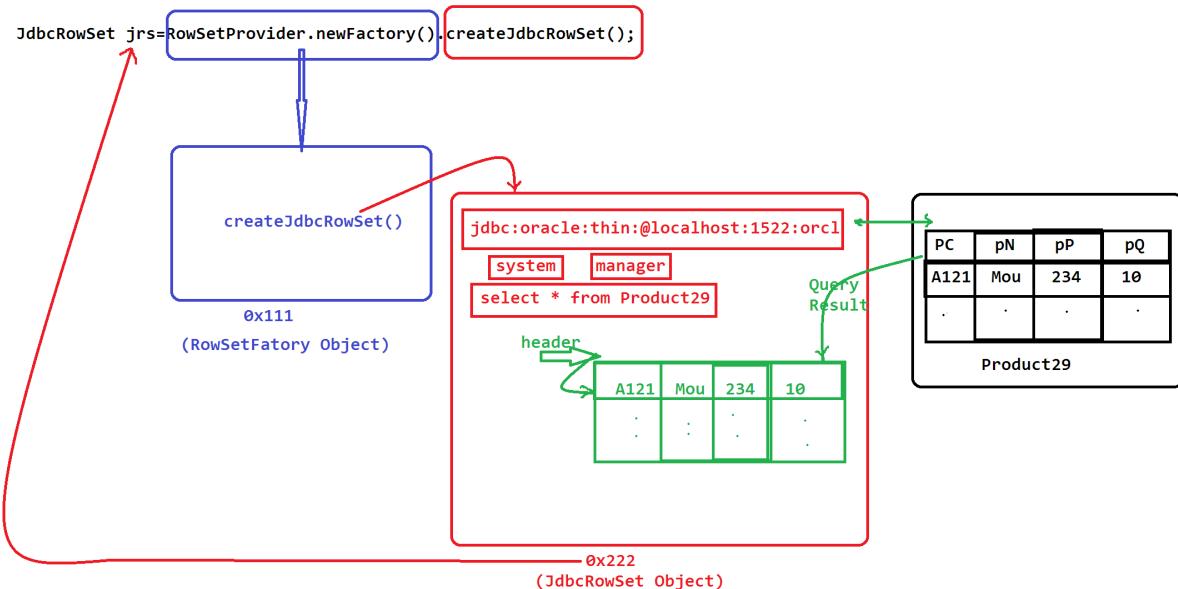
jrs.close();

}catch(ClassNotFoundException | SQLException ob){ob.printStackTrace();}

}

```

Execution flow of above program:



Note:

(i) `newFactory()` method is an static method of `javax.sql.rowset.RowSetProvider` class and which return the ref of `javax.sql.rowset.RowSetFactory` interface.

method signature:

```
public static RowSetFactory newFactory() throws SQLException;
```

(ii)createJdbcRowSet() method is available from

**javax.sql.rowset.RowSetFactory and which returns the ref of
javax.sql.rowset.JdbcRowSet**

method signature:

```
public abstract JdbcRowSet createJdbcRowSet() throws SQLException;
```

syntax:

```
JdbcRowSet jrs = RowSetProvider.newFactory().createJdbcRowSet();
```

Note:

=>JdbcRowSet object will hold url,username,password,Query and also
Query result.(which binds all together)

Dt : 26/12/2020

The following are the Objects generated from JDBC:

1.Connection Object

2.Statement Object

3.PreparedStatement Object

4.CallableStatement Object

5.Non-Scrollable ResultSet Object

6.Scrollable ResultSet Object

7.DatabaseMetaData Object

8.ParameterMetaData Object

9.ResultSetMetaData Object

*

10.RowSet Object

11.Connection Pooling Object

*imp

Streams with DB Product:

define Stream?

=>The Contineous flow of data is known as Stream.

Streams in Java are categoeized into two types:

1.Binary Stream

2.Charater Stream

1.Binary Stream:

=>The Contineous flow of 8-bit data is known as Binary Stream or

Byte Stream.

Note:

=>Binary Stream supports all the multimedia data formats like Text,
Audio,Video,Image and Animation.

2.Charater Stream:

=>The Contineous flow of 16-bit data is known as Character Stream.

Note:

=>Character Stream is best support for text data and not preferable

for Audio, Video, Image and Animation.

=>We use the following in JDBC to store Stream data onto DB Storage:

(a)BLOB

(b)CLOB

Dt : 29/12/2020

=>We use the following in JDBC to store Stream data onto DB Storage:

(a)BLOB

(b)CLOB

(a)BLOB:

=>BLOB stands for 'Binary Large OBjects' and which is used to store Binary Stream data.(Byte Stream data)

Exp program:

step1 : Create one table with name 'BStore29'

```
create table BStore29(id varchar2(10),BFile BLOB,primary key(id));
```

step2 : Write JDBC Appl to store image on to DB Table

```
package test;

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

public class BinaryStore {

    public static void main(String[] args) {
        try{
            Scanner s = new Scanner(System.in);
            System.out.println("Enter the fPath&fName(Source):");
            File f = new File(s.nextLine());
            FileInputStream fis = new FileInputStream(f);
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");
        }
    }
}
```

```
PreparedStatement ps = con.prepareStatement  
("insert into BStore29 values(?,?)");  
ps.setString(1, "A111");  
ps.setBinaryStream(2,fis,(int)f.length());  
int k = ps.executeUpdate();  
if(k>0){  
System.out.println("Image Stored Successfully...");  
}  
  
con.close();  
s.close();  
}catch(Exception e){e.printStackTrace();}  
}  
}
```

O/P:

Enter the fPath&fName(Source):

C:\Users\Public\Pictures\Sample Pictures\Koala.jpg

Image Stored Successfully...

Exp program:

write JDBC Appl to retrive image from DB Table based on 'id'?

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

```
import java.sql.*;
import java.io.*;

public class BinaryRetrieve {
    public static void main(String[] args) {
try{
Scanner s = new Scanner(System.in);
System.out.println("Enter the id:");
String id = s.nextLine();
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");
PreparedStatement ps = con.prepareStatement
("select * from BStore29 where id=?");
ps.setString(1,id);
ResultSet rs = ps.executeQuery();
if(rs.next()){
Blob b = rs.getBlob(2);
byte by[] = b.getBytes(1,(int)b.length());
FileOutputStream fos = new FileOutputStream
("E:\\OnlineData\\Images\\TT.jpg");
fos.write(by);
fos.close();
System.out.println("Image retrieved successfully...");
}else{
System.out.println("Invalid id..");
}
}
}
```

```

    }

    con.close();

    s.close();

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

}

```

}

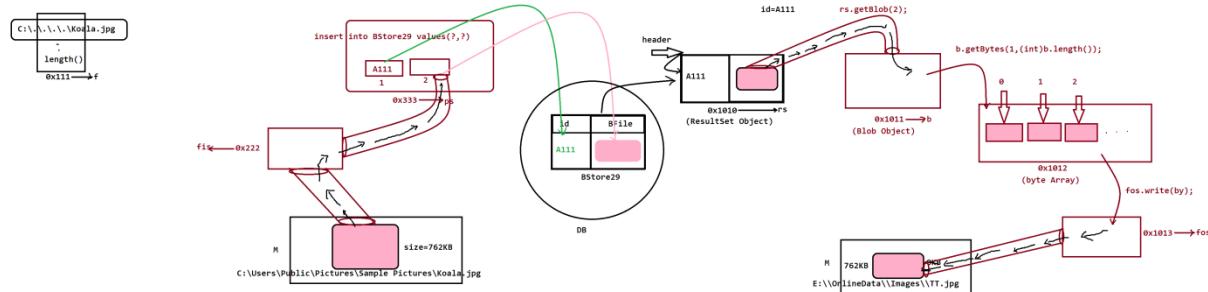
O/P:

Enter the id:

A111

Image retrieved successfully...

Execution flow of above programs:



Dt : 30/12/2020

define FileInputStream?

=>FileInputStream class is from java.io package and which is used to

find the file and opens the file to read Binary Stream.

syntax:

```
FileInputStream fis = new FileInputStream("fPath&fName");
```

define setBinaryStream() method?

=>**setBinaryStream()** method specify the Stream and its length to the index-parameter.

=>**This method is available from "PreparedStatement" interface.**

Method Signature:

```
public abstract void setBinaryStream(int,java.io.InputStream,int)  
throws java.sql.SQLException;
```

syntax:

```
ps.setBinaryStream(2,fis,(int)f.length())
```

define FileOutputStream?

=>**FileOutputStream** class is from **java.io** package and which is used to create file and opens the file to record Binary Stream.

syntax:

```
FileOutputStream fos = new FileOutputStream("fPath&fName");
```

define getBlob() method?

=>**getBlob()** method will create the implementation object of

'java.sql.Blob' interface and the object is linked to the Stream column.

=>This method is available from 'ResultSet' interface.

Method Signature:

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

syntax:

```
Blob b = rs.getBlob(2);
```

define getBytes() method?

=>getBytes() method will take the stream and convert in Byte Array.

=>This method is available from 'java.sql.Blob' interface.

Method Signature:

```
public abstract byte[] getBytes(long,int) throws java.sql.SQLException;
```

syntax:

```
byte by[] = b.getBytes(1,(int)b.length());  
=====
```

***imp**

Object State onto DB Storage:

=>In the process of storing Object State onto DB Storage, the Object State must be available in the form Binary Stream.

define Serialization process?

=>The process of converting Object State into Binary Stream is known as **Serialization** process.

=>we use **writeObject()** method from '**java.io.ObjectOutputStream**' class to perform **Serialization** process.

Method Signature:

```
public final void writeObject(java.lang.Object)  
    throws java.io.IOException;
```

syntax:

```
ObjectOutputStream oos = new ObjectOutputStream(fos);  
oos.writeObject(obj);
```

define DeSerialization process?

=>The process of converting Binary Stream into Object State is known as **DeSerialization** process.

=>we use **readObject()** method from '**java.io.ObjectInputStream**' class to perform **DeSerialization** process.

Method Signature:

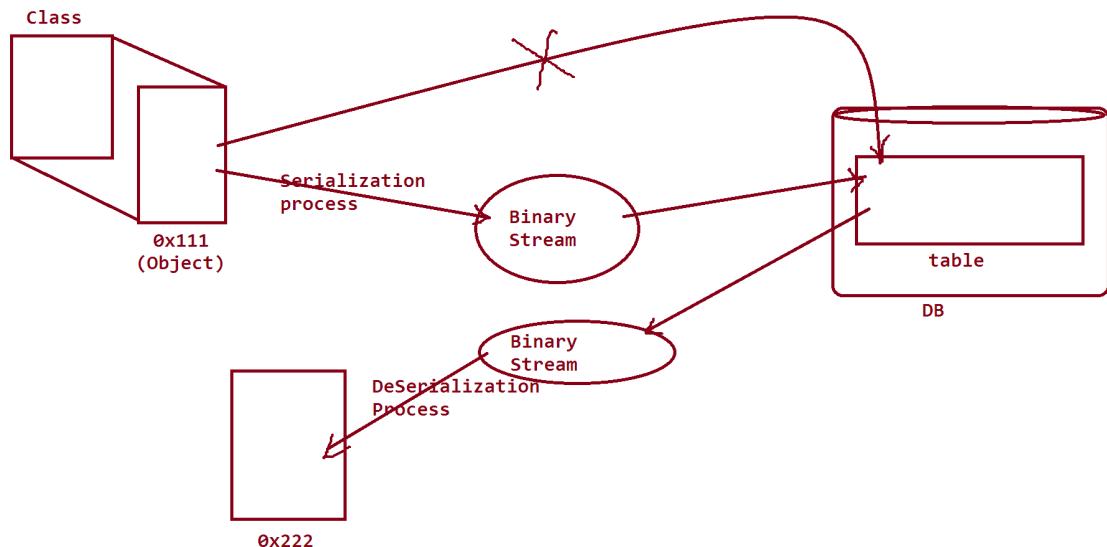
```
public final java.lang.Object readObject() throws java.io.IOException,  
    java.lang.ClassNotFoundException;
```

syntax:

```
ObjectInputStream ois = new ObjectInputStream(fis);  
Object ob = ois.readObject();
```

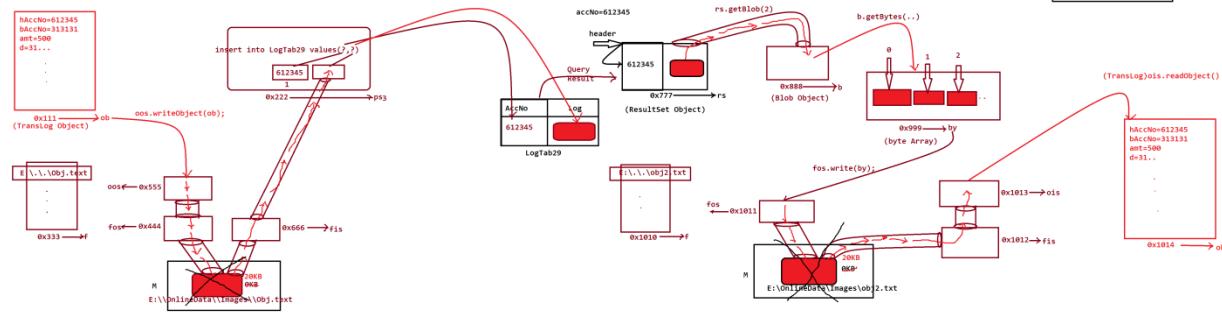
Note:

=>To perform **Serialization** and **DeSerialization** process the class must be implemented from '**java.io.Serializable**' interface.



dt : 31/12/2020

Diagram:



faq:

define Immutable class?

=>The class which is declared with the following rules, is known as **Immutable class**.

Rule-1 : The class must be final class

Rule-2 : The variables within the class must be private and final

Rule-3 : The class must be declared with only 'Getter methods'.

Rule-4 : These Getter methods must be final methods.

Note:

=>These **Immutable classes** will generate **Immutable objects**.

define Immutable objects?

=>The Objects once created cannot be modified are known as **Immutable Objects.(Secured Objects)**

=====

dt : 2/1/2021

(b)CLOB:

=>CLOB stands for 'Character Large OBjects' and which is used to store Character Stream.

Exp program:

wap to store Character Stream(Text data) on to DB Storage.

step1 : Create table with name 'CStore29'

```
create table CStore29(id varchar2(10),CFile CLOB,primary key(id));
```

step2 : JDBC App to store Character Stream

```
package test;

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

public class CharacterStore {

    public static void main(String[] args) {
        try{
            Scanner s = new Scanner(System.in);
            System.out.println("Enter the fPath&fName(Source):");

            File f = new File(s.nextLine());
            FileReader fr = new FileReader(f);
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");
        }
    }
}
```

```
PreparedStatement ps = con.prepareStatement  
("insert into CStore29 values(?,?)");  
  
ps.setString(1, "A111");  
  
ps.setCharacterStream(2,fr,(int)f.length());  
  
int k = ps.executeUpdate();  
  
if(k>0){  
  
System.out.println("Text data Stored Successfully...");  
  
}  
  
con.close();  
  
s.close();  
  
}catch(Exception e){e.printStackTrace();}  
  
}  
  
}
```

O/P:

Enter the fPath&fName(Source):

E:\OnlineData\Images\Text1.txt

Text data Stored Successfully...

Exp program:

wap to retrive Character Stream from the DB Storage.

```
package test;

import java.util.*;

import java.sql.*;

import java.io.*;

public class CharacterRetrieve {

    public static void main(String[] args) {

try{

Scanner s = new Scanner(System.in);

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

String id = s.nextLine();

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");

PreparedStatement ps = con.prepareStatement

("select * from CStore29 where id=?");

ps.setString(1,id);

ResultSet rs = ps.executeQuery();

if(rs.next()){

Clob c = rs.getBlob(2);

Reader r = c.getCharacterStream();

FileWriter fw = new FileWriter

("E:\\OnlineData\\Images\\TT2.txt");

int ch;

while((ch=r.read())!=-1){

fw.write(ch);

}}}}}
```

```
//end of loop  
  
fw.close();  
  
System.out.println("Text data retrieved successfully...");  
  
}else{  
  
System.out.println("Invalid id..");  
  
}  
  
con.close();
```

```
s.close();  
  
}catch(Exception e){e.printStackTrace();}  
  
}  
  
}
```

O/P:

Enter the id:

A111

Text data retrieved successfully...

define FileReader?

=>FileReader class is from java.io package and which is used to find the file and opens the file to read character Stream.

syntax:

```
FileReader fr = new FileReader("fPath&fName");
```

define FileWriter?

=>FileWriter class is from java.io package and which is used to

create file and opens the file to write Character Stream.

syntax:

FileWriter fw = new FileWriter("fPath&fName");

Note:

=>In realtime CLOB is less used when compared to BLOB.

Dt : 4/1/2021

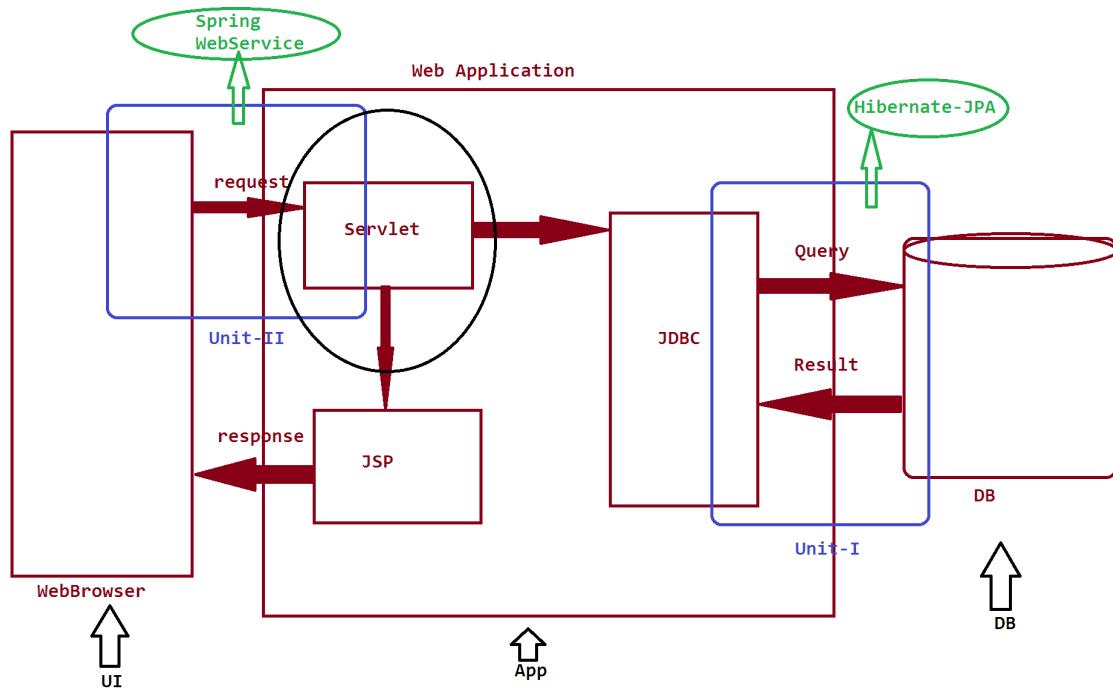
Summary of JDBC:

=>The following are the 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.DatabaseMetaData object**
 - 8.ParameterMetaData object**
 - 9.ResultSetMetaData object**
 - 10.RowSet Object**
- *
- 11.Connection Pooling Object**
-

*imp

Servlet Programming(Unit-II):



define Servlet Program?

=>The Java Program which is executing in Server environment and providing the service, is known as **Servlet Program**.

define Service?

=>The process of accepting the request and providing the response is known as **Service**.

Note:

=>Servlet program means **Server Side program**.

define Application?

=>The 'set-of-programs' collected together to perform defined action

is known as Application.

=>The following are some important applications:

1.StandAlone applications

2.Web Applications

3.Enterprise Applications

4.Mobile Applications

1.StandAlone applications:

=>The applications which are installed in one computer and performs actions in the same computer are known as StandAlone applications.

Note:

=>The applications which do not use HTML i/p,server and DB are known as StandAlone applications.

Based on User interaction the StandAlone applications are categorized into two types:

(i)CUI Applications

(ii)GUI Applications

(i)CUI Applications:

**=>In CUI applications the user interact through CommandPrompt.
(CUI - Console User Interface)**

(ii)GUI Applications:

=>In GUI Applications the user interacts through GUI Components.

(GUI - Graphical User Interface)

Note:

=>**StandAlone applications**

Starting point : main() method

Environment : JVM

Command : java

define Applet program?

=>The process of binding bytecode into HTML Code is known as Applet program.

=>In the process of constructing applet programs we use the following

LifeCycle methods:

(i)init() - Initialization

(ii)start() - starting the applet

(iii)stop() - stopping the applet

(iv)destroy() - destroying the applet

Note:

=>**Applet programs**

Starting point : init() method

Environment : AE(Applet Engine)

Command : appletviewer

Dt : 5/1/2021

***imp**

2.Web Applications:

=>The application which is executing in web environment or internet environment is known as Web Application or Internet Application.

=>We use the following to construct WebApplications:

(i)JDBC

(ii)Servlet

(iii)JSP

Note:

=>Web Applications

Starting point : init()

Environment : WebContainer

Command : url-pattern

3.Enterprise Applications:

=>The applications which are available in distributed environment and depending on the features like LoadBalancing,Security and Clustering are known as Enterprise Applications.

4. Mobile Applications:

=>The applications which are executed in Mobile environment are known as Mobile Applications

Dt : 7/1/2021

***imp**

define WebContainer?

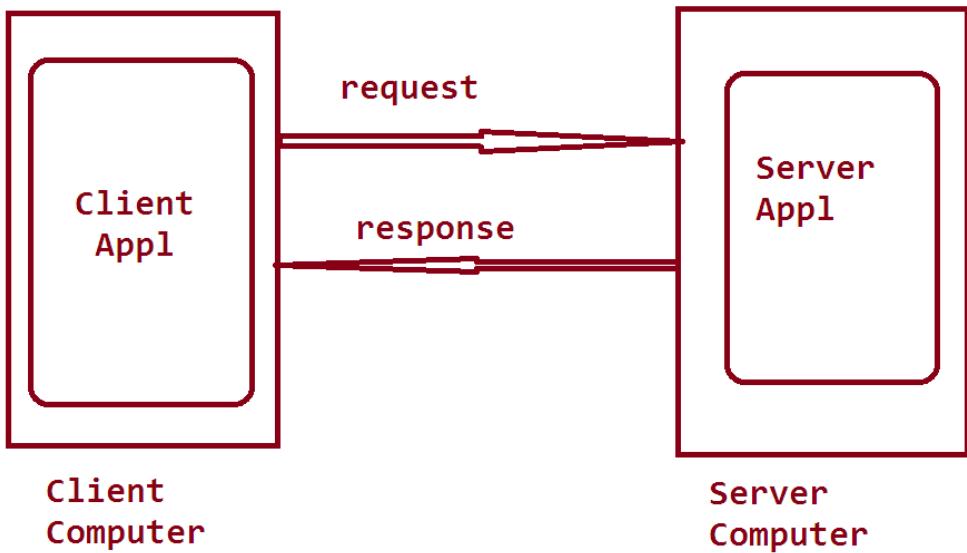
=>The s/w component in which the WebAppl is deployed for execution is known as WebContainer.

=>This WebContainer will hold Web Components.

=>This WebContainer is available from WebServers and Application Servers.

define Server?

=>The S/W component which provides Service is known as Server.



define WebServer?

=>The Server which holds only WebContainer to execute WebApps is known as WebServer.

Exp:

Tomcat Server

define Application Server?

=>The Server which holds both WebContainer and EJB Container is known as Application Server.

=>In Application Server we can execute both WebApps and Enterprise Apps.

(EJB - Enterprise Java Bean)

Exp:

JBoss

Glassfish

WebLogic

WebSphere

***imp**

Installing Tomcat Server:

step1 : Download Tomcat9.x WebServer

webserver - Tomcat 9.x

(Compatable 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

step2 : Install Tomcat Server

while Installation process,

Select the type of Install : Full (click next)

Server Shutdown port No : 8088(Any port no)

HTTP/1.1 Connector port : 8081 or 8082 ..(Any port no)

User Name : venkatesh

Password : nit

(click on next)

step3 : Start the Tomcat Server

=>Click 'startup' or 'Tomcat9w' from 'bin' to start the TomcatServer

C:\Tomcat 9.0\bin

**step4 : Use the following address from WebBrowser to check the Tomcat
is working or not.**

<http://localhost:8082>



step5 : Stop the Tomcat Server

=>Click 'shutdown' or 'Tomcat9w' from 'bin' to stop the TomcatServer

C:\Tomcat 9.0\bin

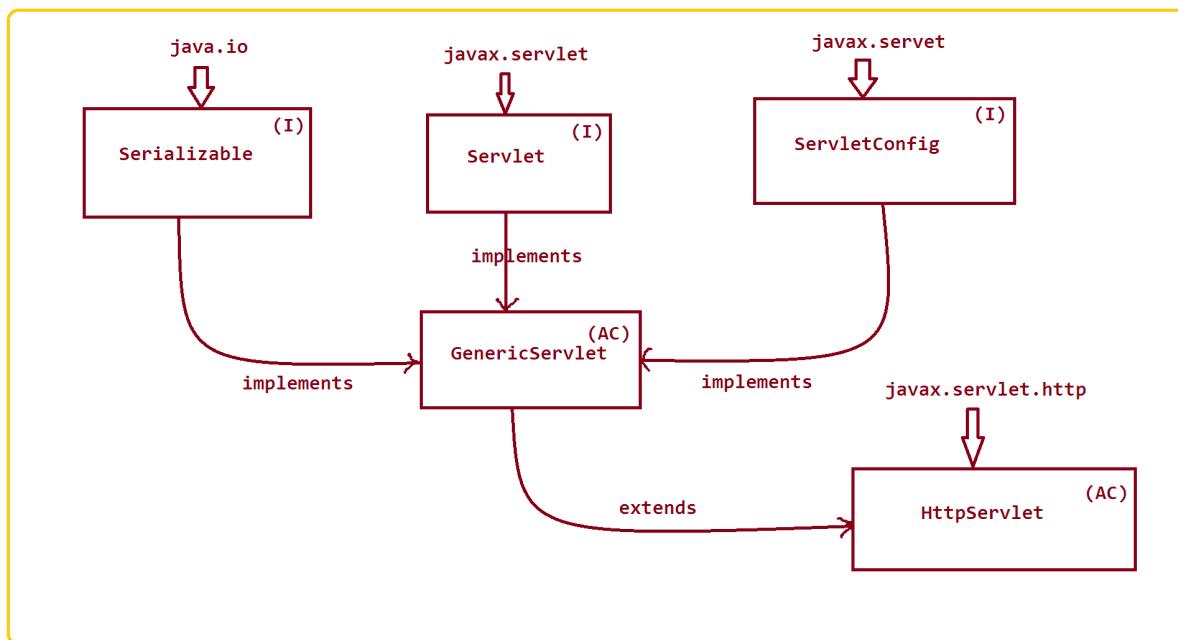
Dt : 8/1/2021

Servlet API:

=>The Servlet programming related classes and interfaces are available from 'javax.servlet' package and which is known as 'Servlet API'.

=>The root of Servlet API is 'javax.servlet.Servlet' Interface.

Hierarchy of Servlet API:



Note:

=>In the process of Constructing Servlet program,the class must be extended from 'GenericServlet' or 'HttpServlet'.

*imp

Case-1 : Constructing Servlet program using 'GenericServlet'

=>The User defined class must be extended from 'GenericServlet' and in this process the 'GenericServlet' provides the following LifeCycle methods:

1.init()

2.service()

3.destroy()

1.init():

=>using init() method we can perform initialization process,which means making the programming components ready for service() method.

Method Signature:

public void init() throws javax.servlet.ServletException;

public void init(javax.servlet.ServletConfig)

throws javax.servlet.ServletException;

2.service():

=>Using service() method we provide services to the end user.which means accepting the request and providing the response.

Method Signature:

public abstract void service(javax.servlet.ServletRequest,

javax.servlet.ServletResponse) throws javax.servlet.ServletException,

java.io.IOException;

3.destroy():

=>Using destroy method we can close resources opened for the Servlet program.

Method Signature:

public void destroy();

define LifeCycle methods?

=>The methods which are executed automatically are known as LifeCycle method.

=>The above LifeCycle methods are executed in the following order:

init()

service()

destroy()

Note:

=>when we want to make Servlet API available to JDK then we must copy 'servlet-api.jar' file from Tomcat to JDK.

Location of servlet-api.jar :

C:\Tomcat 9.0\lib

Location of 'ext' in JDK:

C:\Program Files\Java\jdk1.8.0_251\jre\lib\ext

Note:

=>Before Constructing Dynamic Web Project from IDE Eclipse,stop the Tomcat Server.

Dt : 11/1/2021

*imp

Constructing Dynamic Web Project(WebApplication) from IDE Eclipse:

step1 : Open Eclipse,while opening name the workspace(folder) and click 'ok'

step2 : Create Dynamic Web Project

Click on File->new->Project->Web->select 'Dynamic web project' and click 'next'->name the project and click 'finish'.

step3 : Add 'servlet-api.jar' to the Dynamic Web Project

RightClick on Project->Built-path->Configure Build path->Libraries->Add External Jars->Browse and select 'Servlet-api.jar' from 'lib' of Tomcat Server->Open->Apply->Ok

step4 : Create package

RightClick on src->new->package,name the package and click 'finish'.

step5:Create HTML file in WebContent

RightClick on WebContent->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">
ProductCode:<input type="text" name="PCODE"><br>
ProductName:<input type="text" name="PNAME"><br>
ProductPrice:<input type="text" name="PPRICE"><br>
ProductQty:<input type="text" name="PQTY"><br>
<input type="submit" value="Display">
</form>
</body>
</html>

```

step6 : Construct Servlet program to read data from HTML form.

RightClick on package->new->class, name the class and click 'finish'

DisServlet.java

```

package test;

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

@SuppressWarnings("serial")

public class DisServlet extends GenericServlet{

    public void init(){
        //NoCode
    }

    public void service(ServletRequest req,ServletResponse res) throws
    ServletException,IOException{
        PrintWriter pw = res.getWriter();

```

```

res.setContentType("text/html");

String pC = req.getParameter("PCODE");
String pN = req.getParameter("PNAME");
float pP = Float.parseFloat(req.getParameter("PPRICE"));
int pQ = Integer.parseInt(req.getParameter("PQTY"));

pw.println("ProdCode:" + pC);
pw.println("<br>ProdName:" + pN);
pw.println("<br>ProdPrice:" + pP);
pw.println("<br>ProdQty:" + pQ);

}

```

```

public void destroy(){

    //NoCode

}

```

**step7 : Construct 'web.xml' file to hold Servlet url-pattern,part of
'WEB-INF'**

**RightClick on WEB-INF->new->other->XML->XML file,name the file and
click finish**

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <servlet>
        <servlet-name>DisServlet</servlet-name>

```

```
<servlet-class>test.DisServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>DisServlet</servlet-name>
    <url-pattern>/dis</url-pattern>
</servlet-mapping>

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

step8 : Add WebServer(Tomcat Server) to the IDE Eclipse

To Add Server,Click 'Servers'->Click this link to create new Server->

Browse and select Tomcat Server from Apache->click 'next'->

Browser Tomcat Server Folder and click finish.

step9 : Deploy WebApp on to Tomcat Server for execution.

RightClick on WebApp->Run As->Run on Server->Click 'finish'

Dt : 12/1/2020

Exp Appl2:

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="dis" method="post">
BookCode:<input type="text" name="bcode"><br>
BookName:<input type="text" name="bname"><br>
BookAuthor:<input type="text" name="bauthor"><br>
BookPrice:<input type="text" name="bprice"><br>
BookQty:<input type="text" name="bqty"><br>
<input type="submit" value="Display">
</form>
</body>
</html>
```

DisServlet.java

```
package test;

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

@SuppressWarnings("serial")
public class DisServlet extends GenericServlet{
    public void init(){
        //NoCode
    }
    public void service(ServletRequest req,ServletResponse res) throws
    ServletException,IOException{
```

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

res.setContentType("text/html");

String bC = req.getParameter("bcode");

String bN = req.getParameter("bname");

String bA = req.getParameter("bauthor");

float bP = Float.parseFloat(req.getParameter("bprice"));

int bQ = Integer.parseInt(req.getParameter("bqty"));

pw.println("BookCode:"+bC);

pw.println("<br>BookName:"+bN);

pw.println("<br>BookAuthor:"+bA);

pw.println("<br>BookPrice:"+bP);

pw.println("<br>BookQty:"+bQ);

}

public void destroy(){

//NoCode

}

}

web.xml
```

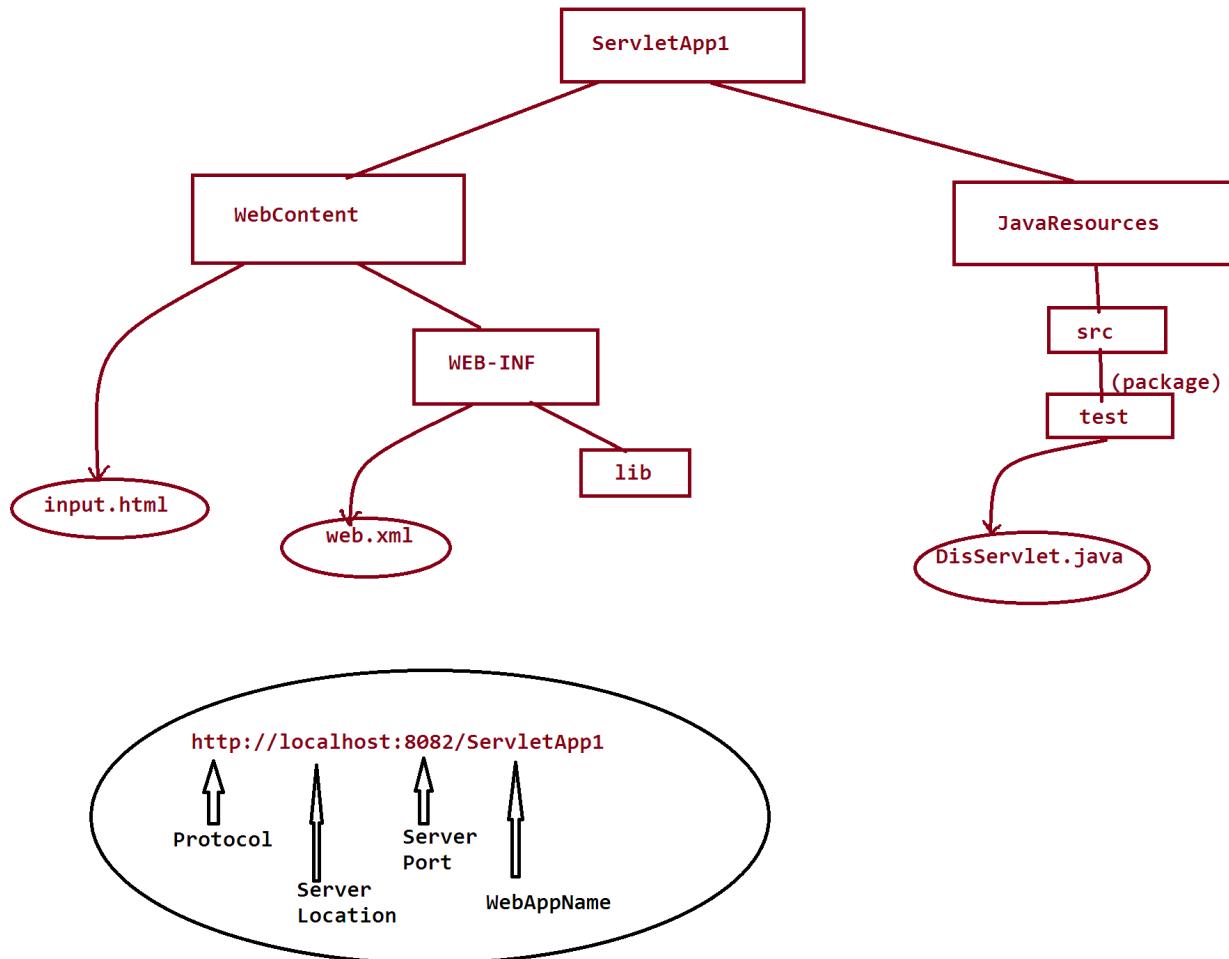
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <servlet>
        <servlet-name>DisServlet</servlet-name>
        <servlet-class>test.DisServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>DisServlet</servlet-name>
        <url-pattern>/dis</url-pattern>
    </servlet-mapping>
```

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

Dt : 18/1/2021

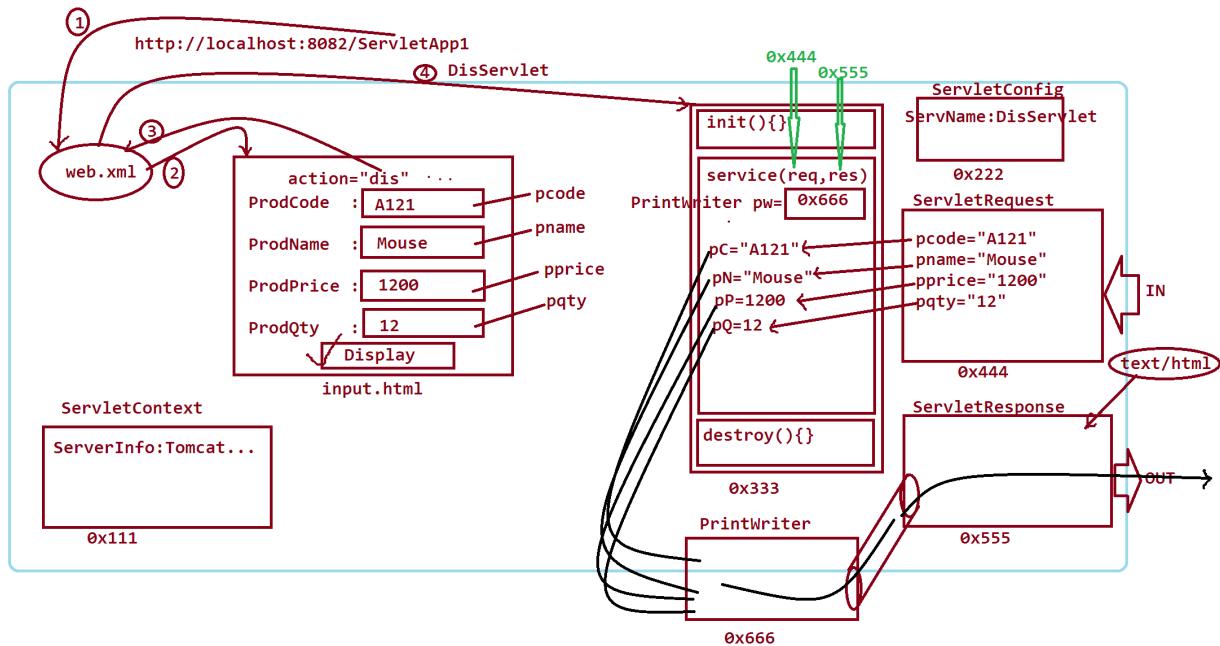
Execution flow of above application:(ServletApp1)

Deployment Directory Structure:(Eclipse)



Execute the application as follows:

<http://localhost:8082/ServletApp1>



Dt : 19/1/2021

define getWriter() method?

=>This getWriter() method is used to create object for 'java.io.PrintWriter' class and this method is available from 'javax.servlet.ServletResponse' interface.

Method Signature:

```
public abstract java.io.PrintWriter getWriter() throws  
        java.io.IOException;
```

syntax:

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

define setContentType() method?

=>This setContentType() method will specify the type of data sent through response object and this method is available from 'ServletResponse' interface.

Method Signature:

```
public abstract void setContentType(java.lang.String);
```

syntax:

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

define getParameter() method?

=>This getParameter() method is used to get the data from the request object and which is available from 'javax.servlet.ServletRequest' Interface.

Method Signature:

public abstract java.lang.String getParameter(java.lang.String);

syntax:

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

***imp**

define 'ServletContext'?

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

=>This 'ServletContext' object is loaded with Server information.

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

=>This getServletContext() method is available from GenericServlet, ServletRequest and ServletConfig.

Method Signature from GenericServlet:

public javax.servlet.ServletContext getServletContext();

syntax:

ServletContext sct = this.getServletContext();

Note:

=>we use <context-param> tag part of web.xml to initialize parameters to the ServletContext object.

syntax:

```
<web-app>
  <context-param>
    <param-name>name</param-name>
    <param-value>value</param-value>
  </context-param>
  <servlet>
    </servlet>
  <servlet-mapping>
    </servlet-mapping>
  .
  .
</web-app>
```

***imp**

define ServletConfig?

=>'ServletConfig' is an interface from javax.servlet package and
which is instantiated automatically when the Servlet program is loaded.

=>This ServletConfig object loaded with **Servlet_name**.

=>we use **getServletConfig** method to access the reference of
ServletConfig object.

=>This **getServletConfig** method is available from '**GenericServlet**'.

Method Signature:

```
public javax.servlet.ServletConfig getServletConfig();
```

syntax:

```
ServletConfig scf = this.getServletConfig();
```

Note:

=>we use <init-param> tag part of web.xml to initialize parameters in
ServletConfig object.

syntax:

```
<web-app>
```

```
  <servlet>
```

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

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

```
    <init-param>
```

```
      <param-name>name</param-name>
```

```
      </param-value>value</param-value>
```

```
    </init-param>
```

```
  </servlet>
```

```
  <servlet-mapping>
```

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

```
.
```

```
.
```

```
</web-app>
```

Exp App : Demonstrate ServletContext and ServletConfig

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="dis" method="post">
Name:<input type="text" name="name"><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>100</param-value>
</context-param>
<servlet>
<servlet-name>DisServlet</servlet-name>
<servlet-class>test.DisServlet</servlet-class>
<init-param>
<param-name>b</param-name>
<param-value>200</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>DisServlet</servlet-name>
<url-pattern>/dis</url-pattern>
</servlet-mapping>

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

DisServlet.java

```
package test;

import java.io.*;

import javax.servlet.*;

@SuppressWarnings("serial")
public class DisServlet extends GenericServlet{

    public void init(){

        //NoCode

    }

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

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

        ServletContext sct = this.getServletContext();

        ServletConfig scf = this.getServletConfig();

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

        int a = Integer.parseInt(sct.getInitParameter("a"));

        int b = Integer.parseInt(scf.getInitParameter("b"));

        pw.println("WELCOME : "+name);

        pw.println("<br>=====ServletContext=====");

        pw.println("<br>ServerInfo:"+sct.getServerInfo());

        pw.println("<br>The value a:"+a);

        pw.println("<br>=====ServletConfig=====");

        pw.println("<br>ServletName:"+scf.getServletName());

        pw.println("<br>The value b:"+b);
    }
}
```

```
}

public void destroy(){

    //NoCode

}

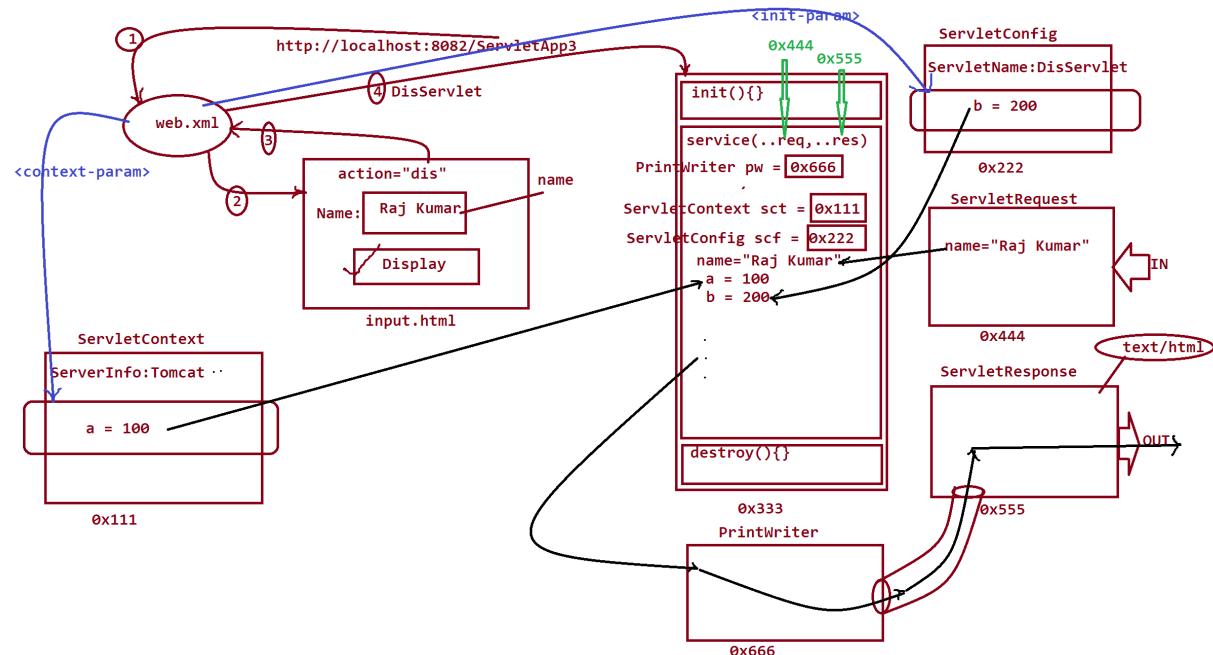
}
```

Dt : 20/1/2021

Execution flow of above application:

Execute the application as follows:

<http://localhost:8082/ServletApp3>



faq:

define getInitParameter()?

=>`getInitParameter()` method is used to read parameter values from `ServletContext` and `ServletConfig` objects.

=>This method is available from `ServletContext` and `ServletConfig`.

Method Signature:

```
public abstract java.lang.String getInitParameter(java.lang.String);
```

faq:

wt is the diff b/w

(i)getParameter()

(ii)getInitParameter()

=>**getParameter()** method will read the data from request object.

=>**getInitParameter()** method will read data from Context and Config objects.

faq:

wt is the diff b/w

(i)<context-param>

(ii)<init-param>

=>**<context-param>** will initialize the parameter to the **ServletContext Object** and **<init-param>** will initialize the parameter to the **ServletConfig object**.

=>**<context-param>** is declared part of **<web-app>** tag and **<init-param>** is declared part of **<servlet>** tag.

***imp**

Servlet program to DB Product:

=>In the process of establishing communication b/w Servlet and DB product, we use **DAO(Data Access Object)** Layer.

=>DAO Layer will have DB Connection and DB Access details.

=>Part of DAO Layer we use the following code for DB Connection:

```
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:1522:orcl","system","maneger");

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

    }//end of static block

    public static Connection getCon(){

        return con;

    }

}
```

Note:

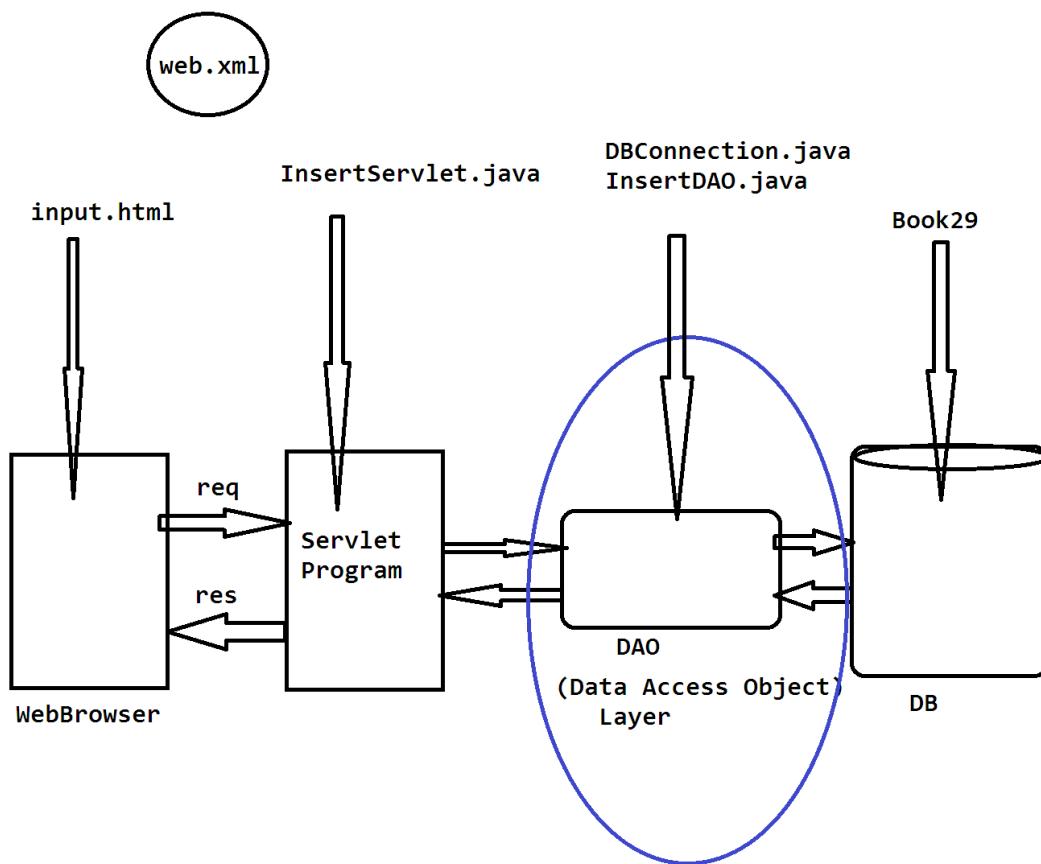
=>To establish connection to DB Product,the DB Jar must be copied in

'lib' folder of 'WEB-INF'.

Dt : 21/1/2021

Exp Application:

(Insert Book details to DB Table 'Book29')



input.html

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

```

<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <form action="insert" method="post">
BookCode:<input type="text" name="bcode"><br>
BookName:<input type="text" name="bname"><br>
BookAuthor:<input type="text" name="bauthor"><br>
BookPrice:<input type="text" name="bprice"><br>
BookQty:<input type="text" name="bqty"><br>
<input type="submit" value="InsertBookData">
    </form>
</body>
</html>

```

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:1522:orcl","system","ma
nager");
            }catch(Exception e){e.printStackTrace();}
        }//end of block
    public static Connection getCon(){
        return con;
    }
}

```

InsertDAO.java

```
package test;

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

public class InsertDAO {

    public int k;

    public int insert(ServletRequest req){

        try{

            Connection con = DBConnection.getCon();

            PreparedStatement ps = con.prepareStatement
                ("insert into Book29 values(?,?,?,?,?)");

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

            ps.setString(2,req.getParameter("bname"));

            ps.setString(3,req.getParameter("bauthor"));

            ps.setFloat(4,Float.parseFloat(req.getParameter("bprice")));

            ps.setInt(5,Integer.parseInt(req.getParameter("bqty")));

            k = ps.executeUpdate();

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

        return k;
    }
}
```

InsertServlet.java

```
package test;

import java.io.*;
import java.sql.*;

import javax.servlet;
```

```
@SuppressWarnings("serial")

public class InsertServlet extends GenericServlet{

    public InsertDAO id=null;

    public void init(){

        id = new InsertDAO();

    }

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

PrintWriter pw = res.getWriter();

res.setContentType("text/html");

int k = id.insert(req);

    if(k>0){

pw.println("BookDetails inserted Successfully...");

    }

}

public void destroy(){

    try{

        Connection con = DBConnection.getCon();

        con.close();

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

}

}
```

web.xml

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

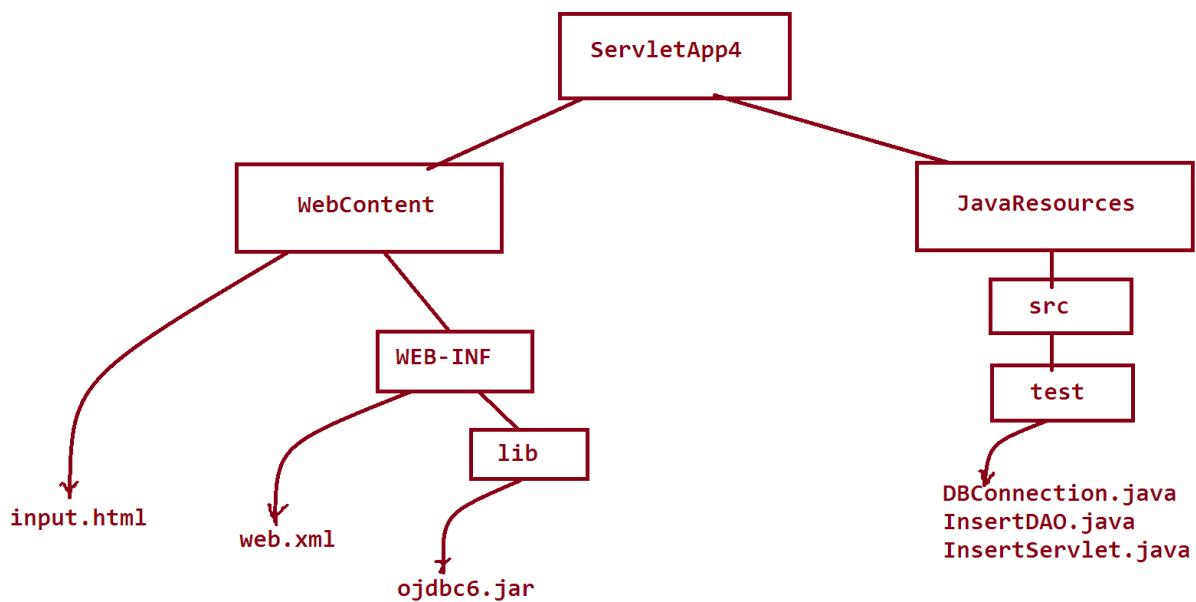
```

<servlet>
    <servlet-name>InsertServlet</servlet-name>
    <servlet-class>test.InsertServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>InsertServlet</servlet-name>
    <url-pattern>/insert</url-pattern>
</servlet-mapping>

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

```

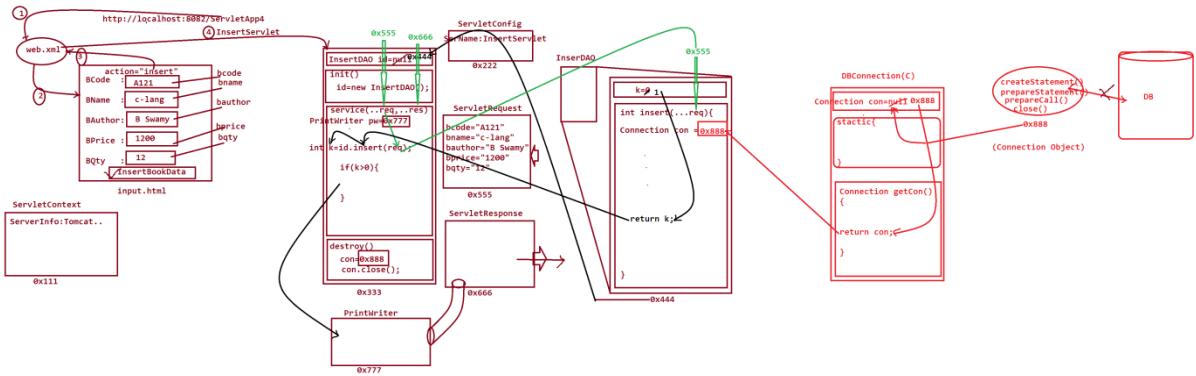
Deployment directory Structure:(Eclipse)



Execute the appl as follows:

<http://localhost:8082/ServletApp4>

Execution flow of above application:



Assignment:

Update 'ServletApp1' by inserting the data to DB Table Product29.

Dt : 22/1/2021

Exp Application:

Servlet Application to retrieve Book Details based on bCode?

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <form action="retrieve" method="post">
Enter the BookCode:<input type="text" name="bcode"><br>
<input type="submit" value="RetrieveBookData">
    </form>
</body>
</html>
```

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:1522:orcl","system","ma
nager");
        }catch(Exception e){e.printStackTrace();}
    }//end of block
    public static Connection getCon(){
        return con;
```

```
    }  
}
```

RetrieveDAO.java

```
package test;  
  
import java.sql.*;  
  
import javax.servlet.*;  
  
public class RetrieveDAO {  
  
    public ResultSet rs=null;  
  
    public ResultSet retrieve(ServletRequest req){  
  
        try{  
  
            Connection con = DBConnection.getCon();  
  
            PreparedStatement ps = con.prepareStatement  
("select * from Book29 where bcode=?");  
  
            ps.setString(1,req.getParameter("bcode"));  
  
            rs = ps.executeQuery();  
  
        }catch(Exception e){e.printStackTrace();}  
  
        return rs;  
    }  
}
```

RetrieveServlet.java

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

```
@SuppressWarnings("serial")

public class RetrieveServlet extends GenericServlet{

    public RetrieveDAO rd=null;

    public void init(){

        rd = new RetrieveDAO();

    }

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

        try{

PrintWriter pw = res.getWriter();

res.setContentType("text/html");

ResultSet rs = rd.retrieve(req);

if(rs.next()){

pw.println(rs.getString(1)+" &nbsp"+rs.getString(2)+" &nbsp"
        +rs.getString(3)+" &nbsp"+rs.getFloat(4)+" &nbsp"
        +rs.getInt(5));

}else{

pw.println("Invalid bookCode...");

}

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

}

public void destroy(){

try{

    Connection con = DBConnection.getCon();

    con.close();

}
```

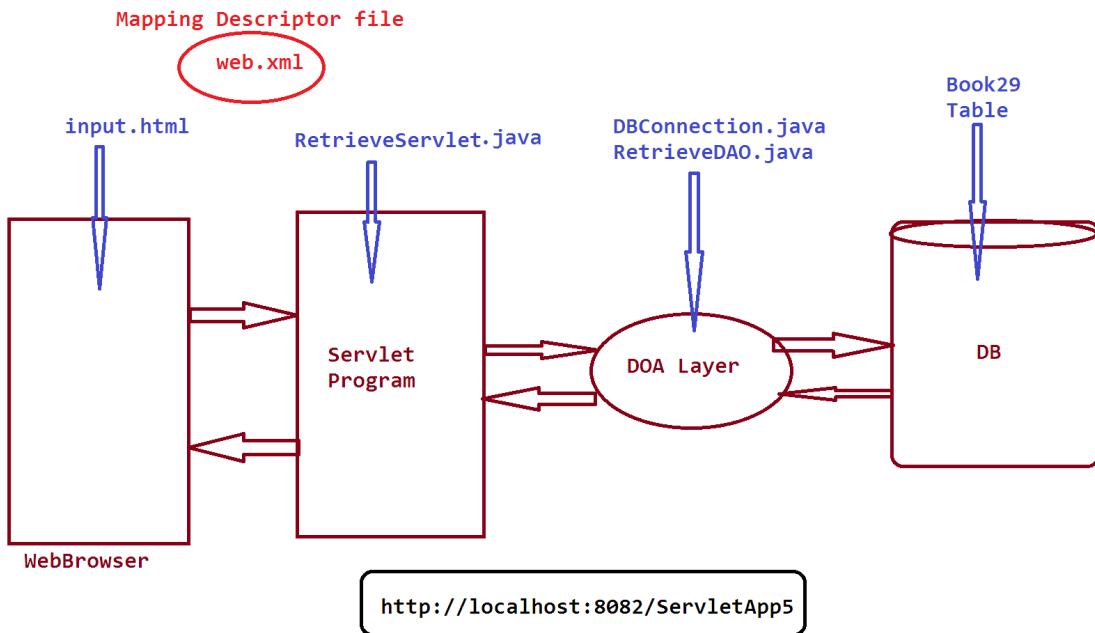
```
        }catch(Exception e){e.printStackTrace();}  
    }  
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app>  
    <servlet>  
        <servlet-name>RetrieveServlet</servlet-name>  
        <servlet-class>test.RetrieveServlet</servlet-class>  
    </servlet>  
    <servlet-mapping>  
        <servlet-name>RetrieveServlet</servlet-name>  
        <url-pattern>/retrieve</url-pattern>  
    </servlet-mapping>  
  
    <welcome-file-list>  
        <welcome-file>input.html</welcome-file>  
    </welcome-file-list>  
</web-app>
```

Execute the application as follows:

<http://localhost:8082/ServletApp5>



Assignment:

Servlet Application to retrieve product details based on pCode?

(DB Table - Product29)

Assignment based on DB Table : UserReg29

(i) Servlet Application to insert User details into UserReg29

**(ii) Servlet Application to retrieve User details based on
uname and pword**

Dt : 23/1/2021

*imp

Servlet Communications:

=>The process of establishing communication b/w ServletProgram to ServletProgram or ServletProgram to HTMLFile or ServletProgram to JSPFile is known as Servlet Communication.

=>These Servlet Communications are categorized into two types:

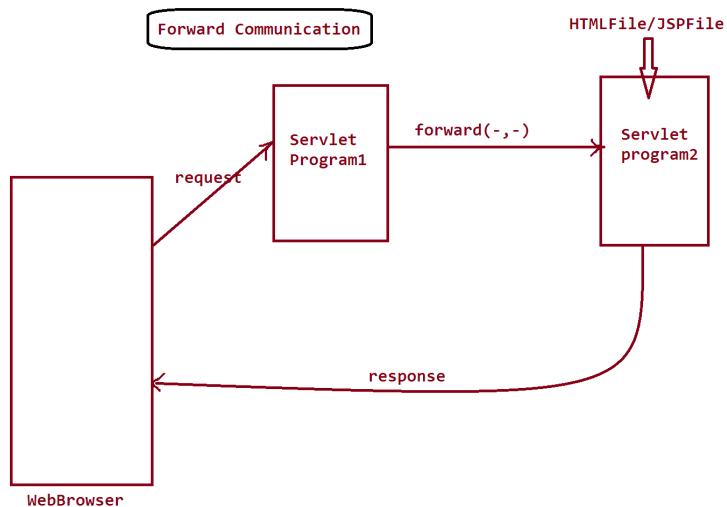
(a)forward communication

(b)include communication

(a)forward communication:

=>In forward communication process ServletProgram1 will take the request and forwards the request to the ServletProgram2,in this process the response is generated from ServletProgram2.

Diagram:



=>we use **forward()** method from '**javax.servlet.RequestDispatcher**' interface to perform forward communication.

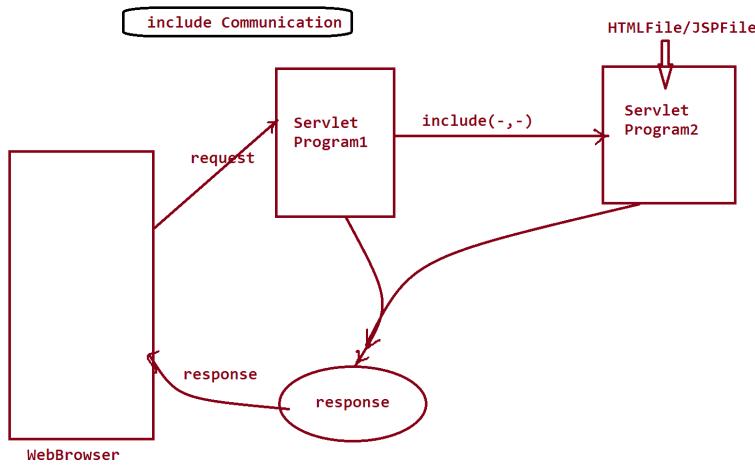
Method Signature:

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

(b)include communication:

=>In include communication process **ServletProgram1** will take the request and generates the response, but the response is added(included) with the response of **ServletProgram2**.

Diagram:



=>we use **include()** method from 'javax.servlet.RequestDispatcher' interface to perform include communication.

Method Signature:

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

Note:

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

Method Signature:

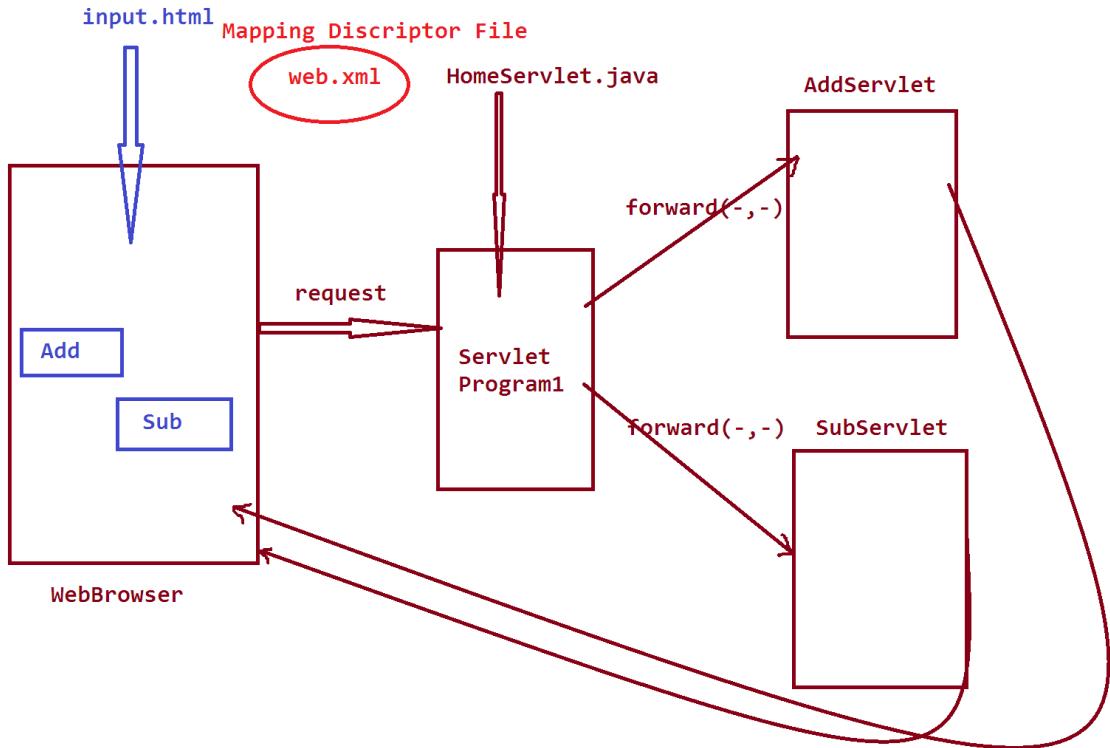
```
public abstract javax.servlet.RequestDispatcher getRequestDispatcher  
(java.lang.String);
```

syntax:

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

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

Exp Application:



input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
  <form action="home" method="post">
Enter the value1:<input type="text" name="v1"><br>
Enter the value2:<input type="text" name="v2"><br>
<input type="submit" value="Add" name="s1">
<input type="submit" value="Sub" name="s1">
  </form>
</body>
</html>
```

HomeServlet.java(home)

```
package test;

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

@SuppressWarnings("serial")
public class HomeServlet extends GenericServlet{

    public void init(){
        //NoCode
    }

    public void service(ServletRequest req,ServletResponse res) throws
    ServletException,IOException{
        String s1 = req.getParameter("s1");
        if(s1.equals("Add")){
            RequestDispatcher rd = req.getRequestDispatcher("pqr");
            rd.forward(req,res);
        }else{
            RequestDispatcher rd = req.getRequestDispatcher("xyz");
            rd.forward(req,res);
        }
    }

    public void destroy(){
        //NoCode
    }
}

AddServlet.java(pqr)
```

```
package test;

import java.io.*;

import javax.servlet.*;

@SuppressWarnings("serial")
public class AddServlet extends GenericServlet{

    PrintWriter pw = null;

    public void init(){

        //NoCode

    }

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

        try{

            pw = res.getWriter();

            res.setContentType("text/html");

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

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

            int z = x+y;

            pw.println("Sum:"+z+"<br>");

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

            rd.include(req,res);

        }catch(NumberFormatException nfe){

            pw.println("Enter Only Integer Values...<br>");

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

            rd.include(req,res);

        }

    }

}
```

```
}

public void destroy(){

    //NoCode

}

}
```

SubServlet.java(xyz)

```
package test;

import java.io.*;

import javax.servlet.*;

@SuppressWarnings("serial")

public class SubServlet extends GenericServlet{

    PrintWriter pw = null;

    public void init(){

        //NoCode

    }

    public void service(ServletRequest req,ServletResponse res) throws

    ServletException,IOException{

        try{

            pw = res.getWriter();

            res.setContentType("text/html");

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

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

            int z = x-y;

            pw.println("Sub:"+z+"<br>");

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


```

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

        }catch(NumberFormatException nfe){

pw.println("Enter only Integer values...<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>
    <servlet>
        <servlet-name>HomeServlet</servlet-name>
        <servlet-class>test.HomeServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HomeServlet</servlet-name>
        <url-pattern>/home</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>AddServlet</servlet-name>
        <servlet-class>test.AddServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>AddServlet</servlet-name>
        <url-pattern>/pqr</url-pattern>
    </servlet-mapping>
```

```

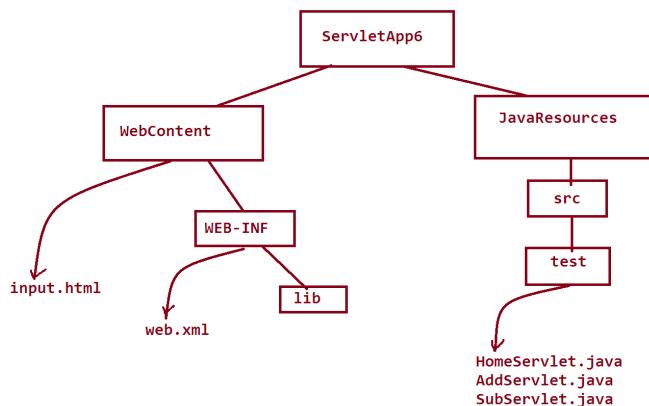
<servlet>
    <servlet-name>SubServlet</servlet-name>
    <servlet-class>test.SubServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>SubServlet</servlet-name>
    <url-pattern>/xyz</url-pattern>
</servlet-mapping>

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

```

Dt : 25/1/2021

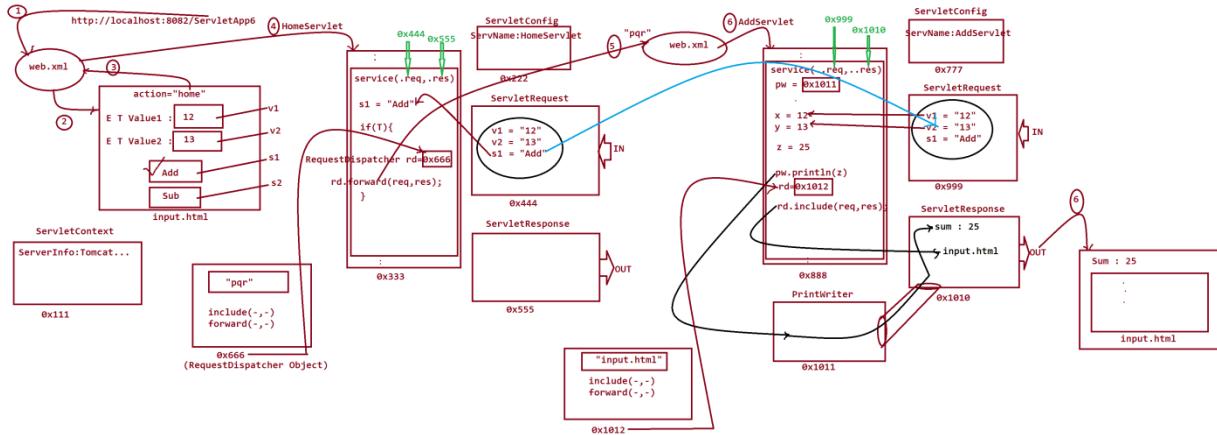
Deployment Directory Structure:



Execute the application as follows:

<http://localhost:8082/ServletApp6>

Execution flow of above application:



Note:

=>In forward communication process the data from request object of **ServletProgram1** will be loaded into the request object of **ServletProgram2**.

Assignment:

Update above application by adding the following operations:

MulServlet

DivServlet

ModDivServlet

Dt : 26/1/2021

*imp

"attribute" in Servlet Programming:

=>"attribute" is a variable in Servlet programming and which can be added to the ServletContext,ServletRequest and HttpSession.

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

- (i)setAttribute()
- (ii)getAttribute()
- (iii)removeAttribute()

(i)setAttribute():

=>setAttribute() method is used to add the attribute to the objects.

Method Signature:

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

(ii)getAttribute():

=>getAttribute() method is used to get the attribute from the objects.

Method Signature:

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

(iii)removeAttribute():

=>removeAttribute() method is used to delete the attribute from the objects.

Method Signature:

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

Note:

=>These methods are available in **ServletContext**,**ServletRequest** and **HttpSession**.

Scope of 'attribute':

=>'attribute' in **ServletContext** can be accessed by all the Servlet programs in WebApplication.

=>'attribute' in **ServletRequest** can be accessed by the next Servlet program in forward communication process.

=>'attribute' in **HttpSession** can be accessed with in the Session.

(User login to logout)

*imp

define Bean class?

=>The class which is constructed using the following rules is known as Bean Class:

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

Rule-2 : The Variables within the class must be 'private'.

Rule-3 : The Class must be declared with 0-argument constructor or 0-parameter constructor.

Rule-4 : The Class must be declared with 'setter methods' and 'getter methods'.

define Getter methods?

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

define **Setter** methods?

=>The methods which are used to set the data to the object are known as **Setter** methods.

Example Bean Class:

```
import java.io.*;  
  
public class Employee implements Serializable  
{  
    private String id,name;  
  
    public Employee(){}
  
    public void setId(String id)  
    {  
        this.id=id;  
    }
  
    public String getId()  
    {  
        return id;  
    }
  
    public void setName(String name)  
    {  
        this.name=name;  
    }
  
    public String getName()
```

```
{  
    return name;  
}  
}
```

define Bean Objects?

=>The Objects which are generated from the Bean classes are known as Bean objects.

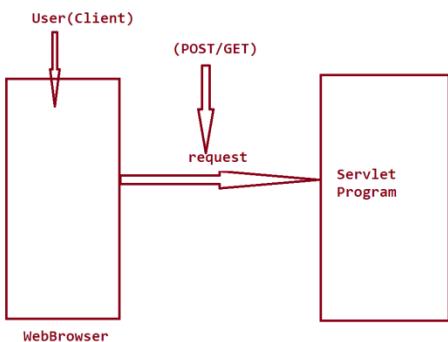
Note:

=>These Bean objects will hold data to be loaded onto DB Table and also these Bean objects hold data coming from the DB Table.

Dt : 27/1/2021

define Request?

=>The query which is generated by the user(client) from the WebBrowser is known as request.



=>The request is categorized into two types:

(a)POST Request

(b)GET Request

(a)POST request:

=>The request which is used to post(send) data to the Server
is known as POST request.

=>The POST request is generated by declaring method="POST" in
<form> tag of HTML file.

syntax:

```
<form action="url" method="post">
```

```
.
```

```
.
```

```
</form>
```

=>we use doPost() method from "javax.servlet.http.HttpServlet"
interface to hold POST request.

Method Signature:

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

Note:

=>Through POST request we can send any type of data(Text,Audio,Video, Image and Animation) and UnLimited data.

=>The data which is sent through the POST request is secured because the data is encapsulated to the body of HTTP Protocol.

(b)GET request:

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

=>The GET request is generated in the following ways:

(i)By declaring method="GET" in <form> tag of HTML file.

syntax:

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

(ii)By using Hyper Link

(iii)By using url-pattern in address bar

Exp:

<http://localhost:8082/ServletApp/pqr>

(iv)Submit the HTML form without declaring "method" attribute

syntax:

```
<form action="url">  
    .  
    .
```

</form>

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

Method Signature:

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

Note:

=>Through GET request we can send only text data and we can send only limited data upto 2kb or 4kb.(Based on Browser)

=>The data which is sent through the GET request is NotSecured because which is attached to the query-string and display in the address bar.

Note:

=>To differentiate the type of request to handle,the class must be extended from "HttpServlet".

Case-1 : Constructing Servlet program using 'HttpServlet'

=>The Class must be extended from 'javax.servlet.http.HttpServlet' and which provides the following LifeCycle methods:

- (i)init()
- (ii)doPost()/doGet()
- (iii)destroy()

=>doPost() method to hold POST request.

=>doGet() method to hold GET request.

*imp

Session Tracking process in Java:

define Session?

=>The time interval between login to logout of an user,is known as Session.

define Session Tracking process?

=>The process of tracking the state of user in the Session,is known as "Session Tracking process".

=>The following are the Session Tracking Techniques used in Servlet Programming:

1.Cookie

2.URL re-writing

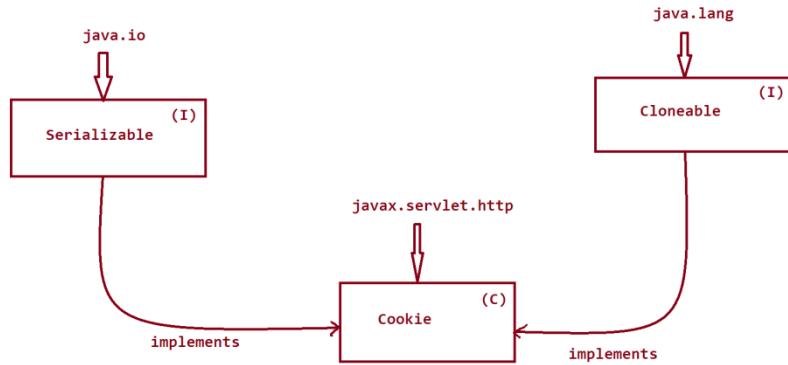
3.Hidden Form fields

4.HttpSession

1.Cookie:

=>'Cookie' is a class from 'javax.servlet.http' package and which is implemented from "Serializable" and "Cloneable" interfaces.

Hierarchy of Cookie:



The following are some important methods of Cookie:

```

public javax.servlet.http.Cookie(java.lang.String,java.lang.String);
public void setValue(java.lang.String);
public java.lang.String getValue();
public java.lang.String getName();
public void setMaxAge(int);
public int getMaxAge();

```

=>The following are the steps used in 'Cookie' Session Tracking process:

step1 : Creating Cookie

=>we pass information in the form of name and value while Object creation of Cookie.

syntax:

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

step2 : Adding Cookie to response

=>we use addCookie() method from "HttpServletResponse" to add the cookie to response

Method Signature:

```
public abstract void addCookie(javax.servlet.http.Cookie);
```

syntax:

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

step3 : Getting the Cookie from the request

=>we use getCookies() method from "HttpServletRequest" to get the cookie from the request object.

Method Signature:

```
public abstract javax.servlet.http.Cookie[] getCookies();
```

syntax:

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

step4 : Destroying the Cookie

=>we use setMaxAge() method from "Cookie" to destroy the Cookie.

syntax:

```
ck.setMaxAge(0);
```

Web Application(Project) : UserApp

DB Table : UserReg29

(uname,pword, fname, lname, addr, phno, mid)

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:1522:orcl","system","ma
nager");
        }catch(Exception e){e.printStackTrace();}
    }//end of block
    public static Connection getCon(){
        return con;
    }
}

```

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="Reg1.html">NewUser?</a>
    </form>
</body>
</html>

```

Reg1.html

```
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <form action="reg1" 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>
<input type="submit" value="NEXT">>>">
    </form>
</body>
</html>
```

Reg2.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <form action="reg2" method="post">
Address:<input type="text" name="addr"><br>
PhoneNo:<input type="text" name="phno"><br>
MailId:<input type="text" name="mid"><br>
<input type="submit" value="Register">
    </form>
</body>
</html>
```

RegBean.java

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

```
@SuppressWarnings("serial")
public class RegBean implements Serializable{
    private String uName, pWord, fName, lName, addr, mId;
    private long phNo;
    public RegBean(){}
    public final String getuName() {
        return uName;
    }
    public final void setuName(String uName) {
        this.uName = uName;
    }
    public final String getpWord() {
        return pWord;
    }
    public final void setpWord(String pWord) {
        this.pWord = pWord;
    }
    public final String getfName() {
        return fName;
    }
    public final void setfName(String fName) {
        this.fName = fName;
    }
    public final String getlName() {
        return lName;
    }
    public final void setlName(String lName) {
        this.lName = lName;
    }
    public final String getAddr() {
        return addr;
    }
    public final void setAddr(String addr) {
        this.addr = addr;
    }
}
```

```

}

public final String getmId() {
    return mId;
}

public final void setmId(String mId) {
    this.mId = mId;
}

public final long getPhNo() {
    return phNo;
}

public final void setPhNo(long phNo) {
    this.phNo = phNo;
}

}

```

InsertDAO.java

```

package test;
import java.sql.*;
public class InsertDAO {
    public int k=0;
    public int insert(RegBean rb){
        try{
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                ("insert into UserReg29 values(?,?,?,?,?,?)");
            ps.setString(1,rb.getuName());
            ps.setString(2,rb.getpWord());
            ps.setString(3,rb.getfName());
            ps.setString(4,rb.getlName());
            ps.setString(5,rb.getAddr());
            ps.setLong(6,rb.getPhNo());
            ps.setString(7,rb.getmId());
        }
    }
}

```

```
k = ps.executeUpdate();
    }catch(Exception e){e.printStackTrace();}
    return k;
}
}
```

Dt : 28/1/2021

RegServlet1.java(reg1)

```
package test;

import java.io.*;

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

@SuppressWarnings("serial")
public class RegServlet1 extends HttpServlet{
    public RegBean rb=null;
    public void init(){
        rb = new RegBean();
    }
    public void doPost(HttpServletRequest req,HttpServletResponse res)
        throws ServletException,IOException{
        rb.setuName(req.getParameter("uname"));
        rb.setpWord(req.getParameter("pword"));
        rb.setfName(req.getParameter("fname"));
        rb.setlName(req.getParameter("lname"));
        ServletContext sct = req.getServletContext();
        sct.setAttribute("bean",rb);
    }
}
```

```
RequestDispatcher rd = req.getRequestDispatcher("Reg2.html");
rd.forward(req,res);
}
public void destroy(){
//NoCode
}
}
```

RegServlet2.java(reg2)

```
package test;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
@SuppressWarnings("serial")
public class RegServlet2 extends HttpServlet{
    public ServletContext sct=null;
    public RegBean rb=null;
    public void init(){
        sct = this.getServletContext();
        rb = (RegBean)sct.getAttribute("bean");
    }
    public void doPost(HttpServletRequest req,HttpServletResponse res)
    throws ServletException,IOException{
    PrintWriter pw = res.getWriter();
    res.setContentType("text/html");
    rb.setAddr(req.getParameter("addr"));
    }
}
```

```

rb.setPhNo(Long.parseLong(req.getParameter("phno")));

rb.setId(req.getParameter("mid"));

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

if(k>0){

pw.println("User Registered Successfully...");

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

rd.include(req,res);

}

}

public void destroy(){

sct.removeAttribute("bean");

}

}

```

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <servlet>
    <servlet-name>RegServlet1</servlet-name>
    <servlet-class>test.RegServlet1</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>RegServlet1</servlet-name>
    <url-pattern>/reg1</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>RegServlet2</servlet-name>
    <servlet-class>test.RegServlet2</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>RegServlet2</servlet-name>

```

```
<url-pattern>/reg2</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>LoginServlet</servlet-name>
    <servlet-class>test.LoginServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/login</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>ViewServlet</servlet-name>
    <servlet-class>test.ViewServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>ViewServlet</servlet-name>
    <url-pattern>/view</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>LogoutServlet</servlet-name>
    <servlet-class>test.LogoutServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>LogoutServlet</servlet-name>
    <url-pattern>/logout</url-pattern>
</servlet-mapping>

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

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="edit">>EditProfile</a>
<a href="Logout">Logout</a>
</body>
</html>
```

LoginDAO.java

```
package test;

import java.sql.*;

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

public class LoginDAO {

    public boolean z=false;

    public boolean login(HttpServletRequest req){

        try{

Connection con = DBConnection.getCon();

PreparedStatement ps = con.prepareStatement

("select * from UserReg29 where uname=? and pword=?");

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

ps.setString(2,req.getParameter("pword"));

ResultSet rs = ps.executeQuery();

if(rs.next()){


```

```

RegBean rb = new RegBean();
rb.setuName(rs.getString(1));
rb.setpWord(rs.getString(2));
rb.setfName(rs.getString(3));
rb.setlName(rs.getString(4));
rb.setAddr(rs.getString(5));
rb.setPhNo(rs.getLong(6));
rb.setmId(rs.getString(7));

ServletContext sct = req.getServletContext();
sct.setAttribute("bean",rb);

z = true;
}

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

return z;

}

}

LoginServlet.java(login)

package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
@SuppressWarnings("serial")

public class LoginServlet extends HttpServlet{
    public void init(){
        //NoCode
    }
}

```

```
}

public void doPost(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException{
PrintWriter pw = res.getWriter();
res.setContentType("text/html");
boolean z = new LoginDAO().login(req);
if(z){
ServletContext sct = req.getServletContext();
RegBean rb =(RegBean)sct.getAttribute("bean");
Cookie ck = new Cookie("fName",rb.getfName());
res.addCookie(ck);
pw.println("WELCOME "+rb.getfName()+"<br>");
RequestDispatcher rd = req.getRequestDispatcher("Link.html");
rd.include(req,res);
}else{
pw.println("InValid UserName or PassWord....<br>");

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

public void destroy(){
//NoCode
}
```

ViewServlet.java(view)

```
package test;

import java.io.*;

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

@SuppressWarnings("serial")
public class ViewServlet extends HttpServlet{

    public 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("Please ! Login first...<br>");
            RequestDispatcher rd = req.getRequestDispatcher("Login.html");
            rd.include(req,res);
        }else{
            String fName = c[0].getValue();
            pw.println("User "+fName+"<br>");
            RequestDispatcher rd = req.getRequestDispatcher("Link.html");
            rd.include(req,res);
            ServletContext sct = req.getServletContext();
            RegBean rb = (RegBean)sct.getAttribute("bean");
            pw.println("<br>"+rb.getfName()+"&nbsp&nbsp"+rb.getlName()+
            "&nbsp&nbsp"+
            "&nbsp&nbsp"+
```

```
rb.getAddr()+" &nbsp"+rb.getPhNo()+" &nbsp"+rb.getmId());  
}  
}  
}
```

LogoutServlet.java(logout)

```
package test;  
  
import java.io.*;  
  
import javax.servlet.*;  
  
import javax.servlet.http.*;  
  
@SuppressWarnings("serial")  
  
public class LogoutServlet extends HttpServlet{  
  
    public 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("Please ! Login First...<br>");  
  
        }else{  
  
            c[0].setMaxAge(0);  
  
            req.getServletContext().removeAttribute("bean");  
  
            pw.println("User Logged out Successfully...<br>");  
  
        }  
  
        RequestDispatcher rd = req.getRequestDispatcher("Login.html");
```

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

Dt : 29/1/2021

UpdateDAO.java

```
package test;
import java.sql.*;
public class UpdateDAO {
    public int k=0;
    public int update(RegBean rb){
        try{
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                ("update UserReg29 set addr=?,phno=?,mid=? where
                 uname=? and pword=?");
            ps.setString(1,rb.getAddr());
            ps.setLong(2,rb.getPhNo());
            ps.setString(3,rb.getmId());
            ps.setString(4,rb.getuName());
            ps.setString(5,rb.getpWord());
            k = ps.executeUpdate();
        }catch(Exception e){e.printStackTrace();}
        return k;
    }
}
```

EditServlet.java(edit)

```
package test;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
@SuppressWarnings("serial")
public class EditServlet extends HttpServlet{
```

```
public 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("Please ! Login first...<br>");
RequestDispatcher rd = req.getRequestDispatcher("Login.html");
rd.include(req,res);
}else{
ServletContext sct = req.getServletContext();
RegBean rb = (RegBean)sct.getAttribute("bean");
pw.println("<form action='update' method='post'>");
pw.println
("Address:<input type='text' name='addr' value='"++
rb.getAddr()+"><br>");
pw.println("PhoneNo:<input type='text' name='phno' value='"++
rb.getPhNo()+"><br>");
pw.println("MailId:<input type='text' name='mid' value='"++
rb.getId()+"><br>");
pw.println("<input type='submit' value='UpdateProfile'>");
pw.println("</form>");
}
}}
```

UpdateServlet.java(update)

```
package test;

import java.io.*;

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

@SuppressWarnings("serial")
public class UpdateServlet extends HttpServlet{

    public void doPost(HttpServletRequest req,HttpServletResponse res)
        throws ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        Cookie c[] = req.getCookies();
        if(c==null){
            pw.println("Please ! Login first...<br>");
            RequestDispatcher rd = req.getRequestDispatcher("Login.html");
            rd.include(req,res);
        }else{
            String fName = c[0].getValue();
            pw.println("USER "+fName+"<br>");
            RequestDispatcher rd = req.getRequestDispatcher("Link.html");
            rd.include(req,res);
            ServletContext sct = req.getServletContext();
            RegBean rb = (RegBean)sct.getAttribute("bean");
            rb.setAddr(req.getParameter("addr"));
            rb.setPhNo(Long.parseLong(req.getParameter("phno")));
            rb.setId(req.getParameter("mid"));
        }
    }
}
```

```
int k = new UpdateDAO().update(rb);

    if(k>0){

pw.println("<br>Profile updated Successfully...");

    }

}

}
```

web.xml(Updated)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <servlet>
        <servlet-name>RegServlet1</servlet-name>
        <servlet-class>test.RegServlet1</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>RegServlet1</servlet-name>
        <url-pattern>/reg1</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>RegServlet2</servlet-name>
        <servlet-class>test.RegServlet2</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>RegServlet2</servlet-name>
        <url-pattern>/reg2</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>test.LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
```

```
<url-pattern>/login</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>ViewServlet</servlet-name>
    <servlet-class>test.ViewServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>ViewServlet</servlet-name>
    <url-pattern>/view</url-pattern>
</servlet-mapping>

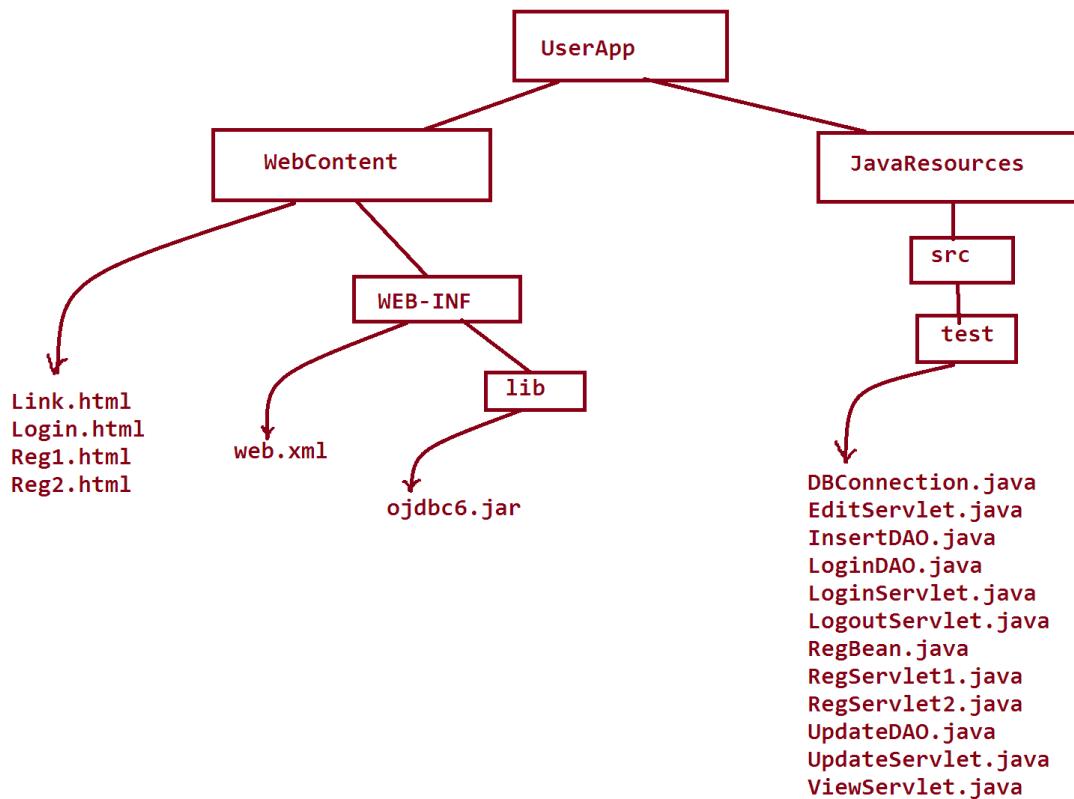
<servlet>
    <servlet-name>LogoutServlet</servlet-name>
    <servlet-class>test.LogoutServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>LogoutServlet</servlet-name>
    <url-pattern>/logout</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>EditServlet</servlet-name>
    <servlet-class>test.EditServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>EditServlet</servlet-name>
    <url-pattern>/edit</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>UpdateServlet</servlet-name>
    <servlet-class>test.UpdateServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>UpdateServlet</servlet-name>
    <url-pattern>/update</url-pattern>
</servlet-mapping>
```

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

Deployment Directory Structure:(IDE Eclipse)



Execute the application as follows:

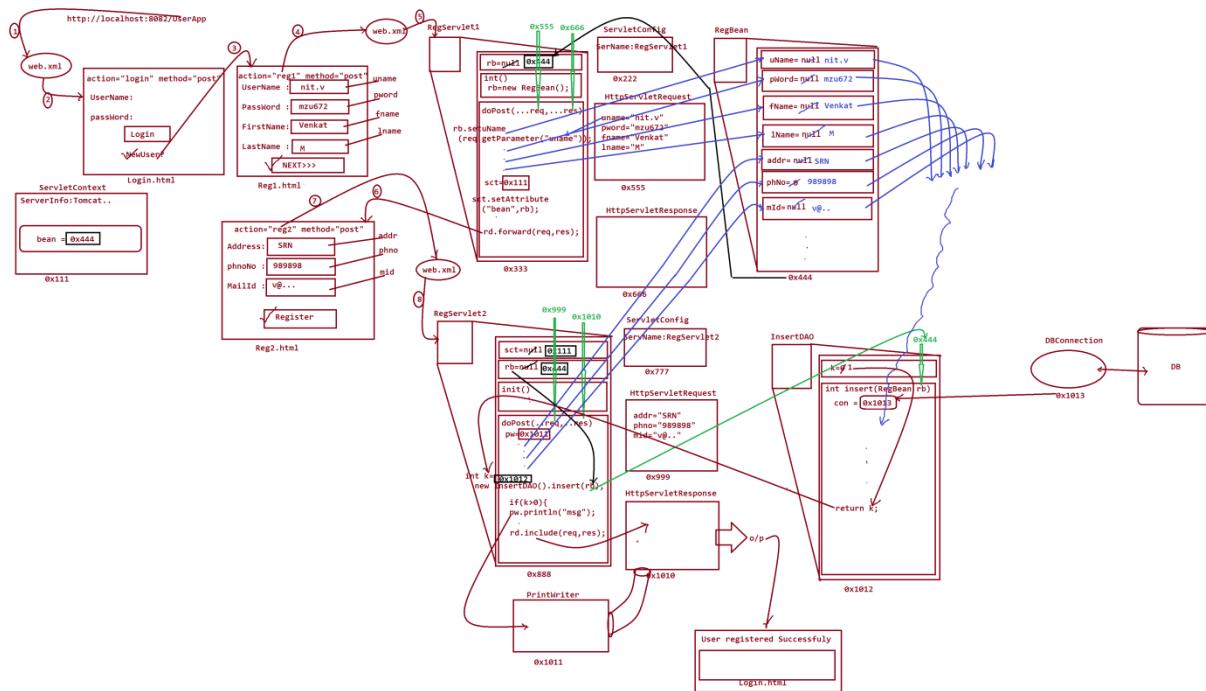
<http://localhost:8082/UserApp>

Dt : 1/2/2021

Execution flow of above application:

part-1 : User Registration process

<http://localhost:8082/UserApp>



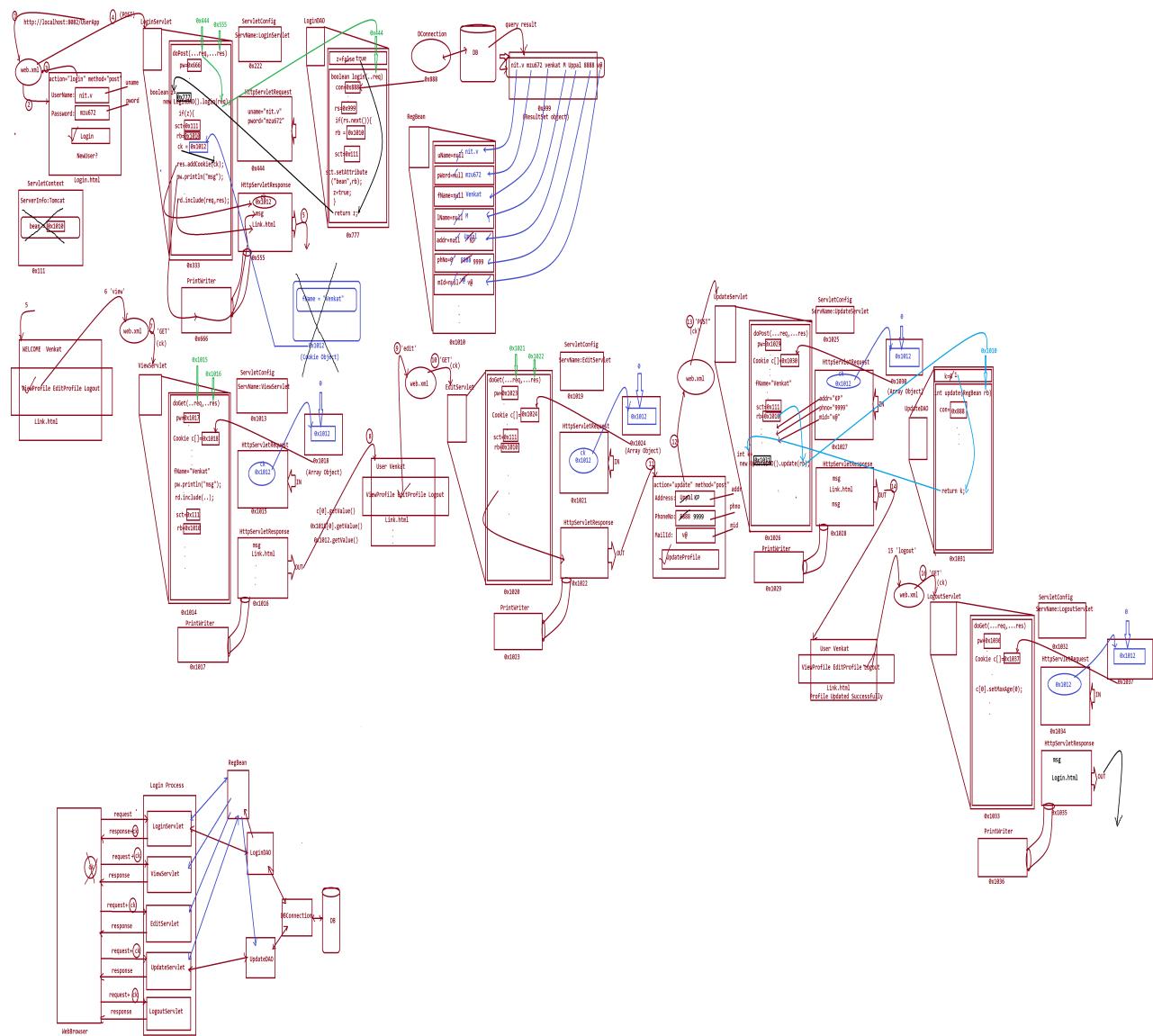
Dt : 2/2/2021

Execution flow of above application:

part-2 : User Login process

<http://localhost:8082/UserApp>

Dt : 3/2/2021



Dt : 4/2/2021

Note:

=>cookie is stored in WebBrowser and added for all the multiple requests generated from the EndUser(Client)

faq:

wt is diff b/w

(i)ServletContext

(ii)ServletConfig

=>ServletContext is instantiated when WebAppl deployed into Server, but ServletConfig is instantiated when the Servlet program is loaded for execution.

=>Only one ServletContext object is created for WebAppl, but every Servlet in the WebAppl will have its own ServletConfig object.

=>ServletContext object will hold Server information, but ServletConfig object will hold ServletName.

Note:

=>This ServletContext can be used by all the multiple Servlet programs of WebAppl.

Assignment using "Cookie" in Session Tracking:

WebApplication Name : OnlineBookStore

DBTables : UserReg29,Admin29,Book29

UserReg29

(uname,pword,fname,lname,addr,phno,mid)

Admin29

(uname,pword,fname,lname,addr,phno,mid)

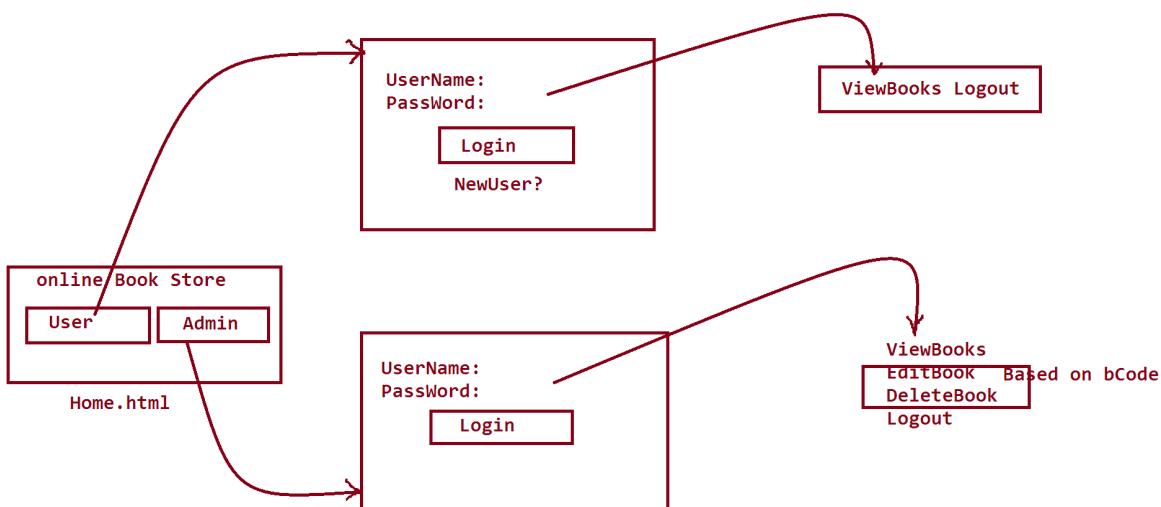
Book29

(bcode,bname,bauthor,bprice,bqty)

Note:

=>UserReg29 and Book29 must be empty(0 records)

=>Insert one record to Admin29 from SQL CommandLine



=====

Dt : 5/2/2021

2.URL re-writing:

=>The process of adding para-value to the Servlet url_pattern is known as URL re-writing process.

Syntax:

url_pattern?para=value¶=value&...

Note:

=>Using URL re-writing process we can send data from one Servlet to another Servlet in Servlet Communication process.

Exp application:

WebAppName : URWrite

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:1522:orcl","system","ma
nager");
        }catch(Exception e){e.printStackTrace();}
    }//end of block
    public static Connection getCon(){
        return con;
    }
}
```

EmpBean.java

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class EmpBean implements Serializable{
    private String eId,eName,eDesg;
    private int eSal;
    public EmpBean(){}
    public final String geteId() {
        return eId;
    }
    public final void seteId(String eId) {
        this.eId = eId;
    }
    public final String geteName() {
        return eName;
    }
    public final void seteName(String eName) {
        this.eName = eName;
    }
    public final String geteDesg() {
        return eDesg;
    }
    public final void seteDesg(String eDesg) {
        this.eDesg = eDesg;
    }
    public final int geteSal() {
        return eSal;
    }
    public final void seteSal(int eSal) {
        this.eSal = eSal;
    }
}
```

LoginDAO.java

```
package test;

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

public class LoginDAO {

    public String fName=null;

    public String login(HttpServletRequest req){

        try{

            Connection con = DBConnection.getCon();

            PreparedStatement ps = con.prepareStatement
                ("select * from UserReg29 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);

                PreparedStatement ps2 = con.prepareStatement
                    ("select * from Emp29");

                ResultSet rs2 = ps2.executeQuery();

                ArrayList<EmpBean> al= new ArrayList<EmpBean>();

                while(rs2.next()){

                    EmpBean eb = new EmpBean();

                    eb.seteId(rs2.getString(1));

                    eb.seteName(rs2.getString(2));

```

```

eb.seteDesg(rs2.getString(3));
eb.seteSal(rs2.getInt(4));
al.add(eb);
}//end of while
req.getServletContext().setAttribute("list",al);
}//end of if
}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="Login" method="post">
UserName:<input type="text" name="uname"><br>
Password:<input type="text" name="pword"><br>
<input type="submit" value="Login">
    </form>
</body>
</html>

```

LoginServlet.java(login)

```

package test;

```

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
@SuppressWarnings("serial")
public class LoginServlet extends HttpServlet{
    @SuppressWarnings("unchecked")
    public void doPost(HttpServletRequest req,HttpServletResponse res)
        throws ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        String fName = new LoginDAO().login(req);
        if(fName==null){
            pw.println("Invalid UserName or PassWord...<br>");
            RequestDispatcher rd = req.getRequestDispatcher("Login.html");
            rd.include(req,res);
        }else{
            pw.println("WELCOME "+fName+"<br>");
            ArrayList<EmpBean> al = (ArrayList<EmpBean>)
                req.getServletContext().getAttribute("list");
            if(al.size()==0){
                pw.println("No records available...<br>");
            }else{
                al.forEach((k)->
{

```

```

    EmpBean eb = (EmpBean)k;

pw.println("<a href='dis?eid="+eb.getId()+"'>"+eb.getId()+"</a>");

});

}//end of else

}//end of else

}

}

```

DisServlet.java(dis)

```

package test;

import java.io.*;
import java.util.*;
import java.util.stream.Collectors;
import javax.servlet.*;
import javax.servlet.http.*;

@SuppressWarnings("serial")
public class DisServlet extends HttpServlet{
    @SuppressWarnings({ "unchecked", "rawtypes" })
    public void doGet(HttpServletRequest req,HttpServletResponse res)
        throws ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        String eid = req.getParameter("eid");
        ArrayList<EmpBean> al = (ArrayList<EmpBean>)
            req.getServletContext().getAttribute("list");
        List l = al.stream().filter((p)->p.getId().equals(eid));

```

```

        collect(Collectors.toList());

pw.println("Emp details based on eid...<br>");

l.forEach(z->

{
    EmpBean eb = (EmpBean)z;

pw.println(eb.getId()+"  "+eb.getName()+"  "+
eb.getDesg()+"  "+eb.getSal());

});

}
}

```

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>test.LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>

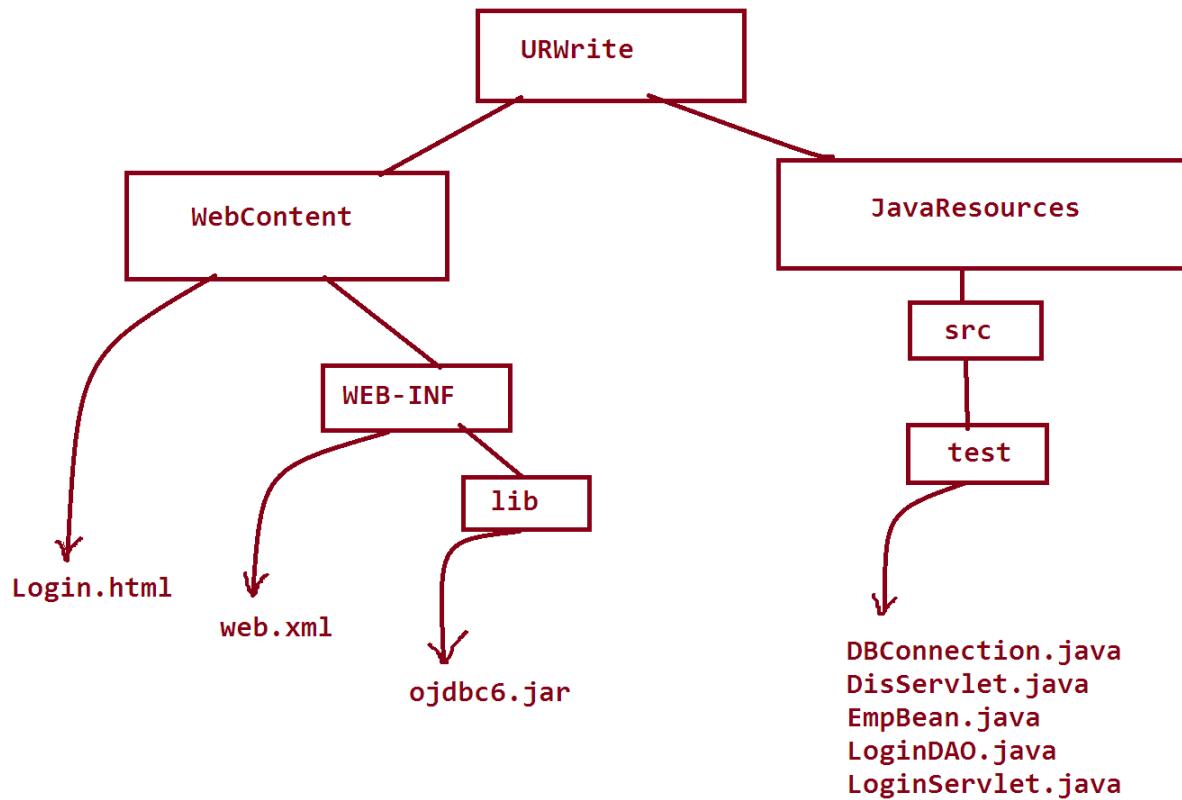
    <servlet>
        <servlet-name>DisServlet</servlet-name>
        <servlet-class>test.DisServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>DisServlet</servlet-name>
        <url-pattern>/dis</url-pattern>
    </servlet-mapping>

    <welcome-file-list>

```

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

Deployment directory Structure:

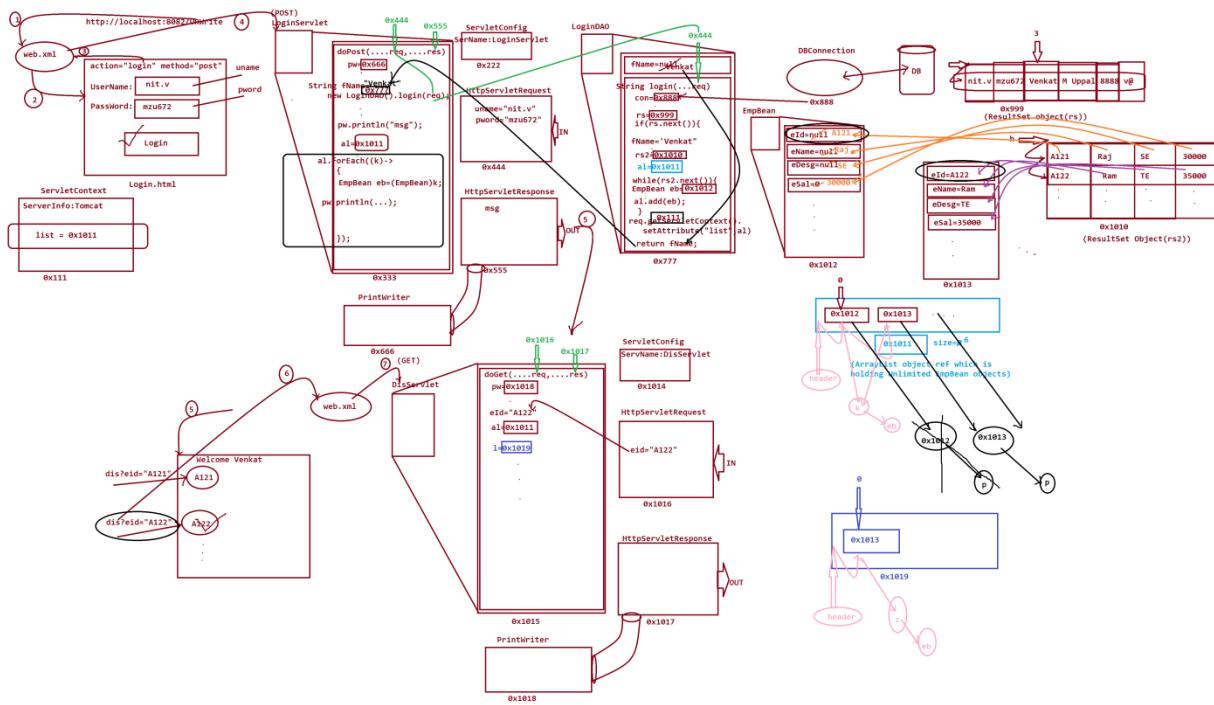


Execute the application as follows:

<http://localhost:8082/URWrite>

Dt : 8/2/2021

Execution flow of above application:



define forEach() method?

=>forEach() method introduced by Java8 version and which is used to retrieve elements from Collection and Map objects.

Method Signature of forEach() on Collection<T>:

```
public void forEach(java.util.function.Consumer<? super T>);
```

Method Signature of forEach() on Map<K,V>:

```
public void forEach  
(java.util.function.BiConsumer<? super K,? super V>);
```

define Consumer<T>?

=>**Consumer<T>** is a functional interface introduced by Java8 version and which provides the abstract method signature to hold LambdaExpression passed as parameter to the forEach() method on Collection<T> object.

Structure of Consumer<T>:

```
public interface java.util.function.Consumer<T>
{
    public abstract void accept(T);
}
```

define BiConsumer<T>?

=>**BiConsumer<T>** is a functional interface introduced by Java8 version and which provides the abstract method signature to hold LambdaExpression passed as parameter to the forEach() method on Map<K,V> object.

Structure of BiConsumer<T,U>:

```
public interface java.util.function.BiConsumer<T,U>
{
    public abstract void accept(T,U);
}
```

Dt : 9/2/2021

<http://localhost:8082/URWrite/dis?eid=A122>

**https://accounts.google.com/ServiceLogin/
signinchooser?service=mail
&passive=true
&rm=false
&continue=https%3A%2F%2Fmail.google.com%2Fmail%2F
&ss=1
&scc=1
<mpl=default
<mplcache=2
&emr=1
&osid=1
&flowName=GlfWebSignIn
&flowEntry=ServiceLogin**

Note:

**=>The para-value which is sent through url-pattern is available in
the request object of Second Servlet.**

3.Hidden Form fields

**=>The process of declaring "hidden form field" part of HTML code and
using this "hidden form field" we can pass para-value from one Servlet
to another Servlet is known as "Hidden form field".**

syntax:

```
<form action="url" method="post/get">
```

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

Note:

=>The value which is assigned for hidden form field will not be displayed to the EndUser on WebBrowser.

Dt : 10/2/2021

Exp Program:

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>
```

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:1522:orcl","system","ma
nager");
        }catch(Exception e){e.printStackTrace();}
    }//end of block
    public static Connection getCon(){
        return con;
```

```
    }  
}
```

LoginDAO.java

```
package test;  
  
import java.sql.*;  
  
import javax.servlet.http.*;  
  
public class LoginDAO {  
  
    public String fName=null;  
  
    public String login(HttpServletRequest req){  
  
        try{  
  
            Connection con = DBConnection.getCon();  
  
            PreparedStatement ps = con.prepareStatement  
("select * from UserReg29 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);  
  
            }//end of if  
  
            }catch(Exception e){  
  
                e.printStackTrace();  
  
            }  
  
            return fName;  
    }  
}
```

```
}
```

LoginServlet.java(login)

```
package test;

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

@SuppressWarnings("serial")
public class LoginServlet extends HttpServlet{
    public void doPost(HttpServletRequest req,HttpServletResponse res)
        throws ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        String fName = new LoginDAO().login(req);
        if(fName==null){
            pw.println("Invalid UserName or PassWord...<br>");
            RequestDispatcher rd = req.getRequestDispatcher("Login.html");
            rd.include(req,res);
        }else{
            pw.println("<form action='second' method='post'>");
            pw.println("<input type='hidden' name='fname' value='"+fName+"'><br>");
            pw.println("<input type='submit' value='Display'>");
            pw.println("</form>");
        }
    }
}
```

```
}
```

SecondServlet.java(second)

```
package test;

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

@SuppressWarnings("serial")
public class SecondServlet extends HttpServlet{

    public void doPost(HttpServletRequest req,HttpServletResponse res)
        throws ServletException,IOException{
        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");
        String fName = req.getParameter("fname");
        pw.println("WELCOME "+fName);
    }
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>test.LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>

    <servlet>
```

```
<servlet-name>SecondServlet</servlet-name>
<servlet-class>test.SecondServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>SecondServlet</servlet-name>
    <url-pattern>/second</url-pattern>
</servlet-mapping>

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

Execute the application as follows:

http://localhost:8082/HiddenFormField

Note:

=>The para-value which is sent through "Hidden form fields" are available in request object of second Servlet.

=====

*imp

4.HttpSession:

=>HttpSession is an interface from "javax.servlet.http" package and which is instantiated to hold information used in Session Tracking process.

=>The following are some important methods of HttpSession:

```
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();
```

Note:

=>we use **getSession()** method from "HttpServletRequest" to instantiate **HttpSession**.

Method Signature:

```
public abstract javax.servlet.http.HttpSession getSession(boolean);  
public abstract javax.servlet.http.HttpSession getSession();
```

Dt : 11/2/2021

WebAppName : HttpSessionApp

DB Table : Product29

(PCode,pname,pprice,pqty)

(Insert min 5 records)

=>when the login is successful then the user can view the products.

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>
```

Link.html

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

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:1522:orcl","system","ma
nager");
    }catch(Exception e){e.printStackTrace();}
}//end of block
public static Connection getCon(){
    return con;
}
}

```

LoginDAO.java

```

package test;

import java.sql.*;
import javax.servlet.http.*;
public class LoginDAO {
    public String fName=null;
    public String login(HttpServletRequest req){
        try{
Connection con = DBConnection.getCon();
PreparedStatement ps = con.prepareStatement
("select * from UserReg29 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);

}//end of if

}catch(Exception e){

    e.printStackTrace();

}

return fName;

}
```

LoginServlet.java(login)

```
package test;

import java.io.*;

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

@SuppressWarnings("serial")
public class LoginServlet extends HttpServlet{

    public void doPost(HttpServletRequest req,HttpServletResponse res)
        throws ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        String fName = new LoginDAO().login(req);

        if(fName==null){
            pw.println("Invalid UserName or PassWord...<br>");
            RequestDispatcher rd = req.getRequestDispatcher("Login.html");
            rd.include(req,res);
        }else{
```

```

HttpSession hs = req.getSession();

hs.setAttribute("fname", fName);

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

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

rd.include(req,res);

}//end of else

}

}

```

Product.java(BeanClass)

```

package test;
import java.io.*;
@SuppressWarnings("serial")
public class Product implements Serializable{
    private String pCode,pName;
    private float pPrice;
    private int pQty;
    public Product(){}
    public final String getpCode() {
        return pCode;
    }
    public final void setpCode(String pCode) {
        this.pCode = pCode;
    }
    public final String getpName() {
        return pName;
    }
    public final void setpName(String pName) {
        this.pName = pName;
    }
}

```

```
public final float getpPrice() {
    return pPrice;
}
public final void setpPrice(float pPrice) {
    this.pPrice = pPrice;
}
public final int getpQty() {
    return pQty;
}
public final void setpQty(int pQty) {
    this.pQty = pQty;
}

}
```

RetriveDAO.java

```
package test;

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

public class RetriveDAO {

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

    public ArrayList<Product> retrieve(){

        try{

            Connection con = DBConnection.getCon();

            PreparedStatement ps = con.prepareStatement
                ("select * from Product29");

            ResultSet rs = ps.executeQuery();

            while(rs.next()){

                Product p = new Product();

```

```
p.setpCode(rs.getString(1));
p.setpName(rs.getString(2));
p.setpPrice(rs.getFloat(3));
p.setpQty(rs.getInt(4));
al.add(p);
}//end of loop

}catch(Exception e){e.printStackTrace();}
return al;
}
```

ViewServlet.java(view)

```
package test;

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

@SuppressWarnings("serial")
public class ViewServlet extends HttpServlet{

    public void doGet(HttpServletRequest req,HttpServletResponse res)
        throws ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        HttpSession hs = req.getSession(false);
        if(hs==null){
            pw.println("Please ! Login first...<br>");
        }
    }
}
```

```
RequestDispatcher rd = req.getRequestDispatcher("Login.html");
rd.include(req,res);
}else{
String fName = (String)hs.getAttribute("fname");
ArrayList<Product> al = new RetrieveDAO().retrieve();
pw.println("User page of "+fName+"<br>");
al.forEach((k)->
{
    Product p = (Product)k;
    pw.println(p.getCode()+" &ampnbsp"+p.getName()+" &ampnbsp"+
p.getPrice()+" &ampnbsp&ampnbsp"+p.getQty()+"<br>");

});
pw.println("<a href='logout'>Logout</a>");
}
}
```

LogoutServlet.java(logout)

```
package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
@SuppressWarnings("serial")
public class LogoutServlet extends HttpServlet{
    public void doGet(HttpServletRequest req,HttpServletResponse res)
    throws ServletException,IOException{
```

```

PrintWriter pw = res.getWriter();

res.setContentType("text/html");

HttpSession hs = req.getSession(false);

if(hs==null){

pw.println("Please ! Login first...<br>");

}else{

hs.invalidate();

pw.println("Logged Out Successfully...<br>");

}

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

rd.include(req,res);

}

}

```

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>test.LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>ViewServlet</servlet-name>
        <servlet-class>test.ViewServlet</servlet-class>
    
```

```
</servlet>
<servlet-mapping>
    <servlet-name>ViewServlet</servlet-name>
    <url-pattern>/view</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>LogoutServlet</servlet-name>
    <servlet-class>test.LogoutServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>LogoutServlet</servlet-name>
    <url-pattern>/logout</url-pattern>
</servlet-mapping>

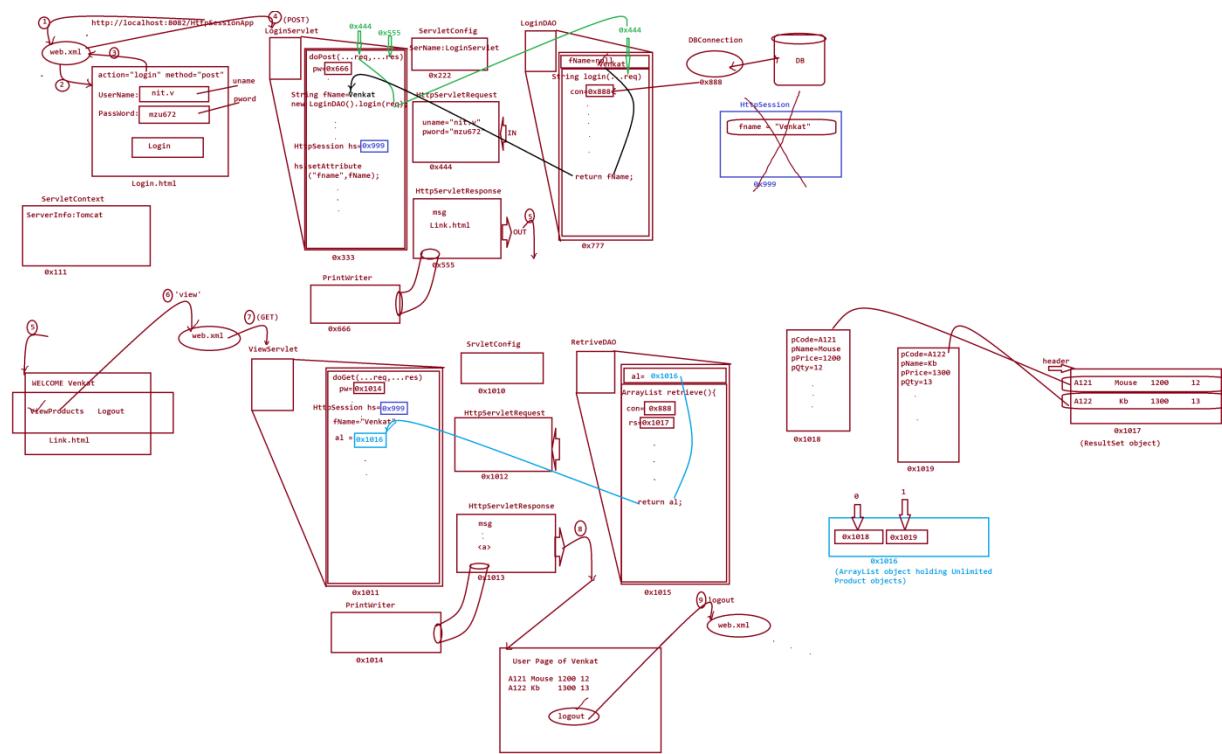
<welcome-file-list>
    <welcome-file>Login.html</welcome-file>
</welcome-file-list>
</web-app>
```

Execute the appl as follows:

<http://localhost:8082/HttpSessionApp>

dt : 12/2/2021

Execution flow of above application:



Dt : 13/2/2021

Cookie in Session Tracking:

=>Cookie Session Tracking process is WebBrowser dependent,which

means the Cookies are stored in WebBrowser.

DisAdvantage:

=>when we disable Cookies from WebBrowser then the Cookie Session projects will not work.

HttpSession in Session Tracking:

=>HttpSession session tracking process is Server depenedent,which means the HttpSession is available within the Server.

faq:

wt is the diff b/w

(i)getSession()

(ii)getSession(boolean)

(i)getSession():

=>getSession() method will search the HttpSession object,

=>If it is available access the reference of HttpSession object.

=>If not available create the new reference of HttpSession.

(ii)getSession(boolean):

=>getSession(false)

=>This method is used search the HttpSession object,

=>If it is available access the reference of HttpSession object.

=>If not available the new reference is not created.

=>getSession(true)

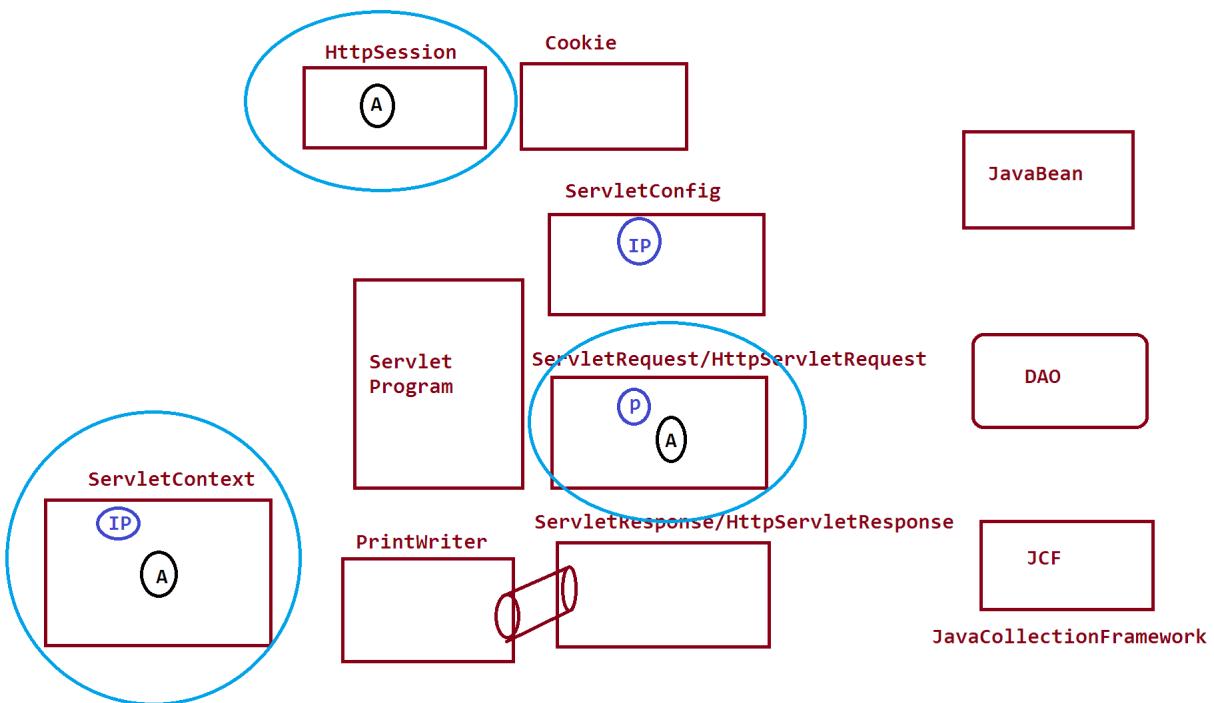
=>This method is used search the HttpSession object,

=>If it is available access the reference of HttpSession object.

=>If not available the new reference is created.

Summary of Objects in Servlet Programming:

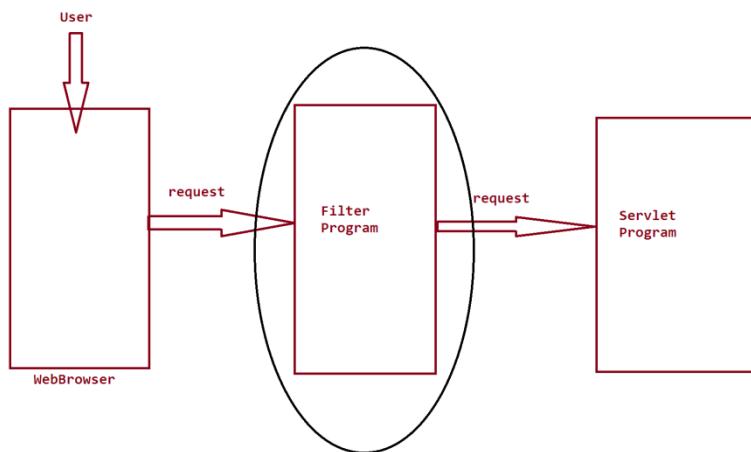
- 1.ServletContext**
- 2.ServletConfig**
- 3.ServletRequest/HttpServletRequest**
- 4.ServletResponse/HttpServletResponse**
- 5.PrintWriter**
- 6.Cookie**
- 7.HttpSession**
- 8.JavaBean**
- 9.DAO**
- 10.JCF**



*imp

Filters in Servlet programming:

=>Filter is a pre-processing component and which is executed before
Servlet program.



Use of Filters in Servlet programming:

- (i)Filters are used for recording all incoming requests.**
 - (ii)Filter "logs" the IP address of the computer from where the request is generated.**
 - (iii)Filters are used for conversion process.**
 - (iv)Filters are used for data compression process.**
 - (v)Filters used for encryption and decryption process.**
 - (vi)Filters are also used for input validation process.**
-

=>The following are the components used in constructing Filter programs:

- (a)Filter**
- (b)FilterChain**
- (c)FilterConfig**

(a)Filter:

=>'Filter' is an interface from javax.servlet package and which provides the following LifeCycle methods in constructing Filter programs

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

(i)init():

=>init() method perform initialization process.

Method Signature:

public void init(javax.servlet.FilterConfig)

```
throws javax.servlet.ServletException;
```

(ii)doFilter():

=>doFilter() method will provide service by accepting the request and providing the response and this method also provide link to next Servlet program using "FilterChain".

Method Signature:

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

(iii)destroy():

=>destroy() method will perform destroying process.

(closing the resources)

Method Signature:

```
public void destroy();
```

Note:

=>In the process of constructing Filter program the user defined class must be implemented from "Filter".

(b)FilterChain:

=>'FilterChain' is an interface from javax.servlet package and which provides "doFilter()" method to link the Servlet program.

Method Signature:

```
public abstract void doFilter(javax.servlet.ServletRequest,
```

```
javax.servlet.ServletResponse) throws java.io.IOException,  
        javax.servlet.ServletException;
```

(c)FilterConfig:

=>'FilterConfig' is an interface from javax.servlet package and which is instantiated automatically when Filter program is loaded onto WebContainer and this FilterConfig object is loaded with Filter name.

Note:

=>we use the following tags in web.xml related to FilterProgram:

```
<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>
```

Exp Application on Filter:

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>
```

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:1522:orcl","system","ma
nager");
        }
    }
}
```

```
        }catch(Exception e){e.printStackTrace();}  
    }//end of block  
    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 UserReg29 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);  
  
            } //end of if  
  
        }catch(Exception e){  
  
            e.printStackTrace();  
        }  
    }  
}
```

```
    return fName;  
}  
}
```

LoginFilter.java(login)

```
package test;  
  
import java.io.*;  
  
import javax.servlet.*;  
  
public class LoginFilter implements Filter{  
  
    public FilterConfig fc;  
  
    public LoginDAO Id;  
  
    public void init(FilterConfig fc) throws ServletException{  
  
        this.fc=fc;  
  
        Id = new LoginDAO();  
    }  
  
    public void doFilter(ServletRequest req, ServletResponse res,  
                        FilterChain chain) throws ServletException, IOException{  
  
        PrintWriter pw = res.getWriter();  
  
        res.setContentType("text/html");  
  
        String fName = Id.login(req);  
  
        if(fName==null){  
  
            pw.println("Please ! Login first...<br>");  
  
            RequestDispatcher rd = req.getRequestDispatcher("Login.html");  
  
            rd.include(req,res);  
  
        }else{  
  
            int a = Integer.parseInt(fc.getInitParameter("a"));  
        }  
    }  
}
```

```
req.setAttribute("fname",fName);

req.setAttribute("val",new Integer(a));

chain.doFilter(req,res);

}

}

public void destroy(){

//NoCode

}

}
```

WelcomeServlet.java(login)

```
package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

public class WelcomeServlet extends HttpServlet{

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

    PrintWriter pw = res.getWriter();

    res.setContentType("text/html");

    String fName = (String)req.getAttribute("fname");

    Integer a = (Integer)req.getAttribute("val");

    pw.println("WELCOME "+fName+"  
");

    pw.println("Value:"+a);

    }

}
```

```
}
```

web.xml

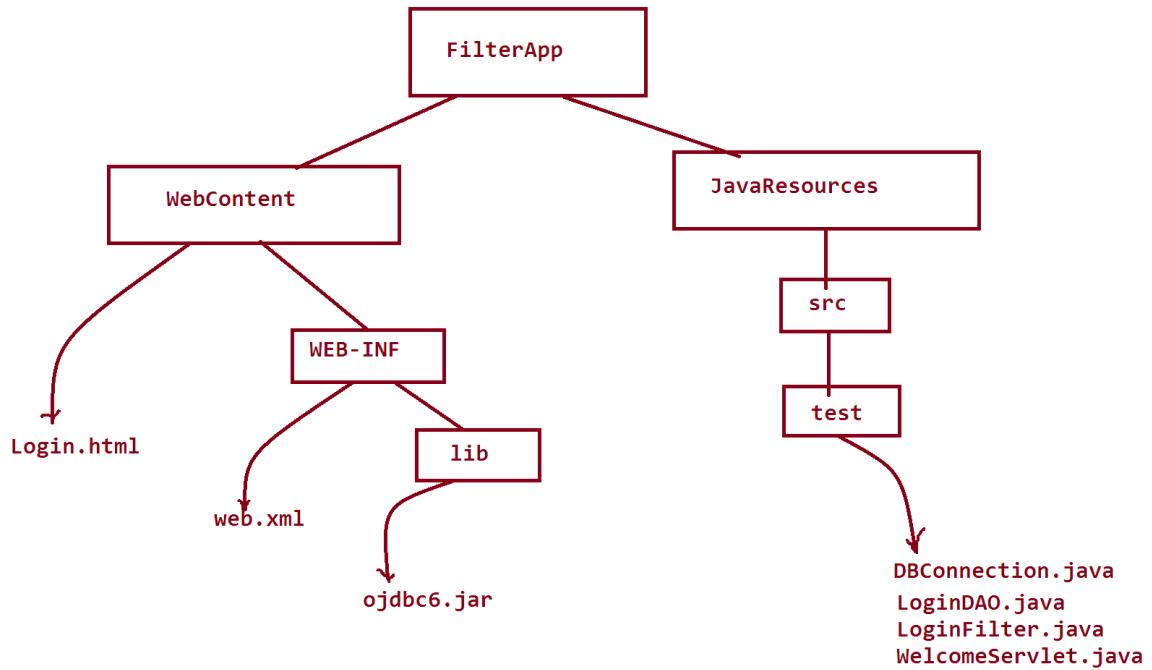
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <servlet>
        <servlet-name>WelcomeServlet</servlet-name>
        <servlet-class>test.WelcomeServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>WelcomeServlet</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>

    <filter>
        <filter-name>LoginFilter</filter-name>
        <filter-class>test.LoginFilter</filter-class>
        <init-param>
            <param-name>a</param-name>
            <param-value>2000</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>LoginFilter</filter-name>
        <url-pattern>/login</url-pattern>
    </filter-mapping>

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

Dt : 15/2/2021

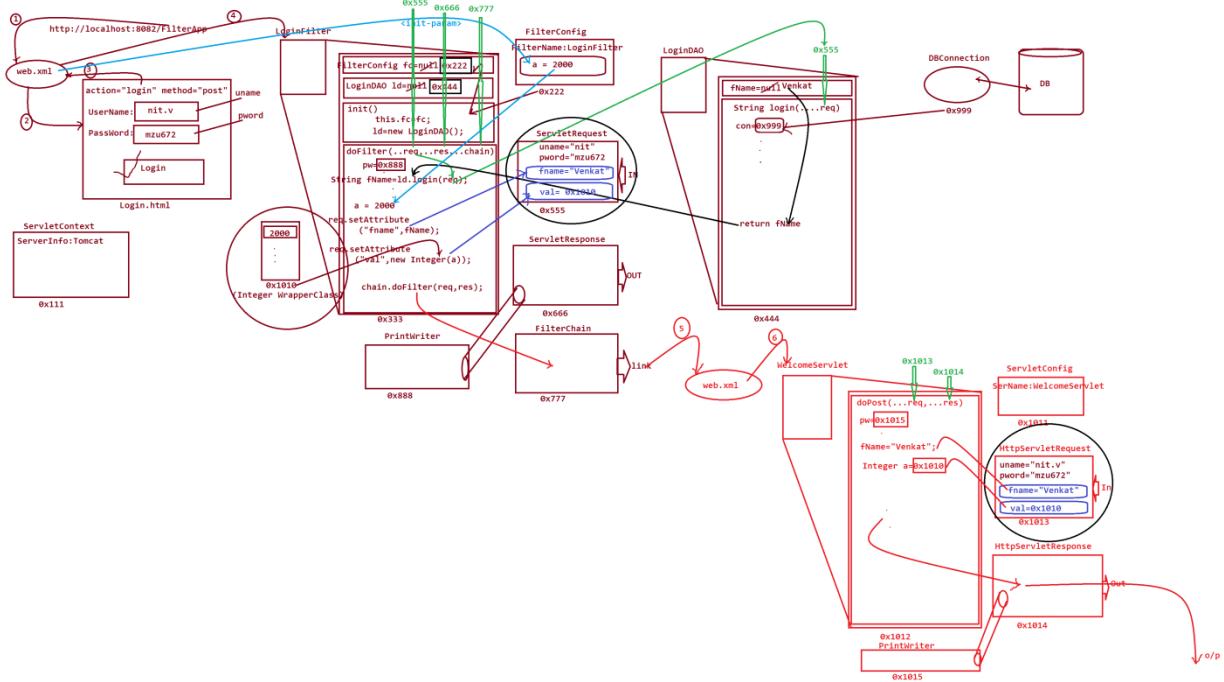
Deployment Directory Structure:



Execute the application as follows:

http://localhost:8082/FilterApp

Execution flow of above application:



faq;

wt is the diff b/w

(i)ServletConfig

(ii)FilterConfig

(i)ServletConfig:

=>ServletConfig is instantiated automatically when the Servlet program is loaded onto WebContainer and which is loaded with Servlet name.

(ii)FilterConfig:

=>FilterConfig is instantiated automatically when the Filter program is loaded onto WebContainer and which is loaded with Filter name.

Note:

=>Every Servlet program will have its own ServletConfig and every Filter program will have its own FilterConfig.

*imp

Listeners and Event Handling process in Servlet Programming:

define Event?

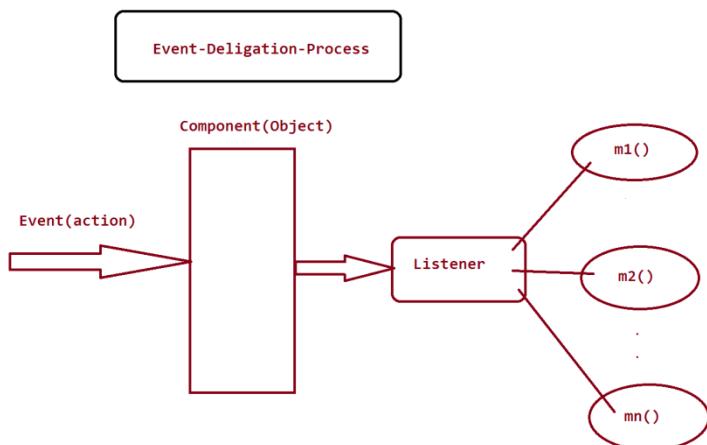
(i)The action which is performed on the component(object) is known as Event.

(ii)The process which is used to handle the event is known as 'Event handling process'.

(iii)We use Listeners to handle events in 'Event handling process'.

Note:

=>The User performs event on Source Component and which is accepted by the Listener and the Listener will call the related method to handle the event, is known as "Event-deligation-process"



The following are used in Event Handling process:

(a)Source object:

=>The Object on which the event raised is known as Source object.

(b)Event class:

=>Type of event raised is known as Event class.

(c)Event Listener:

=>Event Listener will detect and hold the raised event, and also responds to the Event.

(d)Event handling method:

=>The method which we use to handle the event is known as 'Event Handling Method'.

The following are three types of Source Objects in Servlet programming

where the Events are Handled:

1.ServletRequest object

2.ServletContext object

3.HttpSession object

1.ServletRequest object:(source object)

Event Class: ServletRequestEvent

Event Listener : ServletEventListener

Event Handling methods:

requestInitialized()

requestDestroyed()

2.ServletContext object:(source object)

Event Class: ServletContextEvent

Event Listener : ServletContextListener

Event Handling methods:

contextInitialized()

contextDestroyed()

*imp

3.HttpSession object:(Source object)

Event Class: HttpSessionEvent

Event Listener: HttpSessionListener

EventHandling Methods:

sessionCreated()

sessionDestroyed()

Note:

we use the following tag part in web.xml to add listener:

```
<listener>
    <listener-class>
        User-Def-class-name
    </listener-class>
</listener>
```

Dt : 16/2/2021

Exp Application:

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 {  
    public void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        response.setContentType("text/html");  
        PrintWriter out = response.getWriter();  
        String n=request.getParameter("uname");  
        out.print("Welcome "+n);  
        HttpSession session=request.getSession();  
        session.setAttribute("uname",n);  
        ServletContext ctx=this.getServletContext();  
        int t=(Integer)ctx.getAttribute("totalusers");  
        int c=(Integer)ctx.getAttribute("currentusers");  
        out.print("<br>total users= "+t);  
        out.print("<br>current users= "+c);  
        out.print("<br><a href='logout'>logout</a>");  
        out.close();  
    }  
}
```

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 {

    public void doGet(HttpServletRequest request, HttpServletResponse
response)

            throws ServletException, IOException {

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();

        HttpSession session=request.getSession(false);

        session.invalidate();

        out.print("You are successfully logged out");

        RequestDispatcher rd = request.getRequestDispatcher("index.html");

        rd.include(request,response);

        out.close();

    }

}
```

CountUserListener.java

```
package test;

import javax.servlet.*;

import javax.servlet.http.*;

public class CountUserListener implements HttpSessionListener{
```

```
public static ServletContext ctx=null;  
public static int total=0,current=0;  
  
public void sessionCreated(HttpSessionEvent e) {  
    total++;  
    current++;  
  
    ctx=e.getSession().getServletContext();  
    ctx.setAttribute("totalusers", total);  
    ctx.setAttribute("currentusers", current);  
  
    System.out.println("Session Created...");  
}  
}
```

```
public void sessionDestroyed(HttpSessionEvent e) {  
    current--;  
    ctx.setAttribute("currentusers",current);  
  
    System.out.println("Session Destroyed...");  
}  
}
```

index.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>

PassWord : <input type="password" name="pword"><br>

<input type="submit" value="Login">

</form>

</body>

</html>
```

web.xml

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

<web-app>

<listener>

<listener-class>

test.CountUserListener

</listener-class>

</listener>
```

```
<welcome-file-list>

<welcome-file>index.html</welcome-file>

</welcome-file-list>

</web-app>
```

Execute the application as follows:

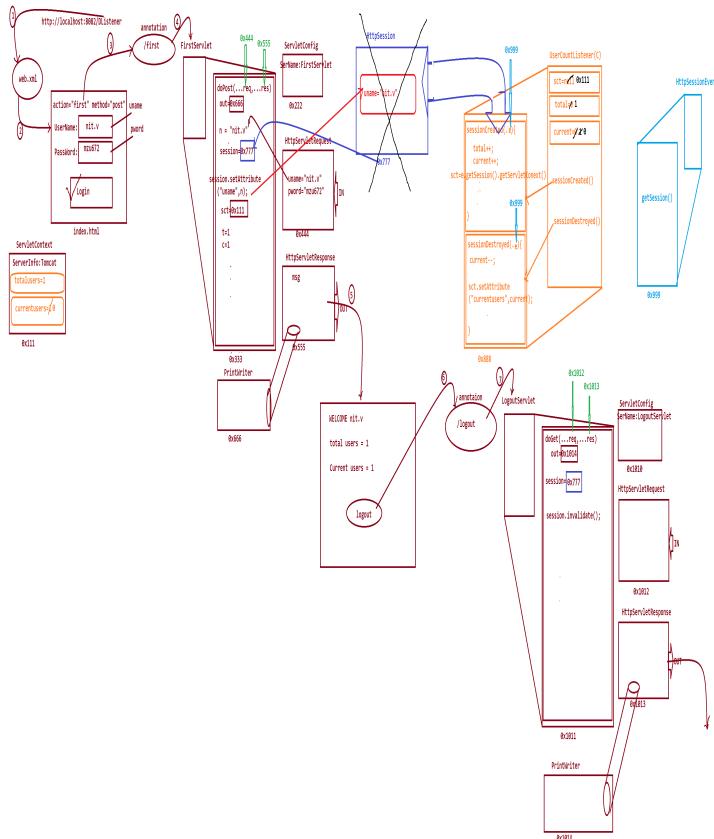
<http://localhost:8082/DListener>

Note:

=>In realtime Listeners are used part of Web Services.

Execution flow of above application to demonstrate Listener:

http://localhost:8082/DListener



Listeners in WebApp are categorized into three types:

1.Application Level Listener

2.Request Level Listener

3.Session Level Listener

1.Application Level Listener:

=>The process of adding Listener to the ServletContext object is

known as Application Level Listener.

2.Request Level Listener:

=>The process of adding Listener to the ServletRequest object is known as Request Level Listener.

3.Session Level Listener:

=>The process of adding Listener to the HttpSession object is known as Session Level Listener.

=====

structure of web.xml from Servlet Programming:

```
<web-app>
  <context-param></context-param>
  <listener></listener>
  <servlet>
    <servlet-name></servlet-name>
    <servlet-class></servlet-class>
    <init-param></init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name></servlet-name>
    <url-pattern></url-pattern>
  </servlet-mapping>
  <filter>
    <filter-name></filter-name>
    <filter-class></filter-class>
    <init-param></init-param>
```

```
</filter>

<filter-mapping>
    <filter-name></filter-name>
    <url-pattern></url-pattern>
</filter-mapping>

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

</web-app>
```

Annotations in Servlet Programming:

define Annotation?

=>The tag based information which is added to the programming component like class,Interface,method and variable is known as annotation.

=>These Annotations are represented using "@" symbol.

=>These Annotations will specify the information to the compiler at compilation stage or Execution controls at execution stage.

Exp:

@ Override

@ SuppressWarnings("unchecked")

@ SuppressWarnings("rawtypes")

=>The following are some important annotations used in Servlet programming:

(a)@ WebServlet

(b)@ WebFilter

(c)@ WebInitParam

(a)@ WebServlet:

=>This '@ WebServlet' is declared to the ServletProgram and used for identifying Servlet program in execution process.

(b)@ WebFilter:

=>This '@ WebFilter' is declared to FilterProgram and used for identifying Filter program in execution process.

(c)@ WebInitParam:

=>This '@ WebInitParam' is used to initialize the parameters with ServletConfig and FilterConfig objects.

Note:

=>When we use annotations in Servlet programming, the Servlet programs can be executed without web.xml mapping file.

=>All the annotations related to ServletProgramming are available from "javax.servlet.annotation" package.

Dt : 17/2/2021

Exp Application:

```
DBConnection.java  
package test;  
import java.sql.*;  
  
public class DBConnection {
```

```

private static Connection con=null;//Class variable or static variable

private DBConnection(){}

static{

    try{

        Class.forName("oracle.jdbc.driver.OracleDriver");

        con=DriverManager.getConnection

            ("jdbc:oracle:thin:@localhost:1522:orcl","system","manager");

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

}//end of static block

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 UserReg28 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;

}

}
```

LoginFilter.java

```
package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.annotation.*;

@WebFilter(
    urlPatterns="/login",
    initParams= @WebInitParam(name="a",value="200"))

public class LoginFilter implements Filter{

    public FilterConfig fc;

    @Override

    public void init(FilterConfig fc) throws ServletException{

        this.fc=fc;

    }

    @Override
```

```
public void doFilter(ServletRequest req,ServletResponse res,FilterChain fch)
throws IOException,ServletException{
PrintWriter pw = res.getWriter();
res.setContentType("text/html");
String a = fc.getInitParameter("a");
String fName = new LoginDAO().login(req);
if(fName==null){
pw.println("InValid UserName or PassWord...<br>");
RequestDispatcher rd = req.getRequestDispatcher("Login.html");
rd.include(req,res);
}else{
req.setAttribute("fName",fName);
req.setAttribute("a",a);
fch.doFilter(req,res);
}
}
```

```
@Override
public void destroy(){
java.sql.Connection con = DBConnection.getCon();
try{
con.close();
}catch(Exception e){e.printStackTrace();}
}
}
```

WelcomeServlet.java

```
package test;

import java.io.*;

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

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

public class WelcomeServlet extends HttpServlet{

    @ Override

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

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

        String a = (String)req.getAttribute("a");

        String fName = (String)req.getAttribute("fName");

        pw.println("WELCOME : "+fName+"  
");

        pw.println("Init Param : "+a);

    }

}

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>
```

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 : 18/2/2021

faq:

define sendRedirect() method?

=>sendRedirect() method is used to establish communication b/w two

Servlet programs running under different Web Applications.In this process the different Web Applications can be from Same WebServer or different WebServers.

=>This `sendRedirect()` method is available from "HttpServletResponse":

Method Signature:

```
public abstract void sendRedirect(java.lang.String)  
                      throws java.io.IOException;
```

Exp Application:

Web Application1 : 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>
```

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>
```

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{
    @Override
    public 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?uname="+uName+"&mid="+mId);
    }
}
```

Web Application2 : TestApp2

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{  
    @Override  
    public void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException{  
        PrintWriter pw = res.getWriter();  
        res.setContentType("text/html");  
        String uName = req.getParameter("uname");  
        String mId = req.getParameter("mid");  
  
        pw.println("=====SecondServlet=====<br>");  
        pw.println("UserName:" + uName + "<br>");  
        pw.println("MailId:" + mId);  
    }  
}
```

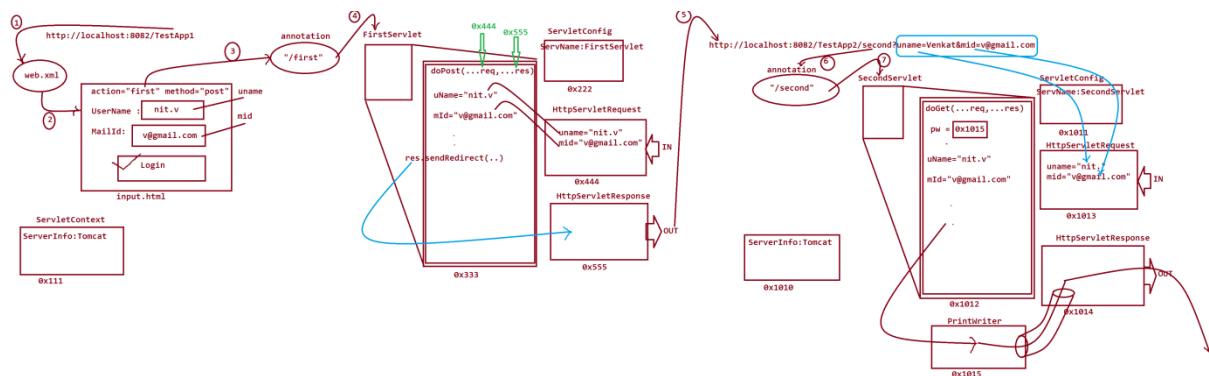
Execute the application as follows:

<http://localhost:8082/TestApp1>

<http://localhost:8082/TestApp2/second?uname=Venkat&mid=v@gmail.com>

Dt : 20/2/2021

Execution flow of above application:



Note:

=>when we use `sendRedirect()` method the data is sent through
webBrowser and the request is GET request.

Dt : 22/2/2021

faq:

define Servlet Collaboration?

=>**Servlet Collaboration is a process in which multiple Servlet programs working together to achieve a defined task.**

(for common business purpose)

=>**In Servlet Collaboration process the multiple Servlets are linked together or integrated together.**

=>**we use the following to integrate Servlet programs:**

(a)forward() method

(b)include() method

(c)sendRedirect() method

(a)forward() method:

=>**forward() method is used to establish forward communication b/w servlets,in this process first servlet will take the request and second servlet will give the response.**

(b)include() method:

=>**include() method is used to establish include communication b/w servlets,in this process first servlet response is added with the response of second servlet.**

Note:

=>**include() and forward() methods are used to establish communication b/w Servlet programs of Same web Application.**

(c)sendRedirect() method:

=>**sendRedirect()** method is used to establish communication b/w Servlet programs available in different web applications.

faq:

define Servlet Life-Cycle:

=>**Servlet Life-Cycle** demonstrates different states of Servlet program from Starting-of-the-Servlet execution to ending-of-the-servlet execution.

=>The following are the stages in Servlet Life-Cycle:

1.Loading

2.Instantiation

3.Initialization

4.Handling request

5.destroying

1.Loading:

=>The process of identifying the Servlet program based on url-pattern and loading onto WebContainer is known as "Loading process".

Note:

=>The url-pattern can be mapped using web.xml or annotation.

2.Instantiation:

=>The process of creating object for Servlet program automatically by the WebContainer is known as Instantiation process.

Note:

=>After Instantiation process the object will have Life cycle methods

GenericServlet:

=>init()
=>service()
=>destroy()

HttpServlet:

=>init()
=>doPost()/doGet()
=>destroy()

Filter:

=>init()
=>doFilter()
=>destroy()

define Life-Cycle methods?

=>The methods which are automatically called by the WebContainer for execution are known as Life-Cycle methods.

3.Initialization:

=>The process of making the programming components ready for request handling process is known as Initialization process.

Note:

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

=>This init() method will be executed only once while object creation and performs initialization.

4.Handling request:

=>The process of accepting the request and providing the response is known as "request handling process"

Note:

=>we use service() or doPost() or doGet() or doFilter() methods in request handling process.

=>These methods will be executed for all the multiple requests generated from the multiple users.

5.destroying:

=>The process of destroying the servlet instance from the WebContainer is known as "destroying process".

Note:

=>we use destroy() method to perform destroying process.

Generating WAR file(Eclipse IDE) and deploying into Tomcat server:

step1 : Generate WAR file

Right Click on WebApp->Export->WAR file,browse the destination folder and click 'finish'

step2 : Deploy the WAR file into Tomcat Server

Start the Tomcat Server->Open the WebBrowser and run the Tomcat

(<http://localhost:8082>)> Click on Manager App and "sign in" ->

click on Choose file from "WAR file to deploy" ->

Browse the WAR file and select->Open->click "deploy".

Execute the appl as:

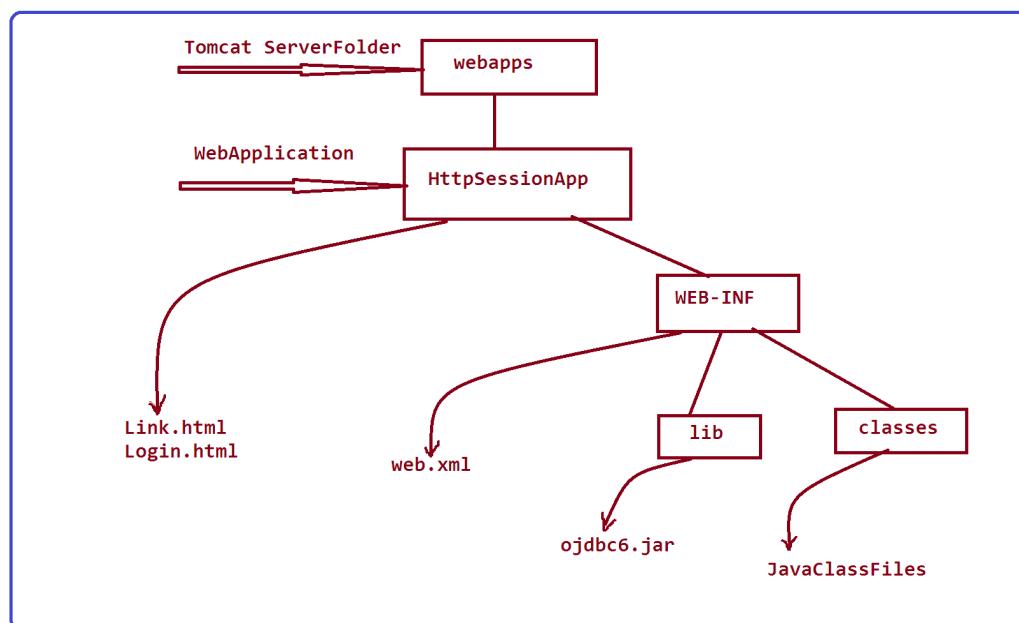
<http://localhost:8082/HttpSessionApp>

Note:

=>The WebAppl is deployed into "webapps" folder of Tomcat server.

=>The following is the deployment directory structure of Tomcat Server:

Diagram:



faq:

wt is the diff b/w

(i)JAR

(ii)WAR

(iii)EAR

(i)JAR:

=>JAR stands for Java Archive,which holds Java Class files.

(ii)WAR:

=>WAR stands for Web Archive,which holds Servlets,JSP,HTML,CSS,XML, Images and Web Components.

Note:

=>WAR will have WEB-INF to hold jar files.

(iii)EAR:

=>EAR stands for Enterprise Archive,which holds EJB components, Connectors,External Libraries,...

Note:

=>EAR will have APP-INF to hold Jar files and WAR files.

=====

Note:

=>WebServer can be deployed with WAR files for execution.

=>Applications Servers can be deployed with both WAR Files and EAR files for execution.

=====

Note:

=>WAR file is executed on WebServers.

=>WAR file and EAR file is executed on Application servers.

=====

Dt 23/2/2021

*imp

JSP Programming:

=>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

2.Directive tags

3.Action tags

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 the JSP program.

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"%>
```

```
=====
```

Exp program1:

JSP Application to calculate factorial of given number.

Note:

=>JSP files are created part of WebContent.

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="Display.jsp" method="post">
Enter the value:<input type="text" name="n"><br>
<input type="submit" value="CalculateFactorial">
</form>
</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>
```

Display.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" import="java.util.Date"
errorPage="Error.jsp"%>
<%= new Date() %>
<%! 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"));
out.println("<br>Factorial="+factorial(n)+"<br>");
%>
<%@ include file="input.html" %>
```

Error.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isErrorPage="true"%>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
<%= exception %>
<br>
<h2>Enter only Integer value...</h2>
<br>
<%@ include file="input.html" %>
</body>
```

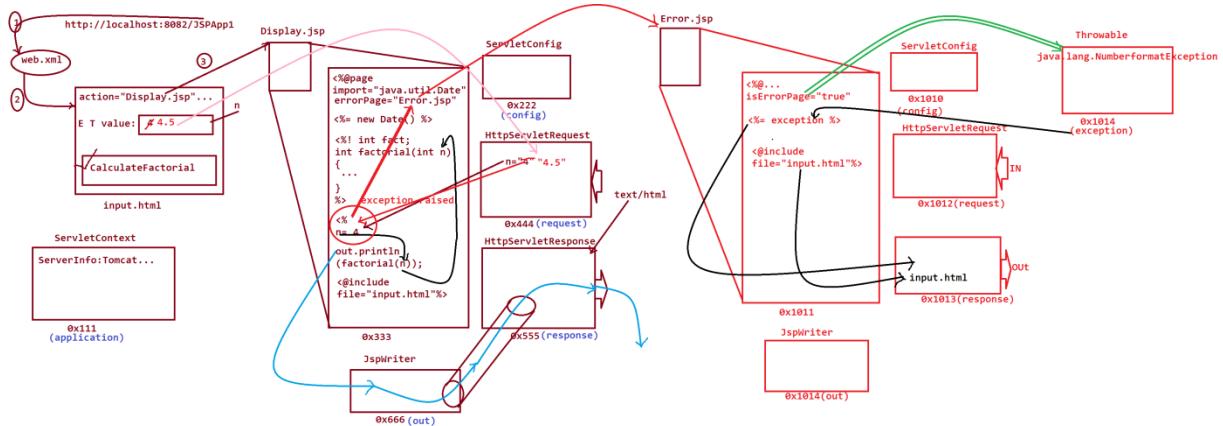
```
</html>
```

Execute the application as follows:

http://localhost:8082/JSPApp1

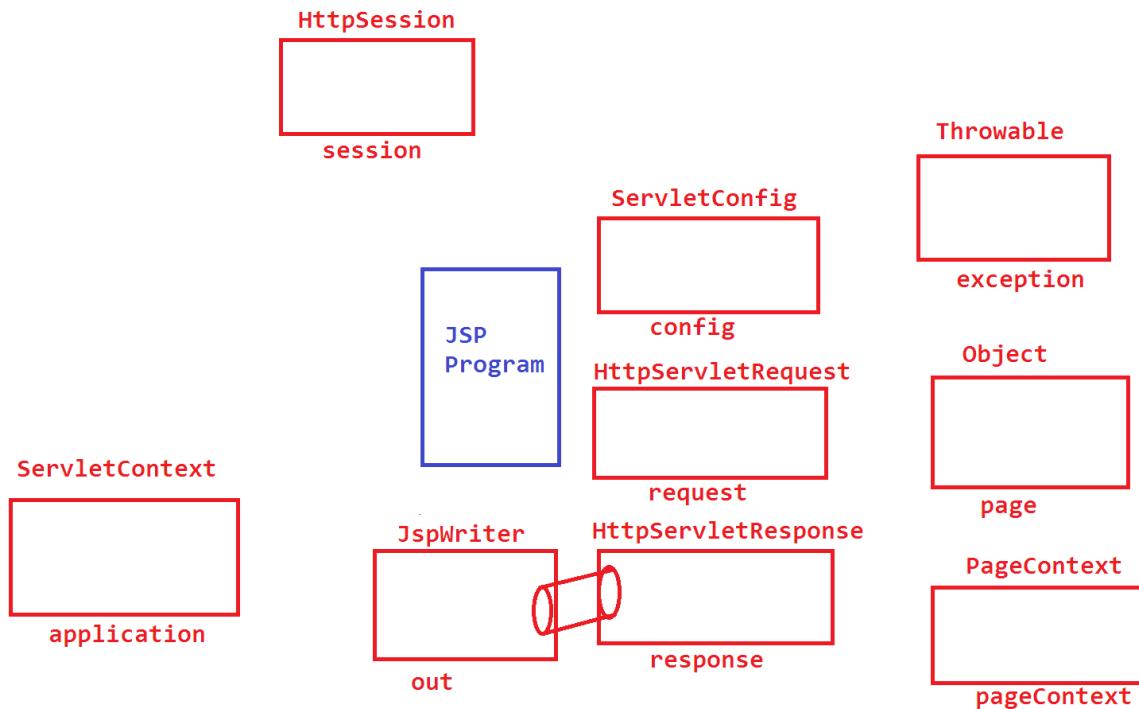
Dt : 24/2/2021

Execution flow of above program:



=>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 a 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.

sntax:

```
<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 or bean 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 : 25/2/2021

Exp:

JSP Application to demonstrate jsp:include,jsp:forward and jsp:param.

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="Home.jsp" method="post">
Enter the Value1:<input type="text" name="v1"><br>
Enter the Value2:<input type="text" name="v2"><br>
<input type="submit" value="Add" name="s1">
<input type="submit" value="Sub" name="s1">
</form>
</body>
</html>
```

Home.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
<%
String s1 = request.getParameter("s1");
if(s1.equals("Add")){
%>
<jsp:forward page="Addition.jsp"/>
<%
}else{
```

```

    %>
    <jsp:forward page="Subtraction.jsp"/>
    <%
    %
%>
</body>
</html>

```

Addition.jsp

```

<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" errorPage="Error.jsp"%>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
<%
int x = Integer.parseInt(request.getParameter("v1"));
int y = Integer.parseInt(request.getParameter("v2"));
int z = x+y;
out.println("Sum:"+z+"<br>");
%>
<jsp:include page="input.html"/>
</body>
</html>

```

Subtraction.jsp

```

<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" errorPage="Error.jsp"%>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
<%
int x = Integer.parseInt(request.getParameter("v1"));

```

```

int y = Integer.parseInt(request.getParameter("v2"));
int z = x-y;
out.println("Sub:"+z+"<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"%>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
<%= exception %>
<br>
<h2>Enter only Integer values</h2>
<br>
<jsp:include page="input.html"/>
</body>
</html>

```

web.xml

```

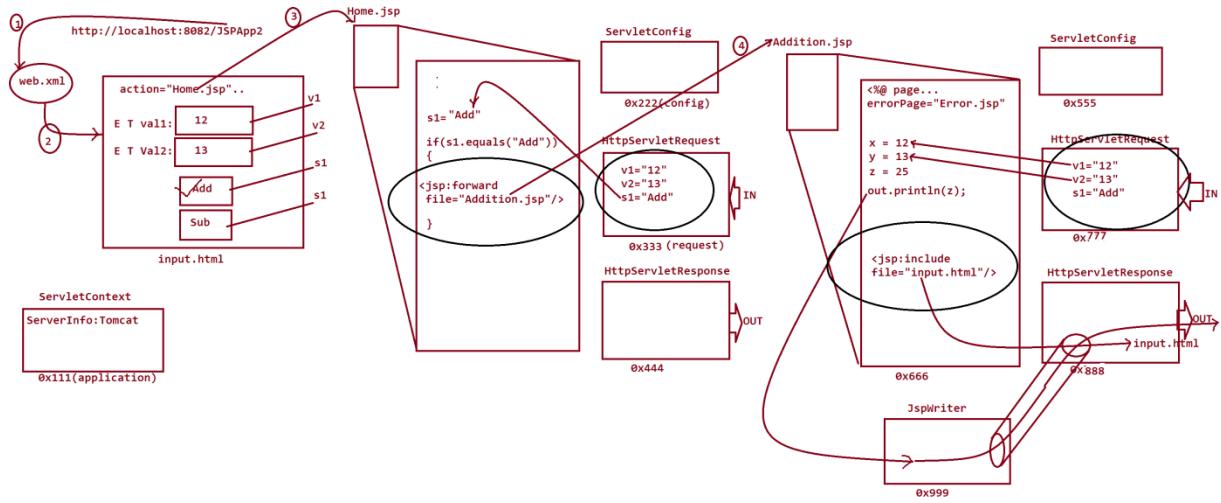
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>input.html</welcome-file>
    </welcome-file-list>
</web-app>

```

Execute the appl as follows:

<http://localhost:8082/JSPApp2>

Execution flow of above application:



Note:

- =>This `jsp:forward` and `jsp:include` action tags are used to establish communication b/e JSP pages(files).
- =>In `jsp:forward` communication the data in request object of first JSP page is copied to the request object of second JSP page.
- =>The parameter-value which is sent using `jsp:param` tag is available in the request object of Second JSP page.

Assignment:

Update above application with

Mul

Div

ModDiv

Dt : 1/3/2021

Exp Application:

**JSP Application to demonstrate jsp:useBean,jsp:setProperty and
jsp:getProperty.**

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="Load.jsp" method="post">
ProductCode:<input type="text" name="PCODE"><br>
ProductName:<input type="text" name="PNAME"><br>
ProductPrice:<input type="text" name="PPRICE"><br>
ProductQty:<input type="text" name="PQTY"><br>
<input type="submit" value="LoadProductData">
</form>
</body>
</html>
```

Product.java(Bean class)

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class Product implements Serializable{
    private String pCode,pName;
    private float pPrice;
    private int pQty;
```

```

public Product(){}
public final String getpCode() {
    return pCode;
}
public final void setpCode(String pCode) {
    this.pCode = pCode;
}
public final String getpName() {
    return pName;
}
public final void setpName(String pName) {
    this.pName = pName;
}
public final float getpPrice() {
    return pPrice;
}
public final void setpPrice(float pPrice) {
    this.pPrice = pPrice;
}
public final int getpQty() {
    return pQty;
}
public final void setpQty(int pQty) {
    this.pQty = pQty;
}

}

```

Load.jsp

```

<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>

```

```

<title>Insert title here</title>
</head>
<body>
<jsp:useBean id="bean" class="test.Product"
scope="session"/>
<jsp:setProperty property="pCode" param="pcode"
name="bean"/>
<jsp:setProperty property="pName" param="pname"
name="bean"/>
<jsp:setProperty property="pPrice" param="pprice"
name="bean"/>
<jsp:setProperty property="pQty" param="pqty" name="bean"/>
<h4>Product data loaded to Bean Object...</h4>
<br>
<a href="Retrieve.jsp">RetrieveDataFromBean</a>
</body>
</html>

```

Retrieve.jsp

```

<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
<jsp:useBean id="bean" type="test.Product" scope="session"/>
ProductCode:<jsp:getProperty property="pCode"
name="bean"/><br>
ProductName:<jsp:getProperty property="pName"
name="bean"/><br>
ProductPrice:<jsp:getProperty property="pPrice"
name="bean"/><br>
ProductQty:<jsp:getProperty property="pQty"
name="bean"/><br>
</body>
</html>

```

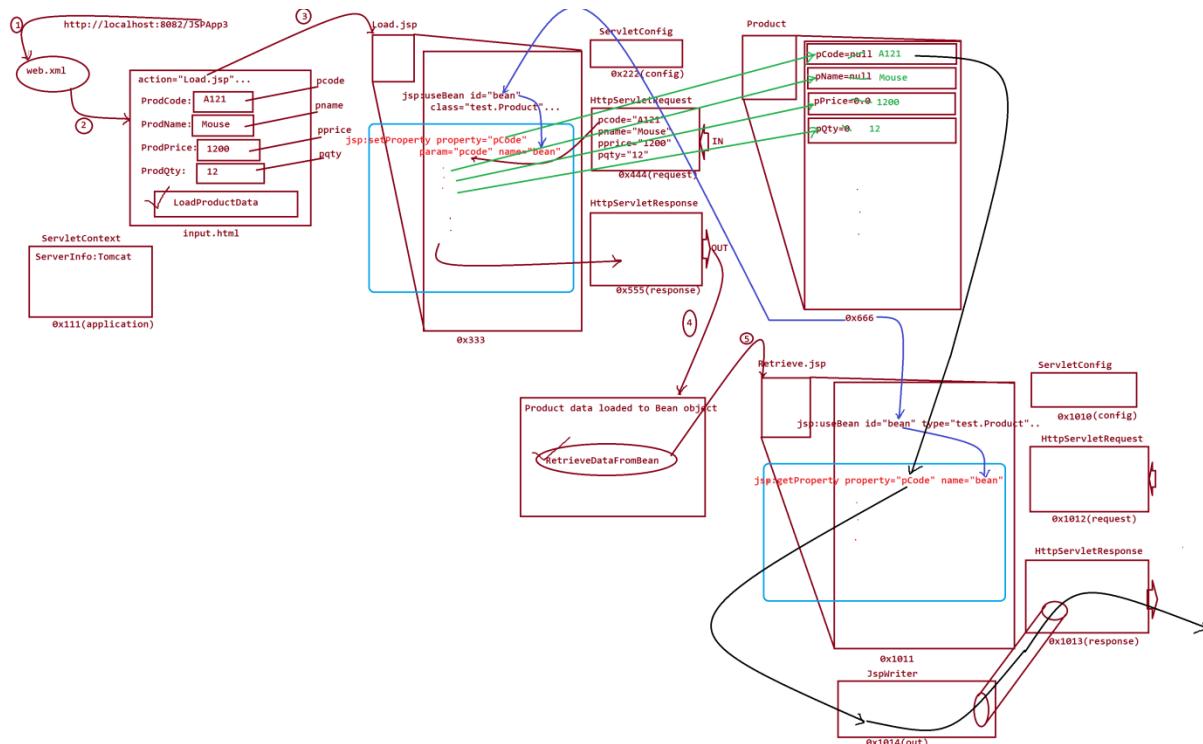
web.xml

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

Execute the application as follows:

<http://localhost:8082/JSPApp3>

Execution flow of above application:



=====
====

DT : 2/3/2021

JSP LifeCycle:

=>JSP LifeCycle demonstrates diff states of JSP program from Starting of the program to the ending of the program.

=>The following are the states of JSP program in LifeCycle:

(1)Translation process

(2)Compilation process

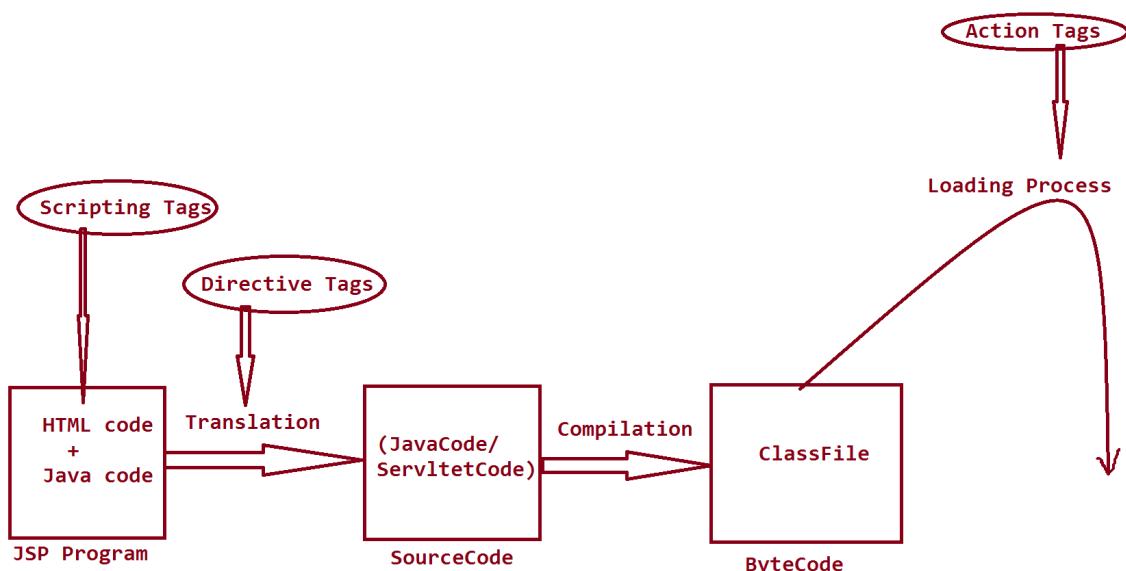
(3)Loading process

(4)Instantiation process

(5)Initialization process

(6)Request Handling process

(7)Destroying process



(1)Translation process:

=>The process of separating JavaCode from JSP program is known as Translation process.

=>After translation process SourceCode is generated.

(2)Compilation process:

=>The process of compiling the SourceCode and generating the ByteCode is known as Compilation process.

(3)Loading process:

=>The process of loading the JSP program for execution is known as Loading process.

(4)Instantiation process:

=>The process of creating object for JSP program is known as Instantiation process.

=>After Instantiation process we can find the following implicit lifeCycle methods:

(i)_jspInit()

(ii)_jspService()

(iii)_jspDestroy()

(5)Initialization process:

=>The process of making the programming components ready for service is known as Initialization process.

(6)Request Handling process:

=>The process of handling the request and giving the response is known as Request Handling process.

(7)Destroying process:

=>The process of destroying the instances from the WebContainer is known as Destroying process.

=====

Dt : 3/3/2021

*imp

Web Architecture Models:(Web Application Architectures)

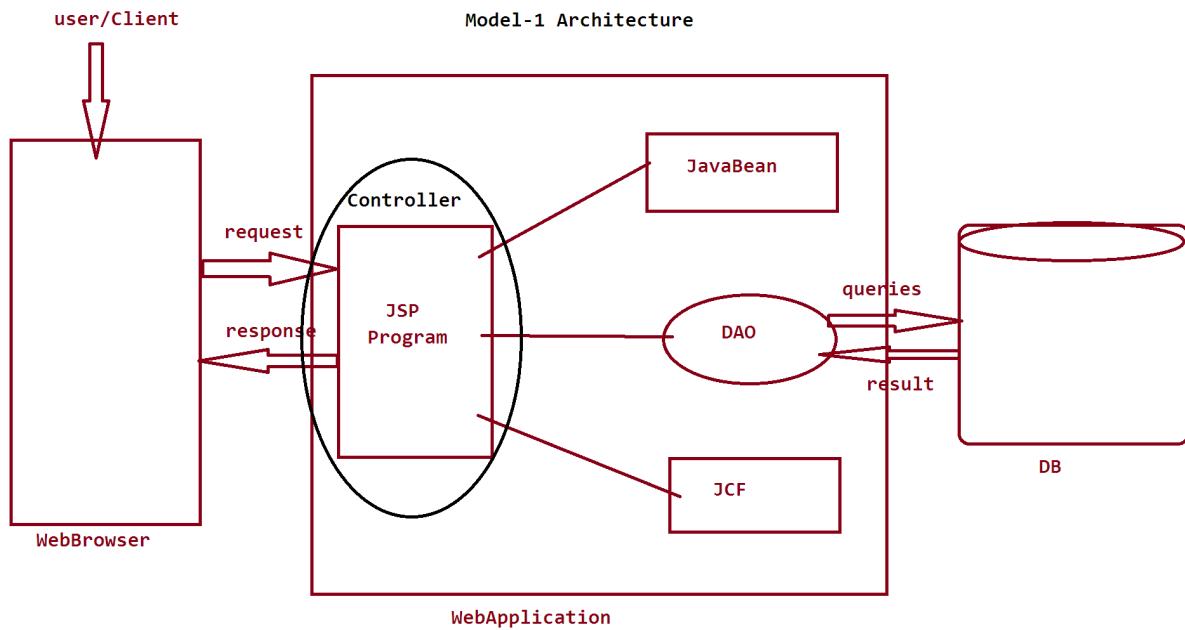
Two types of development models are used in Java for Web applications and these models are classified based on the different approaches used to develop Web applications.

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.

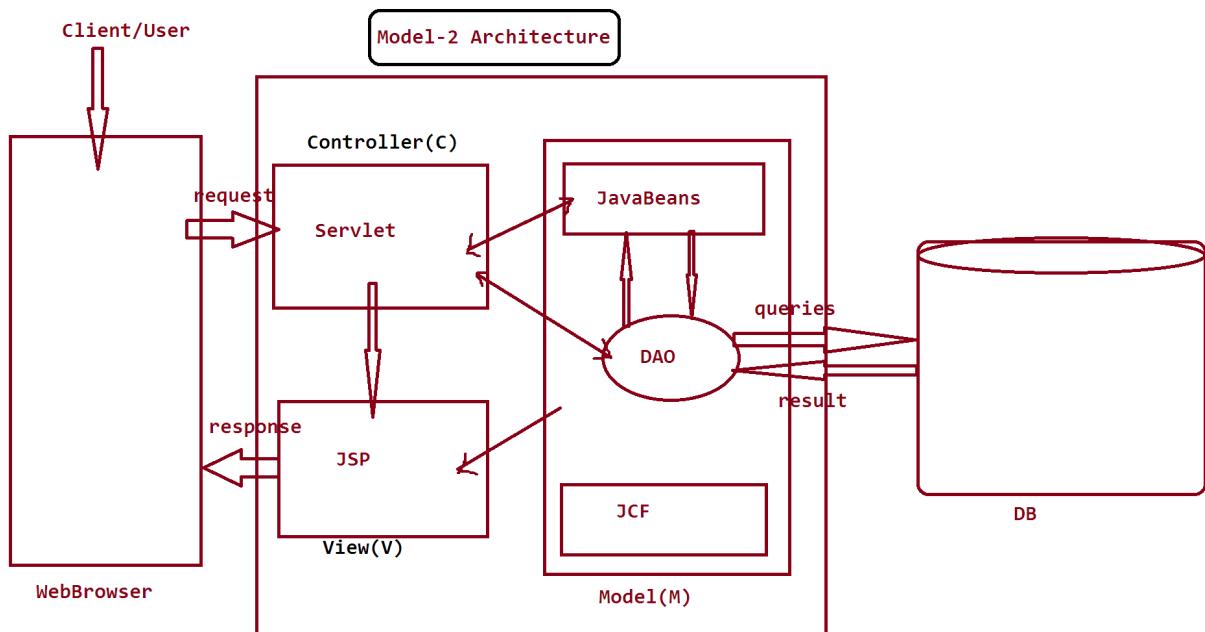
=>MVC Stands for Model View Controller.

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.

Exp application:

MVC Application : OnlineBookStore

DBTables : UserReg29,Admin29,Book29

UserReg29

(uname,pword, fname, lname, addr, phno, mid)

Admin29

(uname,pword, fname, lname, addr, phno, mid)

Book29

(bcode, bname, bauthor, bprice, bqty)

Note:

=>UserReg29 and Book29 must be empty(0 records)

=>Insert one record to Admin29 from SQL CommandLine

```
create table Admin29(uname varchar2(15),pword varchar2(15),
fname varchar2(15),lname varchar2(15),addr varchar2(15),
phno number(15),mid varchar2(25),primary key(uname,pword));
```

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:1522:orcl","system","ma
nager");
        }catch(Exception e){e.printStackTrace();}
    }//end of block
    public static Connection getCon(){
        return con;
    }
}
```

Home.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <form action="home" method="post">
        <input type="submit" value="Admin" name="s1">
        <input type="submit" value="User" name="s1">
    </form>
</body>
</html>
```

AdminLogin.html

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

UserLogin.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="uLogin" 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>
    PhoneNO<input type="text" name="phno"><br>
    MailId:<input type="text" name="mid"><br>
    <input type="submit" value="Register">
  </form>
</body>
</html>

```

HomeServlet.java(home)

```

package test;

import java.io.*;

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

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

public class HomeServlet extends HttpServlet{

  @Override

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

    String s1 = req.getParameter("s1");

    req.getServletContext().setAttribute("s1", s1);
}

```

```
if(s1.equals("Admin")){
    req.getRequestDispatcher("AdminLogin.html").forward(req,res);
}else{
    req.getRequestDispatcher("UserLogin.html").forward(req,res);
}
}
```

InsertDAO.java

```
package test;

import java.sql.*;
import javax.servlet.http.*;
public class InsertDAO {
    public int k=0;
    public int insert(HttpServletRequest req){
        try{
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                ("insert into UserReg29 values(?,?,?,?,?,?)");
            ps.setString(1,req.getParameter("uname"));
            ps.setString(2,req.getParameter("pword"));
            ps.setString(3,req.getParameter("fname"));
            ps.setString(4,req.getParameter("lname"));
            ps.setString(5,req.getParameter("addr"));
            ps.setLong(6,Long.parseLong(req.getParameter("phno")));
            ps.setString(7,req.getParameter("mid"));
        }
    }
}
```

```
        k = ps.executeUpdate();

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

    return k;

}

}
```

DT : 4/3/2021

RegServlet.java(reg)

```
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{

    @Override

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

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

        if(k>0){

            req.getRequestDispatcher("Success.jsp").forward(req,res);

        }

    }

}
```

Success.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
<h3>Registration Successful...</h3>
<%@include file="UserLogin.html" %>
</body>
</html>
```

LoginDAO.java

```
package test;

import java.sql.*;
import javax.servlet.http.*;
public class LoginDAO {
    public String fName=null;
    public String login(HttpServletRequest req){
        String s1 = (String)req.getServletContext().getAttribute("s1");
        Connection con = DBConnection.getCon();
        try{
            if(s1.equals("Admin")){
                PreparedStatement ps = con.prepareStatement
                    ("select * from Admin29 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);
        }
    }else{
        PreparedStatement ps=con.prepareStatement
        ("select * from UserReg29 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;
}
}

```

UserLoginServlet.java(ulogin)

```

package test;

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

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

```

```

public class UserLoginServlet extends HttpServlet{

    @Override

    public void doPost(HttpServletRequest req,HttpServletResponse res)
    throws ServletException,IOException{
        String fName = new LoginDAO().login(req);

        if(fName==null){
            req.getRequestDispatcher("LoginFail.jsp").forward(req,res);
        }else{
            HttpSession hs = req.getSession();
            hs.setAttribute("fName",fName);
            req.getRequestDispatcher("LoginSuccess.jsp").forward(req,res);
        }
    }
}

```

LoginSuccess.jsp

```

<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
<%
String fName = (String)session.getAttribute("fName");
String s1 = (String)application.getAttribute("s1");
out.println("WELCOME "+fName+"<br>");
    if(s1.equals("Admin")){
    %>
    <%@include file="Link2.html" %>

```

```

<%
}else{
%>
<%@include file="Link1.html" %>
<%
}
%>
</body>
</html>

```

Link1.html

```

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

```

Link2.html

```

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<a href="view">ViewBooks</a>
<a href="Book.html">AddBook</a>
<a href="BookCode1.html">UpdateBook</a>
<a href="BookCode2.html">DeleteBook</a>
<a href="Logout">Logout</a>
</body>
</html>

```

LoginFail.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
<h3>Invalid UserName or Password...</h3>
<%@include file="Home.html" %>
</body>
</html>
```

LogoutServlet.java(logout)

```
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{

    @Override

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

        HttpSession hs = req.getSession(false);
        hs.invalidate();
        req.getRequestDispatcher("Logout.jsp").forward(req,res);
    }
}
```

```
}
```

Logout.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
<h3>Logged Out Successfully...</h3>
<%@include file="Home.html" %>
</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>
```

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:

pageScope : It maps the given attribute name with the value, set in the page scope

requestScope : It maps the given attribute name with the value set in the request scope

sessionScope : It maps the given attribute name with the value set in the session scope

applicationScope : It maps the given attribute name with the value set in the application scope

param : It maps the request parameter to the single value

paramValues :

It maps the request parameters to the array of values

header : It maps the request header name to the single value

headerValues : It maps the request header names to the array of values

cookie : It maps the cookie name to the cookie value

initParam : It maps the Initialization parameter

pageContext : It provides access to many objects request,session,...

Note:

|->Scriptlet tag can be replaced by EL Tag
in JSP Programming.

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 Throwable 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 delimiters.

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.

Exp program1:

AppI to read data from HTML form to "EL and JSTL":

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <form action="Display.jsp"
method="post">
        Operand1:<input type="text"
name="op1"><br>
        Operand2:<input type="text"
name="op2"><br>
        <input type="submit" value="Add"
name="submit">
```

```
<input type="submit" value="Sub"
name="submit">
</form>
</body>
</html>
```

Display.jsp

```
<%@taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
<title>EL App1</title>
</head>
<body>
<c:choose>
<c:when test="${param.submit eq 'Add'}">
Sum of <c:out value="${param.op1}" />
and
<c:out value="${param.op2}" /> is
<c:out value="${param.op1 + param.op2}" />
</c:when>

<c:otherwise>
<c:out value="${param.op1}" /> and
<c:out value="${param.op2}" /> is
<c:out value="${param.op1 - param.op2}" />
</c:otherwise>
```

```
</c:choose>

<jsp:include page="input.html"/>
</body>
</html>
```

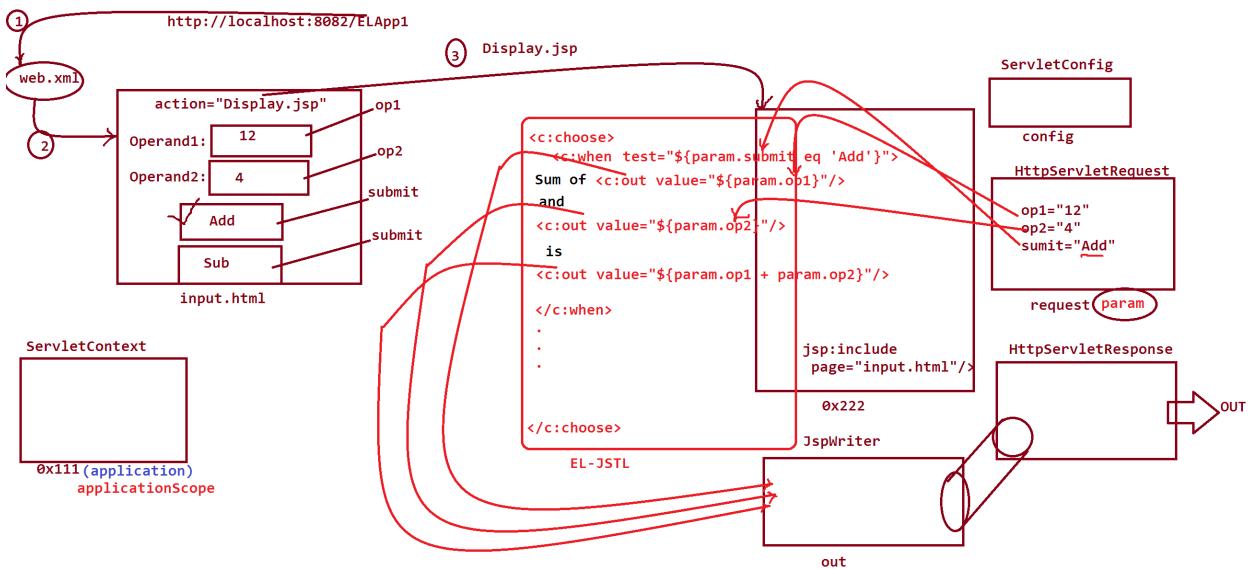
web.xml

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

Execute the Appl as follows:

<http://localhost:8082/ELApp1>

Execution flow of above application:



Exp program2 on EL&JSTL

`input.html`

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

```
</html>
```

ControllerServlet.java

```
package test;

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

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

public class ControlServlet extends HttpServlet{

    @Override
    public void doPost(HttpServletRequest req,
                       HttpServletResponse res)
        throws ServletException, IOException{
        ServletContext sct = this.getServletContext();

        HttpSession hs = req.getSession();
        sct.setAttribute("a",100);
        hs.setAttribute("b",200);
        req.setAttribute("c", 300);

        RequestDispatcher rd = req.getRequestDispatcher("Display.jsp");
        rd.forward(req,res);
    }
}
```

Display.jsp

```

<%@taglib prefix="c"
  uri="http://java.sun.com/jsp/jstl/core"
%>
<%
pageContext.setAttribute(" fName ", "Raju");
%>
<c:set var="name" value="${param.name} "/>
WELCOME : <c:out value="${name} "/><br>
<c:set var="a"
value="${applicationScope.a} "/>
<c:set var="b" value="${sessionScope.b}" />
<c:set var="c" value="${requestScope.c}" />
<c:set var="d" value="${pageScope.fName}" />

```

```

ContextVal:<c:out value="${a} "/><br>
SessionVal:<c:out value="${b} "/><br>
RequestVal:<c:out value="${c} "/><br>
PageVal:<c:out value="${d} "/><br>

```

web.xml

```

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

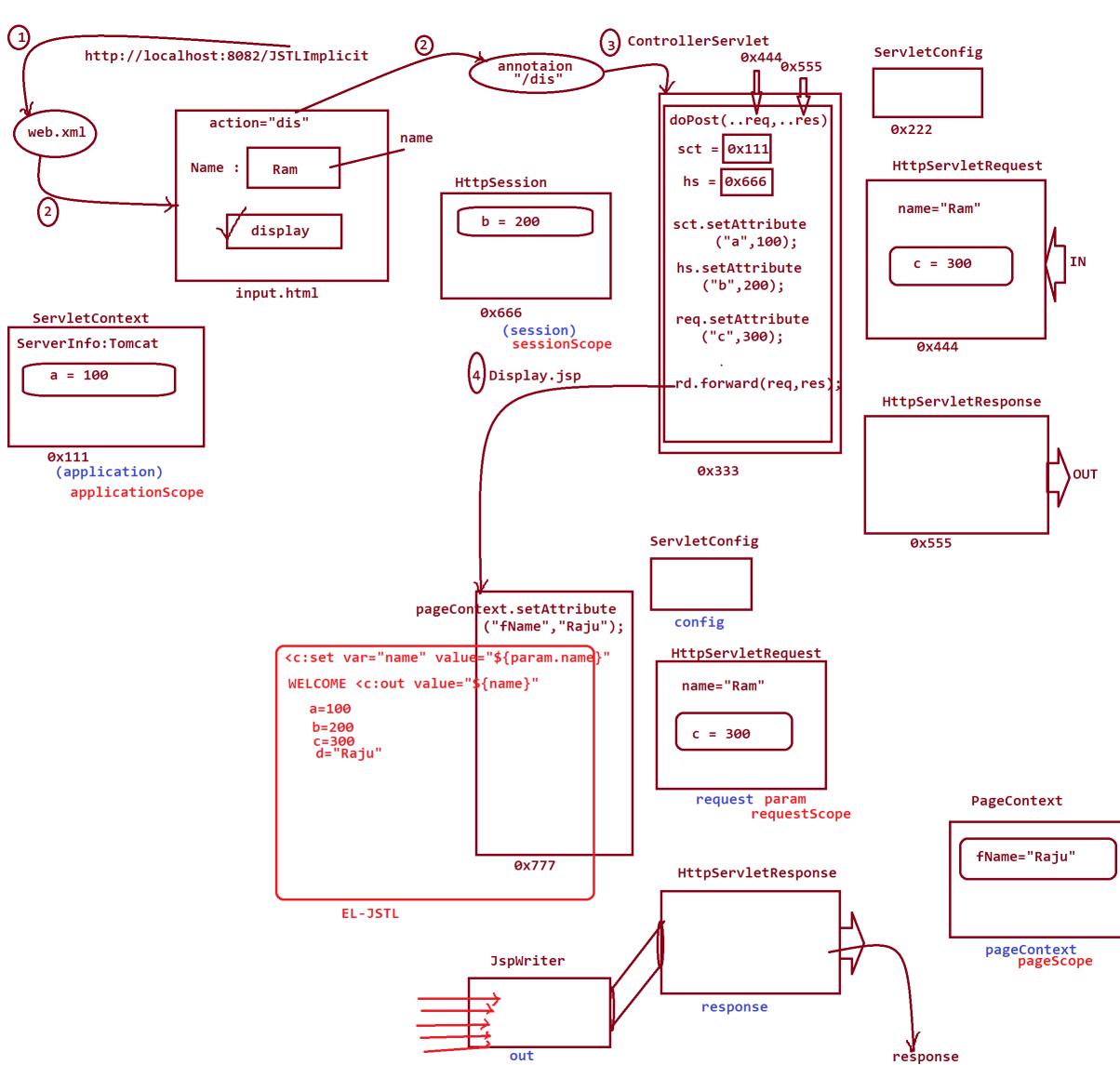
```

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

Execute the appl as follows:

http://localhost:8082/JSTLImplicit

Execution flow of above application:



Note:

=>In forward communication b/w Servlet program and JSP program, the data available in the request object of Servlet program is also available within the request object of JSP program.

Exp program3 on EL&JSTL:

input.html

```
<!doctype html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <form method="post"
action="CoreTags.jsp">
        Enter the String:<input type="text"
name="str"><br>
        <input type="submit" value="DispLay">
    </form>
</body>
</html>
```

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>
```

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>
```

Execute the appl as follows:

<http://localhost:8082/JSTLCoreTags>

(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.

Exp program:

input.html

```
<!doctype html>
<html lang="en">
  <head>
    <title>Document</title>
  </head>
  <body>
    <form method="post"
action="FunctionTag.jsp">
      Enter the String:<input type="text"
name="str"><br>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

FunctionTag.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>

```

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>
```

Execute the above program as follows:

<http://localhost:8082/JSTL3>

(3)Formatting Tags:

=>The 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.

Exp program:

FormatTag1.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

```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app>  
    <welcome-file-list>  
        <welcome-  
file>FormatTag1.jsp</welcome-file>  
    </welcome-file-list>  
</web-app>
```

Execute the appl as follows:

<http://localhost:8082/JSTL4/>

Note: |-In realtime JSTL Formating tags are mostly used for Date and Time formats.

(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 methods of pageContext object:

```
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.

Summery 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

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

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

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.Connection pooling 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:

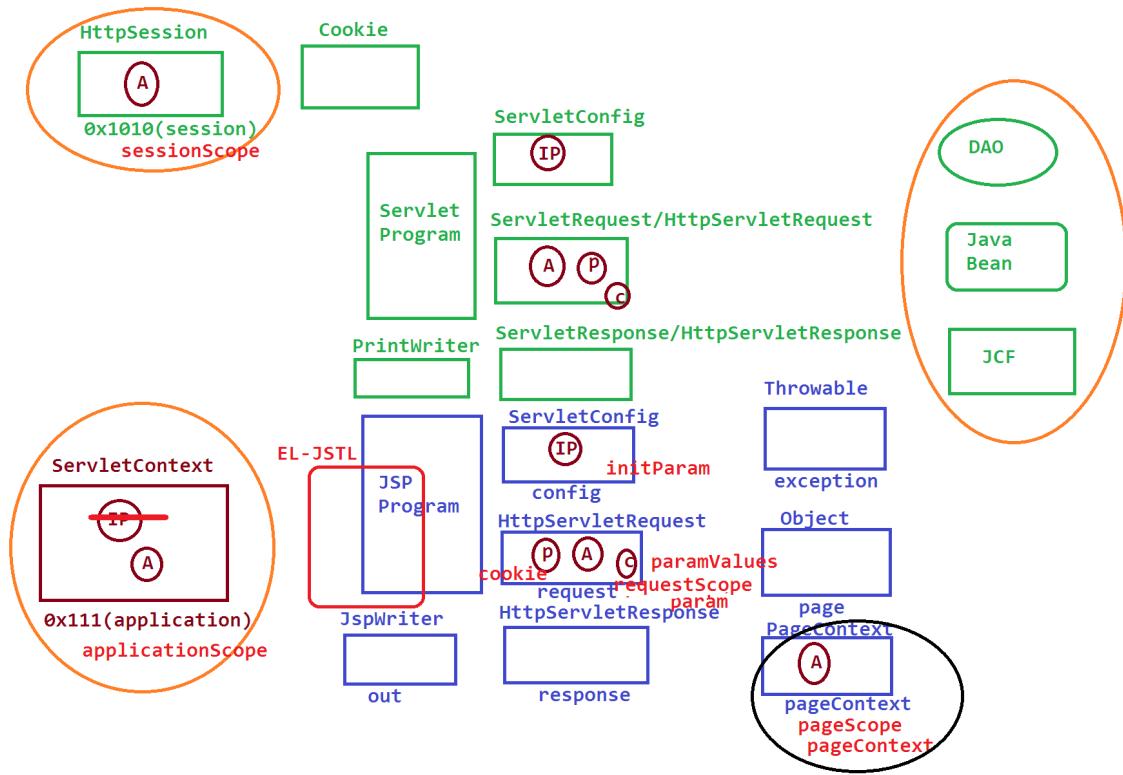
- 1.application**
 - 2.config**
 - 3.request**
 - 4.response**
 - 5.out**
 - 6.session**
 - 7.exception**
 - 8.page**
 - 9.pageContext**
-

JSP EL objects:

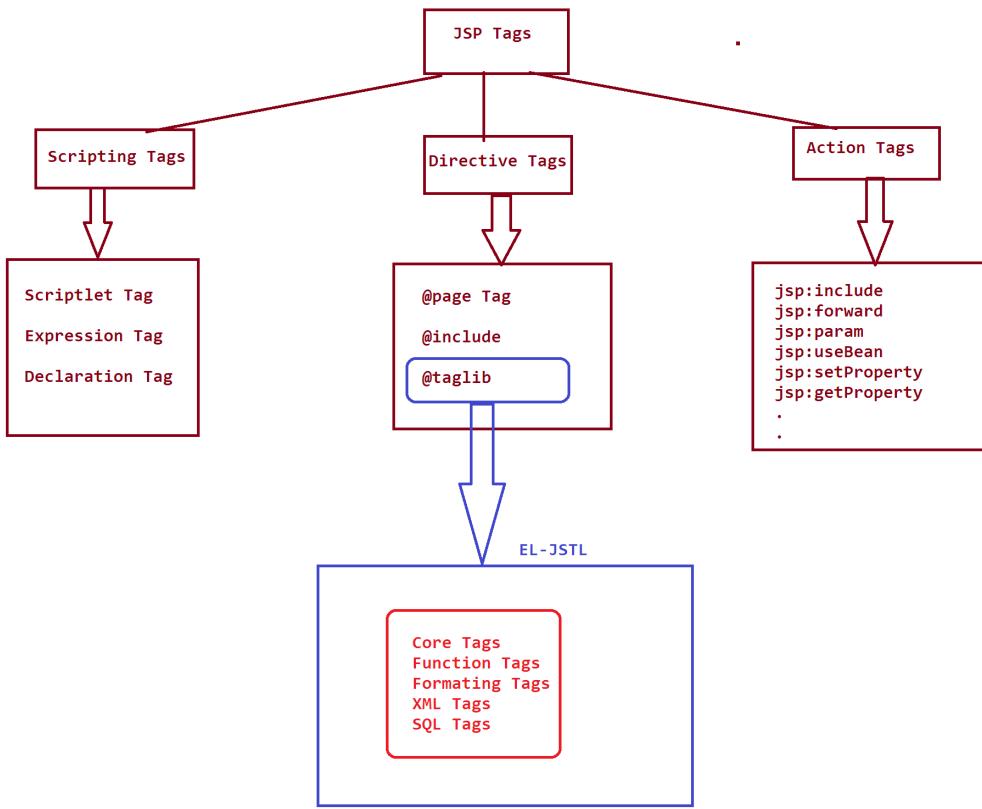
- 1.pageScope**
 - 2.sessionScope**
 - 3.requestScope**
 - 4.applicationScope**
 - 5.param**
 - 6.paramValues**
 - 7.cookie**
 - 8.pageContext**
 - 9.header**
 - 10.headerValues**
 - 11.initParam**
-

Note:

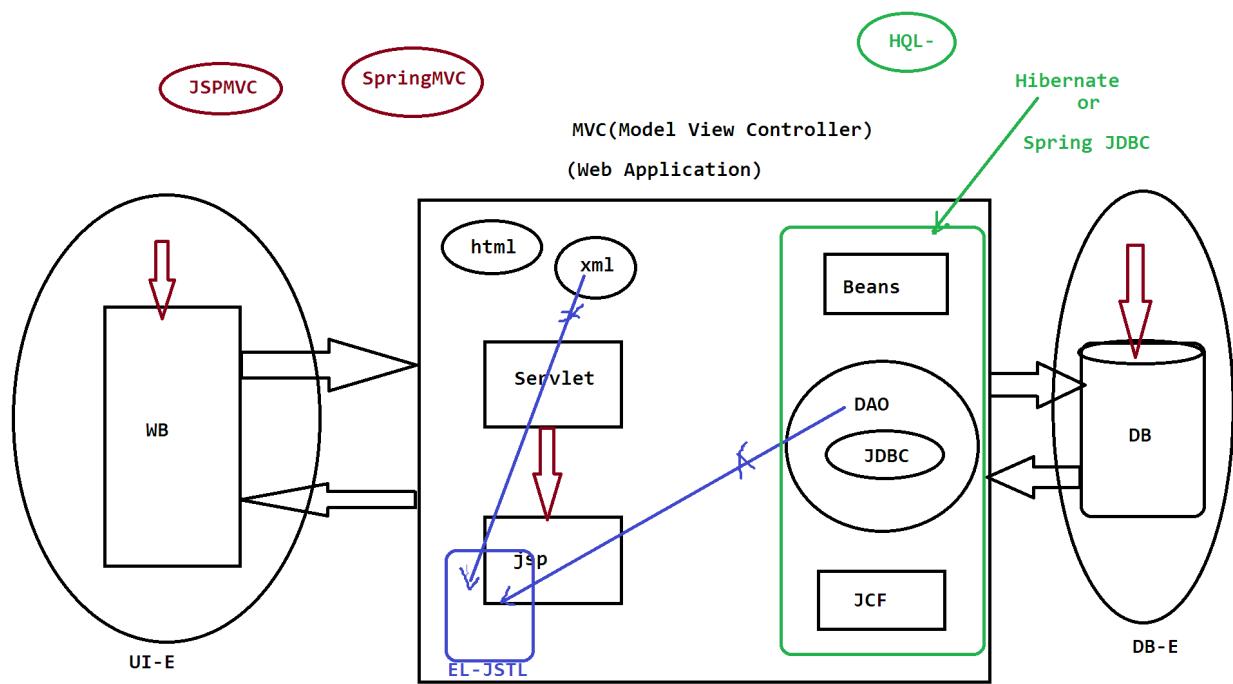
=>In realtime JSPEL replaces with SpEL.



=====



ist



=====END=====