

**Dt : 29/12/2021(Day-1)**

**define Application?**

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

**define Web Application?**

=>*The application which is executing in Web Environment or Internet Environment is known as Web Application.*

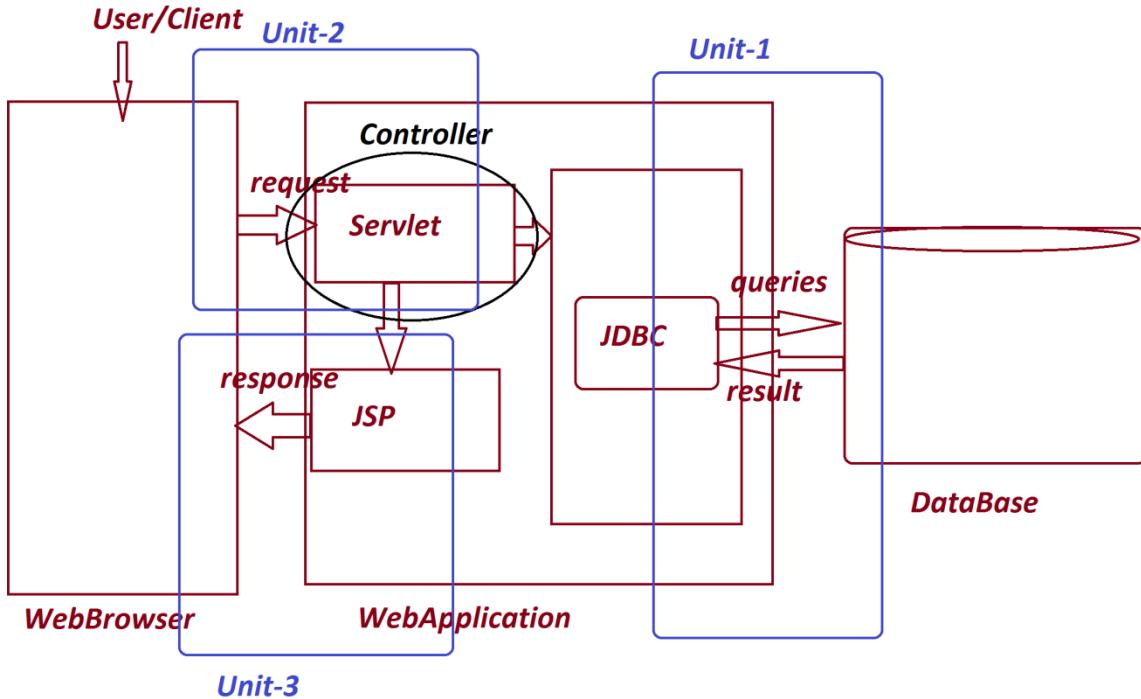
=>*The following technologies are used to construct Web Application:*

**1.JDBC**

**2.Servlet**

**3.JSP**

**Structure of Web Application:**



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

=>'Servlet' is a Server program used to accept the request from user.

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

=====

\*imp

**Types of Storages:**

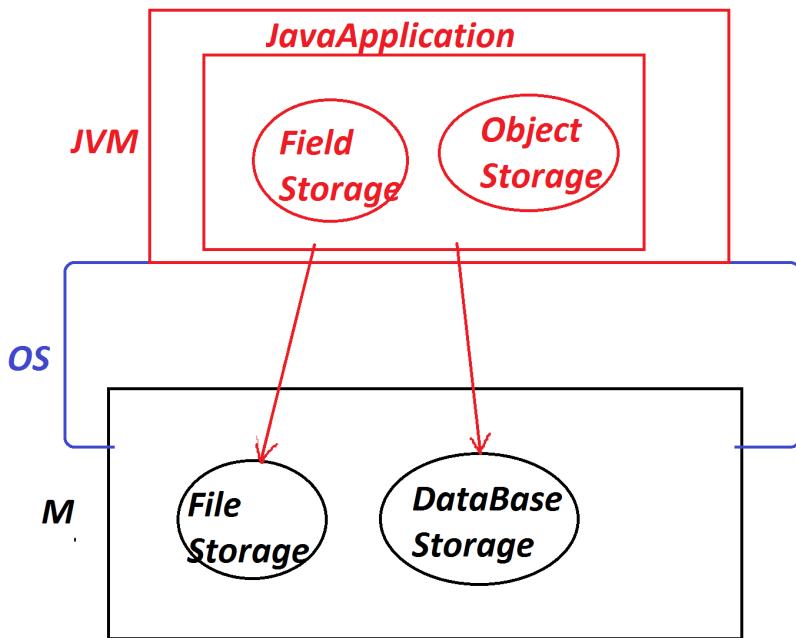
=>According to JavaApplication development the storages are categorized into four types:

**1.Field Storage**

**2.Object Storage**

**3.File Storage**

**4.DataBase Storage**



### 1. Field Storage:

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

Note:

=>Field Storages are generated when we use Primitive DataType variables  
`(byte,short,int,long,float,double,char,boolean)`

=>Field Storages can be part of Classes(static variables) or part of Objects  
`(Instances Variables)` or part of methods(Local variables)

### 2. Object Storage:

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

**Note:**

=>*Object Storages are generated when we use NonPrimitive DataTypes.*

*(Class,Interface,Array,Enum)*

---

**Note:**

=>*The Field Storages and Object Storages which are generated part of Java Application will be destroyed automatically when JVM ShutDowns.*

=>*Because of this reason we have to take the support of any one of the following permanent storage:*

*(i)File Storage*

*(ii)DataBase Storage*

---

*Dt : 30/12/2021(Day-2)*

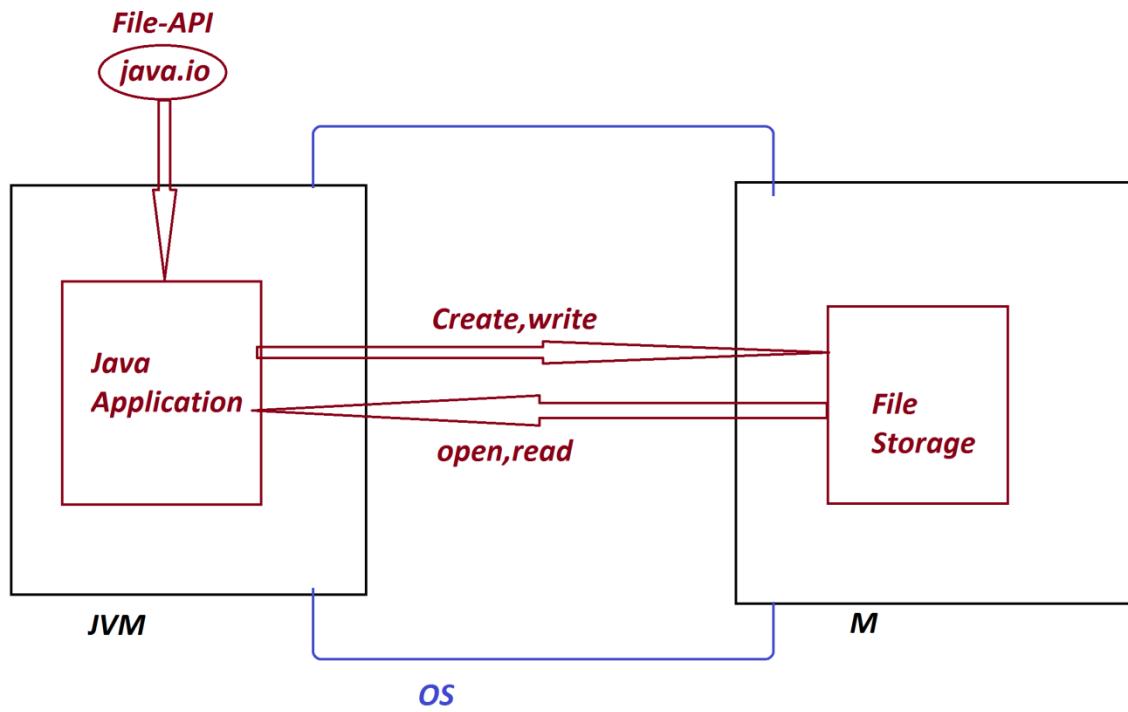
**3.File Storage::**

=>*The small permanent storage of ComputerSystem which is controlled and managed by the OperatingSytstem(OS) is known as File Storage.*

**Note:**

=>*When we want to establish communication b/w JavaProgram and File Storage, the JavaProgram must be constructed using classes and interfaces available from java.io package known as File-API.*

**Diagram:**



*faq:*

*define API?*

=>*API Stands for 'Application Programming Interface' and which is collection of related classes and Interfaces used for application development.*

=>*According to JavaDeveloper API means package.*

*AdvJava:*

*java.sql - DataBase Connection package(JDBC-API)*

*javax.servlet - Servlet Programming package(Servlet-API)*

*javax.servlet.jsp - JSP Programming package(JSP-API)*

### ***Dis-Advantages of File Storage:***

- 1. Data Redundancy**
- 2. Data Inconsistency**
- 3. Difficulty in Accessing Data**
- 4. Limited Data Sharing**

#### ***1. Data Redundancy:***

=>*In File Storage there is a choice of duplicate data known as Data Redundancy.*

#### ***2. Data Inconsistency:***

=>*Because of Data Redundancy there will be inconsistent data, known as Data InConsistency.*

#### ***3. Difficulty in Accessing Data:***

=>*Accessing data in File Storage is very difficult because the data is available in scattered form. (diff folders, files, ...)*

#### ***4. Limited Data Sharing:***

=>*Data Sharing in File Storage is not possible when compared to DataBase.*

=====

**Dt : 31/12/2021(Day-3)**

**Note:**

=>*The Dis-Advantages of File Storage can be overcome using DataBase storage.*

\*imp

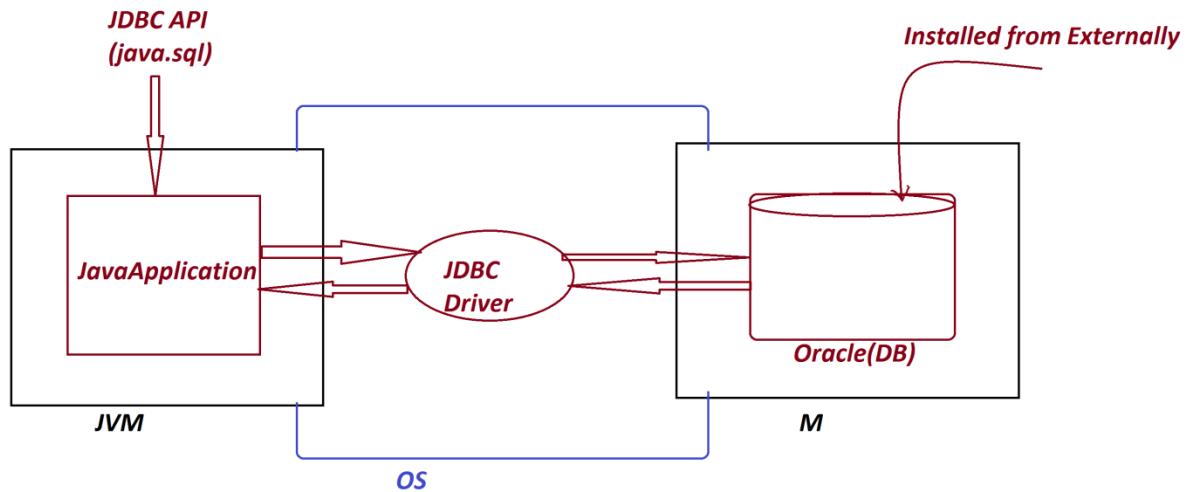
#### 4. DataBase Storage:

=>The largest permanent storage which is installed into ComputerSystem from externally is known as DataBase Storage.

Note:

=>In the process of establishing communication b/w JavaProgram and DataBase product, the JavaProgram must be constructed using 'classes and Interfaces' available from 'java.sql' package(JDBC-API) and the JavaProgram must take the support of JDBC driver.

Diagram:



faq:

define driver?

=>The small s/w programs part of Operating System used to control resources of Computer System are known as 'driver'.

Ex:

**Audio driver**

**Video driver**

**N/W driver**

...

**faq:**

**define JDBC driver?**

=>The driver which support Java Program to interact with DataBase product is known as JDBC driver.(Java DataBase Connectivity driver)

=>These JDBC drivers are categorized into four types:

(a)Type-1 : JDBC-ODBC Bridge Driver

(b)Type-2 : Native API Driver

(c)Type-3 : Network Protocol Driver

(d)Type-4 : Thin Driver

**Note:**

=>In realtime we use Type-4 : Thin driver.

=====

Dt : 3/1/2022

\*imp

*creating System Environment ready for constructing JDBC Applications:*

**step-1 : DownLoad and Install DataBase Product(Oracle)**

**step-2 : Perform Login Process**

*Click on Start->Search->Type 'SQL Command Line' (press enter)*

*SQL>connect (press Enter)*

*Enter User-name : system (press enter)*

*Enter password : manager (press enter)*

*connected.*

*SQL>*

**step-3 : Create table in DataBase product**

*DB Table : Product41(pcode,pname,pprice,pqty)*

```
create table Product41(pcode varchar2(10),pname varchar2(15),
pprice number(10,2),pqty number(10),primary key(pcode));
```

```
SQL> create table Product41(pcode varchar2(10),pname varchar2(15),
2 pprice number(10,2),pqty number(10),primary key(pcode));
```

*Table created.*

```
SQL> desc Product41;
```

| Name   | Null? Type            |
|--------|-----------------------|
| PCODE  | NOT NULL VARCHAR2(10) |
| PNAME  | VARCHAR2(15)          |
| PPRICE | NUMBER(10,2)          |
| PQTY   | NUMBER(10)            |

```
SQL> commit;
```

*Commit complete.*

```
SQL>
```

**step-4 : Insert records into DB-Table**

```
SQL> insert into Product41 values('A122','Mouse',235.67,12);
```

*1 row created.*

*SQL> insert into Product41 values('A121','KB',135.67,10);*

*1 row created.*

*SQL> insert into Product41 values('A120','FDD',234.67,9);*

*1 row created.*

*SQL> insert into Product41 values('A111','CDR',454.67,12);*

*1 row created.*

*SQL> insert into Product41 values('A555','GH',45.67,9);*

*1 row created.*

*SQL>commit;*

*Commit complete.*

*SQL>*

*step-5 : View the records from DB-Table(Select-query)*

*select \* from Product41;*

*SQL> select \* from Product41;*

| <i>PCODE</i> | <i>PNAME</i> | <i>PPRICE</i> | <i>PQTY</i> |
|--------------|--------------|---------------|-------------|
| <i>A122</i>  | <i>Mouse</i> | <i>235.67</i> | <i>12</i>   |
| <i>A121</i>  | <i>KB</i>    | <i>135.67</i> | <i>10</i>   |

|             |            |               |           |
|-------------|------------|---------------|-----------|
| <b>A120</b> | <b>FDD</b> | <b>234.67</b> | <b>9</b>  |
| <b>A111</b> | <b>CDR</b> | <b>454.67</b> | <b>12</b> |
| <b>A555</b> | <b>GH</b>  | <b>45.67</b>  | <b>9</b>  |

*SQL>*

=====

**Assignment:**

**Create one DB Table : Book41(bcode,bname,bauthor,bprice,bqty)**

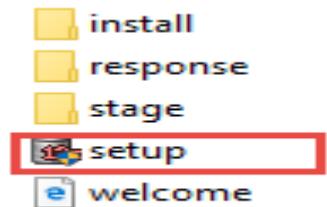
**primary key : bcode**

**Insert min 5 Book details**

**View the Books**

=====

The following picture shows the structure of the folder of the Oracle installation files after extraction.



**Step 1.** The installer asks you to provide your email address to get the latest security issues and updates. You can ignore it by clicking the Next button



Oracle Database 12c Release 1 Installer - Step 1 of 9



ORACLE  
DATABASE

## Configure Security Updates



### Configure Security Updates



### Installation Option



System Class



Oracle Home User Selection



Installation Location



Prerequisite Checks



Summary



Install Product



Finish

Provide your email address to be informed of security issues, install the product and initiate configuration manager. [View details.](#)

Email:

Easier for you if you use your My Oracle Support email address/username

I wish to receive security updates via My Oracle Support.

My Oracle Support Password:

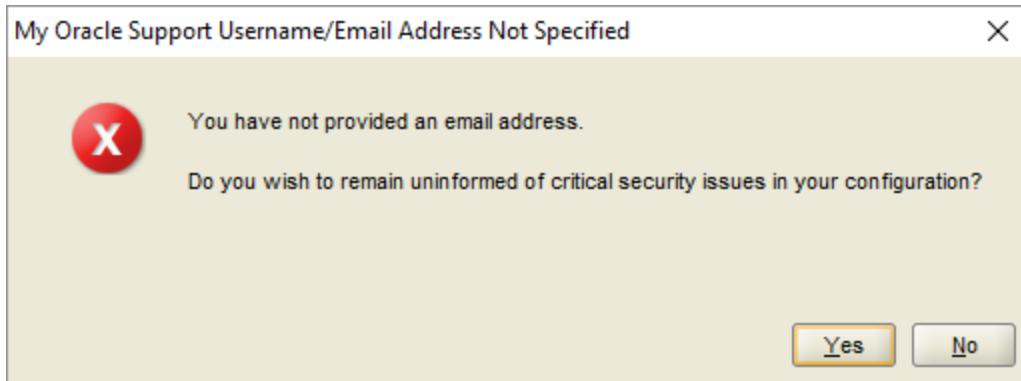
[Help](#)

[< Back](#)

[Next >](#)

[Install](#)

Because I didn't provide the email address, the Oracle database installer confirm it, you just need to click the No button to continue.



**Step 2.** In step 2, Oracle installer ask you to whether you want to create and configure a database, install database software only or just upgrade an existing database. Because you install the Oracle database at the first time, choose the option 1 and click the Next button.



Oracle Database 12c Release 1 Installer - Step 2 of 9



ORACLE  
DATABASE

## Select Installation Option

[Configure Security Updates](#)

**Installation Option**

[System Class](#)

Oracle Home User Selection

Installation Location

Prerequisite Checks

Summary

Install Product

Finish

Select any of the following install options.

[Create and configure a database](#)

[Install database software only](#)

[Upgrade an existing database](#)

[Help](#)

[< Back](#)

[Next >](#)

[Install](#)

**Step 3.** The installer allows you to choose the system class. Because you install Oracle on your computer, not a server, therefore, you choose the first option: desktop class and click the Next button.

**System Class**

- [Configure Security Updates](#)
- [Installation Option](#)
- System Class**
- [Oracle Home User Selection](#)
- Installation Location
- Prerequisite Checks
- Summary
- Install Product
- Finish

 **Desktop class**

Choose this option if you are installing on a laptop or desktop class system. This option includes a standard Oracle Home and allows minimal configuration.

 **Server class**

Choose this option if you are installing on a server class system, which Oracle defines as a system used for production data center. This option allows for more advanced configuration options.

[Help](#)[< Back](#)[Next >](#)[Install](#)

**Step 4.** This step allows you to specify the Windows user account to install and configure Oracle Home for enhanced security. Choose the third option: “Use Windows Built-in Account”.

**Specify Oracle Home User**

- [Configure Security Updates](#)
- [Installation Option](#)
- [System Class](#)
- Oracle Home User Selection**
- [Installation Location](#)
- Prerequisite Checks
- Summary
- Install Product
- Finish

Oracle recommends that you specify a standard Windows User Account (not an Administrator account) and configure the Oracle Home for enhanced security. This account is used for running the Windows Service for the Oracle Home. Do not log in using this account to perform administrative tasks.

Use Existing Windows User

User Name:

Password:

Create New Windows User

User Name:

Password:

Confirm Password:

The newly created user is denied Windows logon privileges.

Use Windows Built-in Account

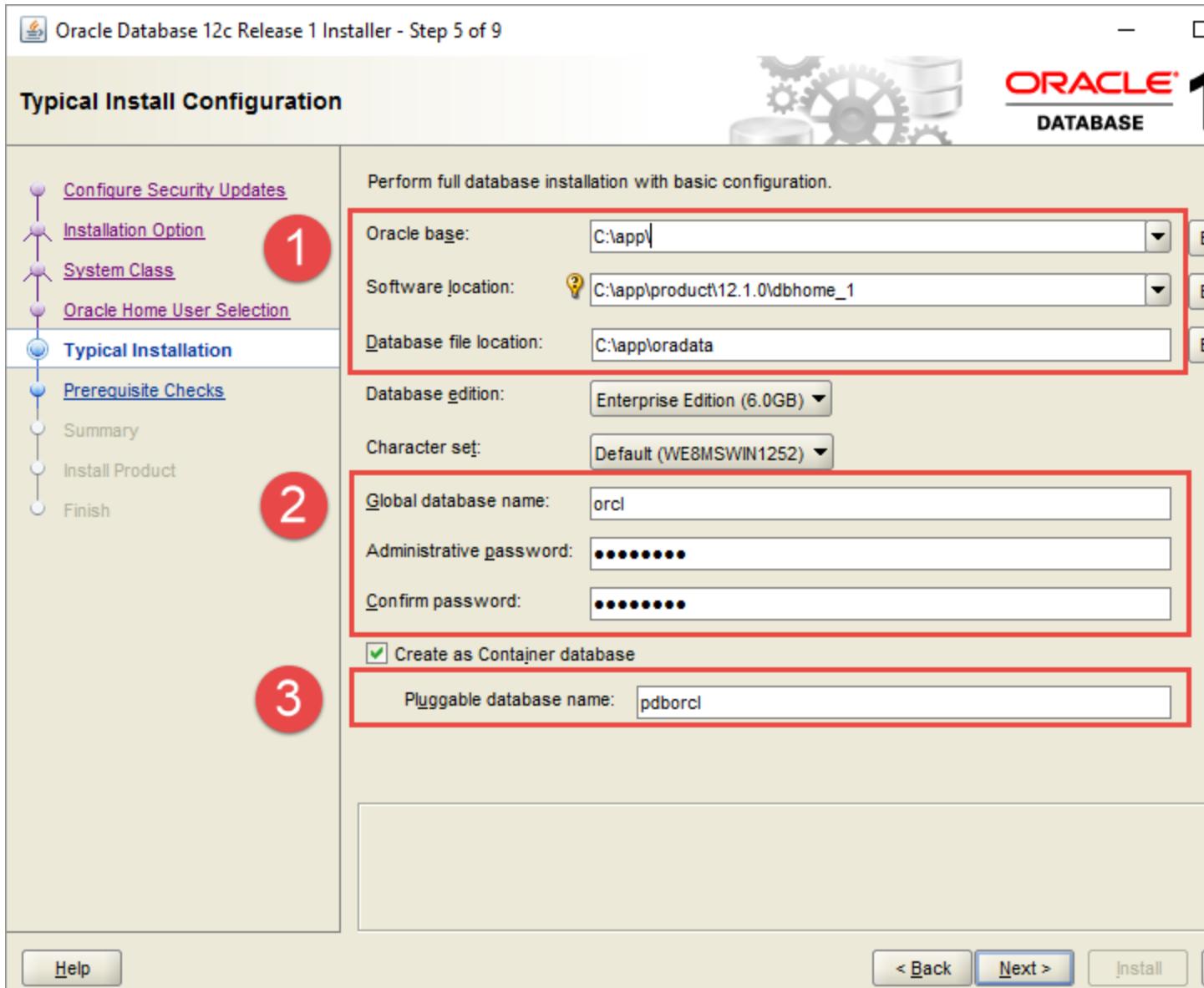
[Help](#)

[< Back](#)

[Next >](#)

[Install](#)

**Step 5.** in this step you can (1) choose the folder on which Oracle database will be installed, (2) Global database name and password, (3) pluggable database name.



**Step 6.** The installer performs the prerequisite check.

**Perform Prerequisite Checks**

- Configure Security Updates
- Installation Option
- System Class
- Oracle Home User Selection
- Typical Installation
- Prerequisite Checks**
- Summary
- Install Product
- Finish

Verifying that the target environment meets minimum installation and configuration requirements for products you have selected. This can take time. Please wait.

67%

Checking Architecture ...

[Help](#)[< Back](#)[Next >](#)[Install](#)

**Step 7.** The installer shows you the summary of the information such as global settings, database information, etc. You need to review the information and click the install button if everything is fine.

The screenshot shows the Oracle Database 12c Release 1 Installer at Step 7 of 9, titled "Summary". On the left, a sidebar lists steps: "Configure Security Updates", "Installation Option", "System Class", "Oracle Home User Selection", "Typical Installation", "Prerequisite Checks", "Summary" (which is selected), "Install Product", and "Finish". The main pane displays "Oracle Database 12c Release 1 Installer" settings under two sections: "Global settings" and "Database information".

**Global settings**

- Disk space: required 6.0 GB available 29.88 GB [[Edit](#)]
- Oracle Home User Selection: NT AUTHORITY\SYSTEM
- Source location: D:\software\winx64\_12102\_database\_1of2\database\install\..\stage\produ...
- Install method: Desktop installation [[Edit](#)]
- Database edition: Enterprise Edition (Create and configure a database) [[Edit](#)]
- Oracle base: C:\app\ [Edit]
- Software location: C:\app\product\12.1.0\dbhome\_1 [[Edit](#)]
- Oramts Port Number: 49152

**Database information**

- Configuration: General Purpose / Transaction Processing
- Global database name: orcl [[Edit](#)]
- Oracle system identifier (SID): orcl [[Edit](#)]
- Pluggable database name: pdborcl [[Edit](#)]
- Allocated memory: 3236 MB
- Automatic memory management option: FALSE
- Database character set : West European (WE8MSWIN1252) [[Edit](#)]
- Management method: Database express
- Database storage mechanism: File system

At the bottom right are buttons: "Save Response", "< Back", "Next >", and "Install".

**Step 8.** The installer starts installing Oracle database. It will take a few minutes to complete, depending on your computer.



Oracle Database 12c Release 1 Installer - Step 8 of 9

## Install Product

**ORACLE<sup>®</sup>**  
DATABASE

- Configure Security Updates
- Installation Option
- System Class
- Oracle Home User Selection
- Typical Installation
- Prerequisite Checks
- Summary
- Install Product**
- Finish

### Progress

0%

Central Inventory is not locked.

### Status

| Task                          | Description                          | Status      |
|-------------------------------|--------------------------------------|-------------|
| Oracle Database installation  | • Prepare<br>• Copy files<br>• Setup | In Progress |
| Setup Oracle Base             |                                      | Pending     |
| Oracle Database configuration |                                      | Pending     |

[Details](#)[Retry](#)**ORACLE<sup>®</sup>**  
DATABASE **12<sup>c</sup>**[Help](#)[< Back](#)[Next >](#)[Install](#)



Oracle Database 12c Release 1 Installer - Step 8 of 9



**ORACLE®**  
DATABASE

## Install Product

- Configure Security Updates
- Installation Option
- System Class
- Oracle Home User Selection
- Typical Installation
- Prerequisite Checks
- Summary
- Install Product**
- Finish

### Progress



7%

### Status

|   |                               | In Progress |
|---|-------------------------------|-------------|
| ➡ | Oracle Database installation  | Succeeded   |
| ✓ | • Prepare                     | Pending     |
| ✓ | • Copy files                  | Pending     |
| ✓ | • Setup                       | Pending     |
|   | Setup Oracle Base             | Pending     |
|   | Oracle Database configuration | Pending     |

[Details](#)

[Retry](#)

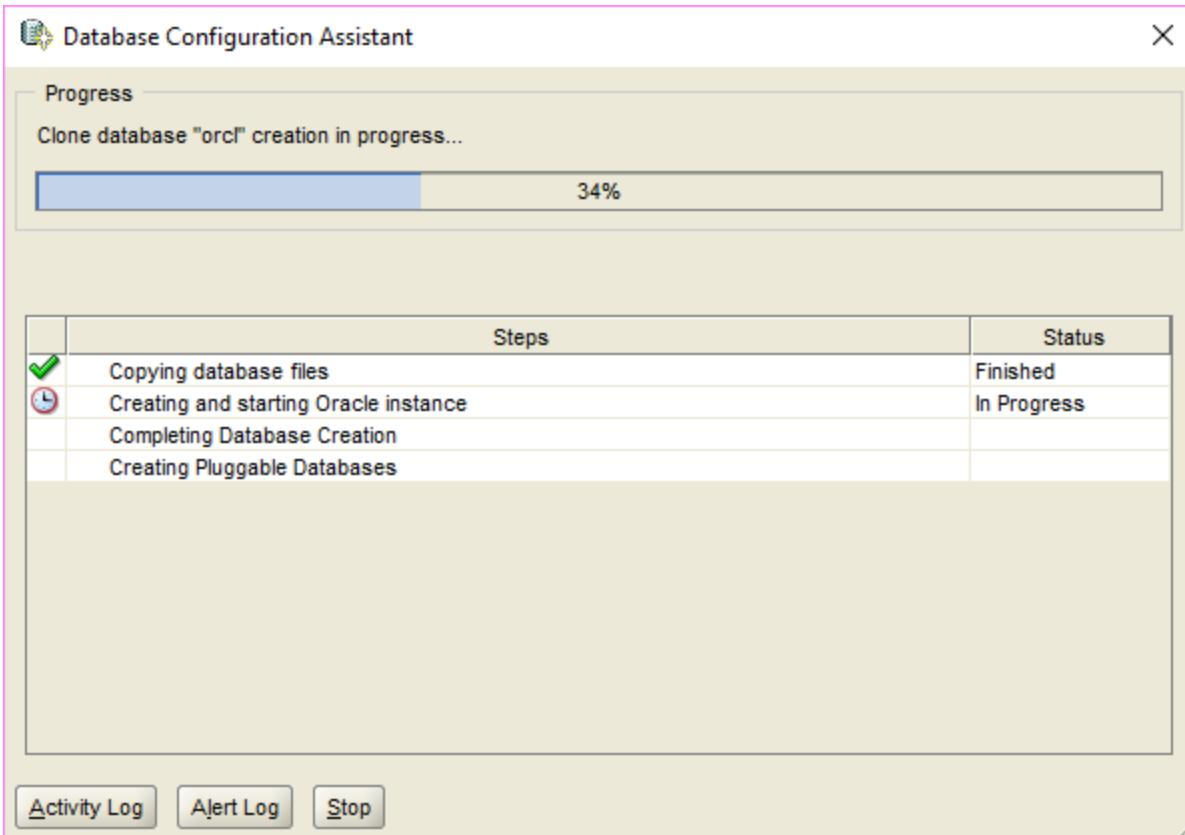
**ORACLE®**  
DATABASE **12<sup>c</sup>**

[Help](#)

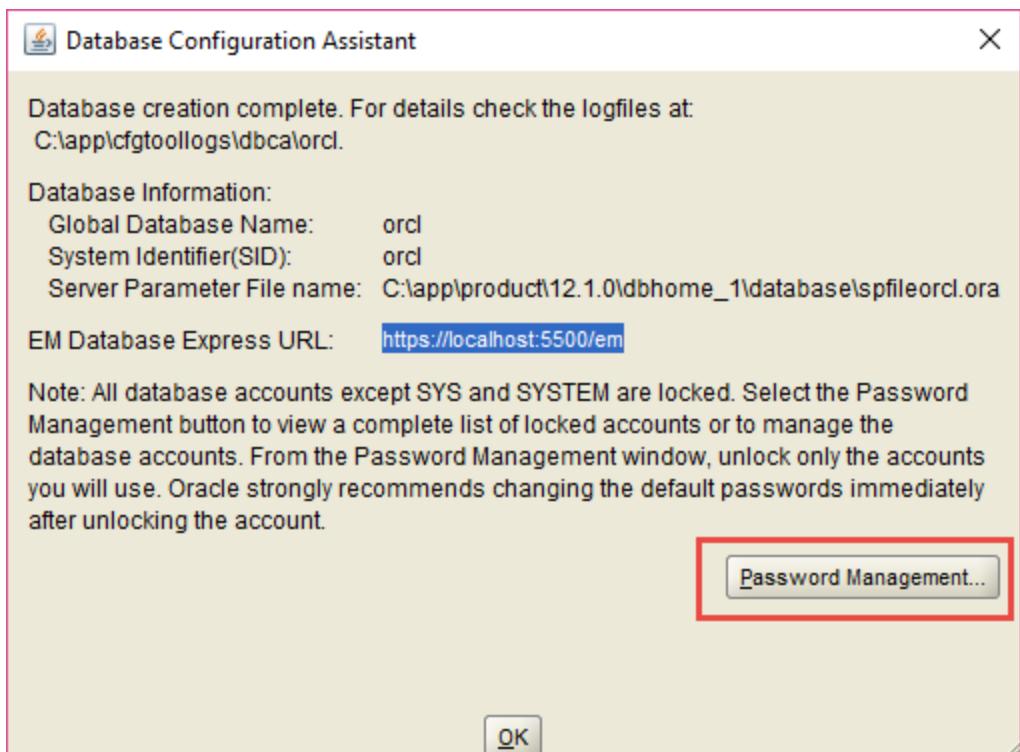
[< Back](#)

[Next >](#)

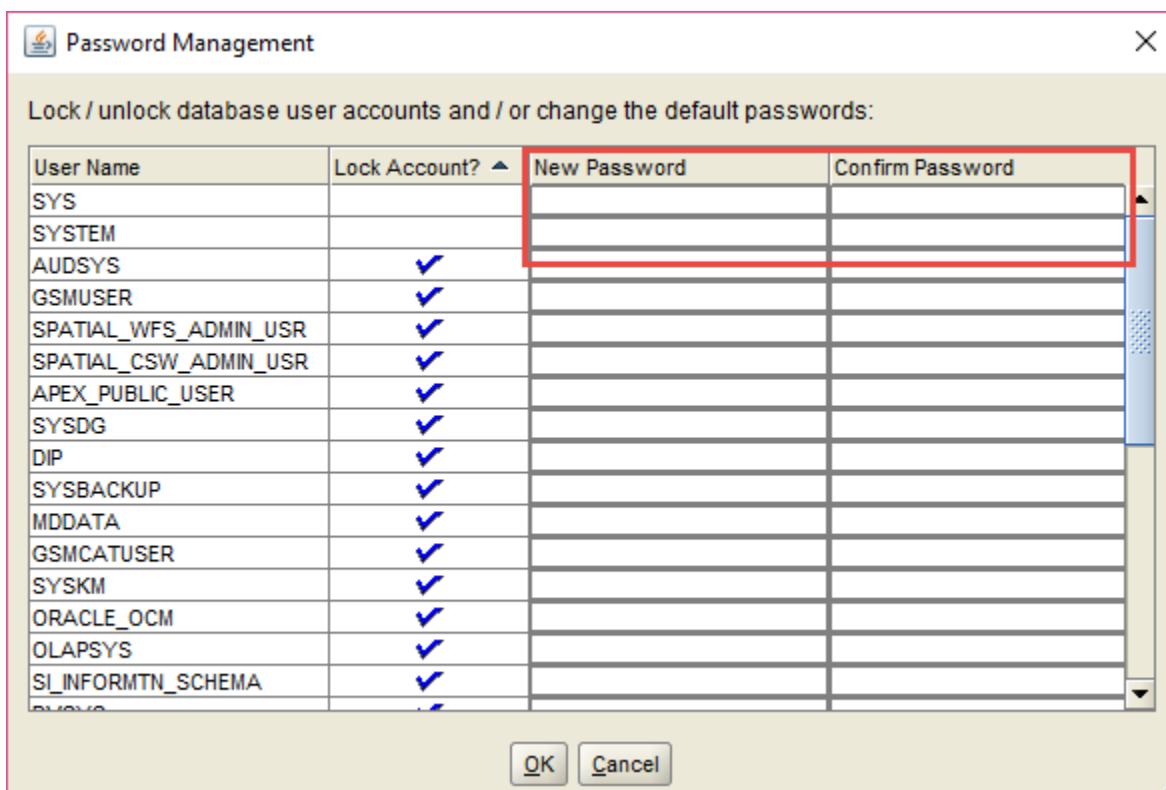
[Install](#)



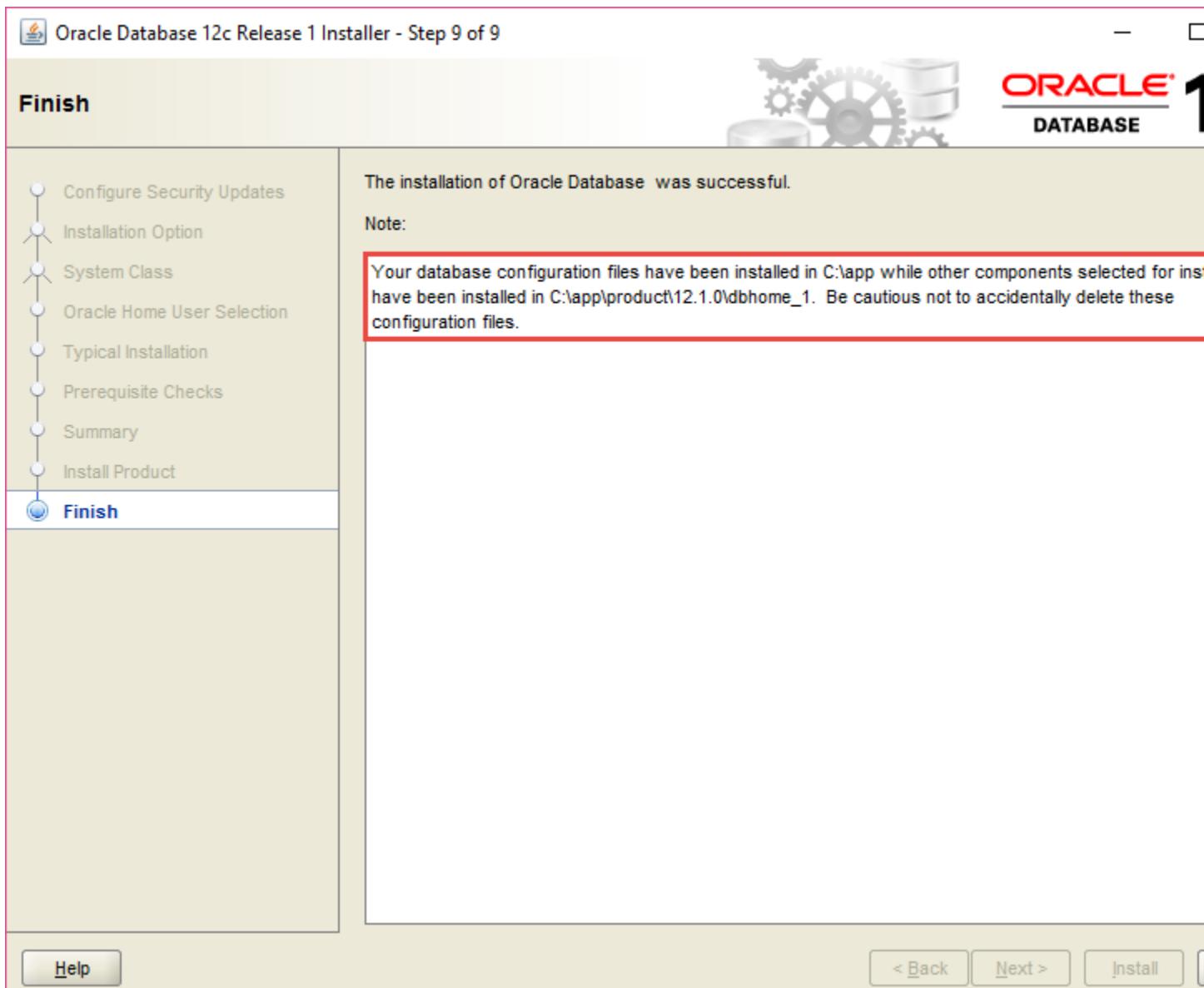
You will see the Database Configuration Assistant window. Click the *Password management...* button to enter the password for Oracle database accounts.



Enter the password for SYS and SYSTEM accounts and then click OK button.



**Step 9.** Once installation completes successfully, the installer will inform you as shown in the following screenshot. Click the Close button to close the window.



# Java Database Connectivity with MySQL

1. **Driver class:** `com.mysql.jdbc.Driver`.
2. **Connection URL:** `jdbc:mysql://localhost:3306/sonoo`
3. **Username:** `root`.
4. **Password:** It is the password given by the user at the time of installing the mysql database.

Dt : 4/1/2022

**Step-6 : Find DB-Jar file from 'lib' of 'JDBC" and copy into one user defined**

**Folder**

**Ex:**

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

**ojdbc6.jar - Oracle11**

**\*imp**

**define DataBase JAR files:**

=>JAR stands for 'Java Archive' and which is compressed format of  
**class files.**

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

**Oracle10 - ojdbc14.jar**

**Oracle11 - ojdbc6.jar**

**oracle12 - ojdbc7.jar,ojdbc8.jar**

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

**MySQL - mysql-connector-java-VERSION.jar**

**SQL Server - sqljdbc41.jar, sqljdbc42.jar**

**PostgreSQL - postgresql-VERSION.jar**

**Apache Derby - derby.jar, derbyclient.jar**

**SQLite - sqlite-jdbc-VERSION.jar**

**Microsoft Access - ucanaccess-VERSION.jar**

**Step-7 : Find Oracle Product PortNo and ServiceName**

=>Open 'tnsnames.ora' file from ADMIN of network to find Oracle PortNo and ServiceName.

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

port : 1521

service\_name : XE

=====

\*imp

**JDBC API?**

=>'java.sql' package is known as JDBC-API and which provide the classes and Interfaces used in constructing JDBC Applications.

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

=>The following are some important methods from 'java.sql.Connection':

1.createStatement()

2.prepareStatement()

3.prepareCall()

4.setAutoCommit()

5.getAutoCommit()

6.setSavepoint()

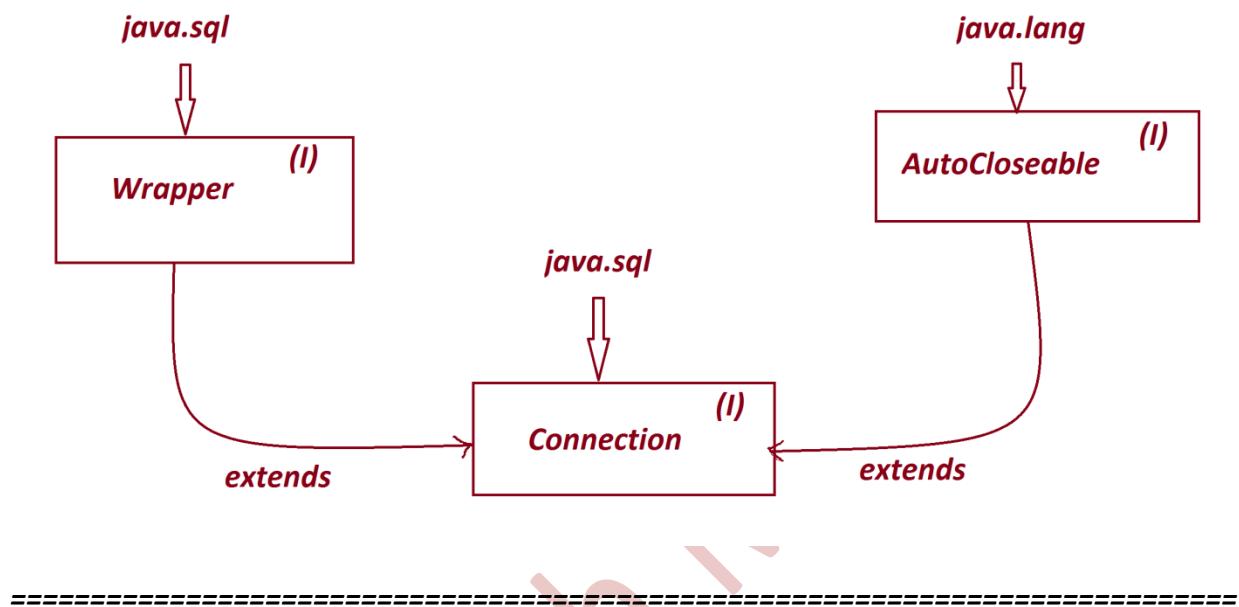
7.commit()

8.rollback()

9.releaseSavepoint()

`10.close()`

*Hierarchy of 'Connection':*



*faq:*

*define `getConnection()` method?*

*=>`getConnection()` method is used to create the implementation object of 'Connection' interface.*

*=>This `getConnection()` method is available from 'java.sql.DriverManager' class.*

*Method signature:*

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

*syntax:*

```
Connection con = DriverManager.getConnection("DB-URL","username","password");
```

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

*username = system*

*password = manager*

*Access  
within  
project*

*return\_type*

*with parameter*

*public static Connection getConnection(String, String, String) throws SQLException;*

*Class\_level*

*method\_name*

*Ignore the exception from  
current method and raise at  
method\_call*

*API*

*driver*

*portNo*

*jdbc:oracle:thin:@localhost:1521:xe*

*Product  
Name*

*Location*

*service\_name*

---

Dt : 5/1/2022

**define forName() method?**

=>forName() method is used to load JDBC-Driver to current running program at runtime or execution time.

=>This forName() method is available from 'java.lang.Class'.

**Method Signature:**

```
public static java.lang.Class<?> forName(java.lang.String)  
throws java.lang.ClassNotFoundException;
```

**syntax:**

```
Class c = Class.forName("JDBC-Driver-Class");
```

(or)

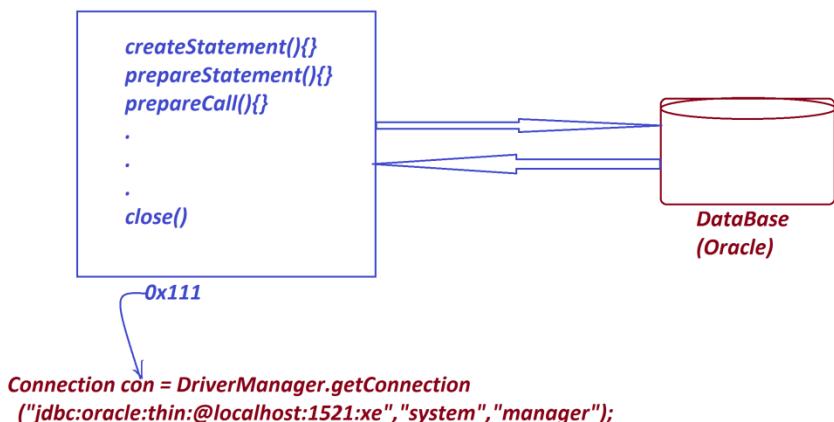
```
Class.forName("JDBC-Driver-Class");
```

**Ex:**

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

---

**Diagram:**



=====

\*imp

### **Types of JDBC Statements:**

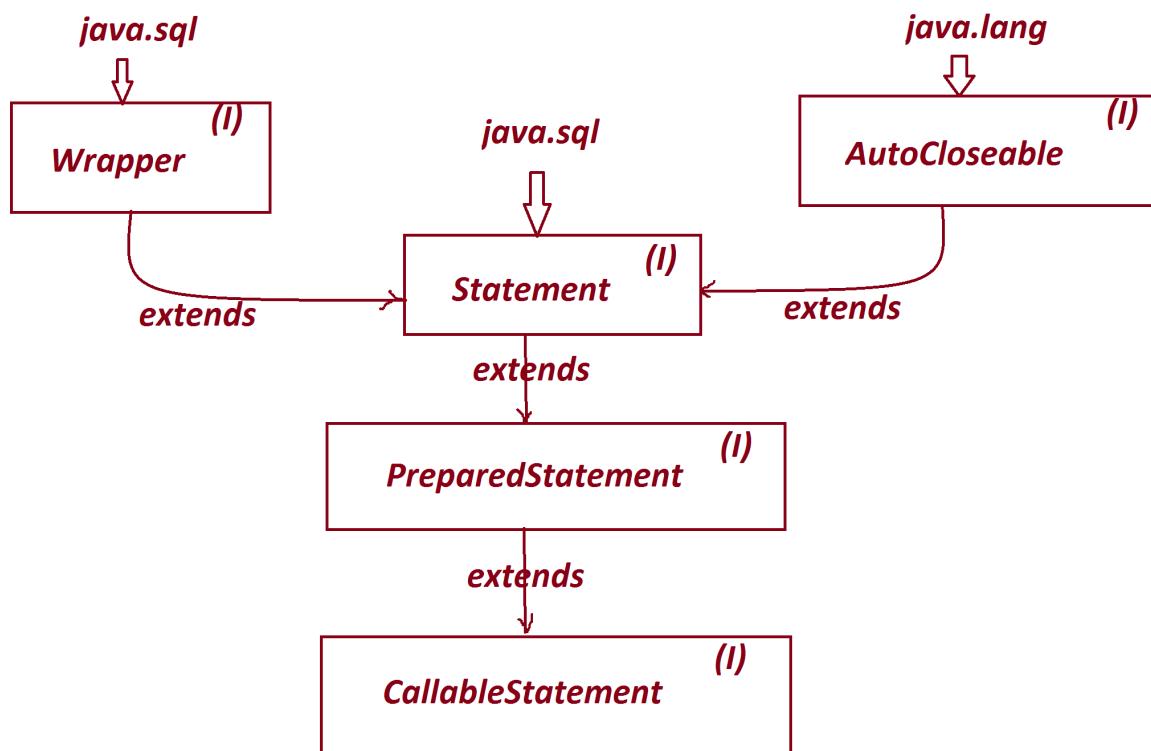
=>JDBC-statements are categorized to three types:

**1.Statement**

**2.PreparedStatement**

**3.CallableStatement**

### **Hierarchy of JDBC-statements:**



=====

**1.Statement:**

=>'Statement' is an interface from `java.sql` package and which is used to execute normal queries(SQL-statements) without IN parameters.

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

**Method Signature of `createStatement()`:**

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

**syntax:**

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

=>The following are some important methods from 'Statement':

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

**(i)`executeQuery()`:**

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

**Method Signature:**

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

**syntax:**

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

**Ex:**

```
ResultSet rs = stm.executeQuery("select * from Product41");
```

*(ii)executeUpdate():*

*=>executeUpdate() method is used to execute Non-Select queries.*

*Method Signature:*

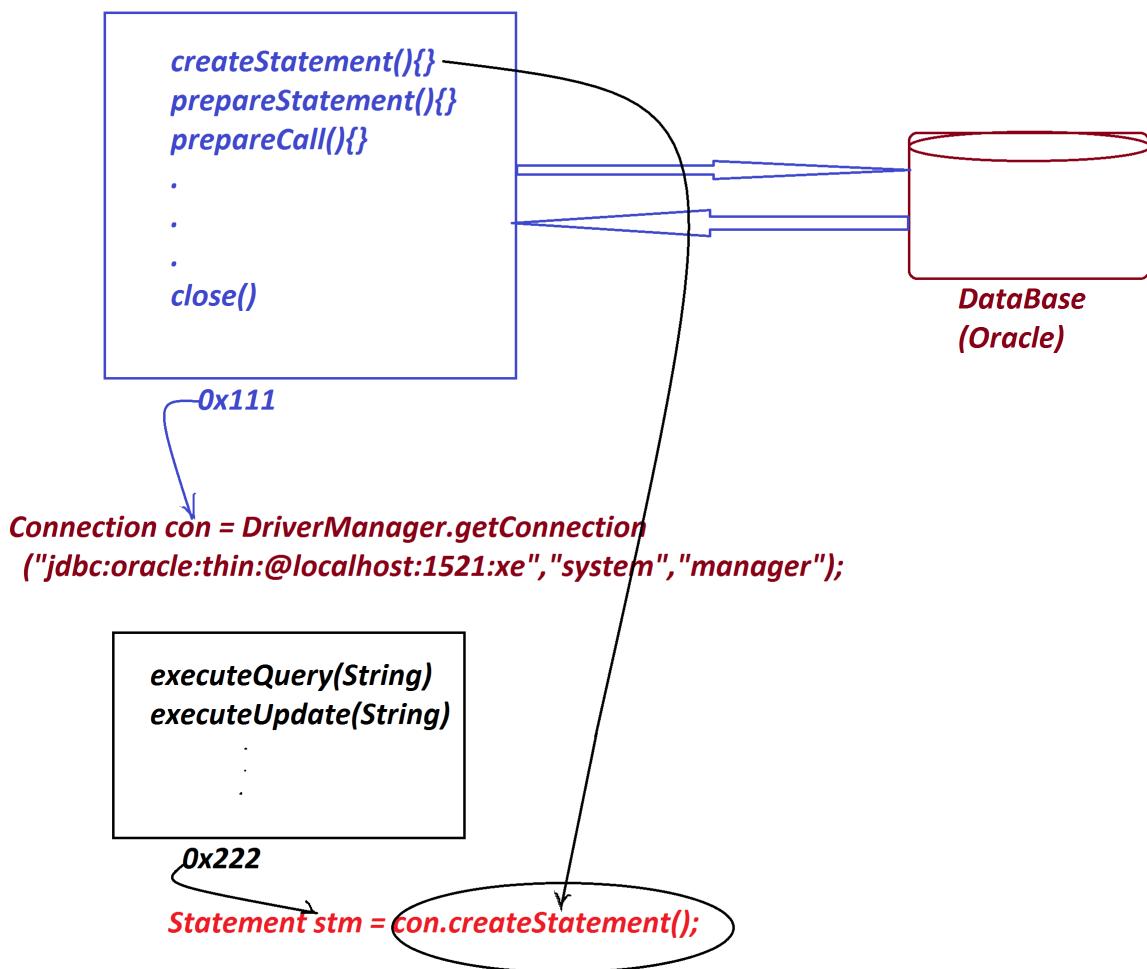
```
public abstract int executeUpdate(java.lang.String)  
                  throws java.sql.SQLException;
```

*syntax:*

```
int k = stm.executeUpdate("Non-Select-Query");
```

---

Dt : 6/1/2022



SQL> select \* from Product41;

| PCODE | PNAME | PPRICE | PQTY |
|-------|-------|--------|------|
| ----- |       |        |      |
| A122  | Mouse | 235.67 | 12   |
| A121  | KB    | 135.67 | 10   |
| A120  | FDD   | 234.67 | 9    |
| A111  | CDR   | 454.67 | 12   |
| A555  | GH    | 45.67  | 9    |

*\*imp*

**Construct JDBC Application using IDE Eclipse to display records from DB table**

**Product41:**

**step-1 : Open IDE Eclipse,while opening name the WorkSpace(folder) and click launch.**

**step-2 : Create Java Project**

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

**step-3 : Add DB-Jar file to Java Project**

**RightClick on JavaProject->Build path->Configure Build path->Libraries-> select classpath and click 'Add External Jars'->Browse and select DB-Jar file from user defined folder->Open->Apply->Apply and close.**

**step-4 : Create package**

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

**step-5 : Create class in package to write JDBC code.**

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

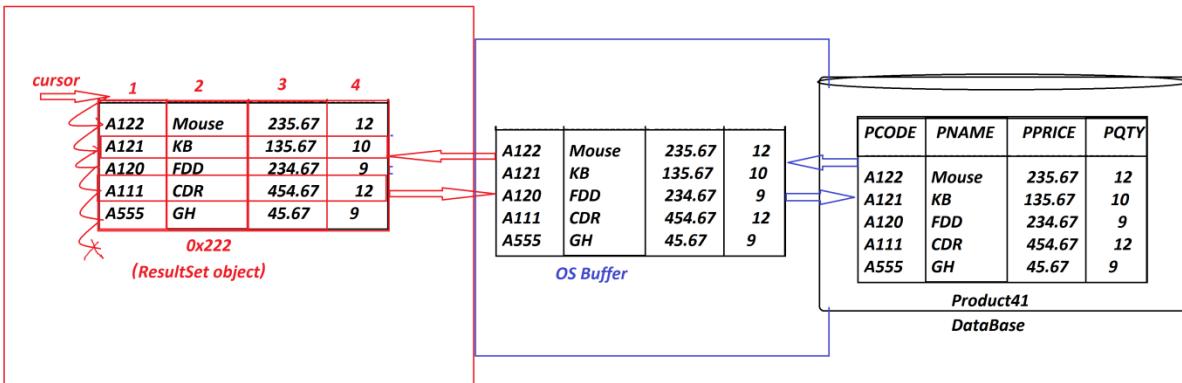
*program : DBCon1.java*

```
package test;
import java.sql.*;
public class DBCon1 {
    public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //Loading Driver
            Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe", "system", "manager");
            //Connection DataBase
            Statement stm = con.createStatement(); //JDBC-Statement
            ResultSet rs = stm.executeQuery("select * from
Product41");
            while(rs.next())
            {
                System.out.println(rs.getString(1)+"\t"+rs.getString(2)
                    +"\t"+rs.getFloat(3)+"\t"+rs.getInt(4));
            } //end of loop
            con.close();
        }catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

*step-6 : Execute the program*

---

*Diagram:*



### Assignment:

Construct JDBC Application to display book details from DB table Book41.

Dt : 7/1/2022

faq:

**define 'ResultSet'?**

=>'ResultSet' is an interface from java.sql package and which is used to hold result generated from Select-query.

=>we use executeQuery() method to execute select-query on DB-Product,in this process it generate implementation object of 'ResultSet' interface and which holds result of select-query.

=>This execute query method also generate cursor(header) automatically pointing before the first row.

=>we use next() method to move the cursor on ResultSet object row-by-row and this next() method boolean method,which means

**row available - true**

**row Not-Available - false**

=>This ResultSet will provide the following getter methods to get the data from the object.

**getString()**

**getFloat()**

**getInt()**

**...**

=>These getter methods will get the data based on Col\_no or Col\_name.

=====

=

**Ex\_Application:**

**DB Table : Employee41(eid,ename,edesg,bsal,totsal)**

**primary key : eid**

```
create table Employee41(eid varchar2(10),ename varchar2(15),edesg varchar2(10),
bsal number(10),totsal number(10,2),primary key(eid));
```

**Construct JDBC Application to read data from Console and insert into DB Table Employee41.**

**Program : DBCon2.java**

```
package test;

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

public class DBCon2 {

    public static void main(String[] args) {
        try {
            Scanner s = new Scanner(System.in);
            System.out.println("Enter the EmpId:");
            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 EmpBSal:");
int bSal = s.nextInt();
float totSal = bSal+(0.93F*bSal)+(0.63F*bSal);
Class.forName("oracle.jdbc.driver.OracleDriver");
//Loading Driver
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
//Connection DataBase
Statement stm = con.createStatement(); //JDBC-Statement
int k = stm.executeUpdate
("insert into Employee41 values('"+eId+"','"+eName+"','"++
eDesg+"','"+bSal+"','"+totSal+"')");
//Compiled and Executed
if(k>0) {
    System.out.println("Employee Details inserted...");
}
s.close();
}catch(Exception e) {
    e.printStackTrace();
}
}
}

o/p:

```

*Enter the EmpId:*

**A121**

**Enter the EmpName:**

**Ram**

**Enter the EmpDesg:**

**SE**

**Enter the EmpBSal:**

**12000**

**Employee Details inserted...**

**Ex\_Application:**

**JDBC Application to insert multiple Employee details based choice.**

**Program : DBCon3.java**

```
package test;

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

public class DBCon3 {

    public static void main(String[] args) {
        try {
            Scanner s = new Scanner(System.in);
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //Loading Driver
            Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
        }
    }
}
```

```
//Connection DataBase

Statement stm = con.createStatement(); //JDBC-Statement

System.out.println("Enter the number of Employees:");

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

//Read in the form of String and Convert into integer

for(int i=1;i<=n;i++)

{

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

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 EmpBSal:");

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

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

int k = stm.executeUpdate

("insert into Employee41 values('"+eId+"','"+eName+"','"+

eDesg+"','"+bSal+"','"+totSal+"')");

//Compiled and Executed

if(k>0) {

System.out.println("Employee Details inserted...");

}

}//end of loop
```

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

*o/p:*

*Enter the number of Employees:*

**2**

*Enter the EmpId:*

**A103**

*Enter the EmpName:*

**Raj**

*Enter the EmpDesg:*

**TE**

*Enter the EmpBSal:*

**13000**

*Employee Details inserted...*

*Enter the EmpId:*

**A101**

*Enter the EmpName:*

**RRR**

*Enter the EmpDesg:*

**ME**

*Enter the EmpBSal:*

**14000**

*Employee Details inserted...*

---

**Assignment-1:**

**step-1 : Create DB Table with name UserReg41**

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

**primary key(uname,pword)**

**step-2 : JDBC application to read user details from Console and insert into**

**DB table UserReg41**

---

**Assignment-2:**

**step-1 : Create DB Table with name Student41**

**(rollNo,stuName,branch,totmarks,per,result)**

**primary key(rollNo)**

**step-2 : JDBC application to read Student details from Console and insert into**

**DB table Srtudent41.**

**read : rollNo,name,branch and 6 sub marks**

**Calculate totMarks,per and result**

---

Dt : 8/1/2022

\*imp

## 2.PreparedStatement:

=>*PreparedStatement is an interface from java.sql package and which is used to execute Normal queries(SQL-statements) with IN Parameters.*

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

*Method Signature of prepareStatement():*

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

*syntax:*

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

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

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

*(i)executeQuery():*

=>*executeQuery() method is used to execute select queries.*

*Method Signature:*

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

*syntax:*

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

*Ex:*

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

*(ii)executeUpdate():*

*=>executeUpdate() method is used to execute Non-Select queries.*

*Method Signature:*

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

*syntax:*

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

---

**Assignment-1:(Solution with PreparedStatement)**

**step-1 : Create DB Table with name UserReg41**

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

*primary key(uname,pword)*

```
create table UserReg41(uname varchar2(15),pword varchar2(15),fname varchar2(15),  
lname varchar2(15),addr varchar2(15),mid varchar2(25),phno number(15),  
primary key(uname,pword));
```

**step-2 : JDBC application to read user details from Console and insert into**

*DB table UserReg41*

*Program : DBCon4.java*

```
package test;

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

public class DBCon4 {

    public static void main(String[] args) {

        try {

            Scanner s = new Scanner(System.in);

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

            String uName = s.nextLine();

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

            String pWord = s.nextLine();

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

            String fName = s.nextLine();

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

            String lName = s.nextLine();

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

            String addr = s.nextLine();

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

            String mId = s.nextLine();

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

            long phNo = s.nextLong();
        }
    }
}
```

```
Class.forName("oracle.jdbc.driver.OracleDriver");  
//Loading Driver  
Connection con = DriverManager.getConnection  
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");  
//Connection DataBase  
PreparedStatement ps=con.prepareStatement  
("insert into UserReg41 values(?,?,?,?,?,?)");  
//Compilation  
ps.setString(1,uName);  
ps.setString(2,pWord);  
ps.setString(3,fName);  
ps.setString(4,lName);  
ps.setString(5,addr);  
ps.setString(6,mId);  
ps.setLong(7,phNo);  
int k = ps.executeUpdate();//Execution  
if(k>0)  
{  
    System.out.println("User registered Successfully...");  
}  
con.close();  
s.close();
```

```
        }catch(Exception e) {  
            e.printStackTrace();  
        }  
    }  
  
o/p:
```

*Enter the UserName:*

*nit.*

*Enter the PassWord:*

*mzu672*

*Enter the FirstName:*

*Venkat*

*Enter the LastName:*

*M*

*Enter the Address:*

*SRNagr*

*Enter the MailId:*

*v@gmail.com*

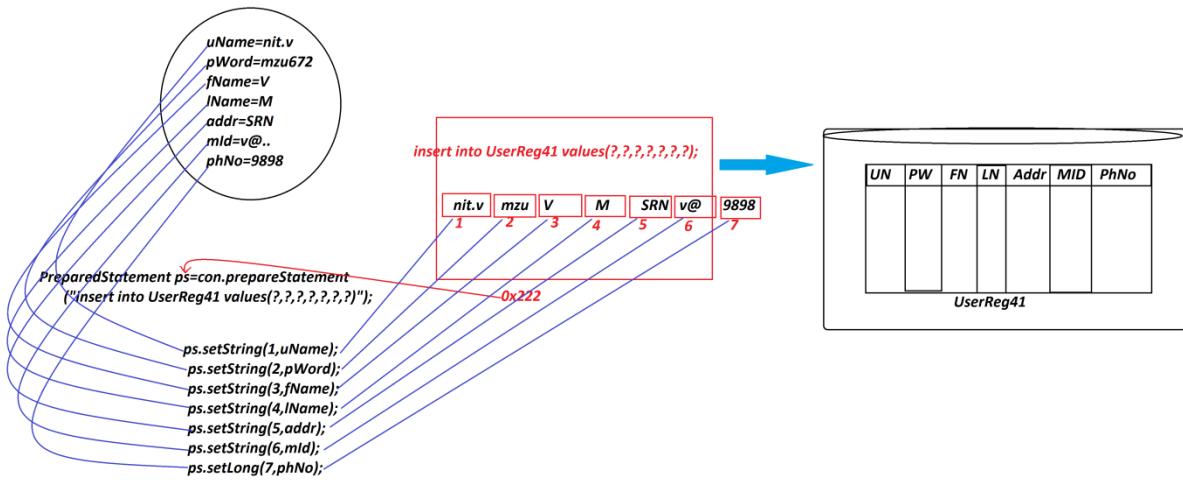
*Enter the PhoneNo:*

*9898981234*

*User registered Successfully...*

---

*Diagram:*



**Note:**

=>prepareStatement() method will compile the query-structure on DB product and if the compilation is successfull then it generate implementation object for 'PreparedStatement' interface with fields equal to the number of Parameter indexes.  
=>This PreparedStatement interface will provide the following 'setter methods' to set the data to fields based on parameter index value.

Ex:

```

    setString()
    setFloat()
    setInt()
    ...
  
```

Dt : 9/1/2022

Noe:

=>In PreparedStatement the query is compiled once and can be executed any number of times with IN Parameters.

=====

**Ex\_Program : JDBC Application to display user details based on uname and pword.**

**Query-Foramts:**

```
select * from UserReg41 where uname='nit.' and pword='mzu672';
```

```
select * from UserReg41 where uname='raj' and pword='raj';
```

**Program : DBCon5.java**

```
package test;

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

public class DBCon5 {

    public static void main(String[] args) {
        try {
            Scanner s = new Scanner(System.in);
            System.out.println("Enter the UserName:");
            String uName = s.nextLine();
            System.out.println("Enter the PassWord:");
            String pWord = s.nextLine();
        }
    }
}
```

```
Class.forName("oracle.jdbc.driver.OracleDriver");  
//Loading Driver  
Connection con = DriverManager.getConnection  
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");  
//Connection DataBase  
  
PreparedStatement ps = con.prepareStatement  
("select * from UserReg41 where uname=? and pword=?");  
ps.setString(1,uName);  
ps.setString(2,pWord);  
  
ResultSet rs = ps.executeQuery();  
if(rs.next()) {  
System.out.println("Login Successfull... ");  
System.out.println("FirstName:"+rs.getString(3));  
System.out.println("LastName:"+rs.getString(4));  
System.out.println("Address:"+rs.getString(5));  
System.out.println("MailId:"+rs.getString(6));  
System.out.println("PhoneNo:"+rs.getLong(7));  
  
}else {  
System.out.println("Invalid UserName and Password.. ");  
}  
s.close();
```

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

*o/p:*

*Enter the UserName:*

*raj*

*Enter the PassWord:*

*raj*

*Login Successfull...*

*FirstName:Raj*

*LastName:Kumar*

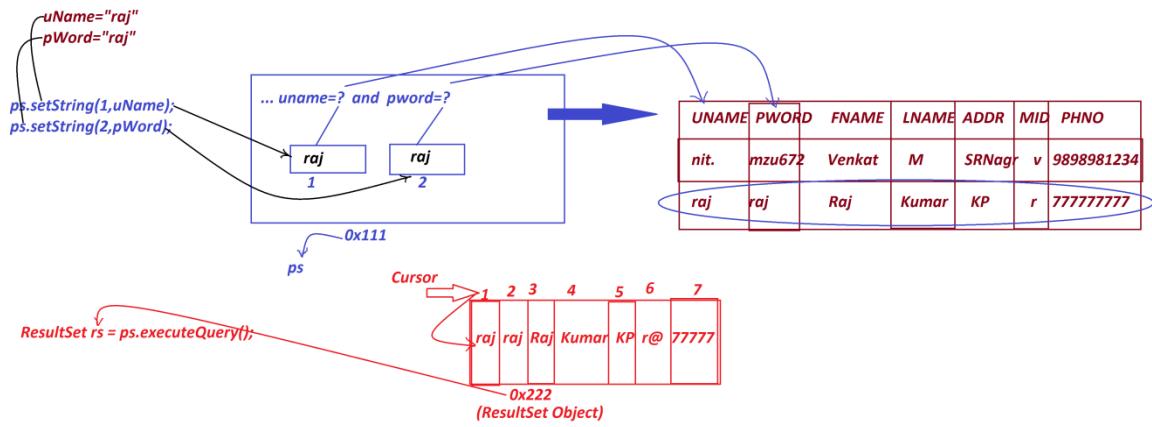
*Address:KP*

*MailId:r@gmail.com*

*PhoneNo:7777777777*

---

*Diagram:*



#### **Assignment-1:**

**JDBC Application to display Student details based on rollNo.**

#### **Assignment-2:**

**JDBC Application to display Product details based on pcode.**

#### **Assignment-3:**

**JDBC Application to display BookDetails based on bCode.**

#### **Assignment-4:**

**JDBC Application to display Employee details based eId.**

#### **Ex\_Application:**

**JDBC Application to update Product price based on Pcode.**

*update Product41 set pprice=? where pcode=?*

*program : DBCon6.java*

*package test;*

*import java.util.\*;*

*import java.sql.\*;*

*public class DBCon6 {*

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

*try {*

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

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

*String pCode = s.nextLine();*

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

*//Loading Driver*

*Connection con = DriverManager.getConnection*

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

*//Connection DataBase*

*PreparedStatement ps = con.prepareStatement*

*("select \* from Product41 where pcode=?");*

*ps.setString(1,pCode);*

*ResultSet rs = ps.executeQuery();*

*if(rs.next()) {*

```
System.out.println("Old price of "+rs.getString(2)
+" is "+rs.getFloat(3));

System.out.println("Enter the New Price:");

float price = s.nextFloat();

PreparedStatement ps2 = con.prepareStatement
("update Product41 set pprice=? where pcode=?");

ps2.setFloat(1,price);

ps2.setString(2,pCode);

int k = ps2.executeUpdate();

if(k>0) {

    System.out.println("Product price Updated... ");

}

else {

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

}

con.close();

s.close();

}catch(Exception e) {

    e.printStackTrace();

}

}

}

o/p:
```

**Enter the ProdCode:**

**A122**

***Old price of Mouse is 235.67***

***Enter the New Price:***

***456.67***

***Product price Updated...***

***Assignment:***

***Construct JDBC Application to update bSal of an Employee.***

Dt : 10/1/2022

**Assignment:**

**Construct JDBC Application to update bSal of an Employee based on Id.**

**Program : DBCon7.java**

```
package test;

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

public class DBCon7 {

    public static void main(String[] args) {
        try {
            Scanner s = new Scanner(System.in);
            System.out.println("Enter the Employee Code:");
            String emplId = s.nextLine();

            Class.forName("oracle.jdbc.driver.OracleDriver");
            //Loading Driver
            Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
            //Connection DataBase

            PreparedStatement ps1 = con.prepareStatement
                ("select * from Employee41 where eid=?");
            ps1.setString(1,emplId);
```

```
ResultSet rs = ps1.executeQuery();  
if(rs.next()) {  
  
    System.out.println("Old bSal:"+rs.getInt(4));  
  
    System.out.println("Old TotSal:"+rs.getFloat(5));  
  
    System.out.print("Enter the new BasicSal:");  
  
    int bSal = s.nextInt();  
  
    float totSal = bSal+(0.93F*bSal)+(0.63F*bSal);  
  
    PreparedStatement ps2 = con.prepareStatement  
        ("update Employee41 set bsal=? , totsal=? where eid=?");  
  
    ps2.setInt(1,bSal);  
  
    ps2.setFloat(2,totSal);  
  
    ps2.setString(3, empId);  
  
    int k = ps2.executeUpdate();  
  
    if(k>0) {  
  
        System.out.println("Employee data Updated..");  
  
    }  
  
}else {  
  
    System.out.println("Invalid Employee Code... ");  
  
}  
  
con.close();  
  
s.close();  
  
}catch(Exception e) {  
  
    e.printStackTrace();  
  
}
```

```
    }  
}
```

*o/p:*

*Enter the Employee Code:*

*A103*

*Old bSal:13000*

*Old TotSal:33280.0*

*Enter the new BasicSal:18000*

*Employee data Updated..*

---

*Assignment:(Solution)*

*step-1 : Create DB Table with name Student41*

*(rollNo,stuName,branch,totmarks,per,result)*

*primary key(rollNo)*

*step-2 : JDBC application to read Student details from Console and insert into*

*DB table Srtudent41.*

*read : rollNo,name,branch and 6 sub marks*

*Calculate totMarks,per and result*

*create table Student41(rollno varchar2(15),stuname varchar2(15),*

*branch varchar2(10),totmarks number(10),per number(10,2),*

```
result varchar2(15),primary key(rollno));
```

**Program :**

**CheckBranch.java**

```
package test;
public class CheckBranch {
    public boolean k=false;
    public boolean verify(String br) {
        switch(br) {
        case "CSE":k=true;
        break;
        case "ECE":k=true;
        break;
        case "EEE":k=true;
        break;
        }//end of switch
        return k;
    }
}
```

**ValidateRollNo.java**

```
package test;
public class ValidateRollNo {
    public boolean z=false;
    public String branch=null;
    public boolean verify(String code,String br) {
        switch(code)
        {
        case "05":branch="CSE";
        break;
        case "04":branch="ECE";
        break;
        case "02":branch="EEE";
        break;
        }//end of switch
        if(branch!=null)//Simple if
        {
            if(branch.equals(br))//simple if
            {
                z=true;
            }
        }
    }
}
```

```
        } //end of if
    } //end of if
    return z;
}
}
```

### Calculate.java

```
package test;
public class Calculate {
    public float per;
    public String result;
    public void cal(int totMarks,boolean p)
    {
        per = (float)totMarks/6; //TypeCasting
        if(p)
        {
            result="Fail";
        }
        else if(per>=70 && per<=100)
        {
            result="Distinction";
        }
        else if(per>=60 && per<70)
        {
            result="FirstClass";
        }
        else if(per>=50 && per<60)
        {
            result="SecondClass";
        }
        else if(per>=35 && per<50)
        {
            result="ThirdClass";
        }
        else
        {
            result="Fail";
        }
    }
}
```

### DBCon8.java

```
package test;
```

```
import java.util.*;
import java.sql.*;
public class DBCon8 {
    public static void main(String[] args) {
        try {
            Scanner s = new Scanner(System.in);
            System.out.println("Enter the Student rollNo:");
            String rollNo = s.nextLine();
            if(rollNo.length()==10) {
                System.out.println("Enter the Student Name:");
                String stuName = s.nextLine();
                System.out.println("Enter the branch:");
                String br = s.nextLine().toUpperCase();
                CheckBranch cb = new CheckBranch();
                boolean k = cb.verify(br);
                if(k)
                {
                    ValidateRollNo vrn = new ValidateRollNo();
                    boolean z = vrn.verify(rollNo.substring(6,8),br);
                    if(z)
                    {
                        System.out.println("==Enter 6 Sub Marks==");
                        int i=1,totMarks=0;
                        boolean p = false;
```

```
while(i<=6)
{
    System.out.println("Enter the Marks of sub"+i);
    int sub = s.nextInt();
    i++;
    if(sub<0 || sub>100)
    {
        System.out.println("Invalid Marks...");
        i--;
        continue;//skip the below lines from loop
    }
    if(sub>=0 && sub<=34)
    {
        p=true;
    }
    totMarks=totMarks+sub;
} //end of loop
Calculate ob = new Calculate();
ob.cal(totMarks, p);
Class.forName("oracle.jdbc.driver.OracleDriver");
//Loading Driver
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
//Connection DataBase
```

```
PreparedStatement ps = con.prepareStatement  
("insert into Student41 values(?,?,?,?,?,?)");  
  
ps.setString(1,rollNo);  
  
ps.setString(2,stuName);  
  
ps.setString(3,br);  
  
ps.setInt(4,totMarks);  
  
ps.setFloat(5,ob.per);  
  
ps.setString(6,ob.result);  
  
int g = ps.executeUpdate();  
  
if(g>0) {  
  
    System.out.println  
("Student details inserted Successfully.. ");  
  
}  
  
}//end of if  
  
else {  
  
    System.out.println  
("RollNo not matched with branch.. ");  
  
}  
  
}//end of if  
  
else {  
  
    System.out.println("Invalid Branch... ");  
  
}  
  
//end of if  
  
else {
```

```
        System.out.println("Invalid RollNo...");  
    }  
  
    s.close();  
}  
}  
}  
}  
}
```

*o/p:*

**Enter the Student rollNo:**

**1234560490**

**Enter the Student Name:**

**Alex**

**Enter the branch:**

**ECE**

**--Enter 6 Sub Marks--**

**Enter the Marks of sub1**

**98**

**Enter the Marks of sub2**

**99**

**Enter the Marks of sub3**

**97**

**Enter the Marks of sub4**

**96**

*Enter the Marks of sub5*

**99**

*Enter the Marks of sub6*

**9**

*Student details inserted Successfully..*

---

Venkatesh Maiopathiji

Dt : 18/1/2022

Ex\_Program :

*JDBC Application to delete Product details based on PCode.*

```
package test;

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

public class DBCon9 {

    public static void main(String[] args) {

        try {

            Scanner s = new Scanner(System.in);

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

            String pCode = s.nextLine();

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

            //Loading Driver

            Connection con = DriverManager.getConnection

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

            //Connection DataBase

            PreparedStatement ps1 = con.prepareStatement

                ("select * from Product41 where pcode=?");//Compilation

            ps1.setString(1,pCode);

            ResultSet rs = ps1.executeQuery();//Execution

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

```
PreparedStatement ps2 = con.prepareStatement  
("delete from Product41 where pcode=?");//Compilation  
ps2.setString(1,pCode);  
int k = ps2.executeUpdate();//Execution  
if(k>0) {  
    System.out.println("Product details deleted Successfully");  
}  
else {  
    System.out.println("Invalid ProductCode... ");  
}  
con.close();  
s.close();  
}catch(Exception e) {  
    e.printStackTrace();  
}  
}  
}
```

*o/p:*

*Enter the ProductCode:*

**A555**

*Product details deleted Successfully*

=====

*\*imp*

**3.CallableStatement:**

=>'CallableStatement' is an interface from java.sql package  
and which is used to execute 'Procedures and Functions' on  
DataBase product.

**define Procedure?**

=>procedure is a set-of-queries executed on DataBase product  
and after execution it will not return any value.

**structure of procedure:**

*create or replace procedure Proc\_name*

*(para\_list) is*

*begin*

*query1;*

*query2;*

*...*

*end;*

*/*

**define Function?**

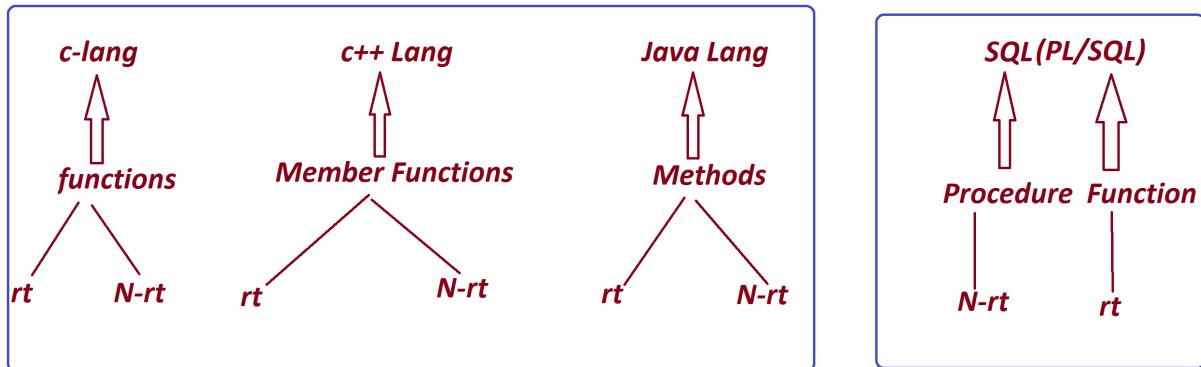
=>Function is a set-of-queries executed on DataBase product and  
after execution it will return the value.

**Note:**

=>Functions in SQL will use 'return' statement to return the value after execution.

**structure of Function:**

```
create or replace function Func_name  
  (para_list) return data_type as var data_type;  
  
begin  
  
query1;  
  
query2;  
  
....  
  
return var;  
  
end;  
/  
=====
```



Dt : 19/1/2022

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

**Method Signature of `prepareCall()`:**

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

**syntax:**

```
CallableStatement cs = con.prepareCall("Procedure/Function");
```

---

\*imp

**Constructing and executing Procedures:**

**step-1 : Create the following DB Tables**

`Bank41(accno,custname,balance,acctype)`

`CustDetails41(accno,address,mid,phno)`

```
create table Bank41(accno number(15),custname varchar2(15),balance number(10,2),  
acctype varchar2(15),primary key(accno));
```

```
create table CustDetails41(accno number(15),address varchar2(15),mid varchar2(25),  
phno number(15),primary key(accno));
```

**step-2 : Construct Procedure to insert data to DB Tables**

```
create or replace procedure CreateAccount41  
(accno number,custname varchar2,bal number,acctype varchar2,address varchar2,  
mid varchar2,phno number) is  
begin  
insert into Bank41 values(accno,custname,bal,acctype);  
insert into CustDetails41 values(accno,address,mid,phno);  
end;  
/
```

**step-3 : Construct JDBC Application to execute Procedure**

**syntax:**

```
CallableStatement cs = con.prepareCall  
("{call CreateAccount41(?,?,?,?,?,?)}");
```

**Program : DBCon10.java**

```
package test;  
import java.sql.*;  
import java.util.*;  
public class DBCon10 {  
    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 MailId:");
String mId = s.nextLine();
System.out.println("Enter the PhoneNo:");
long phNo = Long.parseLong(s.nextLine());

Class.forName("oracle.jdbc.driver.OracleDriver");
//Loading Driver
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe", "system", "manager");
//Connection DataBase

CallableStatement cs = con.prepareCall
("{call CreateAccount41(?, ?, ?, ?, ?, ?, ?, ?)}");

cs.setLong(1, accNo);
cs.setString(2, custName);
cs.setFloat(3, bal);
cs.setString(4, accType);
cs.setString(5, addr);
cs.setString(6, mId);
cs.setLong(7, phNo);

cs.execute(); //Execute procedure with IN-Parameter
values
System.out.println("Procedure Executed
Successfully... ");
con.close();
s.close();
}catch(Exception e) {
e.printStackTrace();
}
}
```

*o/p:*

*Enter the accNO:*

6123456

*Enter the CustName:*

*Raj*

### ***Enter the Balance:***

**12000**

**Enter the AccType:**

**Savings**

**Enter the Address:**

**SRN**

**Enter the MailId:**

**r@gmail.com**

**Enter the PhoneNo:**

**9898981234**

**Procedure Executed Successfully...**

**DB Tables Updated:**

**SQL> select \* from Bank41;**

| <b>ACCNO</b>   | <b>CUSTNAME</b> | <b>BALANCE</b> | <b>ACCTYPE</b> |
|----------------|-----------------|----------------|----------------|
| -----          | -----           | -----          | -----          |
| <b>6123456</b> | <b>Raj</b>      | <b>12000</b>   | <b>Savings</b> |

**SQL> select \* from CustDetails41;**

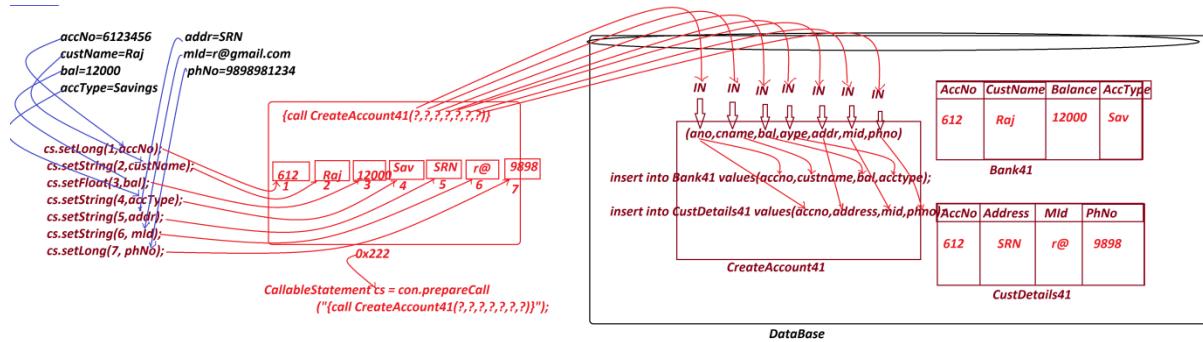
| <b>ACCNO</b> | <b>ADDRESS</b> | <b>MID</b> | <b>PHNO</b> |
|--------------|----------------|------------|-------------|
| -----        | -----          | -----      | -----       |

6123456 SRN

r@gmail.com

9898981234

**Diagram:**



**Assignment:**

**step-1 : Create the following DB Tables**

***EmpData41(eid,ename,edesg)***

***EmpAddress41(eid,hno,sname,city,pincode)***

***EmpContact41(eid,mid,phno)***

***EmpSalary41(eid,bsal,totsal)***

**step-2 : Construct Procedure to insert details into DB Tables**

**step-3 : Construct JDBC Application to execute Procedure.**

Dt : 20/1/2022

**Types of Procedures:**

=>Procedures are categorized into the following:

**1.IN-Parameter Procedures**

**2.Out-Parameter Procedures**

**1.IN-Parameter Procedures:**

=>*The procedures which take the date from Java Programs and load to Db\_product are known as In-Parameter Procedures.*

*Ex:*

*above program*

**2.Out-Parameter Procedures:**

=>*The procedures which take the data from DB\_product and load to Java Programs are known as Out-Parameter Procedures.*

*Ex\_Program : Demontrating OUT-Parameter procedures.*

*step-1 : Construct Procedure to display customer data based on AccNo*

*create or replace procedure RetrieveDetails41*

*(ano number,cname OUT varchar2,bal OUT number,atype OUT varchar2,addr OUT varchar2, mid OUT varchar2,phno OUT number) is*

*begin*

```

select custname,balance,acctype into cname,bal,atype from Bank41 where accno=ano;

select address,mid,phno into addr,mid,phno from CustDetails41 where accno=ano;

end;

/

```

**step-2 : Construct JDBC Application to execute Procedure**

**Program : DBCon11.java**

```

package test;
import java.sql.*;
import java.util.*;
public class DBCon11 {
    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");
            //Loading Driver
            Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe", "system", "manager");
            //Connection DataBase

            CallableStatement cs = con.prepareCall
                ("{call RetrieveDetails41(?,?,?,?,?,?) }");

            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.VARCHAR);
            cs.registerOutParameter(7, 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("MailId:" + cs.getString(6));
        System.out.println("PhoneNo:" + cs.getLong(7));

        con.close();
        s.close();
    }catch(Exception e) {
        System.out.println(e.getMessage());
    }
}
}

```

*o/p:*

*Enter the accNo:*

**6123456**

**AccNo:6123456**

**CustName:Raj**

**Balance:12000.0**

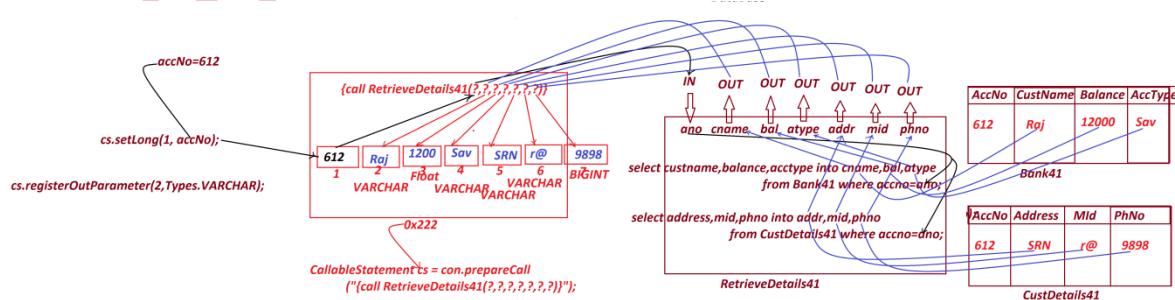
**AccType:Savings**

**Address:SRN**

**MailId:r@gmail.com**

**PhoneNo:9898981234**

*Diagram:*



---

**Assignment:**

**Construct Procedure to display the complete details of employee based on eId.**

---

**faq:**

**define registerOutParameter() method?**

=>*registerOutParameter()* method specify the type of data to be recorded in the field-storage of CallableStatement object.

=>*This registerOutParameter()* method is available from 'CallableStatement' interface.

**Method Signature:**

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

**Ex:**

```
cs.registerOutParameter(2,Types.VARCHAR);
```

---

**faq:**

**define 'Types' in JDBC?**

=>'Types' is a class from `java.sql` package and which is holdind SQL-Types like  
`VARCHAR`

`INTEGER`

`BIGINT`

....

*syntax:*

*Types.VARCHAR*

*Types.INTEGER*

---

Venkatesh Maiopathiji

Dt : 21/1/2022

Ex\_Application : Demonstrating Function.

step-1 : Construct Function to retrieve Balance of a Customer based accNo.

create or replace function RetrieveBalance41

(ano number) return number as bal number;

begin

select balance into bal from Bank41 where accno=ano;

return bal;

end;

/

step-2 : Construct JDBC Application to execute function

syntax:

```
CallableStatement cs = con.prepareCall("{call ?:=RetrieveBalance41(?)}");
```

program : DBCon12.java

```
package test;
```

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

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

```
public class DBCon12 {
```

```
    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");  
  
//Loading Driver  
  
Connection con = DriverManager.getConnection  
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");  
  
//Connection DataBase  
  
CallableStatement cs = con.prepareCall  
("{call ?:=RetrieveBalance41(?)}");  
  
cs.registerOutParameter(1, Types.FLOAT);  
  
cs.setLong(2,accNo);  
  
cs.execute();  
  
System.out.println("AccNo:"+accNo);  
  
System.out.println("Balance:"+cs.getFloat(1));  
  
con.close();  
  
s.close();  
}  
catch(Exception e) {  
    System.out.println(e.getMessage());  
}  
}  
}
```

o/p:

Enter the AccNo:

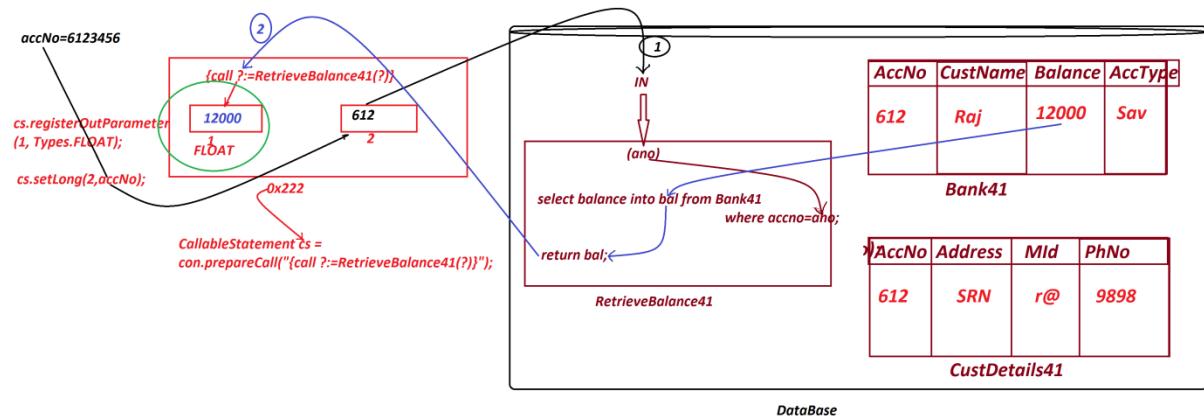
6123456

AccNo:6123456

Balance:12000.0

---

Diagram:



---

Assignment:

Construct Function to retrieve totalSalary of an employee based on eid.

---

faq:

wt is the advantage of Procedures?

=>Procedures are used to execute multiple queries at-a-time on DB-Product

and saves the execution time, and which generate HighPerformance of an application.

---

\*imp

Transaction Management in JDBC:

define Transaction?

=>Transaction is a set-of-statements executed on a resource or resources using ACID properties.

A - Atomicity

C - Consistency

I - Isolation

D - Durability

A - Atomicity

=>The process of completing all the Sub-Transactions Successfully and performing Commit operation is known as Atomicity.

SQL> select \* from Bank41;

| ACCNO   | CUSTNAME | BALANCE | ACCTYPE |
|---------|----------|---------|---------|
| 6123456 | Raj      | 12000   | Savings |
| 313131  | Ram      | 500     | Savings |

SQL>

C - Consistency

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

## I - Isolation

=>In the process of providing Consistency we have to execute users independently, is known Isolation.

## D - Durability

=>when transaction is committed then it is stored and available permanently is known as Durability.

| ACCNO   | CUSTNAME | BALANCE | ACCTYPE |
|---------|----------|---------|---------|
| 6123456 | Raj      | 12000   | Savings |
| 313131  | Ram      | 500     | Savings |

*Bank41*

*Transaction : Transfer amt:3000/- from AccNo:6123456 to AccNo:313131*

*Sub-T1 - Subtract 3000/- from AccNo:6123456*

*Sub-T2- Add 3000/- to AccNo:313131*

*Transaction Successfull means*

*- Sub-T1 is Successfull and Sub-T2 is Successfull.*

Dt : 22/1/2022

define Transaction Management?

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

=>we use the following methods from 'java.sql.Connection' interface in

Transaction Management Process:

(1)setAutoCommit()

(2)getAutoCommit()

(3)commit()

(4)rollback()

(5)setSavepoint()

(6)releaseSavepoint()

(1)setAutoCommit():

=>This method is used to set autoCommit operation to 'false',because JavaApplications will perform commit operation automatically.

Method Signature:

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

(2)getAutoCommit():

=>This method is used to know the status of commit operation.(true/false)

Method Signature:

```
public abstract boolean getAutoCommit() throws java.sql.SQLException;
```

(3)commit():

=>we use commit() method to update data from buffer to DataBase,when transaction is Successfull.

Method Signature:

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

(4) rollback():

=>we use rollback() method to get original state of buffer when the transaction failed and we specify the rollback operation using savepoint.

Method Signature:

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

(5)setSavepoint():

=>This method is used to set savepoint for rollback operation.

Method Signature:

```
public abstract java.sql.Savepoint setSavepoint()  
throws java.sql.SQLException;
```

(6)releaseSavepoint():

=>This method is used to delete save point from the buffer.

Method Signature:

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

---

Ex\_Program:

Program : DBCon13.java

```
package test;  
  
import java.util.*;  
  
import java.sql.*;  
  
public class DBCon13 {
```

```
public static void main(String[] args) {  
    try {  
        Scanner s = new Scanner(System.in);  
        Class.forName("oracle.jdbc.driver.OracleDriver");  
        //Loading Driver  
        Connection con = DriverManager.getConnection  
        ("jdbc:oracle:thin:@localhost:1521:xe","system","manager");  
        //Connection DataBase  
        PreparedStatement ps1 = con.prepareStatement  
        ("select * from Bank41 where accno=?");  
        PreparedStatement ps2 = con.prepareStatement  
        ("update Bank41 set balance=balance+? where accno=?");  
        System.out.println("Commit Status : "+con.getAutoCommit());  
        con.setAutoCommit(false);//stopped Auto Commit Operation  
        System.out.println("Commit Status : "+con.getAutoCommit());  
        Savepoint sp = con.setSavepoint();  
        System.out.println("Enter the homeAccNo:");  
        long hAccNo = s.nextLong();  
        ps1.setLong(1,hAccNo);  
        ResultSet rs1 = ps1.executeQuery();  
        if(rs1.next()) {  
            float bal = rs1.getFloat(3);  
            System.out.println("Enter the beneficiaryAccNo:");  
            long bAccNo = s.nextLong();  
            ps1.setLong(1, bAccNo);  
        }  
    }  
}
```

```
ResultSet rs2 = ps1.executeQuery();

if(rs2.next())
{
    System.out.println("Enter the Amt to be Transferred:");

    int amt = s.nextInt();

    if(amt<=bal)
    {
        ps2.setFloat(1,-amt);

        ps2.setLong(2,hAccNo);

        int i=ps2.executeUpdate();

        ps2.setFloat(1,amt);

        ps2.setLong(2,bAccNo);

        int j=ps2.executeUpdate();

        if(i==1 && j==1)
        {
            con.commit();

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

        }else {
            System.out.println("Transaction Failed...");

            con.rollback(sp);
        }
    }else {
        System.out.println("Insufficient fund...");
    }
}
```

```
        }

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

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

s.close();

}catch(Exception e) {
    System.out.println(e.getMessage());
}

}

}

}

o/p:
```

Commit Status : true

Commit Status : false

Enter the homeAccNo:

6123456

Enter the beneficiaryAccNo:

313131

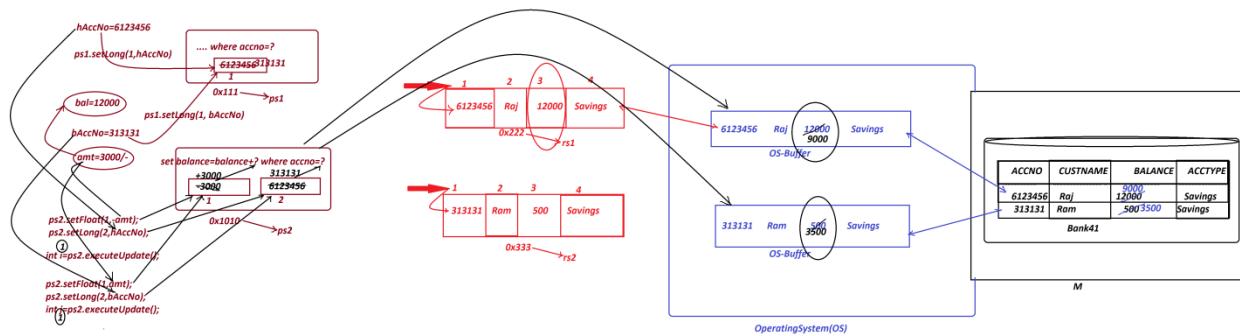
Enter the Amt to be Transferred:

3000

Transaction Successfull...

---

Diagram:



Dt : 24/1/2022

\*imp

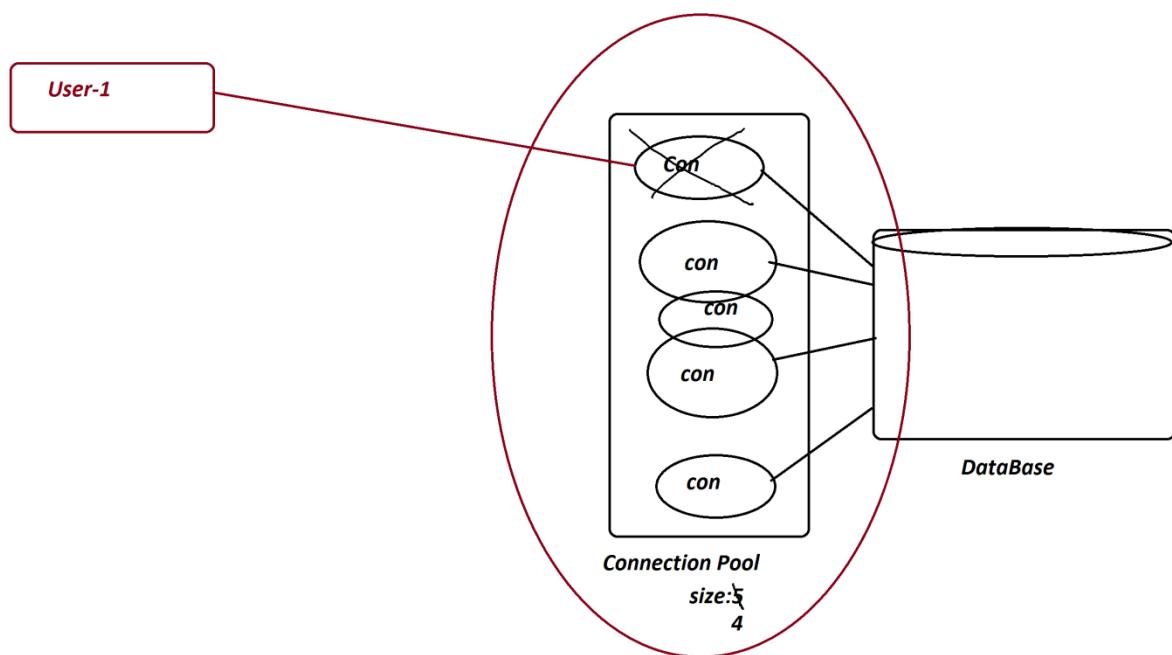
### **Connection Pooling in JDBC:**

=>The process of organizing Pre-Initialized database connections among users is known as 'Connection Pooling'.

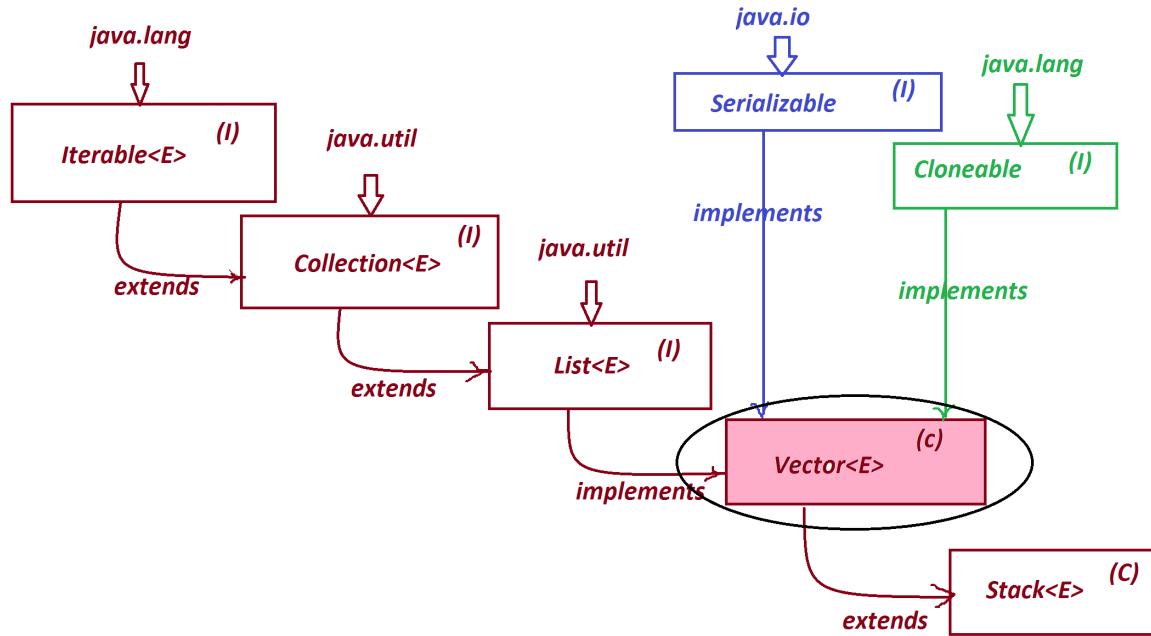
=>In Connection Pooling process, the users use the connection from the pool and after usage the connection is returned back to the pool.

=>The memory location where we organize multipel DB-Connections is known as 'Connection pool'.

=>To create Connection pool memory in JDBC we use `java.util.Vector<E>` class.



### Hierarchy of Vector<E>:



=>'Serializable' interface will support `Vector<E>` object available in the form of

binary stream, so that we can move the `Vector<E>` object on network.

=>'Cloneable' interface will support cloning process, which means we can duplicate the object.

Ex\_program:

Pooling.java

```

package test;
import java.util.*;
import java.sql.*;
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;
    }
    Vector<Connection> v = new Vector<Connection>(); //Connection
    _Pool
    public void createConnection() {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //Loading Driver
            while(v.size()<5) {
                System.out.println("Pool is Not Full... ");
                Connection con =
            DriverManager.getConnection(url,uName,pWord);
                //Creating Connection
                System.out.println(con);
                v.add(con); //Adding Connection to Pool
            } //end of loop
            if(v.size()==5) {
                System.out.println("Pool is Full... ");
            }
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
    public synchronized Connection useConnection() {
        Connection con = v.elementAt(0);
        v.remove(0); //Delete Connection from pool
        return con;
    }
    public synchronized void returnConnection(Connection con) {
        v.add(con);
        System.out.println("Connection added back to the pool... ");
    }
}

```

#### DBCon14.java(MainClass)

```

package test;
import java.sql.*;
public class DBCon14 {
    public static void main(String[] args) {
        try {
            Pooling p = new
Pooling("jdbc:oracle:thin:@localhost:1521:xe",
                    "system","manager"); //Con_Call
            p.createConnection();

```

```

        System.out.println("Pool Size : "+p.v.size());
        Connection con = p.useConnection(); //Using the
Connection
        System.out.println("Pool Size : "+p.v.size());
        System.out.println("Display using : "+con);
        PreparedStatement ps = con.prepareStatement
            ("select * from Product41");
        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 loop
        p.returnConnection(con); //Adding Connection to Pool
        System.out.println("Pool Size : "+p.v.size());
    }catch(Exception e) {
        System.out.println(e.getMessage());
    }
}
}
}

```

*o/p:*

*Pool is Not Full...*

*oracle.jdbc.driver.T4CConnection@31a5c39e*

*Pool is Not Full...*

*oracle.jdbc.driver.T4CConnection@1dd92fe2*

*Pool is Not Full...*

*oracle.jdbc.driver.T4CConnection@6b53e23f*

*Pool is Not Full...*

*oracle.jdbc.driver.T4CConnection@4f9a3314*

*Pool is Not Full...*

*oracle.jdbc.driver.T4CConnection@3b2c72c2*

*Pool is Full...*

*Pool Size : 5*

*Pool Size : 4*

*Display using : oracle.jdbc.driver.T4CConnection@31a5c39e*

**A122 Mouse 456.67 12**

**A121 KB 135.67 10**

**A120 FDD 234.67 9**

**A111 CDR 454.67 12**

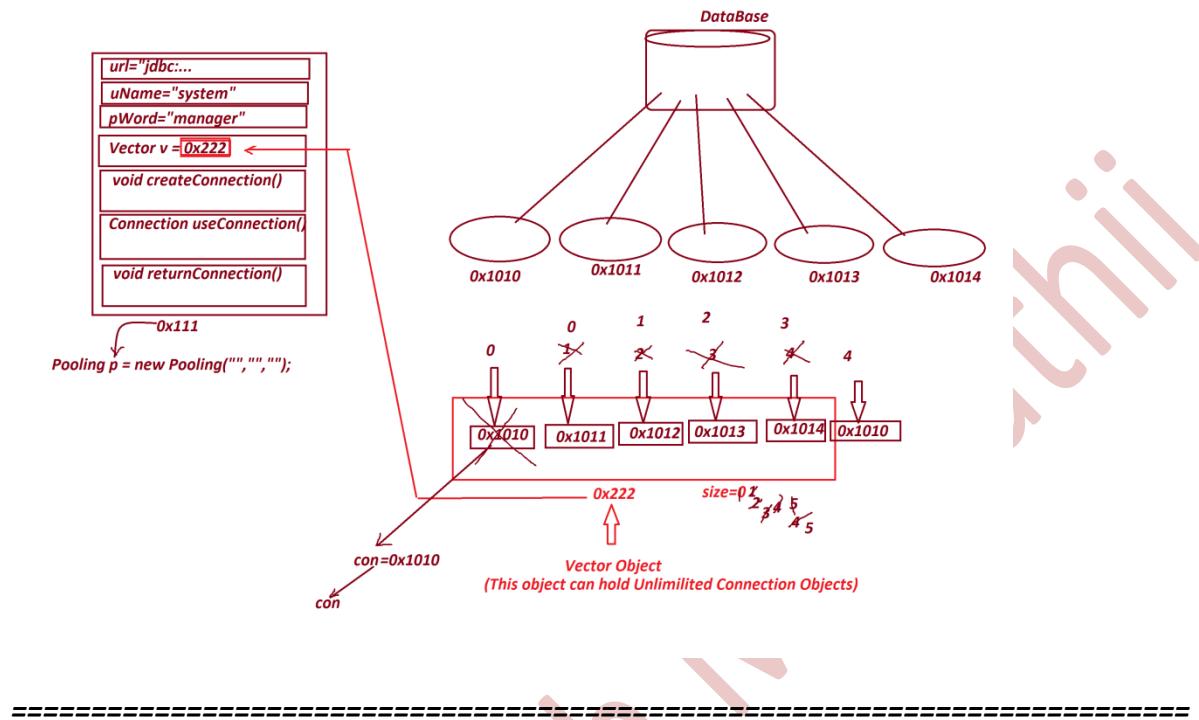
*Connection added back to the pool...*

*Pool Size : 5*

---

Dt : 25/1/2022

**Diagram:**



\*imp

**Batch Processing in JDBC:**

=>The process of collecting multiple queries as a batch and executing on DataBase product is known as 'Batch Processing in JDBC'.

**Case-1 : Batch Processing using 'Statement'**

=>we use the following methods from 'java.sql.Statement' to perform 'Batch Processing':

(a) `addBatch()`

(b) `executeBatch()`

(c) `clearBatch()`

**(a)addBatch():**

=>This method is used to add the query to the batch.

**Method Signature:**

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

**syntax:**

**stm.addBatch("query");**

**(b)executeBatch():**

=>This method is used to execute batch on DataBase Product.

**Method Signature:**

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

**syntax:**

**int k[] = stm.executeBatch();**

**(c)clearBatch():**

=>This method is used to delete(clear) the batch.

**Method Signature:**

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

**syntax:**

**stm.clearBatch();**

*Ex program:*

**DBCon15.java**

```
package test;
import java.sql.*;
public class DBCon15 {
    public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //Loading Driver
            Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe", "system", "manager");
            //Connection DataBase
            Statement stm = con.createStatement();
            con.setAutoCommit(false);
            stm.addBatch("insert into Product41
values('A105', 'XYZ', 435.55, 7)");
            stm.addBatch
                ("insert into Employee41
values('A109', 'Alex', 'SE', 13000, 45000)");
            int k[] = stm.executeBatch();
            for(int i=0;i<k.length;i++) {
                System.out.println("Record Inserted
Successfully... ");
            }
            con.commit();
            stm.clearBatch();
        }catch(Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

---

**Case-2 : Batch Processing using 'PreparedStatement'**

=>we use the following methods to perform Batch processing using  
**PreparedStatement.**

*(a)addBatch() - PreparedStatement*

*(b)executeBatch() - Statement*

*(c)clearBatch() - Statement*

**Note:**

=>Method Signature of addBatch():

*public abstract void addBatch() throws java.sql.SQLException;*

**syntax:**

*ps.addBatch();*

**Ex program:**

**DBCon16.java**

```
package test;
import java.sql.*;
public class DBCon16 {
    public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //Loading Driver
            Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
            //Connection DataBase
            PreparedStatement ps = con.prepareStatement
                ("insert into Product41
values(?, ?, ?, ?)");//Compilation
            con.setAutoCommit(false);
            ps.setString(1, "A106");
            ps.setString(2, "PQR");
            ps.setFloat(3, 345.67F);
            ps.setInt(4, 10);
            ps.addBatch();

            ps.setString(1, "A107");
```

```

        ps.setString(2, "ABC");
        ps.setFloat(3, 745.67F);
        ps.setInt(4, 17);
        ps.addBatch();

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

        con.commit();
        ps.clearBatch();
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
}

```

---

**Note:**

=>'Batch Processing using Statement' we can update multiple DB tables.

=>'Batch Processing using PreparedStatement' we can update same DB Table.

**faq:**

**What is the advantage of Batch processing?**

=>when we use Batch Processing the execution control is transferred to DB product only once and executes all the queries from the batch,in this process execution time is saved and generate HighPerformance of an application.

**faq:**

*wt is the diff b/w*

*(i)Procedure*

*(ii)Batch Processing*

*=>Using procedure we can perform all operations on DB,which means we can perform create,Insert,Select,update and delete.*

*=>Using Batch processing we cannot perform 'select'(retrieve) operations, but we can perform create,insert,update and delete.*

*(Batch processing is also known as Batch Update processing)*

---

Dt: 31/1/2022

**Types of ResultSet objects:**

=>Based on the cursor(header) movement, the ResultSet objects are categorized into two types:

(i) NonScrollable ResultSet object

(ii) Scrollable ResultSet object

(i) NonScrollable ResultSet object:

=>In NonScrollable ResultSet objects the cursor(header) is moved only in one direction from top-of-table-data to Bottom-of-table-data.

(ii) Scrollable ResultSet object:

=>In Scrollable ResultSet objects the cursor(header) is moved in both directions (forward and backward)

---

=>The following are the syntaxes used to generate Scrollable ResultSet object:

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

*PreparedStatement ps = con.prepareStatement("query-structure", type, mode);*

**Type:**

```
public static final int TYPE_FORWARD_ONLY=1003  
public static final int TYPE_SCROLL_INSENSITIVE=1004  
public static final int TYPE_SCROLL_SENSITIVE=1005
```

**Mode:**

```
public static final int CONCUR_READ_ONLY=1007  
public static final int CONCUR_UPDATABLE=1008
```

**Note:**

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

*The following are some methods used to control cursor on Scrollable*

**ResultSet object:**

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

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

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

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

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

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

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

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

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

---

**DB\_Table : Product34**

*SQL> select \* from Product34;*

| <b>PCODE</b> | <b>PNAME</b> | <b>PPRICE</b> | <b>PQTY</b> |
|--------------|--------------|---------------|-------------|
| A121         | Mouse        | 1200          | 12          |
| A122         | KB           | 1300          | 13          |
| A101         | XYZ          | 1400          | 11          |
| A102         | PQR          | 1500          | 11          |
| A100         | ABC          | 1100          | 11          |

**Ex\_Program:DBCon17.java**

```
package test;
import java.sql.*;
public class DBCon17 {
    public static void main(String[] args) {
```

```

try {
    Class.forName("oracle.jdbc.driver.OracleDriver");
    Connection con = DriverManager.getConnection
        ("jdbc:oracle:thin:@localhost:1521:xe", "system", "manager");

    Statement stm =
    con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                      ResultSet.CONCUR_READ_ONLY);
    ResultSet rs = stm.executeQuery("select * from
Product34");
    System.out.println("====Display Products in
reverse====");
    rs.afterLast(); //Cursor pointing after the last row
    while(rs.previous()) {

System.out.println(rs.getString(1)+"\t"+rs.getString(2)+"\t"+
                  rs.getFloat(3)+"\t"+rs.getInt(4));
    } //end of while
    con.close();
} catch(Exception e) {e.printStackTrace();}
}
}

```

*o/p:*

*====Display Products in reverse====*

|      |       |        |    |
|------|-------|--------|----|
| A100 | ABC   | 1100.0 | 11 |
| A102 | PQR   | 1500.0 | 11 |
| A101 | XYZ   | 1400.0 | 11 |
| A122 | KB    | 1300.0 | 13 |
| A121 | Mouse | 1200.0 | 12 |

=====

*Ex\_Program : DBCon18.java*

```

package test;
import java.sql.*;
public class DBCon18 {

```

```

public static void main(String[] args) {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection
        ("jdbc:oracle:thin:@localhost:1521:xe", "system", "manager");
        PreparedStatement ps = con.prepareStatement
        ("select * from
        Product34",ResultSet.TYPE_SCROLL_INSENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
        ResultSet rs = ps.executeQuery();
        System.out.println("====Display Products Normally====");
        while(rs.next()) {

            System.out.println(rs.getString(1)+"\t"+rs.getString(2)+"\t"+
                               rs.getFloat(3)+"\t"+rs.getInt(4));
        }//end of while
        System.out.println("====Display the absolute(3)====");
        rs.absolute(3);

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

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

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

```

*o/p:*

*====Display Products Normally====*

*A121 Mouse 1200.0 12*

*A122 KB 1300.0 13*

*A101 XYZ 1400.0 11*

*A102 PQR 1500.0 11*

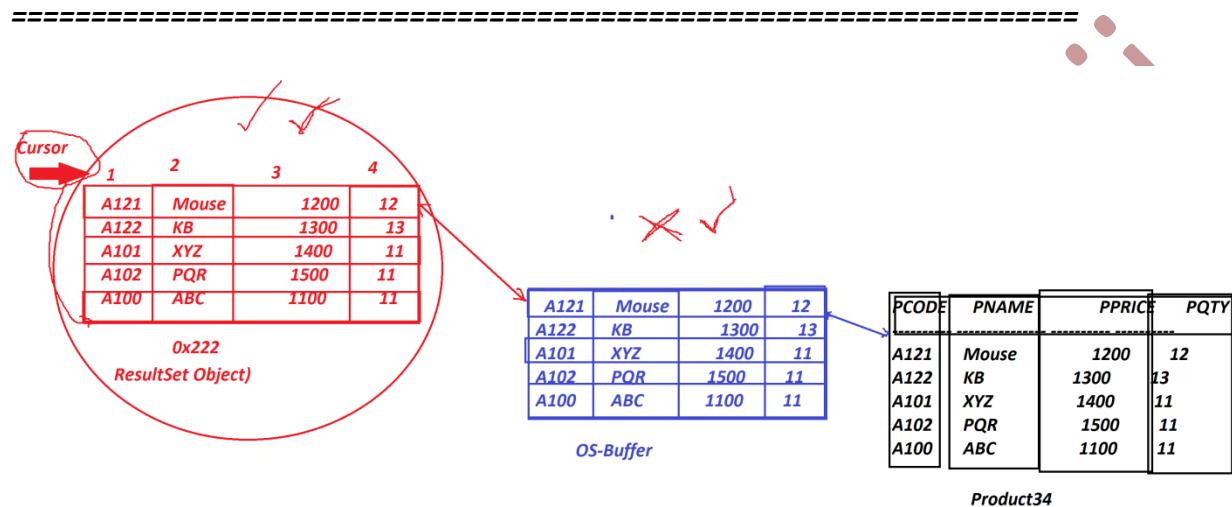
*A100 ABC 1100.0 11*

====Display the absolute(3)=====

A101 XYZ 1400.0 11

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

A102 PQR 1500.0 11

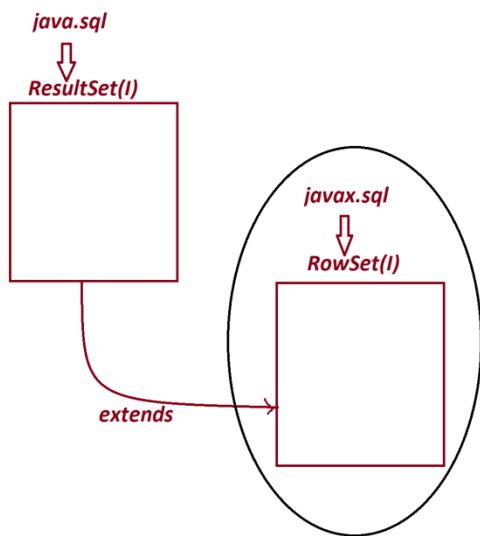


Dt : 1/2/2022

faq:

**define RowSet?**

=>'RowSet' is an interface from `javax.sql` package and which is extended from '`java.sql.ResultSet`' interface.

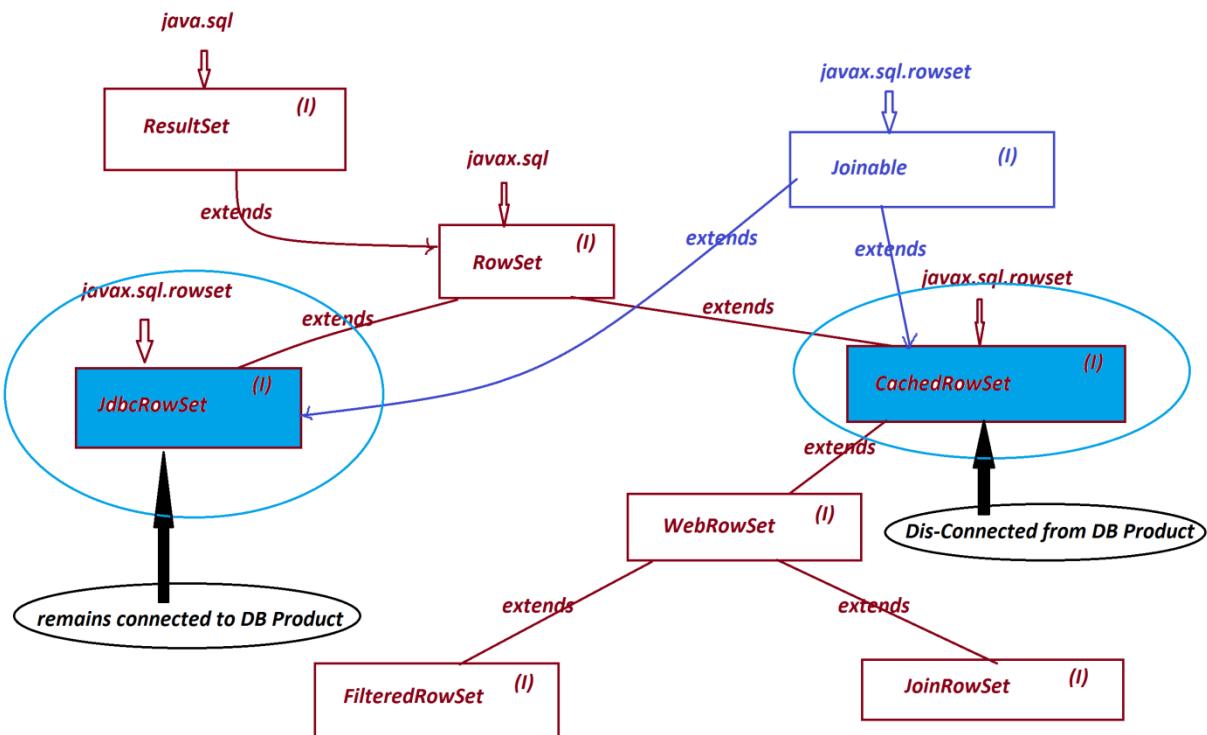


**Types of RowSet:**

=>The following are the five types of RowSet:

1. `JDBCRowSet`
2. `CachedRowSet`
3. `WebRowSet`
4. `FilteredRowSet`
5. `JoinRowSet`

**Hierarchy of RowSet:**



*faq:*

*define RowSetFactory?*

=>*RowSetFactory is an interface from java.sql.rowset package and which provides the following factory methods to create the implementation objects of RowSet.*

*(i)createCachedRowSet():*

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

```
public abstract javax.sql.rowset.CachedRowSet createCachedRowSet()
```

```
throws java.sql.SQLException;
```

*(ii)createFilteredRowSet():*

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

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

*(iii)createJdbcRowSt():*

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

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

*(iv)createJoinRowSet():*

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

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

*(v)createWebRowSet():*

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

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

---

*faq:*

**define RowSetProvider?**

=>*RowSetProvider is a class from javax.sql.rowset package and which provide the following method to create the implementation object of 'RowSetFactory' interface.*

**Method Signature:**

```
public static javax.sql.rowset.RowSetFactory newFactory()  
throws java.sql.SQLException;
```

**syntax:**

```
RowSetFactory rsf = RowSetProvider.newFactory();
```

---

**Ex\_Program : DBCon19.java**

```
package test;  
import javax.sql.rowset.*;  
public class DBCon19 {  
    public static void main(String[] args) {  
        try {  
            Class.forName("oracle.jdbc.driver.OracleDriver");  
            RowSetFactory rsf =  
RowSetProvider.newFactory(); //RowSetFactory Object  
JdbcRowSet jrs = rsf.createJdbcRowSet(); //JdbcRowSet  
Object  
jrs.setUrl("jdbc:oracle:thin:@localhost:1521:xe");  
jrs.setUsername("system");  
jrs.setPassword("manager");  
jrs.setCommand("select * from Product34");  
jrs.execute();  
System.out.println("====Display using JdbcRowSet====");  
while(jrs.next()) {  
  
System.out.println(jrs.getString(1)+"\t"+jrs.getString(2)+"\t"+  
jrs.getFloat(3)+"\t"+jrs.getInt(4));
```

```

} //end of loop
CachedRowSet crs =
rsf.createCachedRowSet(); //CachedRowSet Object
crs.setUrl("jdbc:oracle:thin:@localhost:1521:xe");
crs.setUsername("system");
crs.setPassword("manager");
crs.setCommand("select * from Product34");
crs.execute();
System.out.println("==Display using JdbcRowSet==");
while(crs.next()) {

System.out.println(crs.getString(1)+"\t"+crs.getString(2)+"\t"+
crs.getFloat(3)+"\t"+crs.getInt(4));
}//end of loop
} catch(Exception e){e.printStackTrace();}
}
}

```

*o/p:*

*==Display using JdbcRowSet==*

A121 Mouse 1200.0 12

A122 KB 1300.0 13

A101 XYZ 1400.0 11

A102 PQR 1500.0 11

A100 ABC 1100.0 11

*==Display using CachedRowSet==*

A121 Mouse 1200.0 12

A122 KB 1300.0 13

A101 XYZ 1400.0 11

A102 PQR 1500.0 11

A100 ABC 1100.0 11

=====

**Note:**

=>The RowSet objects are by default Scrollable Objects.

---

Venkatesh Maiopathiji

Dt : 3/2/2022

**Define MetaData?**

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

**Note:**

=>*According to JDBC object holding information about another object is known as MetaData*

=>*The following are the metadata components from JDBC:*

- 1.DatabaseMetaData**
- 2.ParameterMetaData**
- 3.ResultSetMetaData**
- 4.RowSetMetaData**

**1.DatabaseMetaData:**

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

=>*we use getMetaData() method from Connection interface to create the implementation object of DatabaseMetaData.*

**syntax:**

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

**2.ParameterMetaData:**

=>ParameterMetaData is also an interface from java.sql package and which hold the information about PreparedStatement Object.

=>we use getParameterMetaData() method from PreparedStatement interface to create the implementation object of ParameterMetaData.

**syntax:**

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

**3.ResultSetMetaData:**

=>ResultSetMetaData is also an interface from java.sql package and which hold the information about ResultSet object.

=>we use getMetaData() method from ResultSet interface to create the implementation object of ResultSetMetaData interface.

**syntax:**

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

**4.RowSetMetaData:**

=>RowSetMetaData is an interface from javax.sql package and which hold the information about RowSet object.

**syntax:**

```
RowSetMetaData rmd = (RowSetMetaData)jrs.getMetaData();
```

---

*Ex\_Program : DemoMetaData.java*

```
package test;

import java.sql.*;
import javax.sql.RowSetMetaData;
import javax.sql.rowset.*;

public class DemoMetaData {

    public static void main(String[] args) {

        try {

            Class.forName("oracle.jdbc.driver.OracleDriver");
            //Loading Driver

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

            DatabaseMetaData dmd = con.getMetaData();
            System.out.println("DriverName:"+dmd.getDriverName());

            PreparedStatement ps = con.prepareStatement
                ("insert into Product41 values(?,?,?,?,?)");

            ParameterMetaData pmd = ps.getParameterMetaData();
            System.out.println("Number of Parameters:"+pmd.getParameterCount());

            PreparedStatement ps2 = con.prepareStatement
                ("select * from Product41");

            ResultSet rs = ps2.executeQuery();

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

```
System.out.println("Columns:"+rsmd.getColumnCount());  
  
JdbcRowSet jrs =  
  
    RowSetProvider.newFactory().createJdbcRowSet();  
  
jrs.setUrl("jdbc:oracle:thin:@localhost:1521:xe");  
  
jrs.setUsername("system");  
  
jrs.setPassword("manager");  
  
jrs.setCommand("select * from Product41");  
  
jrs.execute();  
  
//RowSetMetaData rmd = (RowSetMetaData)jrs.getMetaData();  
  
    //Compilation_Error  
  
//System.out.println("ColCount:"+rmd.getColumnCount());  
  
}catch(Exception e) {e.printStackTrace();}  
  
}  
  
=====  
=  
  
Summary of Objects:  
  
CoreJava:  
  
1. User defined Class Object  
2. String Objects  
3. WrapperClass Objects  
4. Array Objects  
5. Collection<E> Objects  
6. Map<K,V> Objects
```

**7. *Enum<E> Object***

*JDBC:*

**1. *Connection object***

**2. *Statement Object***

**3. *PreparedStatement Object***

**4. *CallableStatement Object***

**5. *ResultSet Object***

**6. *RowSet Object***

**(a) *JdbcRowSet Object***

**(b) *CachedRowSet Object***

**=>*WebRowSet***

**(i) *FilteredRowSet object***

**(ii) *JoinRowSet Object***

**7. *DatabaseMetaData object***

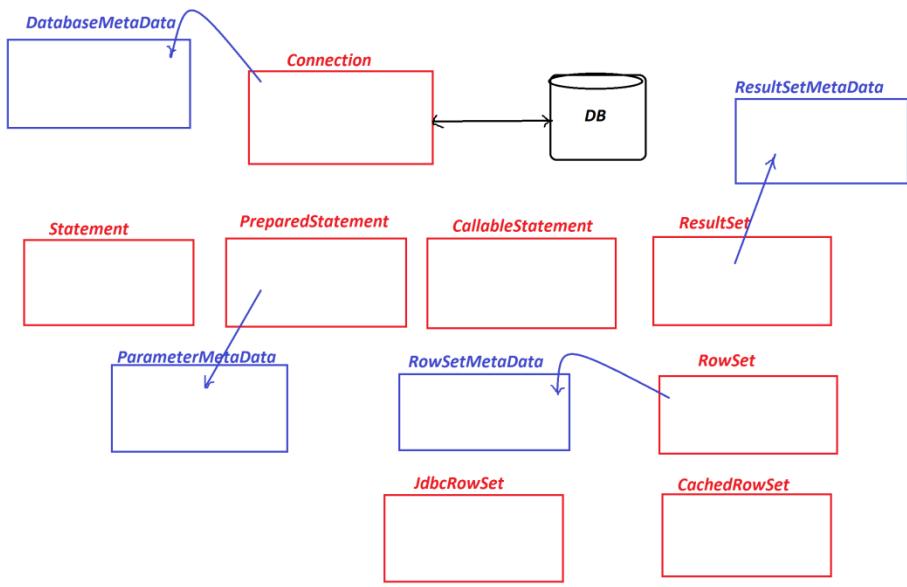
**8. *ParameterMetaData object***

**9. *ResultSetMetaData Object***

**10. *RowSetMetaData Object***

---

*Diagram:*



=====  
Venkatesh Makaraythiji

Dt : 4/2/2022

**JDBC drivers are categorized into four types:**

**(a) Type-1 : JDBC-ODBC Bridge Driver**

**(b) Type-2 : Native API Driver**

**(c) Type-3 : Network Protocol Driver**

**(d) Type-4 : Thin Driver**

**(a) Type-1 : JDBC-ODBC Bridge Driver:**

=>Type-1 driver is used to convert Java Call(JDBC Call) into Native Call and this Native Call is converted into DataBase Specific call using ODBC-driver for DB-Connection.

**Dis-Advantage:**

=>In Type-1 driver we have more number of conversions and consume more execution time and degrades the performance of an application.

**Note:**

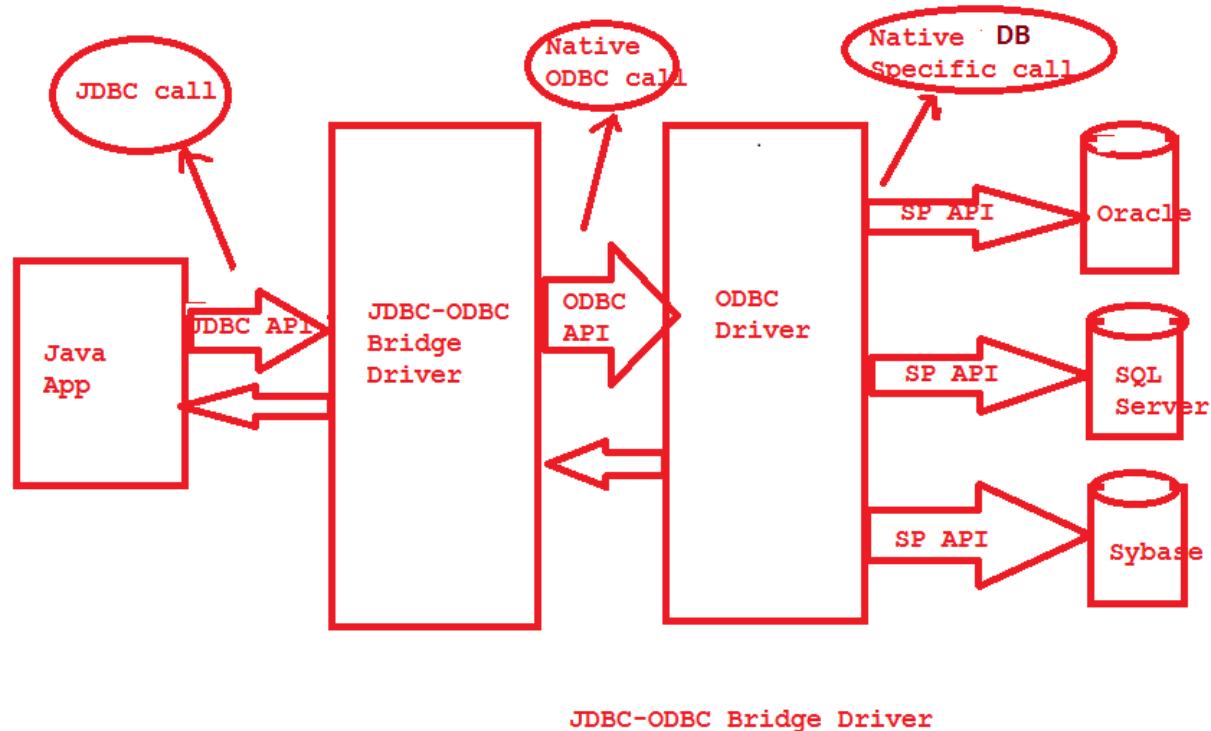
=>From Java8 version onwards Type-1 driver is not available.

**define ODBC?**

=>ODBC stands for 'Open DataBase Connectivity' and this ODBC-driver is used to convert Native calls into DB Specific calls.

=>ODBC driver is Platform dependent because internally uses c or c++ code.

*Diagram:*



*(b) Type-2 : Native API Driver:*

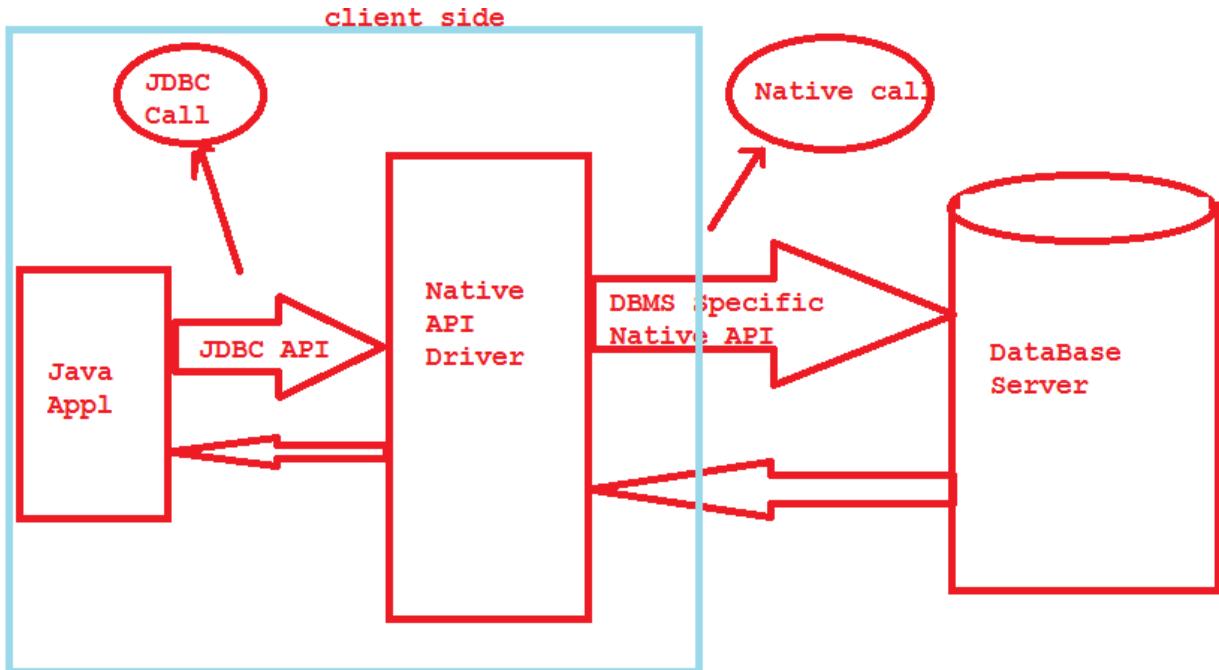
=> In Type-2 driver the Client Machine is installed with DataBase Specific Native API.

=> In this process, the Type-2 driver will convert Java Call(JDBC Call) into DataBase Specific call using 'DataBase Native API' for DB-Connection.

*Dis-Advantage:*

=> In Type-2 driver the applications are DataBase dependent applications.

*Diagram:*



#### (c) Type-3 : Network Protocol Driver:

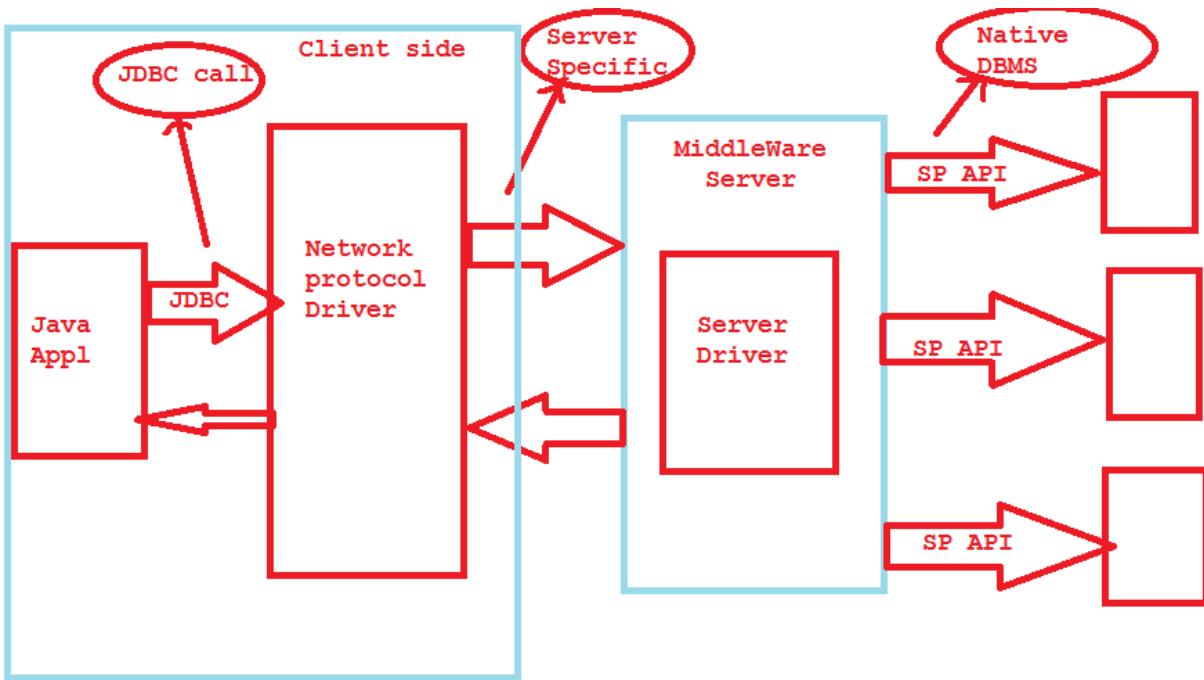
=> In Type-3 driver the Java Call(JDBC call) is converted into Server Call

(Middle-Ware server) and the Server is communicated to DataBase for connection.

*DisAdvantage:*

=> In Type-3 driver Network components are participated in execution process and consumes execution time and degrades the performance of an application.

*Diagram:*



\*imp

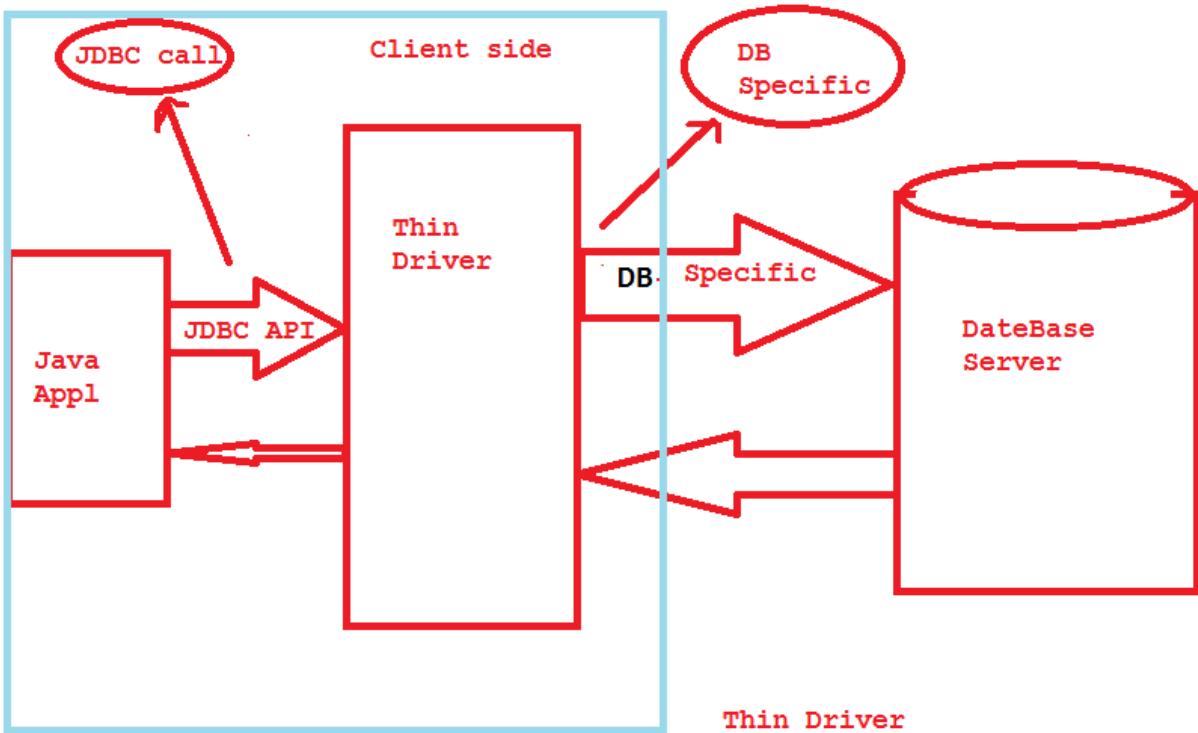
#### (d) Type-4 : Thin Driver

=> In Type-4 driver the Java Call(JDBC call) is converted into DataBase call directly for connection.

=> Type-4 driver is Pure-Java driver and platform independent driver.

=> Type-4 driver is HighPerformance driver and which consume less execution time

Diagram:



-----  
Venkatesh N.

Dt : 5/2/2022

faq:

**define Stream?**

=>*The continuous flow of data is known as Stream.*

**Types of Streams:**

=>*Streams in Java are categorized into two types:*

(a) **Byte Stream(Binary Stream)**

(b) **Character Stream**

**(a) Byte Stream(Binary Stream):**

=>*The continuous flow of data in the form of 8-bits(byte) is known as Byte Stream or Binary Stream.*

**Note:**

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

**(b) Character Stream :**

=>*The continuous flow of data in the form of 16-bits is known as Character Stream.  
(Text Stream)*

**Note:**

=>*Character Stream is preferable for Text data and not preferable for Audio, Video, Image and Animation.*

---

*\*imp*

### ***Streams with DataBase:(SQL-Types)***

*=> DataBase provides the following SQL-Types to record Stream data:*

**1.BLOB**

**2.CLOB**

**1.BLOB:**

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

**2.CLOB:**

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

---

*\*imp*

### ***Servlet Programming:(Unit-2)***

*=> The platform independent java program which is executed in Server environment  
is known as Servlet program or Server program.*

**Note:**

*=> Servlet Programs will extend the functionality of Servers.*

***faq:***

***define Server?***

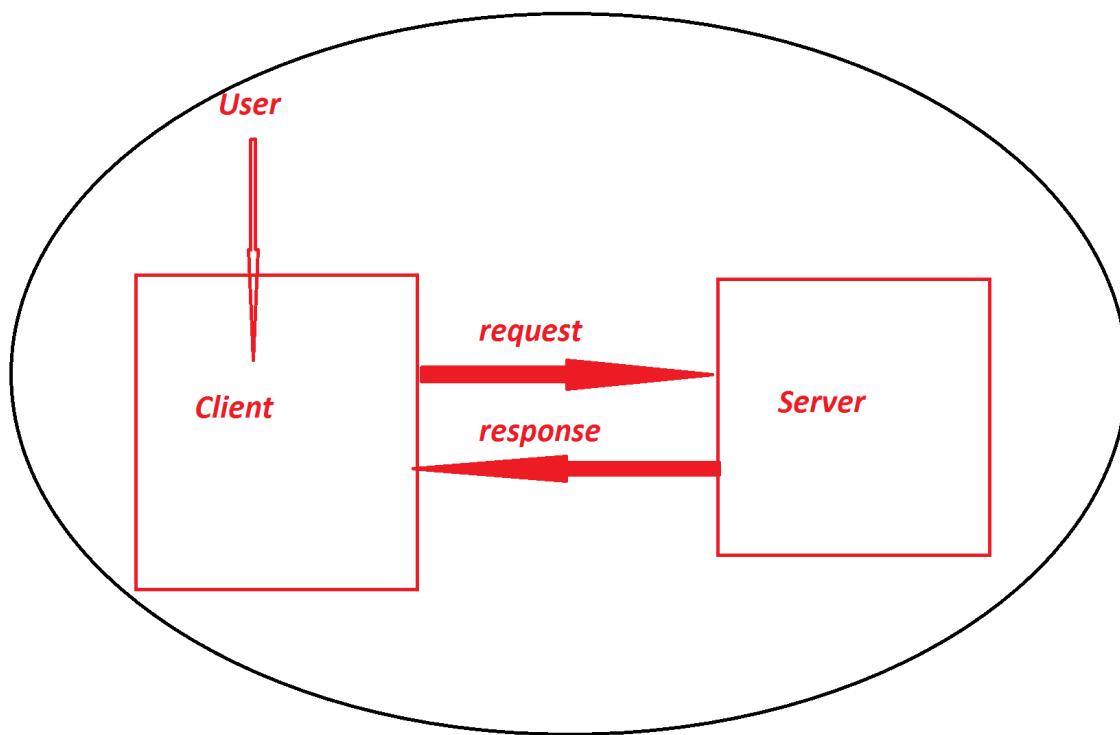
=>Server means service provider, which means accepting the request and providing the response.

faq:

**define Client?**

=>The user who generate request for Server is known as Client.

**Diagram:**



**Types of applications:**

=>Applications are categorized into the following:

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** or **desktop applications** or **Windows applications**.

=>Based on User interaction the **Standalone applications** are categorized into two Types:

**(a) CUI Applications**

**(b) GUI Applications**

**(a) CUI Applications:**

=>The applications in which the user interacts through **Console** are known as **CUI Applications.** (**CUI - Console User Interface**)

**(b) GUI Applications:**

=>The applications in which the user interacts through **GUI components.**  
**(GUI - Graphical User Interface)**

**Ex:**

**AWT - Abstract Window Toolkit**

**Swing**

\*imp

## 2. Web applications:

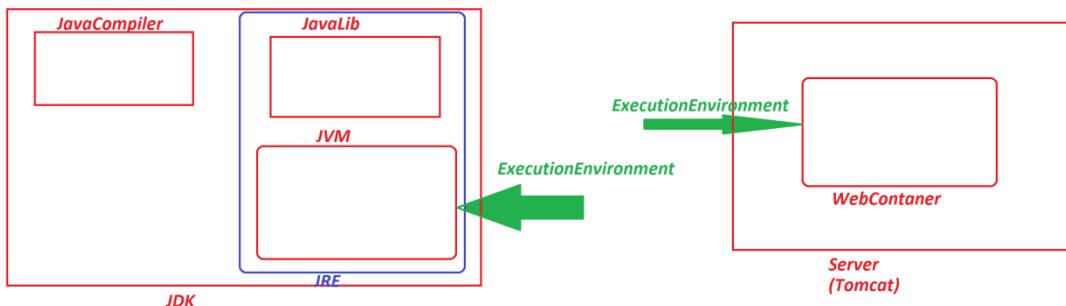
=>The applications which are executed in Web Environment or Internet Environment are known as Web Applications.

=>we use the following technologies to construct Web Applications:

- (a) JDBC
- (b) Servlet
- (c) JSP

=>These WebApplications are executed in WebContainer and which is the execution environment of Servers.

### Diagram:



Dt : 7/2/2022

## 3. Enterprise applications:

=>The applications which are executed in distributed environment and depending on the features like Security, Load Balancing and Clustering process is known as

*Enterprise applications or Enterprise Distributed applications.*

#### **4. Mobile applications:**

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

*\*imp*

#### *Types of Servers:*

=>*Servers are categorized into two types:*

*(i) WebServers*

*(ii) Application Servers*

##### *(i) WebServer:*

=>*Web Server contains only Web container.*

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

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

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

=>*Web Server supports HTTP Protocol*

=>*Apache Web Server.(Tomcat)*

##### *(ii) Application Server:*

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

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

=>*Application server utilizes more resources*

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

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

*(RPC - Remote Procedure call)*

*(RMI - Remote Method Invocation)*

=>*Weblogic, JBoss.*

---

\**imp*

*Installing Tomcat Server:*

*step1 : Download Tomcat9.x WebServer*

*webserver - Tomcat 9.x*

*(Comptable 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*

*step-2 : Install Tomcat Server*

*while Installation process,*

*select the type of install : full (click next)*

*HTTP/1.1 Connector Port : 8081 or 8082 or ...*

*User Name : venkatesh*

*Password : nit*

*(click on next)*

*step-3 : Start the Tomcat Server*

*=> To start the tomcat server, click on 'startup' or 'Tomcat9w' from 'bin' of  
Tomcat folder*

*C:\Tomcat 9.0\bin*

*step-4 : Open WebBrowser and check the Tomcat Server is responding or not using the*

*following url:*

***http://localhost:8082***

***step-5 Stop the Tomcat Server***

***=>To stop the tomcat server,click on 'shutdown' or 'Tomcat9w' from 'bin' of  
Tomcat folder***

***C:\Tomcat 9.0\bin***

---

---

Dt : 8/2/2022

\*imp

**Servlet API:**

=>'javax.servlet' package is known as **Servlet API** and which provide 'classes and interfaces' used in constructing **Servlet** programs.

=>'javax.servlet.Servlet' interface is the root of **Servlet API**

=>The following are the methods of 'Servlet' interface:

- (i) `init()`
- (ii) `service()`
- (iii) `destroy()`
- (iv) `getServletConfig()`
- (v) `getServletInfo()`

=>`init()`,`service()` and `destroy()` methods are known as **LifeCycle methods** and executed automatically in same order.

=>The following are the method Signatures:

```
public abstract void init(javax.servlet.ServletConfig)
    throws javax.servlet.ServletException;

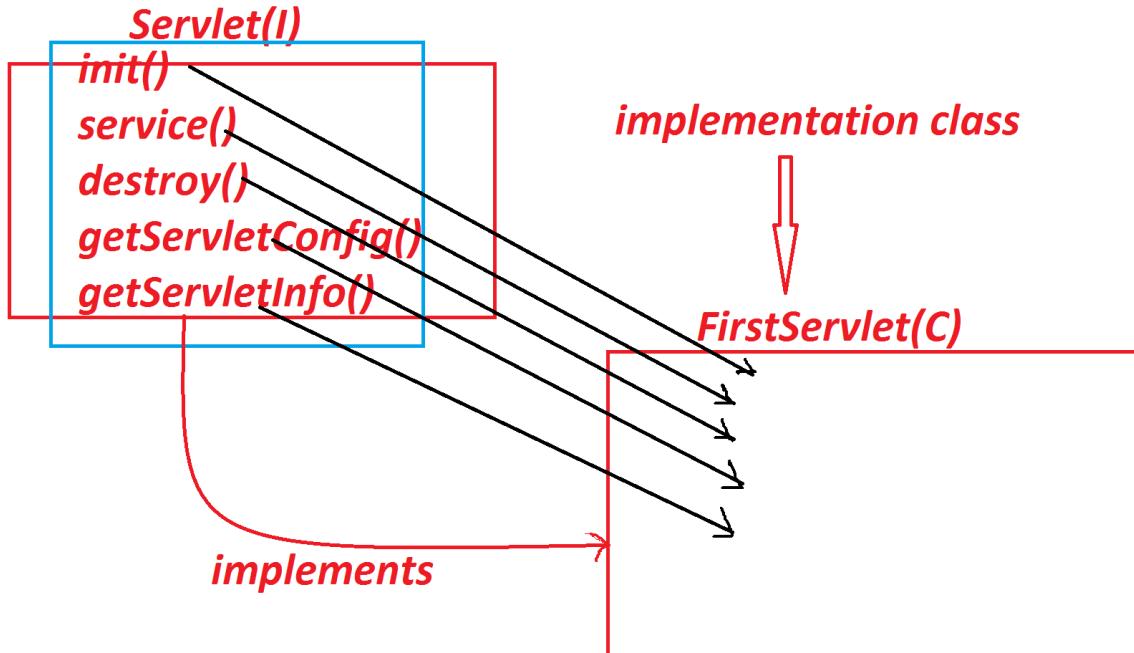
public abstract void service
    (javax.servlet.ServletRequest, javax.servlet.ServletResponse)
    throws javax.servlet.ServletException, java.io.IOException;

public abstract void destroy();
```

```
public abstract javax.servlet.ServletConfig getServletConfig();  
public abstract java.lang.String getServletInfo();
```

=>In the process of constructing Servlet program the user defined class must be implemented from 'javax.servlet.Servlet' interface.

Diagram:



Creating Dynamic Web Project(Web Application) using IDE Eclipse:

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

**step-2 : Create Dynamic Web Project**

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

**step-3 : Add 'servlet-api.jar' file to Dynamic Web Project**

**Right Click on Dynamic Web Project->Built path->Configure Build path->  
Libraries->select classpath and click 'Add External Jars'->Browse and select  
'servlet-api.jar' file from "lib" of Tomcat->Open->Apply->Apply and Close.**

**step-4 : create package in 'src'**

**step-5 : Create Servlet program(class)**

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

**step-6 : Write the following program:**

```
package test;  
  
import java.io.*;  
  
import javax.servlet.*;  
  
import javax.servlet.annotation.*;  
  
@WebServlet("/first")
```

```
public class FirstServlet implements Servlet{  
    public void init(ServletConfig sc)  
        throws javax.servlet.ServletException{  
            System.out.println("Initialization process...");  
        }  
  
    public void service(ServletRequest req, ServletResponse res)  
        throws ServletException, IOException{  
        System.out.println("Request Handling process....");  
        //Output on Console  
        PrintWriter pw = res.getWriter();  
        res.setContentType("text/html");  
        pw.println("Welcome to Servlet Programming...");  
        //response onto WebBrowser  
    }  
  
    public void destroy() {  
        System.out.println("Destroying process....");  
    }  
  
    public ServletConfig getServletConfig(){  
        return this.getServletConfig();  
    }  
  
    public String getServletInfo(){  
        return "FirstServlet";  
    }  
}
```

```
 }  
 }  
  
-----
```

**step-7 : Add Tomcat Server to IDE Eclipse**

*Click on Servers->Click to add new Server->Select Tomcat Server->click on next->*

*Browse and select Tomcat installation folder and click finish.*

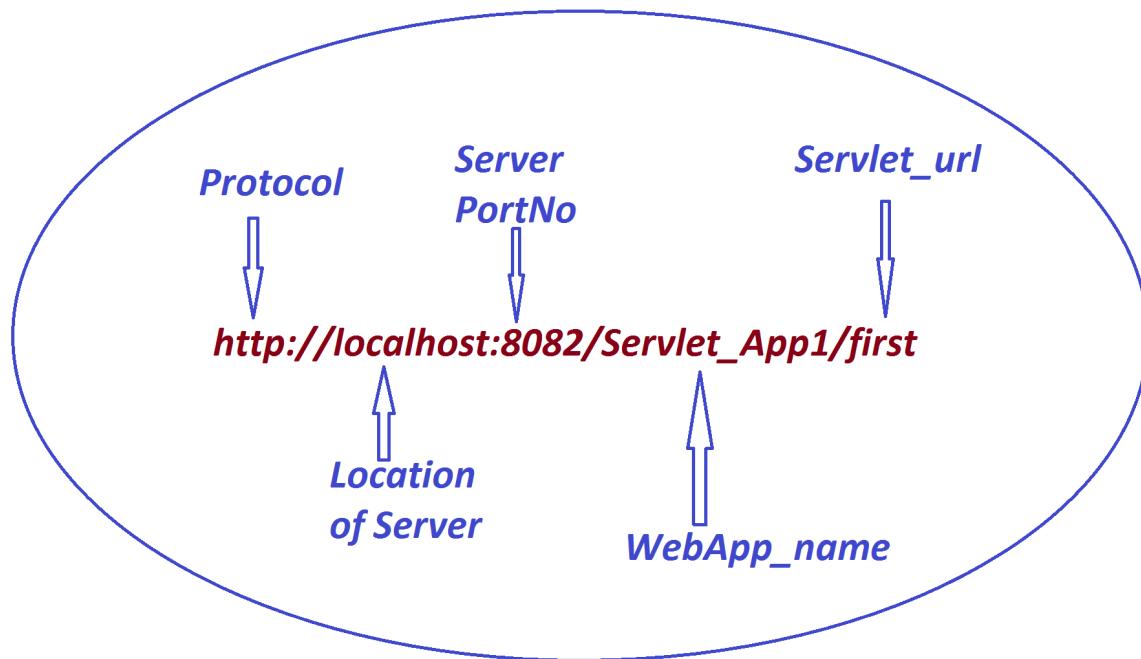
**step-8 : Deploy the application onto Server**

*RightClick on Dynamic Web Project->Run as->Run on Server->Select Tomcat Server  
and click 'finish'*

**Execution URL:**

*[http://localhost:8082/Servlet\\_App1/first](http://localhost:8082/Servlet_App1/first)*

=====

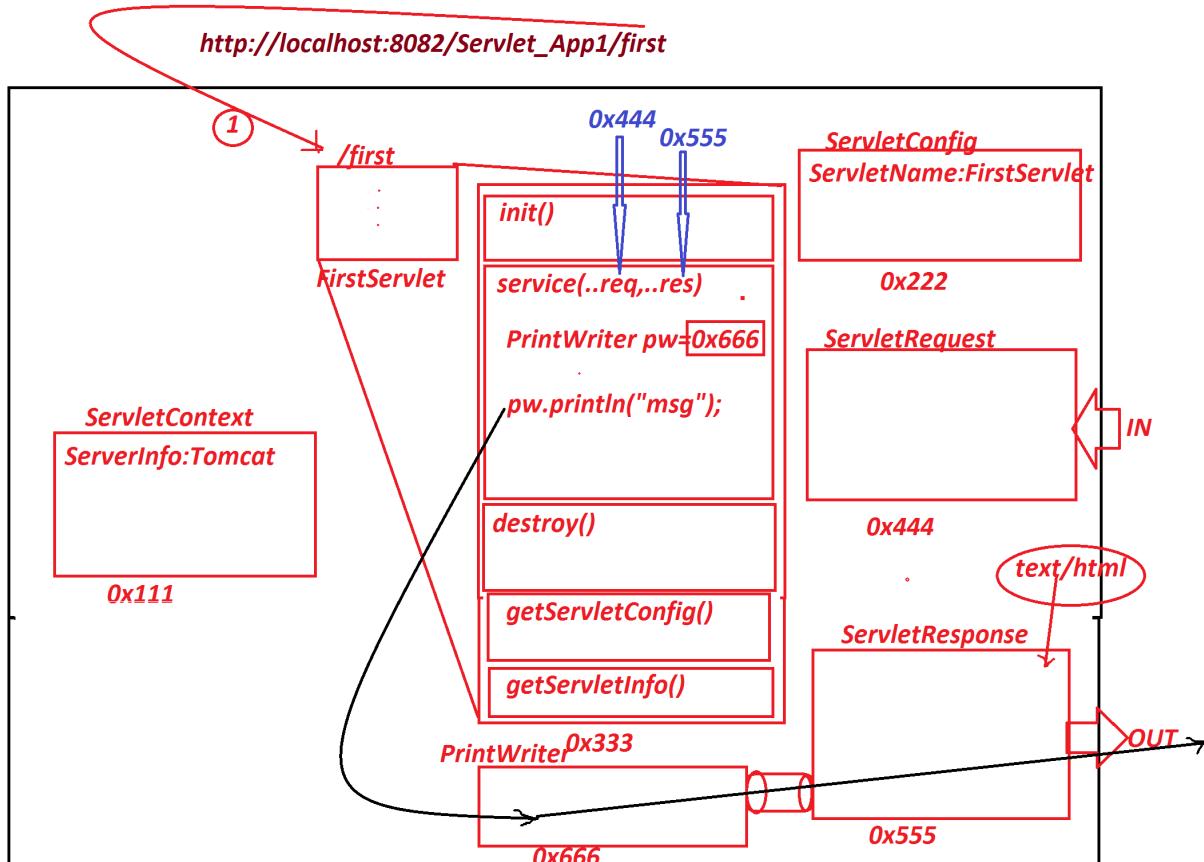


Venkatesh Magadala

Dt : 9/2/2022

\*imp

**Execution flow of above application:**



**ServletContext:**

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

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

**ServletConfig:**

=>*ServletConfig* is also an interface from `javax.servlet` package and which is instantiated automatically when Servlet program loaded onto WebContainer.

=>This `ServletConfig` object is recorded with `Servlet_name`

*ServletRequest:*

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

=>This `ServletRequest` object is recorded with the data submitted from HTML forms.

*ServletResponse:*

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

=>This `ServletResponse` object is recorded with data which is to be sent as response.

*faq:*

*define `getWriter()` method?*

=>`getWriter()` method is from '`ServletResponse`' and which is used to create the object for '`java.io.PrintWriter`' class.

=>This `PrintWriter` object internally linked to `ServletResponse` object to send the output.

*Method Signature:*

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

*Syntax:*

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

*faq:*

*define setContentType() method?*

*=>setContentType() method is from 'ServletResponse' and which specify the type of data we are sending as response.*

*Method Signature:*

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

*syntax:*

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

---

*faq:*

*Life-Cycle of Servlet:*

*=>Life-Cycle of Servlet specifies different states of Servlet program from Starting of program to Ending of Program.*

*=>The following are the states of Servlet Life-Cycle:*

- 1. Loading process*
- 2. Instantiation process*
- 3. Initialization process*
- 4. Request Handling process*
- 5. Destroying process*

*1. Loading process:*

*=>The process of identifying the Servlet program based on url-pattern and*

*loading onto WebContainer for execution is known as 'Loading process'.*

**Note:**

=>*Servlet programs are mapped based on url-pattern.*

=>*we use any one of the following to hold Servlet url-patterns:*

*(i)Annotation*

*syntax:*

`@WebServlet("/first")`

*(ii)web.xml*

**2.Instantiation process:**

=>*Once Servlet program loaded onto WebContainer,then it is automatically instantiated known as 'Instantiation Process'.*

**Note:**

=>*After Instantiation process we can identify the following Life-Cycle methods*

*(i)init()*

*(ii)service()*

*(iii)destroy()*

**3.Initialization process:**

=>*The process of making the programming components ready for service() method is known as Initialization process.*

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

**4.Request Handling process:**

=>*The process of accepting the request from the user and providing the response is known as Request Handling process.*

=>*we use service() method to perform Request handling process.*

#### **5.Destroying process:**

=>*The process of closing the resources or destroying the resources after service() method is known Destroying process.*

=>*we use destroy() method to perform destroying process.*

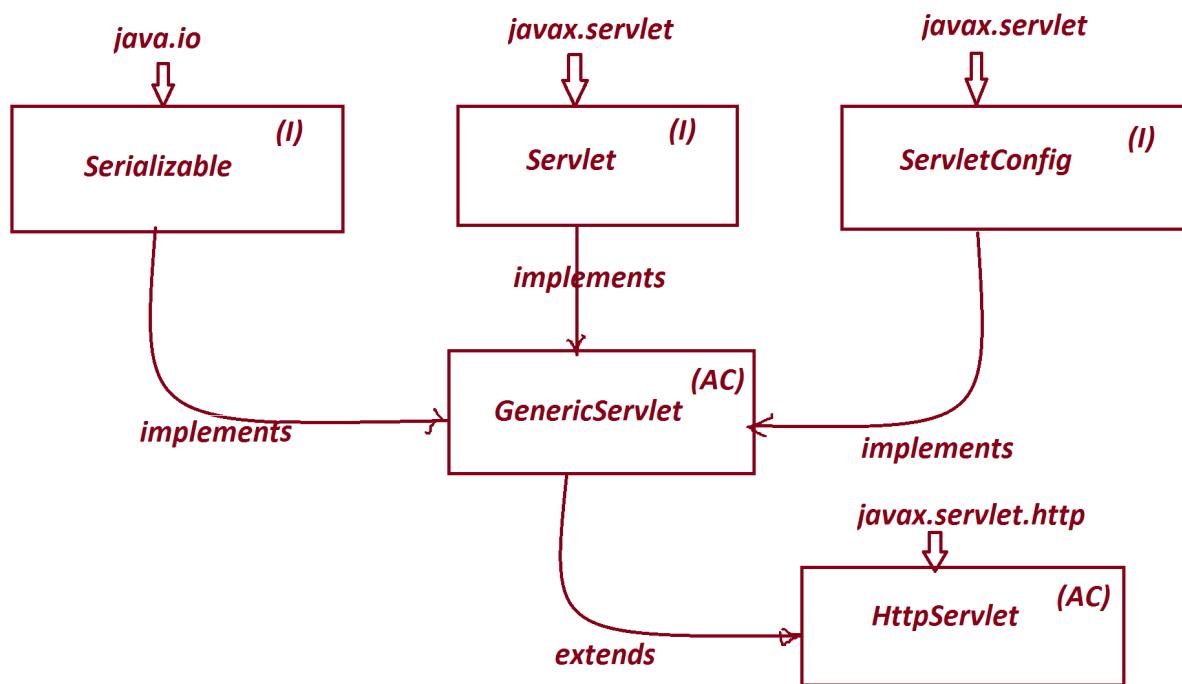
---

-----

Venkatesh Maipathiji

Dt : 10/2/2022

### Hierarchy of Servlet API:



=====

**define Serializable?**

=>'Serializable' interface is from `java.io` package and support

**Serialization process.**

=>**The process of converting Object into binary Stream is know as Serialization process.**

=>**Based on Serialization the Objects are categorized into two types:**

(i)**NonSerializable objects**

(ii)**Serializable objects**

**(i) NonSerializable objects:**

=>The objects which are generated from the class which is not implemented from 'Serializable' interface are known as NonSerializable objects.

**DisAdvantage:**

=>NonSerializable objects cannot be moved on Network.

**(ii) Serializable objects:**

=>The objects which are generated from the class which is implemented from 'Serializable' interface are known as Serializable Objects.

**Advantage:**

=>Serializable objects can be moved on network.

---

**Note:**

**(i)**The Servlet Objects which are generated by implementing from 'javax.servlet.Servlet' interface are NonSerializable objects.

**(ii)**The Servlet Objects which are generated by extending from 'GenericServlet' or by extending from 'HttpServlet' are Serializable objects.

**(iii)**GenericServlet is Protocol independent, which means it will accept the request from any Network protocol.

**(iv)**HttpServlet is Protocol dependent, which means it will accept

*the request from Http Protocol*

---

\*imp

*Constructing Servlet programs using 'GenericServlet':*

=>*GenericServlet is an abstract class from javax.servlet package*

*and which provide the following life-cycle methods:*

*(i) init()*

*(ii) service()*

*(iii) destroy()*

*Method Signatures:*

***public void init(javax.servlet.ServletConfig)***

*throws javax.servlet.ServletException;*

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

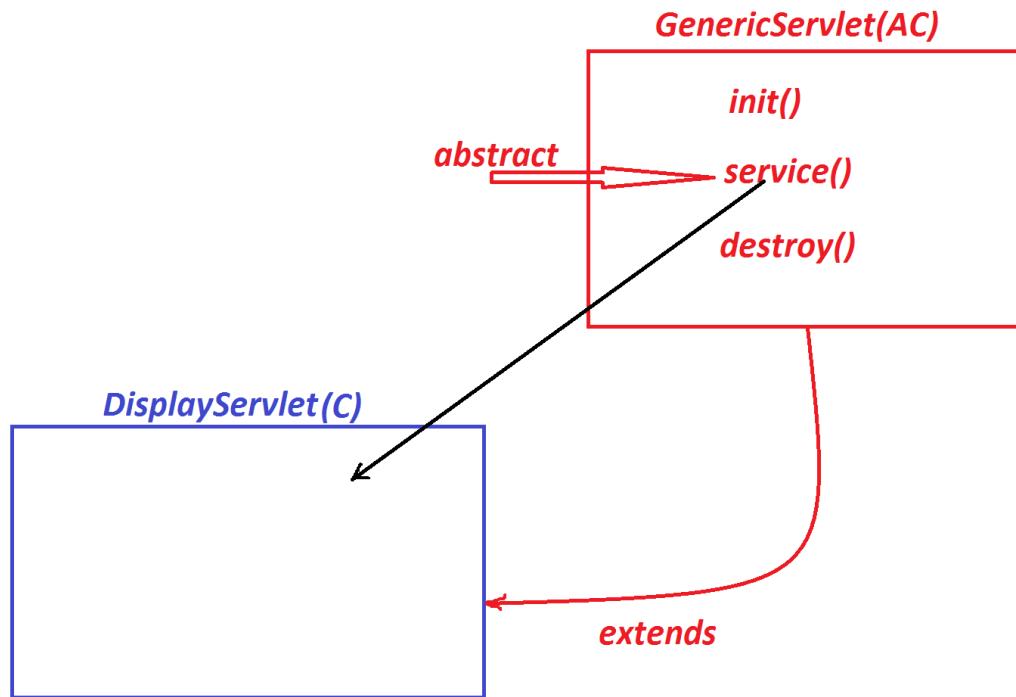
***public abstract void service***

*(javax.servlet.ServletRequest, javax.servlet.HttpServletResponse)*

*throws javax.servlet.ServletException, java.io.IOException;*

***public void destroy();***

*Diagram:*



**Note:**

=>*In the process of constructing Servlet program by extending from 'GenericServlet', we must construct the body for service() method and, init() and destroy() methods are optional.*

=>*If we want initialization process then we use init() method*

=>*If we want destroying process then we use destroy() method*

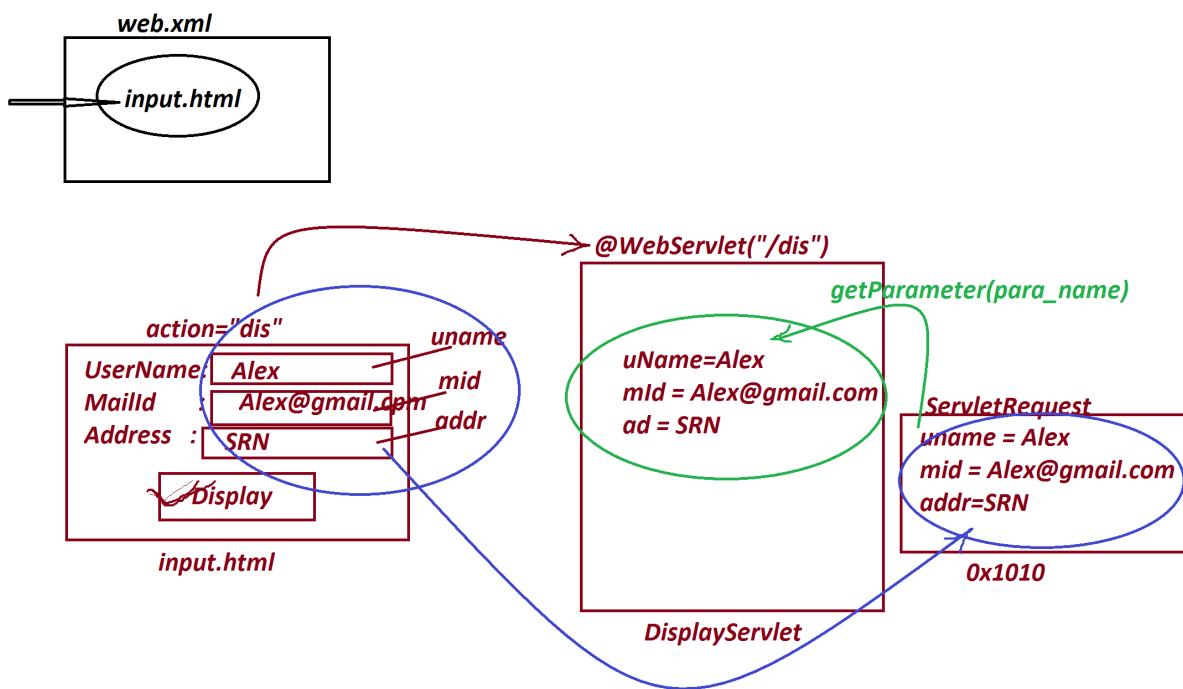
---

Dt : 11/2/2022

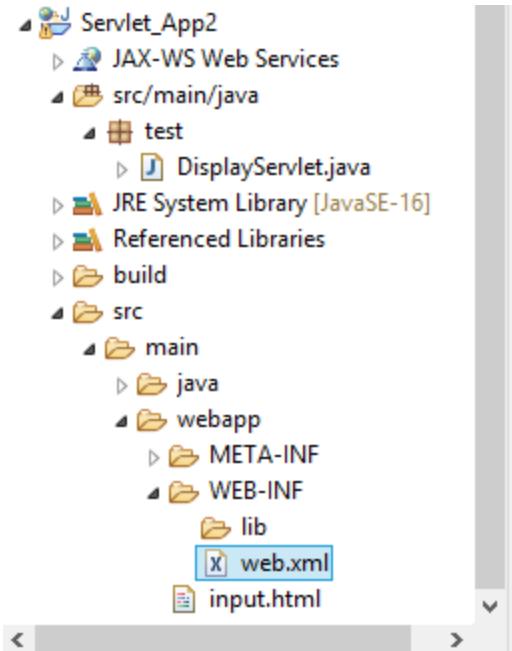
**Ex\_Application-2:**

**Construct Servlet Application to read UserDetails from HTML**

**Form and display the Same.**



Venkatesh



*input.html*

=>Create HTML files in webapp or WebContent folder in IDE

*Eclispe*

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

*'finish'*

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

---

*DisplayServlet.java*

```
package test;

import java.io.*;

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

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

public class DisplayServlet extends GenericServlet{

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

        String uName = req.getParameter("uname");
        String mId = req.getParameter("mid");
        String ad = req.getParameter("addr");
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        pw.println("UserName:"+uName);
        pw.println("<br>MailId:"+mId);
        pw.println("<br>Address:"+ad);
    }
}
```

---

*web.xml*

=>create web.xml file in WEB-INF folder

*Right Click on WEB-INF->new->Other->XML->XML file->name the file and click 'finish'*

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

---

*faq:*

*define getParameter() method?*

*=>getParameter() method is from 'ServletRequest' and which is used to get the data from request object into Servlet program.*

*Method Signature:*

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

*syntax:*

*String var = req.getParameter("para\_name");*

---

*Assignment-1:*

*Construct Servlet program to read and display BookDetails*

*(bcode,bname,bauthor,bprice,bqty)*

*Assignment-2:*

*Construct Servlet program to read and display ProductDetails*

*(pcode,pname,pprice,pqty)*

---

Dt : 12/2/2022

\*imp

*RequestDispatcher:*

=>*RequestDispatcher is an interface from javax.servlet package and which provide the following two methods to perform Servlet Communication process:*

(i)forward()

(ii)include()

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

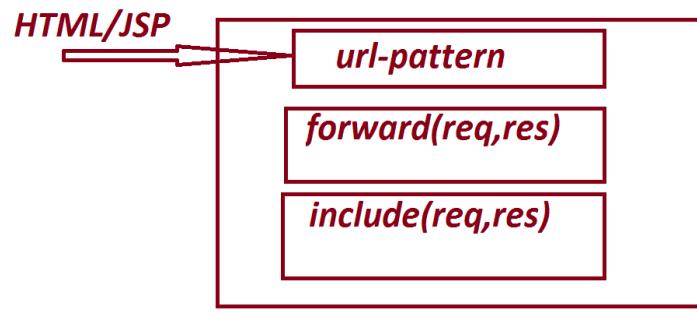
*Method Signature of getRequestDispatcher():*

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

*syntax:*

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

*Diagram:*



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

*rd.forward(req,req);* → *forward communication*  
*rd.include(req,res);* → *include communication*

(i) *forward():*

=> using *forward()* method we can perform *forward communication* process, which means *FirstServlet* will take the request and forward the request to the *SecondServlet*, in this process the response is generated from *SecondServlet*.

*Method Signature of forward():*

```

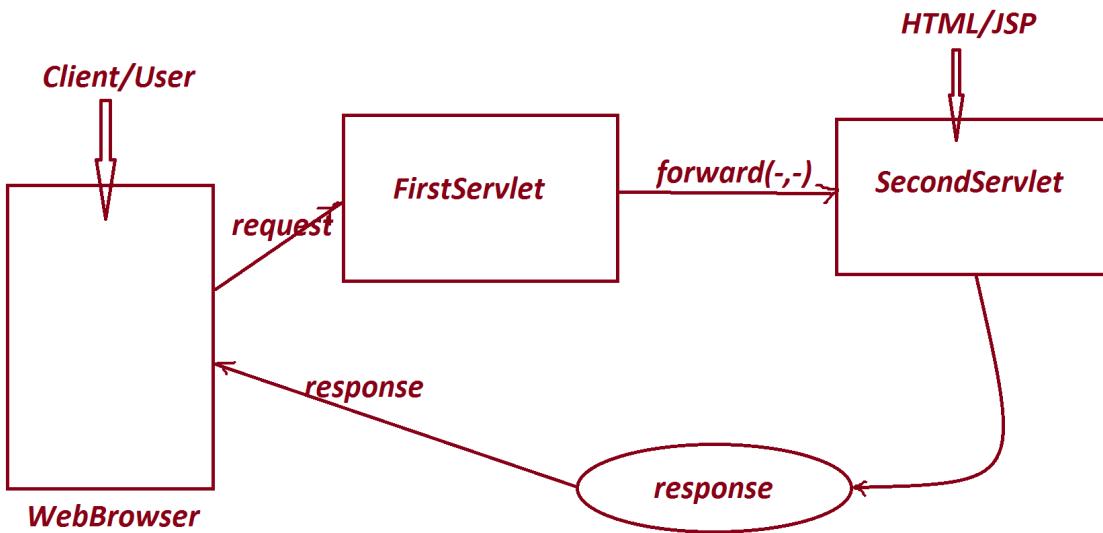
public abstract void forward
(javax.servlet.ServletRequest, javax.servlet.ServletResponse)
throws javax.servlet.ServletException, java.io.IOException;

```

*syntax:*

*rd.forward(req,res);*

*Diagram:*



### (ii) include():

=>using `include()` method we can perform include communication process, which means the `FirstServlet` will generate response but the response of `FirstServlet` is included with the response of `SecondServlet`.

*Method Signature of include():*

```

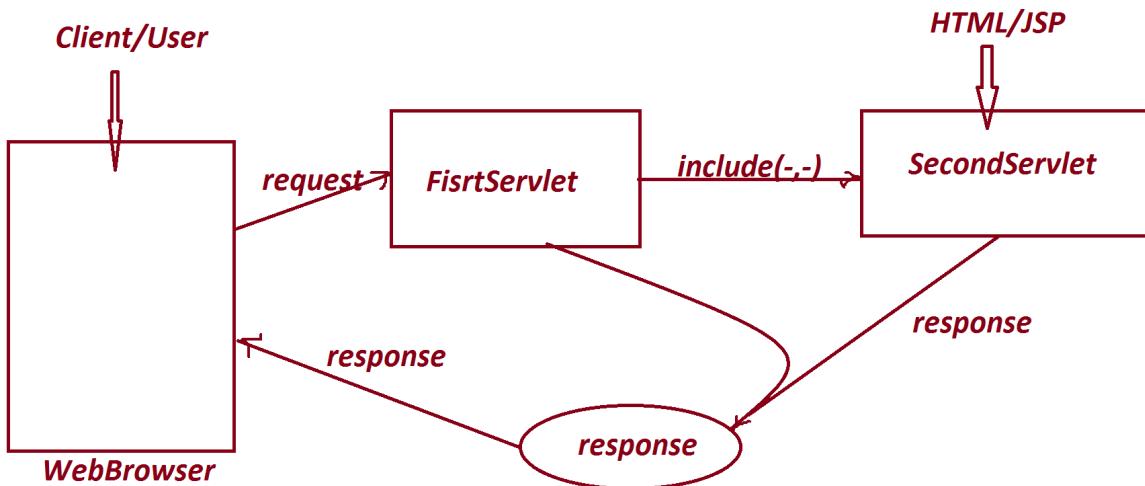
public abstract void include
(javax.servlet.ServletRequest, javax.servlet.ServletResponse)
throws javax.servlet.ServletException, java.io.IOException;

```

*syntax:*

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

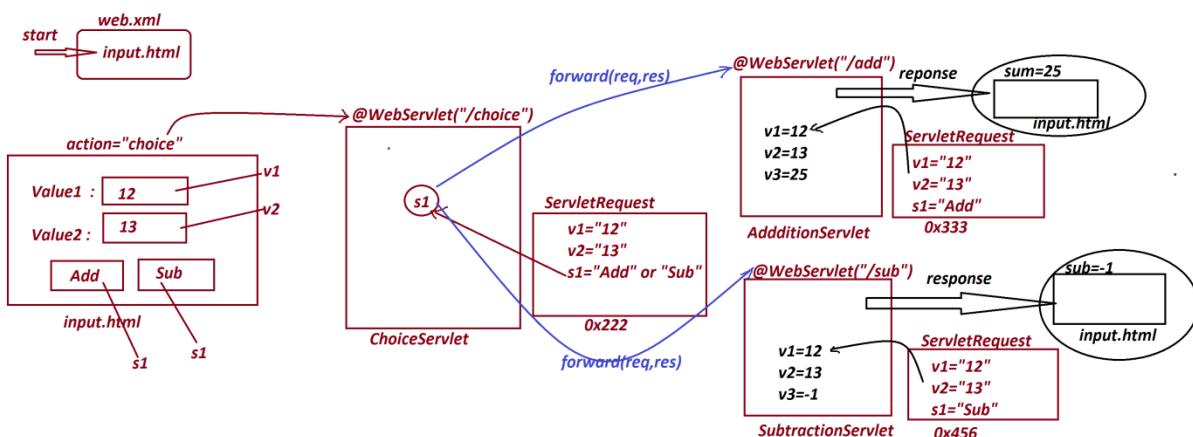
**Diagram:**



**Note:**

=> In forward communication the data which is available in request object of FirstServlet is also available in request object of SecondServlet.

**Ex\_Application:**



*input.html*

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

*ChoiceServlet.java*

```
package test;

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

```
        }else {  
            RequestDispatcher rd =  
                req.getRequestDispatcher("sub");  
            rd.forward(req, res);  
        }  
    }  
  
AdditionServlet.java  
  
package test;  
  
import java.io.*;  
  
import javax.servlet.*;  
  
import javax.servlet.annotation.*;  
  
@SuppressWarnings("serial")  
@WebServlet("/add")  
  
public class AdditionServlet extends GenericServlet{  
  
    public void service(ServletRequest req,ServletResponse res)  
    throws ServletException,IOException{  
  
        int v1 = Integer.parseInt(req.getParameter("v1"));  
        int v2 = Integer.parseInt(req.getParameter("v2"));  
        int v3 = v1+v2;  
  
        PrintWriter pw = res.getWriter();  
        res.setContentType("text/html");  
        pw.println("Sum:"+v3+"<br>");  
        RequestDispatcher rd =
```

```
        req.getRequestDispatcher("input.html");

        rd.include(req, res);

    }

}

SubtractionServlet.java

package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/sub")

public class SubtractionServlet extends GenericServlet{

    public void service(ServletRequest req,ServletResponse res)

        throws ServletException,IOException{

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

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

        int v3 = v1-v2;

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

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

        RequestDispatcher rd =

            req.getRequestDispatcher("input.html");

        rd.include(req, res);

    }

}
```

}

**web.xml**

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

---

**Assignment:**

*Update above applications aby adding the following Servets*

=>*MultiplicationServlet*

=>*DivisionServlet*

=>*ModDivisionServlet*

**Note:**

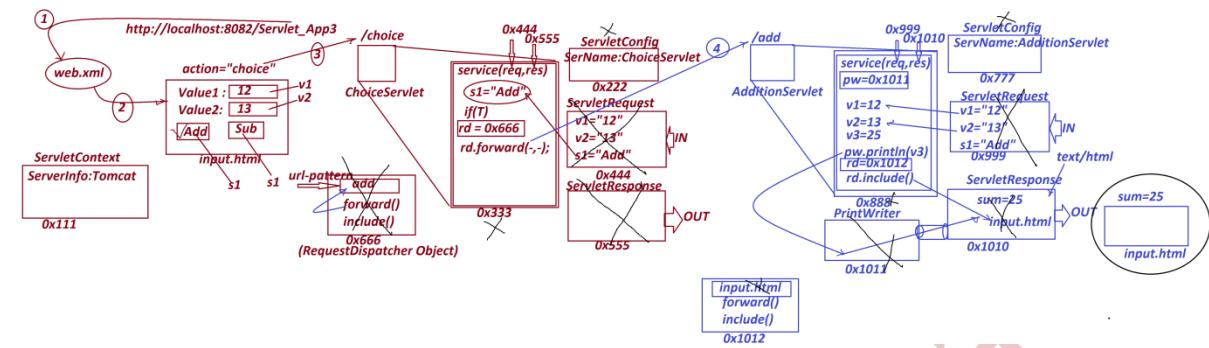
=>*Handle the exception for not int values.*

---

Dt : 14/2/2022

**Execution flow of above application:**

[http://localhost:8082/Servlet\\_App3](http://localhost:8082/Servlet_App3)



=====

\*imp

**ServletContext:**

=>ServletContext is an interface from javax.servlet package

and which is instantiated automatically when the WebApp is deployed into Server.

=>This ServletContext object is loaded with Server information.

=>we use getServletContext() method to access the Object\_reference of ServletContext.

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

**Method signature getServletContext():**

```
public javax.servlet.ServletContext getServletContext();
```

**syntax:**

```
ServletContext sct = this.getServletContext();
```

*Dt : 15/2/2022*

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

*to ServletContext object*

*syntax:*

```
<web-app>
  <context-param>
    <param-name> name </param-name>
    <param-value> value </param-value>
  </context-param>

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

*=>we use getInitParameter() method to access the parameter*

*from ServletContext object.*

*Method Signature:*

```
public abstract java.lang.String
```

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

*syntax:*

```
String var = sct.getInitParameter("para_name");
```

**Note:**

=>The information which is available in ServletContext object

can be used by all the Servlet Programs of WebApplication.

=====

\*imp

**ServletConfig:**

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

=>*This ServletConfig object is loaded with ServletName.*

=>*Every Servlet program in WebApplication will have its own ServletConfig object.*

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

=>*This getServletConfig() method is available from GenericServlet.*

**Method Signature:**

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

**syntax:**

`ServletConfig sc = this.getServletConfig();`

=>*we use <init-param> tag in <servlet> tag of web.xml to initialize parameters in ServletConfig Object*

*syntax:*

```
<web-app>

  <context-param>

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

  <servlet>

    <servlet-name> name </servlet-name>
    <servlet-class> pack_name.Class_name </servlet-class>
    <init-param>
      <param-name> name </param-name>
      <param-value> value </param-value>
    </init-param>
  </servlet>

  <servlet-mapping>
    <servlet-name> name </servlet-name>
    <url-pattern> url-pattern </url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    .
  </welcome-file-list>
```

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

**Note:**

=>we use `getInitParameter()` method to access the parameters  
from `ServletConfig Object`.

```
=====
```

**Ex\_Application:**

*input.html*

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

*DisplayServlet.java*

```
package test;

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

@SuppressWarnings("serial")

public class DisplayServlet extends GenericServlet{

    public void service(ServletRequest req,ServletResponse res)
        throws ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
    }
}
```

```

String uName = req.getParameter("uname");
ServletContext sct = this.getServletContext();
//Accessing ServletContext Object reference
pw.println("UserName:"+uName);
pw.println("<br>====ServletContext====");
pw.println("<br>ServerInfo:"+sct.getServerInfo());
pw.println
("<br>ContextValue:"+sct.getInitParameter("a"));
ServletConfig sc = this.getServletConfig();
//Accessing ServletConfig Object reference
pw.println("<br>====ServletConfig====");
pw.println("<br>ServletName:"+sc.getServletName());
pw.println("<br>ConfigValue:"+sc.getInitParameter("b"));
}
}


```

#### *web.xml*

```

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

    <servlet>
        <servlet-name>DisplayServlet</servlet-name>
        <servlet-class>test.DisplayServlet</servlet-class>
        <init-param>
            <param-name>b</param-name>
            <param-value>2000</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>

```

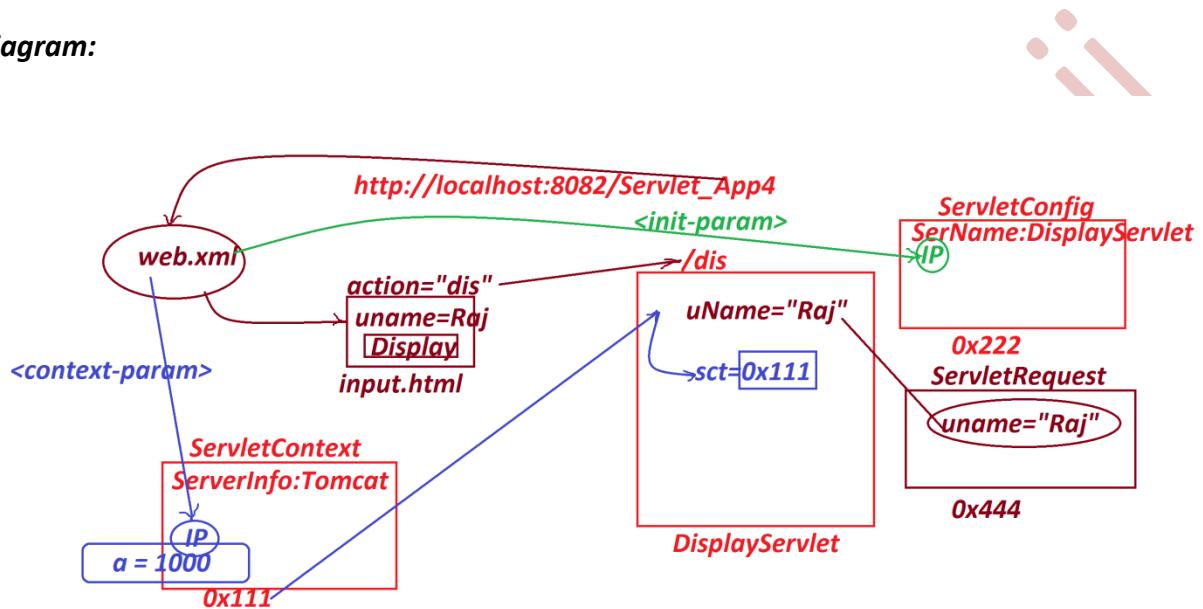
```

<servlet-name>DisplayServlet</servlet-name>
<url-pattern>/dis</url-pattern>
</servlet-mapping>

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

```

*Diagram:*



*faq:*

*wt is the diff b/w*

*(i)ServletContext*

*(ii)ServletConfig*

*=>ServletContext is instantiated automatically when WebApp is*

*deployed onto Server, but ServletConfig is instantiated*

*automatically when the Servlet program loaded onto WebContainer.*

*=>ServletContext object will hold Server information, but*

*ServletConfig will hold Servlet\_name.*

*faq:*

*wt is the diff b/w*

*(i)<context-param>*

*(ii)<init-param>*

*<context-param> will initialize the parameters to ServletContext object, but <init-param> will initialize parameters to ServletConfig Object.*

*faq:*

*wt is the diff b/w*

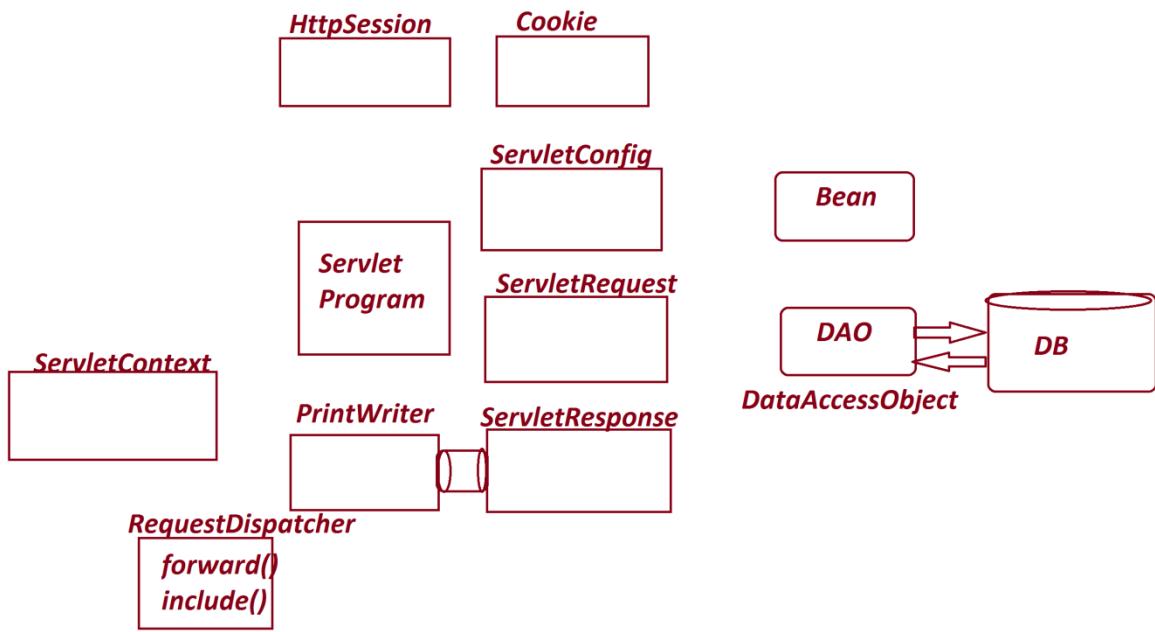
*(i)getParameter()*

*(ii)getInitParameter()*

*=>getParameter() method is used to access the parameter-value from ServletRequest object, but getInitParameter() method is used to access the parameter-value from ServletContext object ServletConfig Object.*

=====

*Object Layout in Servlet programming:*



-----  
Venkatesh MC  
-----

Dt : 16/2/202

### DAO(Data Access Object) Layer:

=>DAO stands for Data Access Object and which is separate layer

in MVC(Model View Controller) to hold database related

operations like:

*creating connection*

*creating table*

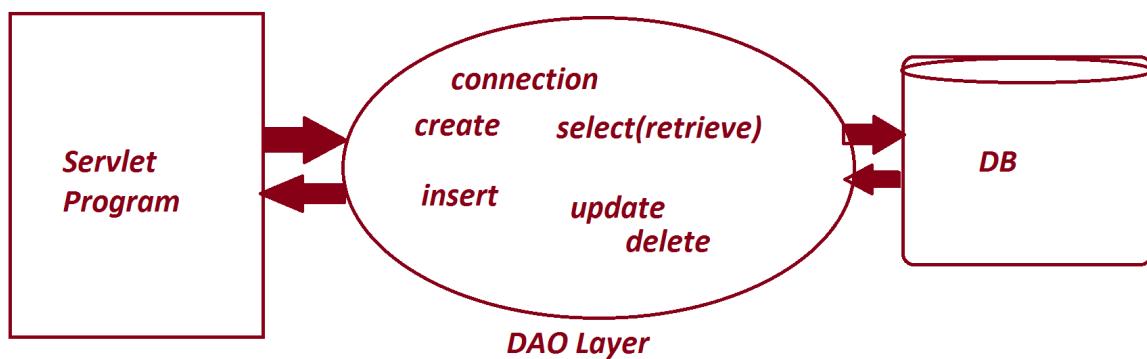
*inserting data*

*retrieving data*

*updating data*

*deleting data*

Diagram:



Note:

=>we use the following 'Singleton class design pattern' to  
hold database connection code:

*package test;*

```

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

```

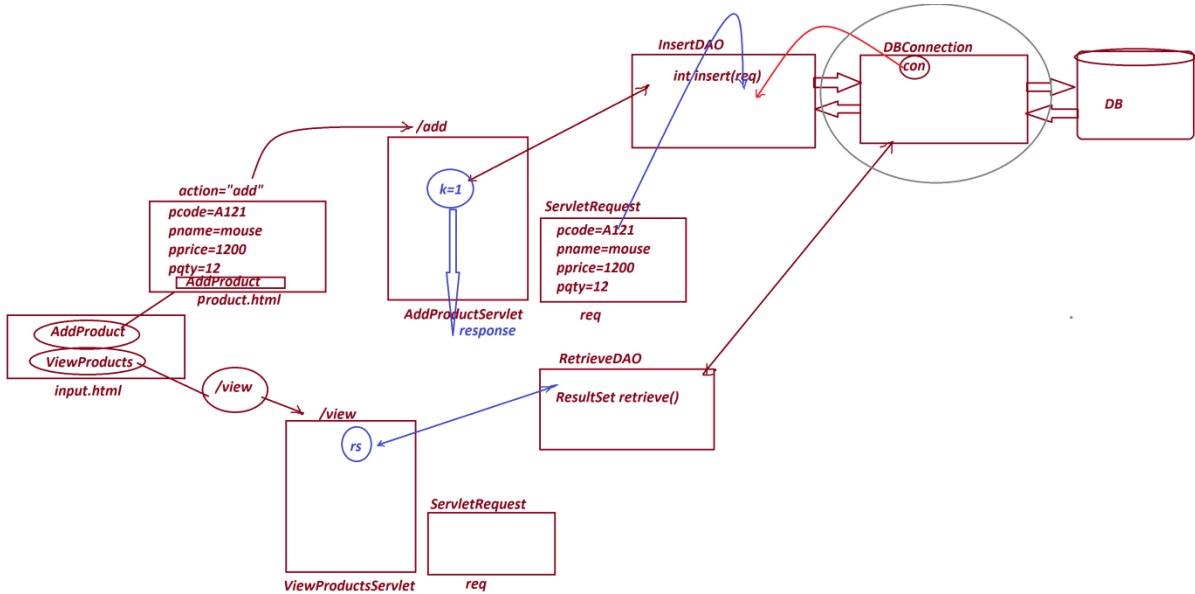
**Note:**

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

---

**Ex\_Application:**

**Diagram:**



### DBConnection.java

```

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

```

### input.html

```

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>

```

```
<body>
<a href="product.html">AddProduct</a>
<a href="view">ViewProducts</a>
</body>
</html>
```

*product.html*

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="add" 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="AddProduct">
</form>
</body>
</html>
```

*InsertDAO.java*

```
package test;

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

public class InsertDAO {

    public int k=0;

    public int insert(ServletRequest req) {
        try {
            Connection con = DBConnection.getCon();
            //Accessing the existing Connection

            PreparedStatement ps = con.prepareStatement
                ("insert into Product41 values(?,?,?,?,?)");

```

```
ps.setString(1,req.getParameter("PCODE"));
ps.setString(2,req.getParameter("PNAME"));
ps.setFloat
(3,Float.parseFloat(req.getParameter("PPRICE")));
ps.setInt
(4,Integer.parseInt(req.getParameter("PQTY")));
k = ps.executeUpdate();
}catch(Exception e){e.printStackTrace();}
return k;
}
}
```

#### AddProductServlet.java

```
package test;
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/add")
public class AddProductServlet extends GenericServlet{
    public InsertDAO ob=null;
    public void init() throws ServletException{
        ob = new InsertDAO(); //Creating object
    }
    public void service(ServletRequest req,ServletResponse res)
```

```
throws ServletException, IOException{

    PrintWriter pw = res.getWriter();

    res.setContentType("text/html");

    int k = ob.insert(req);

    if(k>0) {

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

        RequestDispatcher rd =

            req.getRequestDispatcher("input.html");

        rd.include(req, res);

    }

}

}


```

#### *RetrieveDAO.java*

```
package test;
import java.sql.*;
public class RetrieveDAO {
    public ResultSet rs = null;
    public ResultSet retrieve() {
        try {
            Connection con = DBConnection.getCon();
            //Accessing existing Connection
            PreparedStatement ps = con.prepareStatement
                ("select * from Product41");
            rs = ps.executeQuery();
        }catch(Exception e) {e.printStackTrace();}
        return rs;
    }
}
```

#### *ViewProductsServlet.java*

```
package test;
```

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/view")
public class ViewProductsServlet extends GenericServlet{
    public RetrieveDAO ob=null;
    public void init()throws ServletException{
        ob = new RetrieveDAO();
    }
    public void service(ServletRequest req,ServletResponse res)
    throws ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        try {
            ResultSet rs = ob.retrieve();
            if(rs==null) {
                pw.println("Products not available...<br>");
                RequestDispatcher rd =
                    req.getRequestDispatcher("input.html");
                rd.include(req, res);
            }else {
                while(rs.next()) {

```

```
pw.println(rs.getString(1)+" &nbsp"+
rs.getString(2)+" &nbsp"+
rs.getFloat(3)+" &nbsp"+
rs.getInt(4)+"<br>");

}//end of loop

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

}

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

}

web.xml

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

Dt : 17/2/2022

**Assignment-1:**

*Update above application by adding the following options:*

*input.html*

*=>AddProduct*

*=>ViewProducts*

*=>ViewByProdCode*

*=>DeleteProduct*

**Assignment-2:**

*Servlet Applications to perform the following operations on*

*BookDetails.*

*input.html*

*=>AddBook*

*=>ViewBooks*

*=>ViewByBCode*

*=>DeleteBook*

---

*faq:*

*define Bean class?*

*=>The class which is constructed with the following rules is*

*known as Bean class.*

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

*interface*

**Rule-2 :The variables in 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 'Getter methods' and  
'Setter methods'.**

**Note:**

=>*These bean classes generate 'bean objects' and the bean  
objects are 'Serializable objects'.*

=>*These bean objects will hold data going onto DB-Product and  
these bean objects are also used to hold data retrieved from  
DB-Product.*

---

**faq:**

**define Getter methods?**

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

**define Setter methods?**

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

**Coding Rule:**

=>*In the process of constructing Setter methods and Getter*

*methods in bean classes, every variable will have its own Setter and Getter method.*

---

*\*imp*

*define 'attribute' in Servlet programming?*

*=>'attribute' is a variable in servlet programming which can be added to ServletContext object, ServletRequest object and HttpSession object.*

*=>The following are some important methods related to 'attribute':*

- (a)setAttribute()*
- (b)getAttribute()*
- (c)removeAttribute()*

*(a)setAttribute():*

*=>This method is used to add attribute to the Objects.*

*Method Signature:*

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

*(b)getAttribute():*

*=>This method is used to get the attribute from objects.*

*Method Signature:*

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

**(c)removeAttribute():**

=>*This method is used to delete the attribute from the objects.*

**Method Signature:**

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

---

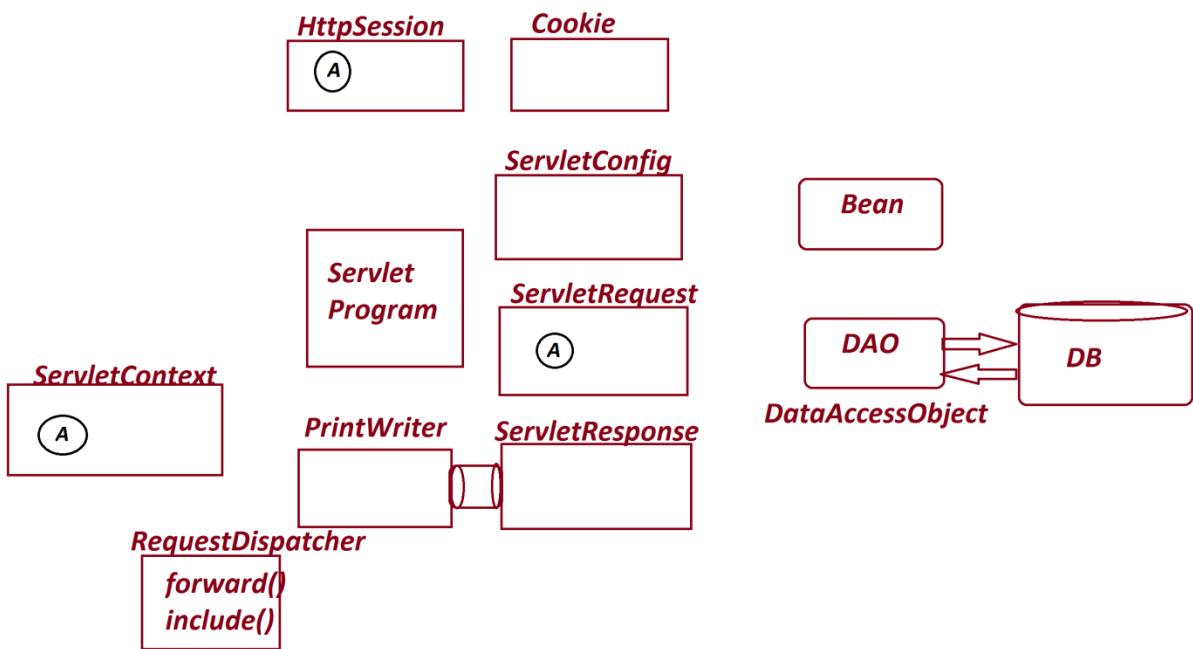
**Scope of 'attribute':**

=>*The attribute in ServletContext object can be used by all the Servlet programs of WebApp.*

=>*The attribute in ServletRequest object can be used by next Servlet program in forward communication process.*

=>*The attribute in HttpSession is available from User-Login to user-Logout.*

---



Venkatesh N.

Dt : 18/2/2022

Note:

=>when we want to have 'Session Tracking process' for applications and when we want to identify the type of request then the Servlet program must be constructed by extending from 'HttpServlet'.

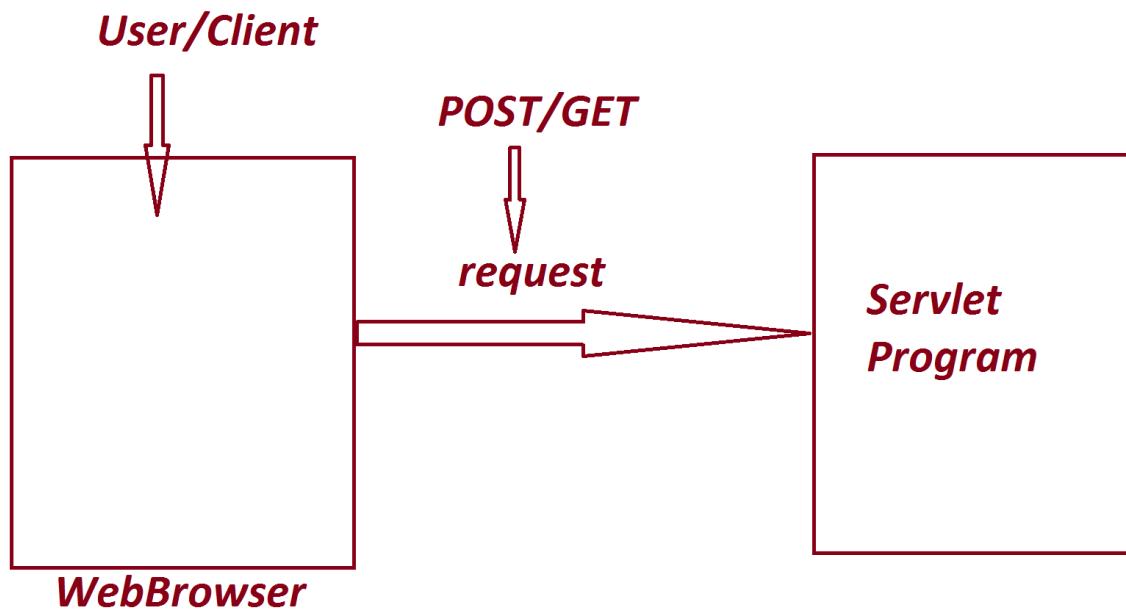
=>when the Servlet program is extended from HttpServlet then we can construct body for doPost() method or doGet() method or service() method.

doPost() - to accept POST request

doGet() - to accept GET request

service() - Any type of request POST/GET

Diagram:



---

*faq:*

*define request?*

=>*The query which is generated by the User/Client to the Servlet Program(Server Program) is known as 'request'.*

=>*'request' is categorized into two types:*

*1.POST request*

*2.GET request*

*(1)POST request:*

=>*The request which is used to post(send) the 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.

---

**(2)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/ServletApp1/first`

**(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 displayed in

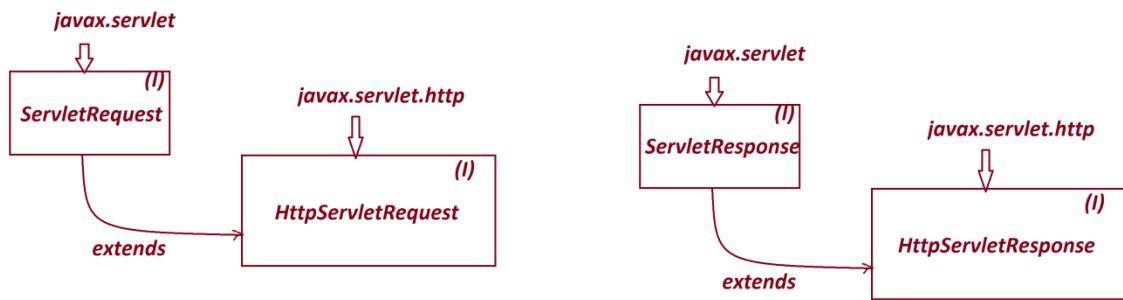
address bar.

---

**Note:**

=>*HttpServletRequest* is extended from *ServletRequest* and  
*HttpServletResponse* is extended from *ServletResponse*.

**Diagram:**



---

**Assignment:**

*Servlet Application to perform the following operations related  
to Employee:*

**1.AddEmployee**

**2.ViewEmployees**

*DB table : Employee41*

*(eid,ename,edesg,hno,sname,city,pincode,bsal,totsal)*

*totsal must be calculated from bsal*

---

Dt : 21/2/2022

\*imp

**Session Tracking in Servlet programming:**

**define Session?**

=>The time interval b/w login\_of\_user to logout\_of\_user is known as 'session'.

**define Session Tracking process?**

=>The process of finding the 'state of user' in session is known as Session Tracking process.

=>The following are the techniques used in Session Tracking

**Process:**

1. *Cookie*
2. *HttpSession*
3. *URL-Rewrite*
4. *Hidden Form fields*

\*imp

**1. Cookie:**

=>The piece of information persisted b/w multiple client requests is known as cookie.

=>Cookie is stored in WebBrowser to track the user.

=>Cookies are categorized into two types:

(i) Persistent Cookie

(ii) NonPersistent Cookie

(i) Persistent Cookie:

=>The Cookies which are stored and available even after WebBroser is closed are known as Persistent Cookies.

(ii) NonPersistent Cookie:

=>The Cookies which will become invalidated when the WebBrowser is closed are known as NonPersistent Cookies.

---

=>we use 'Cookie' class from javax.servlet.http package to construct the functionality of Cookie.

=>The following are some important methods from 'Cookie' class:

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

```
public void setMaxAge(int);
```

```
public int getMaxAge();
```

```
public java.lang.String getName();
```

```
public void setValue(java.lang.String);
```

```
public java.lang.String getValue();
```

---

=>The following are the steps used to construct the functionality of Cookie:

**step-1 : Creating Cookie**

=>Cookie is created when the login process is Successfull.

**syntax:**

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

**step-2 : Adding Cookie to the response**

=>we use addCookie() method to add cookie to the response and through response the cookie is stored in WebBrowser.

**Method signature of addCookie():**

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

**syntax:**

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

**step-3 : Getting Cookies from the request**

=>we use getCookies() method from HttpServletRequest to get the cookies from the request.

**Method Signature of getCookies():**

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

**syntax:**

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

*step-4 : Invalidating Cookie*

*=>we use setMaxAge() method to set time for invalidation.*

*syntax:*

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

---

Dt : 22/2/2022

Ex\_Application : User Registration and Login Process.

DB\_Table : UserReg41

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

link.html(ViewProfile,Logout)

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

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>
        MailId:<input type="text" name="mid"><br>
        PhoneNo:<input type="text" name="phno"><br>
        <input type="submit" value="Register">
    </form>
</body>
</html>
```

UserBean.java

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

```
}

public String getpWord() {
    return pWord;
}

public void setpWord(String pWord) {
    this.pWord = pWord;
}

public String getfName() {
    return fName;
}

public void setfName(String fName) {
    this.fName = fName;
}

public String getlName() {
    return lName;
}

public void setlName(String lName) {
    this.lName = lName;
}

public String getAddr() {
    return addr;
}

public void setAddr(String addr) {
    this.addr = addr;
}

public String getmId() {
    return mId;
}

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

public long getPhNo() {
    return phNo;
}

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

### RegServlet1.java

```
package test;

import java.io.*;
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/reg1")
public class RegServlet1 extends HttpServlet{
    @Override
    protected void doPost(HttpServletRequest req,
        HttpServletResponse res) throws ServletException, IOException{
        UserBean ub = new UserBean();
        ub.setuName(req.getParameter("uname"));
        ub.setpWord(req.getParameter("pword"));
        ub.setfName(req.getParameter("fname"));
        ub.setlName(req.getParameter("lname"));
        ServletContext sct = this.getServletContext();
        //Accessing ServletContext Object
        sct.setAttribute("ubean",ub);
        //Attribute Added to ServletContext
        RequestDispatcher rd =
            req.getRequestDispatcher("reg2.html");
        rd.forward(req, res);
    }
}
DBConnection.java
```

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

### InserDAO.java

```
package test;
import java.sql.*;
public class InsertDAO {
    public int k=0;
    public int insert(UserBean ub) {
        try {
            Connection con = DBConnection.getCon();
            //Accessing the existing Connection
            PreparedStatement ps = con.prepareStatement
            ("insert into UserReg41 values(?,?,?,?,?,?)");
            ps.setString(1,ub.getUserName());
            ps.setString(2,ub.getpWord());
            ps.setString(3,ub.getfName());
            ps.setString(4,ub.getlName());
            ps.setString(5,ub.getAddr());
            ps.setString(6,ub.getmId());
            ps.setLong(7,ub.getPhNo());
            k = ps.executeUpdate();
        }catch(Exception e) {e.printStackTrace();}
        return k;
    }
}
```

### RegServlet2.java

```
package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/reg2")

public class RegServlet2 extends HttpServlet{

    @Override

    protected void doPost(HttpServletRequest req,

                          HttpServletResponse res) throws ServletException, IOException{

        ServletContext sct = this.getServletContext();

        //Accessing ServletContext Object

        UserBean ub = (UserBean)sct.getAttribute("ubean");

        //Getting Attribute from ServletContext

        ub.setAddr(req.getParameter("addr"));

        ub.setMid(req.getParameter("mid"));

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

        int k = new InsertDAO().insert(ub); //Bean as parameter

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

        if(k>0){

            pw.println("User Registration Successfull...<br>");

            RequestDispatcher rd =
```

```
        req.getRequestDispatcher("login.html");  
  
        rd.include(req, res);  
  
    }  
  
}  
  
}  
  
web.xml
```

## *LoginDAO.java*

```
package test;  
  
import java.sql.*;  
  
import javax.servlet.http.*;  
  
public class LoginDAO {  
  
    public UserBean ub=null;  
  
    public UserBean login(HttpServletRequest req) {  
  
        try {  
            Connection con = DBConnection.getCon();  
  
            //Accessing Existing Connection  
  
            PreparedStatement ps = con.prepareStatement(  
                ("select * from UserReg41 where uname=? and  
                ps.setString(1,req.getParameter("uname"));  
            }  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

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

ResultSet rs = ps.executeQuery();

if(rs.next())

{

    ub = new UserBean(); //Creating Bean Object

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

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

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

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

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

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

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

}

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

return ub;

}

}

LoginServlet.java

package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.*;

@SuppressWarnings("serial")
```

```
@WebServlet("/login")  
  
public class LoginServlet extends HttpServlet{  
  
    @Override  
  
    protected void doPost(HttpServletRequest req,  
                         HttpServletResponse res) throws ServletException, IOException{  
  
        PrintWriter pw = res.getWriter();  
  
        res.setContentType("text/html");  
  
        UserBean ub = new LoginDAO().login(req);  
  
        if(ub==null) {  
  
            pw.println("Invalid Login process...<br>");  
  
            RequestDispatcher rd =  
  
                req.getRequestDispatcher("login.html");  
  
            rd.include(req, res);  
  
        }else {  
  
            Cookie ck = new Cookie("fname",ub.getfName());  
  
            ServletContext sct = this.getServletContext();  
  
            //Accessing ServletContext Object  
  
            sct.setAttribute("ubean",ub);  
  
            //Adding Attribute to ServletContext  
  
            pw.println("Welcome User : "+ub.getfName()+"<br>");  
  
            res.addCookie(ck);//Adding Cookie to response  
  
            RequestDispatcher rd =  
  
                req.getRequestDispatcher("link.html");  
  
            rd.include(req, res);  
        }  
    }  
}
```

```
    }

}

}

LogoutServlet.java

package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/logout")

public class LogoutServlet extends HttpServlet{

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

PrintWriter pw = res.getWriter();

res.setContentType("text/html");

Cookie c[] = req.getCookies();

if(c==null){

pw.println("Session Expired..<br>");

} else {

c[0].setMaxAge(0); //Session Invalidated

pw.println("User LoggedOut Successfull...<br>");

}

RequestDispatcher rd =
```

```
req.getRequestDispatcher("login.html");

rd.include(req, res);

}

}

ViewProfileServlet.java

package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/view")

public class ViewProfileServlet extends HttpServlet{

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

PrintWriter pw = res.getWriter();

res.setContentType("text/html");

Cookie c[] = req.getCookies();

if(c==null) {

pw.println("Session Expired...<br>");

RequestDispatcher rd =

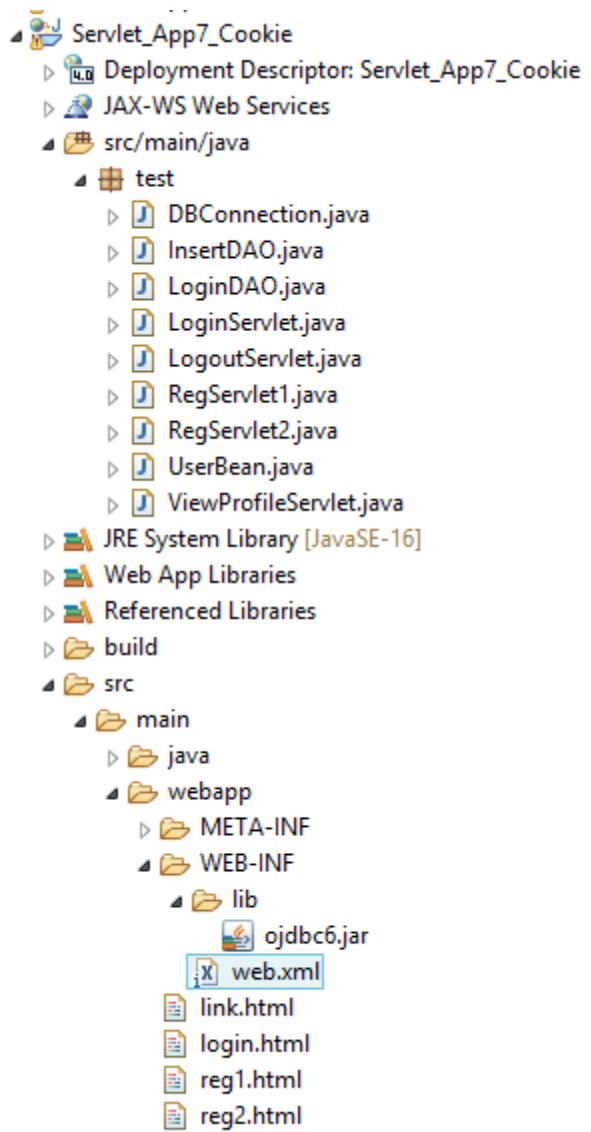
req.getRequestDispatcher("login.html");

rd.include(req, res);

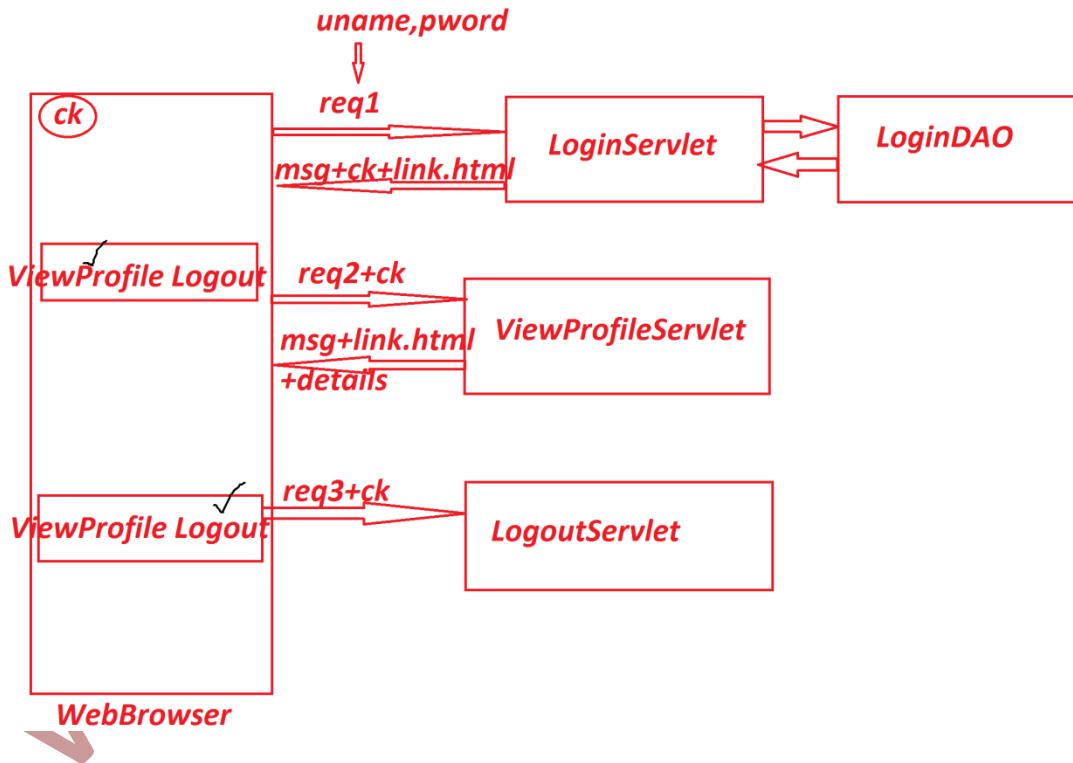
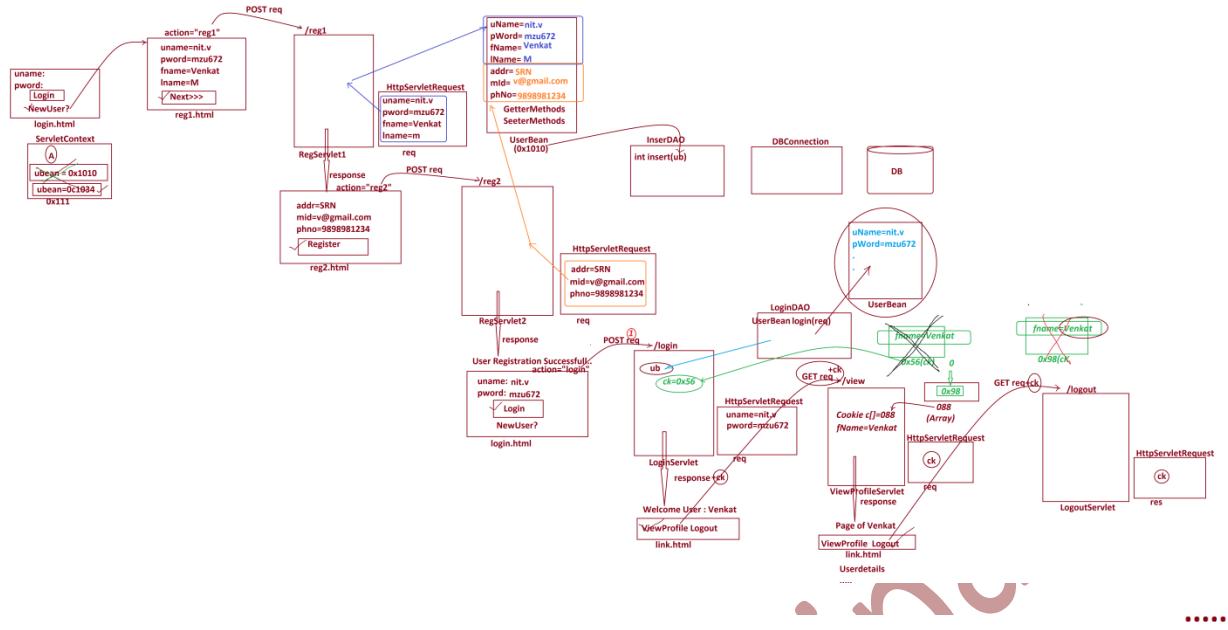
} else {
```

```
String fName = c[0].getValue();  
  
pw.println("Page of "+fName+"<br>");  
  
RequestDispatcher rd =  
  
    req.getRequestDispatcher("link.html");  
  
rd.include(req, res);  
  
ServletContext sct = this.getServletContext();  
  
UserBean ub = (UserBean)sct.getAttribute("ubean");  
  
pw.println("<br>FName:"+ub.getfName());  
  
pw.println("<br>LName:"+ub.getlName());  
  
pw.println("<br>Addr:"+ub.getAddr());  
  
pw.println("<br>MID:"+ub.getMId());  
  
pw.println("<br>PhNo:"+ub.getPhNo());  
  
}  
}  
}
```

---



Venkatesan  
Vaiapathiji



**Assignment-1 :**

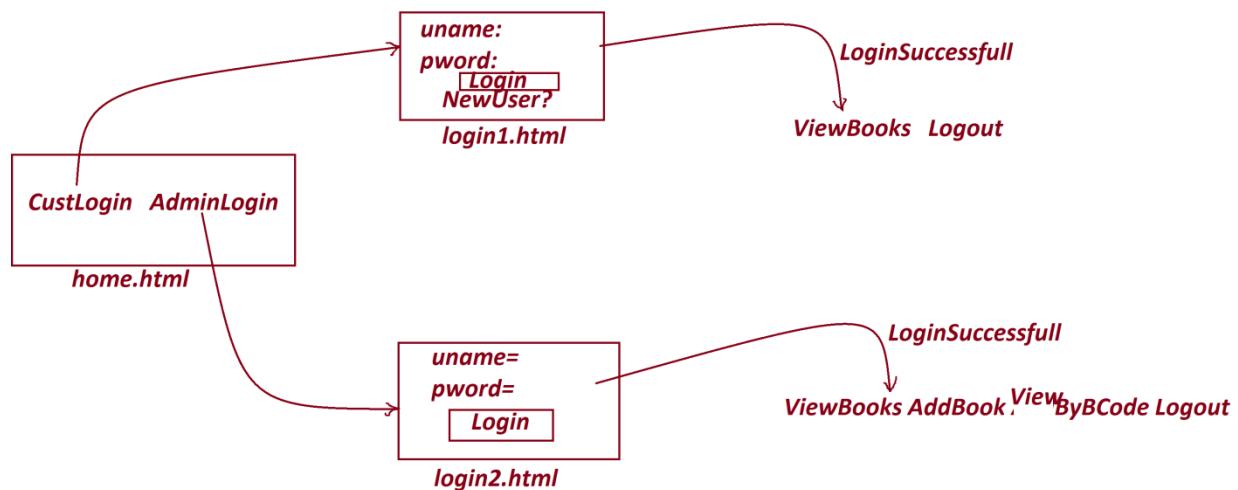
**Update above application with:**

=>UpdateProfile

(update address,mid and phno)

**Assignment-2 :**

**Convert the following layout into codeform:**



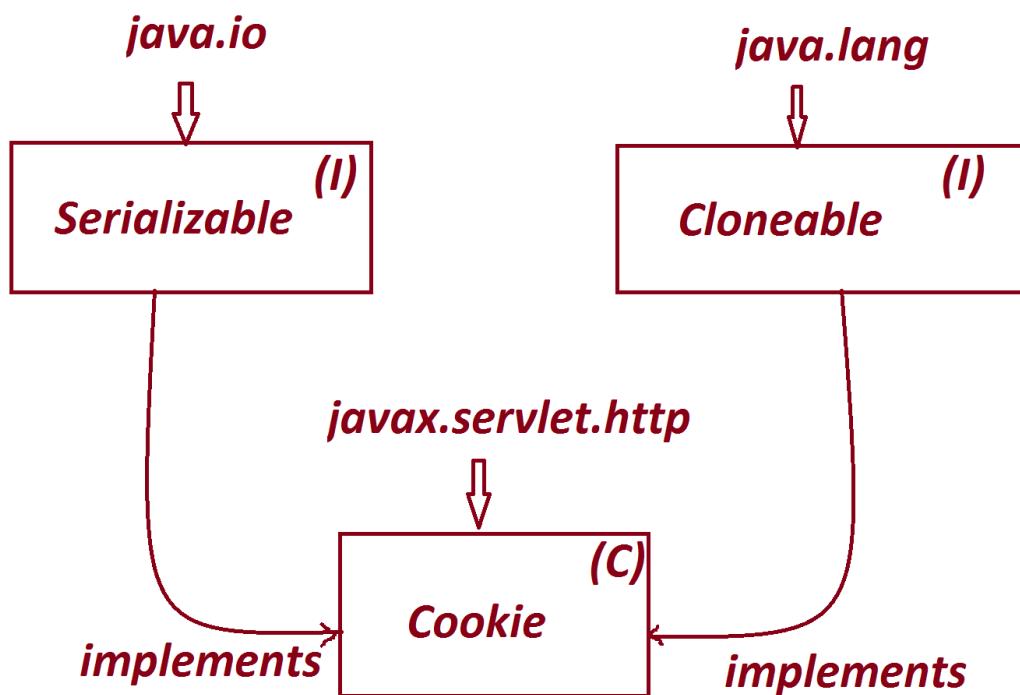
**DB Tables : Book41**

UserReg41(Customer Table)  
AdminReg41(Admin Table)

Ve.

Dt : 25/2/2022

Hierarchy of Cookie:



Note:

=>The `Cookie` objects are `Serializable` objects and `Cloneable` objects.

=>`Serializable` objects means can be converted into binary stream.

=>`Cloneable` objects means can be duplicated part of taking backup of  
objects.

\*imp

2. `HttpSession`:

=>`HttpSession` is an interface from `javax.servlet.http` package and which

*is used in Session tracking process.*

=>*The following are some important methods from 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();*

*public abstract boolean isNew();*

*public abstract javax.servlet.ServletContext getServletContext();*

---

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

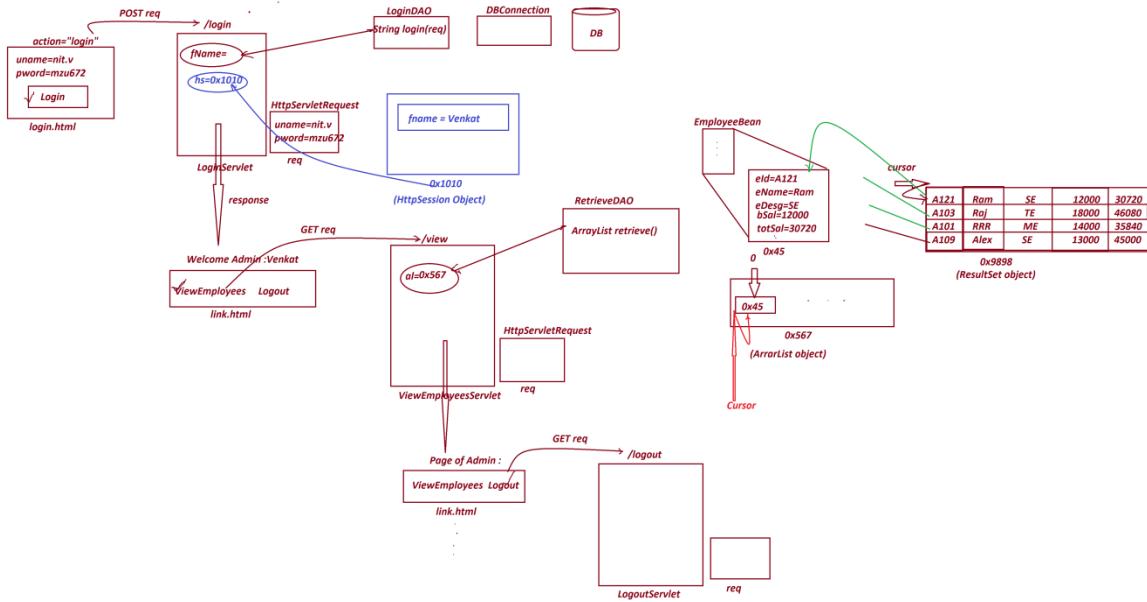
*Method Signature:*

*public abstract javax.servlet.http.HttpSession getSession();*

*public abstract javax.servlet.http.HttpSession getSession(boolean);*

---

*Ex\_Application:*



DBTables : UserReg41,Employee41

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">ViewEmployees</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:1521:xe",
                 "system", "manager");
        } catch (Exception e) {e.printStackTrace();}
    }//end of static
    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();
            //Accessing Existing Connection
            PreparedStatement ps = con.prepareStatement
```

```
("select * from UserReg41 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;

}

}


```

*LoginServlet.java*

```
package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/login")

public class LoginServlet extends HttpServlet{

protected void doPost(HttpServletRequest req,

HttpServletResponse res)throws ServletException,IOException

{
```

```

PrintWriter pw = res.getWriter();

res.setContentType("text/html");

String fName = new LoginDAO().login(req);

if(fName==null) {

    pw.println("Invalid Login process...<br>");

    RequestDispatcher rd =

        req.getRequestDispatcher("login.html");

    rd.include(req, res);

} else {

    HttpSession hs = req.getSession();

    //Creating new Session

    hs.setAttribute("fname",fName);

    //Adding attribute to Session Object

    pw.println("Welcome Admin : "+fName+"<br>");

    RequestDispatcher rd =

        req.getRequestDispatcher("link.html");

    rd.include(req, res);

}

}

}

EmployeeBean.java

```

```

package test;
import java.io.*;
@SuppressWarnings("serial")
public class EmployeeBean implements Serializable{
    private String eId,eName,eDesg;
    private int bSal;

```

```
private float totSal;
public EmployeeBean() {}
public String geteId() {
    return eId;
}
public void seteId(String eId) {
    this.eId = eId;
}
public String geteName() {
    return eName;
}
public void seteName(String eName) {
    this.eName = eName;
}
public String geteDesg() {
    return eDesg;
}
public void seteDesg(String eDesg) {
    this.eDesg = eDesg;
}
public int getbSal() {
    return bSal;
}
public void setbSal(int bSal) {
    this.bSal = bSal;
}
public float getTotSal() {
    return totSal;
}
public void setTotSal(float totSal) {
    this.totSal = totSal;
}
}
```

### RetrieveDAO.java

```
package test;

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

public class RetrieveDAO {
    public ArrayList<EmployeeBean> al=
```

```
new ArrayList<EmployeeBean>();  
  
//ArrayList object can hold UnLimited EmployeeBean objects  
  
public ArrayList<EmployeeBean> retrieve()  
  
{  
  
    try {  
  
        Connection con = DBConnection.getCon();  
  
        PreparedStatement ps = con.prepareStatement  
            ("select * from Employee41");  
  
        ResultSet rs = ps.executeQuery();  
  
        while(rs.next())  
  
        {  
  
            EmployeeBean eb = new EmployeeBean();  
  
            eb.seteId(rs.getString(1));  
  
            eb.seteName(rs.getString(2));  
  
            eb.seteDesg(rs.getString(3));  
  
            eb.setbSal(rs.getInt(4));  
  
            eb.setTotSal(rs.getFloat(5));  
  
            al.add(eb);  
  
            //Adding Bean object to ArrayList Object  
  
        }//end of loop  
  
    }catch(Exception e) {e.printStackTrace();}  
  
    return al;  
  
}
```

*ViewEmployeesServlet.java*

```
package test;

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

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

public class ViewEmployeesServlet extends HttpServlet{
    protected void doGet(HttpServletRequest req,HttpServletResponse res)
        throws ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        HttpSession hs = req.getSession(false);
        //Accessing existing Session object
        if(hs==null) {
            pw.println("Session Expired...<br>");
            RequestDispatcher rd=
                req.getRequestDispatcher("login.html");
            rd.include(req, res);
        }else {
            String fName = (String)hs.getAttribute("fname");
            pw.println("Page of Admin : "+fName+"<br>");
        }
    }
}
```

```
RequestDispatcher rd =  
    req.getRequestDispatcher("link.html");  
  
rd.include(req, res);  
  
pw.println("<br>====EmployeeDetails====<br>");  
  
ArrayList<EmployeeBean> al =  
    new RetrieveDAO().retrieve();  
  
if(al.size()==0) {  
  
    pw.println("No Employee Avaiable..<br>");  
  
}else {  
  
    Iterator<EmployeeBean> it = al.iterator();  
  
    while(it.hasNext()) {  
  
        EmployeeBean eb = (EmployeeBean)it.next();  
  
        pw.println(eb.getId()+" &ampnbsp"+  
                   eb.getName()+" &ampnbsp&ampnbsp"+  
                   eb.getDesg()+" &ampnbsp&ampnbsp"+  
                   eb.getbSal()+" &ampnbsp&ampnbsp"+  
                   eb.getTotSal()+"<br>");  
  
    } //end loop  
}  
}
```

*LogoutServlet.java*

```
package test;
```

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/logout")
public class LogoutServlet extends HttpServlet{
protected void doGet(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException{
PrintWriter pw = res.getWriter();
res.setContentType("text/html");
HttpSession hs = req.getSession(false);
if(hs==null) {
pw.println("Session Expired...<br>");
} else {
hs.invalidate();//Session object is destroyed
pw.println("Admin Loggedout Successfully..<br>");
}
RequestDispatcher rd =
req.getRequestDispatcher("login.html");
rd.include(req, res);
}
}
web.xml
```

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

---

*Assignment:*

*Update above application to display EmployeeCodes and then select  
any one employee code and display details of details of employee*

-----  
*Venkatesh Maipathiji*

Dt : 28/2/2022

### 3. URL-Rewrite:

=>The process of adding parameter-values to the url\_pattern is known as

URL-ReWrite.

**syntax:**

*url\_pattern?para1=value1&para2=value2&...*

"?" => Separator b/w url\_pattern and parameters

"&" => Separator b/w parameter and parameter

**Ex\_Application:**

*input.html*

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

*EmployeeBean.java*

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class EmployeeBean implements Serializable{
    private String eId, eName, eDesg;
    private int bSal;
```

```
private float totSal;
public EmployeeBean() {}
public String getId() {
    return eId;
}
public void setId(String eId) {
    this.eId = eId;
}
public String getName() {
    return eName;
}
public void setName(String eName) {
    this.eName = eName;
}
public String getDesg() {
    return eDesg;
}
public void setDesg(String eDesg) {
    this.eDesg = eDesg;
}
public int getbSal() {
    return bSal;
}
public void setbSal(int bSal) {
    this.bSal = bSal;
}
public float getTotSal() {
    return totSal;
}
public void setTotSal(float totSal) {
    this.totSal = totSal;
}
}
```

### DBConnection.java

```
package test;
import java.sql.*;
public class DBConnection {
    private static Connection con=null;
    private DBConnection() {}
    static {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con=DriverManager.getConnection
```

```
( "jdbc:oracle:thin:@localhost:1521:xe",
    "system", "manager");
}catch(Exception e) {e.printStackTrace();}
}//end of static
public static Connection getCon() {
    return con;
}
}
```

### RetrieveDAO.java

```
package test;

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

public class RetrieveDAO {

    public EmployeeBean eb=null;

    public EmployeeBean retrieve(HttpServletRequest req) {
        try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                ("select * from Employee41 where eid=?");
            ps.setString(1,req.getParameter("eicode"));
            ResultSet rs = ps.executeQuery();
            if(rs.next()) {
                eb = new EmployeeBean(); //Bean object
                eb.seteId(rs.getString(1));
                eb.seteName(rs.getString(2));
                eb.seteDesg(rs.getString(3));
                eb.setbSal(rs.getInt(4));
            }
        }
    }
}
```

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

    }

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

return eb;

}

}

FirstServlet.java

package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/first")

public class FirstServlet extends HttpServlet{

    @Override

    protected void doPost(HttpServletRequest req,HttpServletResponse res)throws

    ServletException,IOException{

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

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

        if(eb==null) {

            pw.println("Invalid EmployeeCode...<br>");

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

```

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

} else {

    pw.print("<a href='second?eid="+eb.getId()+"&ename="+eb.getName()+
        "&edesg="+eb.getDesg()+"&bsal="+eb.getbSal()+
        "&totsal="+eb.getTotSal()+">Display</a>");

}

}

}
```

## *SecondServlet.java*

```
package test;  
  
import java.io.*;  
  
import javax.servlet.*;  
  
import javax.servlet.http.*;  
  
import javax.servlet.annotation.*;  
  
@SuppressWarnings("serial")  
@WebServlet("/second")  
public class SecondServlet extends HttpServlet{  
  
protected void doGet(HttpServletRequest req,HttpServletResponse res) throws  
ServletException,IOException{  
  
PrintWriter pw = res.getWriter();  
  
res.setContentType("text/html");  
  
pw.println("EmpCode:"+req.getParameter("eid"));  
  
pw.println("<br>EmpName:"+req.getParameter("ename"));  
  
pw.println("<br>EmpDesg:"+req.getParameter("edesg"));  
}
```

```
        pw.println("<br>EmpBSal:"+req.getParameter("bsal"));

        pw.println("<br>EmpTotSal:"+req.getParameter("totsal"));

    }

}
```

**web.xml**

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

---

**Ex:**

***http://localhost:8082/ServletApp9\_URLReWrite/second?***

***eid=A121***

***&ename=Ram***

***&edesg=SE***

***&bsal=12000***

***&totsal=30720.0***

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

***q=gmail***

***&oq=gmail***

***&aqs=chrome..69i57j0i433i512l5j69i61j69i60.4808j0j7***

***&sourceid=chrome***

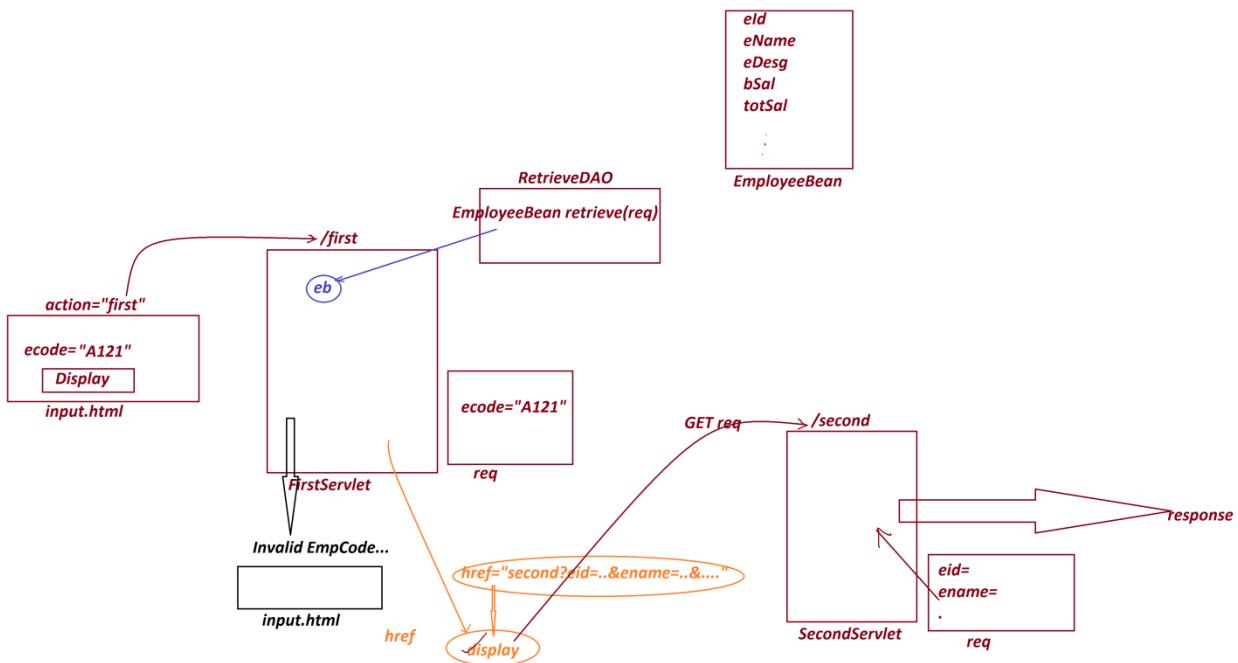
***&ie=UTF-8***

**Note:**

=>Using 'URL-Rewrite' process we can exchange the information b/w

*Servlet programs part of Session Tracking process.*

=====



Venk -

Dt : 1/3/2022

#### 4. Hidden Form fields:

=>The process of declaring `<input type="hidden"....>` in form tag of HTML

is known as Hidden Form Field.

=>The value in Hidden form field is not display to the end user.

=>These Hidden form fields can be used to transfer the information from one Servlet program to another servlet program.

syntax:

```
<form method="post" action="">  
    <input type="hidden" name="course" value="CoreJava"><br>  
    <input type="submit" value="Display">  
</form>
```

Ex\_Application:

*input.html*

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

*EmployeeBean.java*

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class EmployeeBean implements Serializable{
    private String eId,eName,eDesg;
    private int bSal;
    private float totSal;
    public EmployeeBean() {}
    public String geteId() {
        return eId;
    }
    public void seteId(String eId) {
        this.eId = eId;
    }
    public String geteName() {
        return eName;
    }
    public void seteName(String eName) {
        this.eName = eName;
    }
    public String geteDesg() {
        return eDesg;
    }
    public void seteDesg(String eDesg) {
        this.eDesg = eDesg;
    }
    public int getbSal() {
        return bSal;
    }
    public void setbSal(int bSal) {
        this.bSal = bSal;
    }
    public float getTotSal() {
        return totSal;
    }
    public void setTotSal(float totSal) {
        this.totSal = totSal;
    }
}
```

#### DBConnection.java

```
package test;
import java.sql.*;
public class DBConnection {
```

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

#### RetrieveDAO.java

```
package test;

import java.sql.*;
import javax.servlet.http.*;
public class RetrieveDAO {
    public EmployeeBean eb=null;
    public EmployeeBean retrieve(HttpServletRequest req) {
        try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
            ("select * from Employee41 where eid=?");
            ps.setString(1,req.getParameter("ecode"));
            ResultSet rs = ps.executeQuery();
            if(rs.next()) {
                eb = new EmployeeBean(); //Bean object
                eb.setId(rs.getString(1));
            }
        }
    }
}
```

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

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

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

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

    }

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

return eb;

}

}


```

### **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

    protected void doPost(HttpServletRequest req,HttpServletResponse res)throws

    ServletException,IOException{

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

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

```
if(eb==null) {  
  
    pw.println("Invalid EmployeeCode...<br>");  
  
    RequestDispatcher rd = req.getRequestDispatcher("input.html");  
  
    rd.include(req, res);  
  
}  
  
else {  
  
    pw.println("<form action='second' method='post'>");  
  
    pw.println("<input type='hidden' name='eid' value='"+eb.getEid()+"'>");  
  
    pw.println("<input type='hidden' name='ename'  
value='"+eb.getEname()+"'>");  
  
    pw.println("<input type='hidden' name='edesg' value='"+eb.getDesg()+"'>");  
  
    pw.println("<input type='hidden' name='bsal' value='"+eb.getBSal()+"'>");  
  
    pw.println("<input type='hidden' name='totsal' value='"+eb.getTotSal()+"'>");  
  
    pw.println("<input type='submit' value='Display'>");  
  
    pw.println("</form>");  
  
}  
  
}  
  
}  
  
}
```

### *SecondServlet.java*

```
package test;  
  
import java.io.*;  
  
import javax.servlet.*;  
  
import javax.servlet.http.*;  
  
import javax.servlet.annotation.*;  
  
@SuppressWarnings("serial")
```

```

@WebServlet("/second")

public class SecondServlet extends HttpServlet{
    protected void doPost(HttpServletRequest req,HttpServletResponse res) throws
    ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        pw.println("EmpCode:"+req.getParameter("eid"));
        pw.println("<br>EmpName:"+req.getParameter("ename"));
        pw.println("<br>EmpDesg:"+req.getParameter("edesg"));
        pw.println("<br>EmpBSal:"+req.getParameter("bsal"));
        pw.println("<br>EmpTotSal:"+req.getParameter("totsal"));
    }
}

```

#### **web.xml**

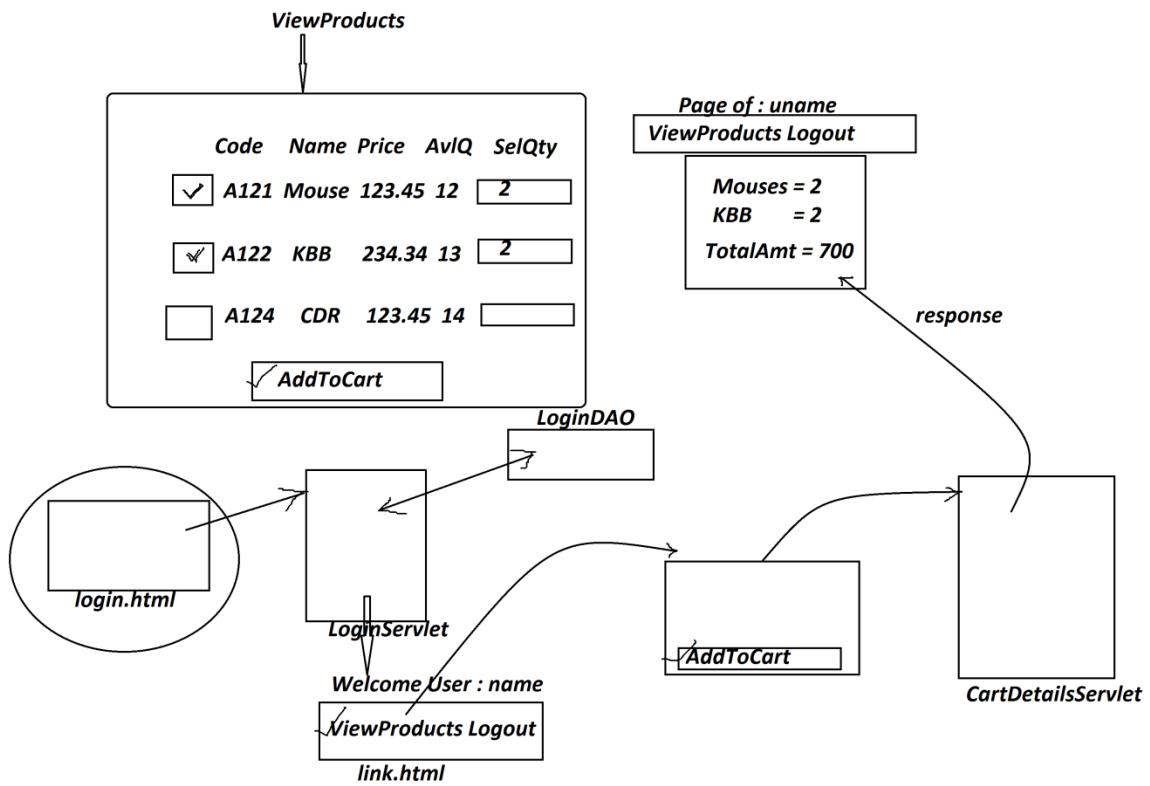
```

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

#### **Assignment:**

**Construct the Servlet Application using the following Layout:**

**Diagram:**



*faq:*

*wt is the diff b/w*

*(i)Cookie*

*(ii)HttpSession*

=>*Cookie is a Class from 'javax.servlet.http' package and used in session tracking process*

*tracking process*

=>*Cookie is browser dependent or Client dependent*

=>*HttpSession is an Interface from 'javax.servlet.http' package and used in Session Tracking process.*

=>*HttpSession is Server dependent.*

=====

*faq:*

*wt is the diff b/w*

*(i)getSession()*

*(ii)getSession(false)*

*(iii)getSession(true)*

*(i)getSession():*

*=>getSession() method will check the session is available or not*

*=>If available access the existing session,if not create new Session.*

*(ii)getSession(false):*

*=>getSession(false) method will check the session is available or not*

*=>If available access the existing session,if not new Session is not  
created.*

*(iii)getSession(true)*

*=>getSession(true) method will check the session is available or not*

*=>If available access the existing session,if not create new Session.*

=====

Dt : 2/3/2022

\*imp

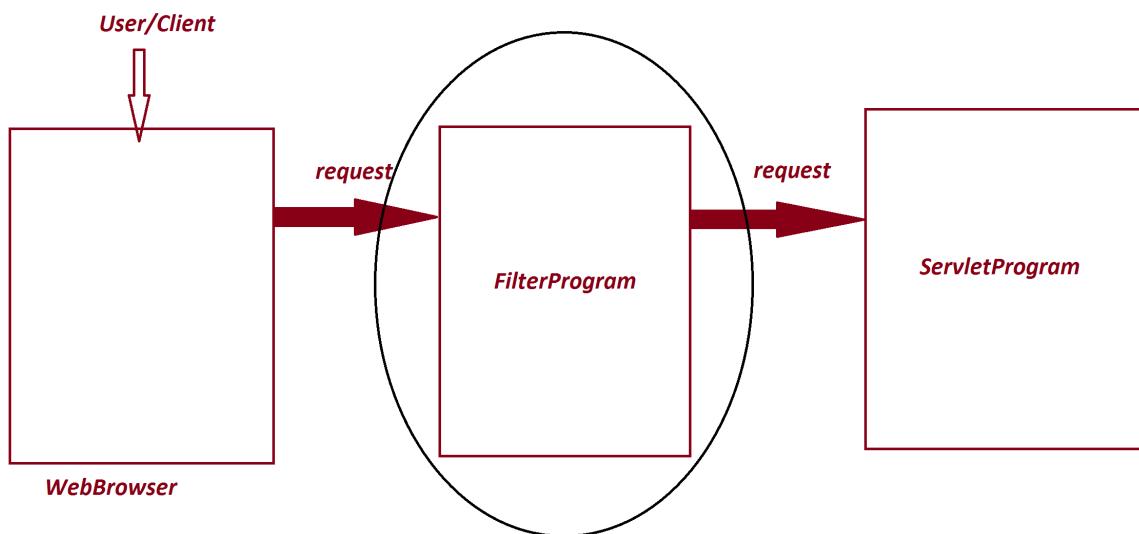
### ***Filters in Servlet Programming:***

=>***Filter is a Pre-Processing component executed before Servlet program.***

=>***Filter and Servlet are executed based on same url-pattern,in this process***

***Filter will have highest priority in execution than Servlet program.***

### ***Diagram:***



=>***The following components are used in constructing filter programs:***

- 1.Filter**
- 2.FilterChain**
- 3.FilterConfig**

### ***1.Filter:***

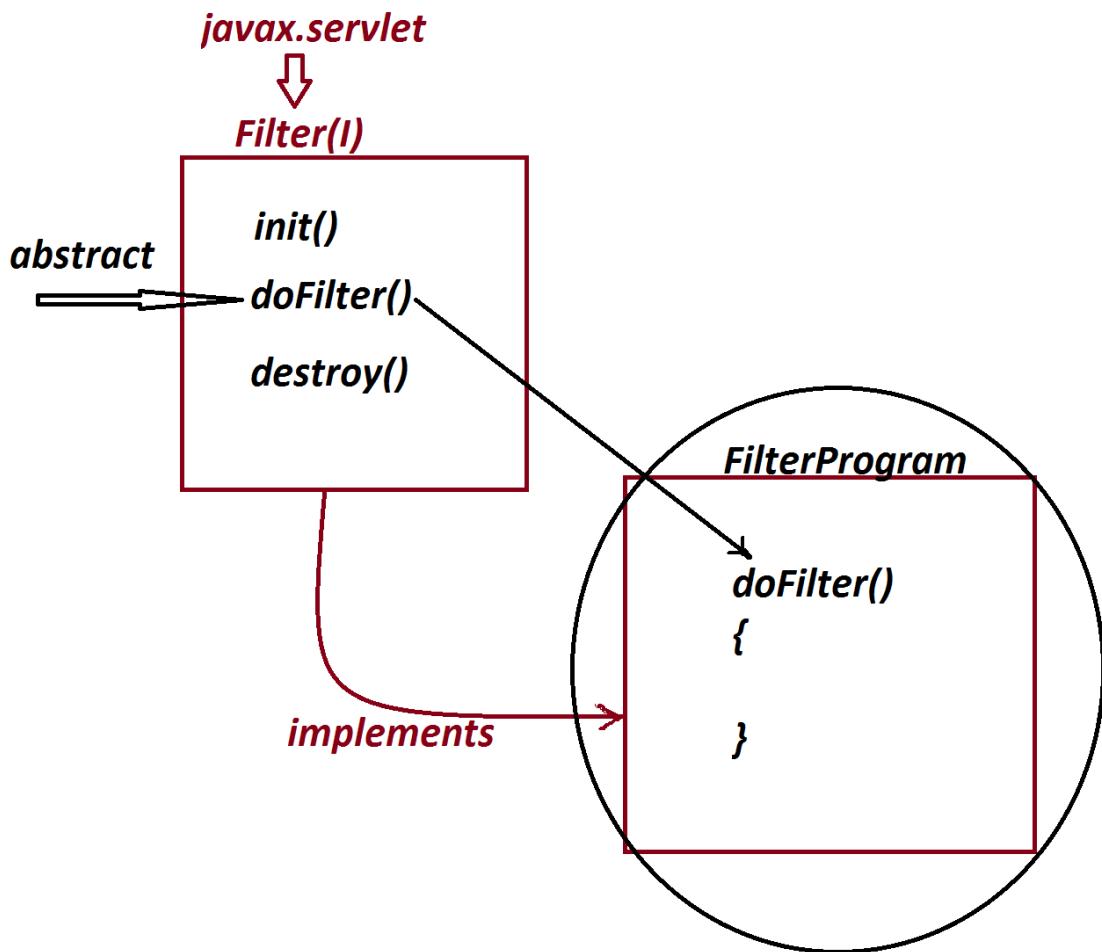
=>***'Filter' is an interface from javax.servlet package and which is***

*implemented while constructing FilterProgram.*

=>*The following are the Life-cycle methods from 'Filter' interface:*

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

*Diagram:*



## 2.FilterChain:

=>*FilterChain is an interface from javax.servlet package and which is instantiated automatically while executing doFilter() method.*

=>*The following is the method from 'FilterChain' interface:*

```

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

=>*Using doFilter() method from FilterChain we establish link to the Servlet*

*program running on Same url-patter.*

---

### **3.FilterConfig:**

*=>FilterConfig is an interface from javax.servlet package and which is instantiated automatically when the Filter program loaded on to WebContainer for execution and this FilterConfig object is loaded with Filter\_name.*

*=>we use init() method to access the reference of FilterConfig object.*

*=>we use <init-param> tag part of <filter> tag to initialize parameters to the FilterConfig object.*

**syntax:**

```
<web-app>
  .
<filter>
  <filter-name> name </filter-name>
  <filter-class> package_name.Class_name </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>
```

```
=====
```

**Ex\_Application-1:**

*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:1521:xe",
             "system", "manager");
        }catch(Exception e) {e.printStackTrace();}
    }//end of static
    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();

                //Accessing Existing Connection

            PreparedStatement ps = con.prepareStatement

                ("select * from UserReg41 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("/login")

public class LoginFilter implements Filter{

    public LoginDAO ob=null;

    @Override

    public void init(FilterConfig fc) throws ServletException{

        ob = new LoginDAO();

    }

    @Override

    public void doFilter(ServletRequest req,ServletResponse res,FilterChain chain) throws

ServletException,IOException{

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

        String fName = ob.login(req);

        if(fName==null) {

            pw.println("Invalid Login process...<br>");

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

            rd.include(req, res);

        }else {

            req.setAttribute("fname", fName); //Adding attribute to request

            chain.doFilter(req, res); //linking Servlet_program
```

```
        }

    }

}

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

    protected void doPost(HttpServletRequest req,HttpServletResponse res) throws

    ServletException,IOException{

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

        String fName = (String)req.getAttribute("fname");

        pw.println("Welcome user : "+fName);

    }

}
```

#### **web.xml**

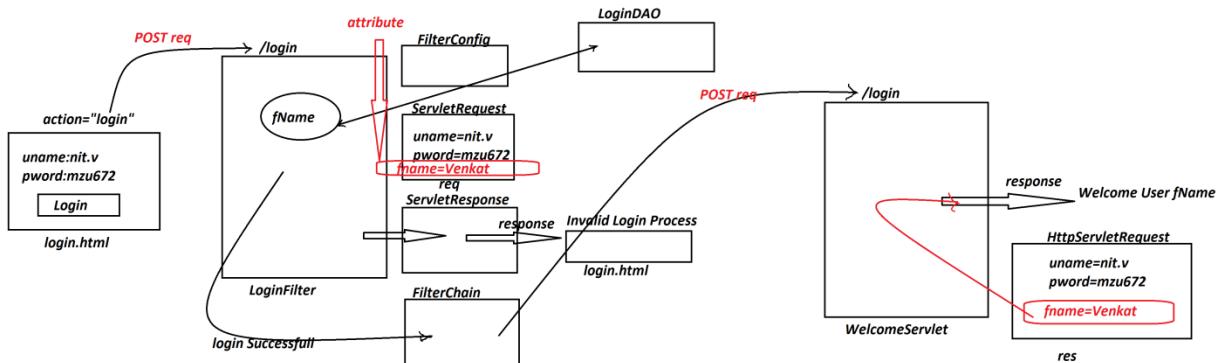
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <welcome-file-list>
        <welcome-file>login.html</welcome-file>
```

```

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

```

**Diagram:**



**Ex\_Application-2:**

*input.html*

```

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

```

*web.xml*

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
<filter>
<filter-name>DisplayFilter</filter-name>
<filter-class>test.DisplayFilter</filter-class>
<init-param>
<param-name>a</param-name>
<param-value>2000</param-value>

```

```
</init-param>
</filter>
<filter-mapping>
    <filter-name>DisplayFilter</filter-name>
    <url-pattern>/dis</url-pattern>
</filter-mapping>

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

### DisplayFilter.java

```
package test;

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

public class DisplayFilter implements Filter{

    public FilterConfig fc=null;

    public void init(FilterConfig fc) throws ServletException{
        this.fc=fc;
    }

    public void doFilter(ServletRequest req,ServletResponse res,FilterChain chain) throws
    ServletException,IOException{
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        String uName = req.getParameter("uname");
        pw.println("Welcome User "+uName+"<br>");
        pw.println("====FilterConfig====");
        pw.println("<br>FilterName:"+fc.getFilterName());
        pw.println("<br>ConfigValue:"+fc.getInitParameter("a"));
    }
}
```

}

}

---

Venkatesh Maiopathiji

Dt : 3/3/2022

faq:

wt is the diff b/w

(i)ServletConfig

(ii)FilterConfig

=>ServletConfig is instantiated automatically when the ServletProgram loaded for execution and ServletConfig object is recorded with Servlet\_name

=>FilterConfig is instantiated automatically when the FilterProgram loaded for execution and FilterConfig object is recorded with Filter\_name.

=>Every ServletProgram will have its own ServletConfig and Every FilterProgram will have its own FilterConfig.

=====

Advantage of Filters:

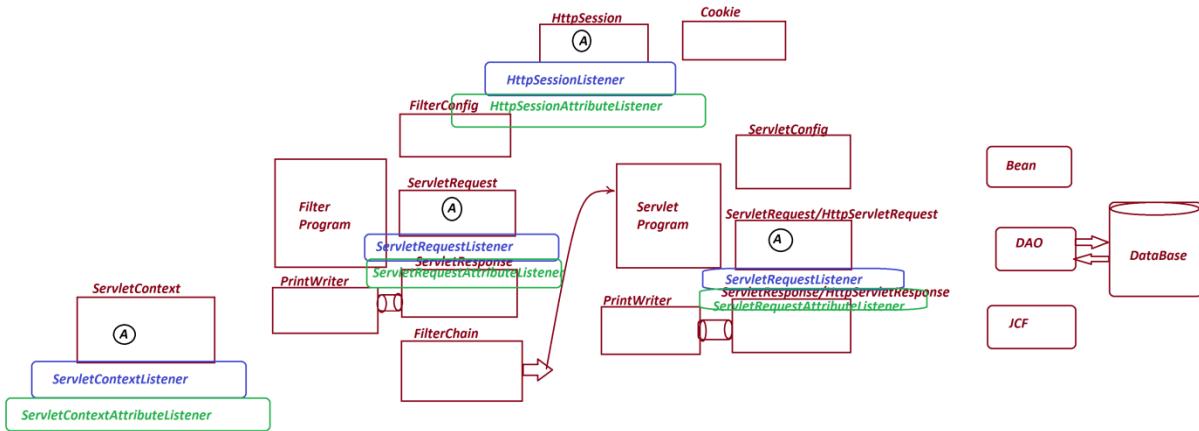
- 1.Authentication process
- 2.Validation process
- 3.Conversion process
- 4.Compression process
- 5.Network Algorithms

Note:

=>Filters are used in Security Layer.

=====

Summary of Objects generated in Servlet Programming:



\*imp

Listeners in Servlet Programmer:

=>The Component which is executed background to the objects is known as Listener.

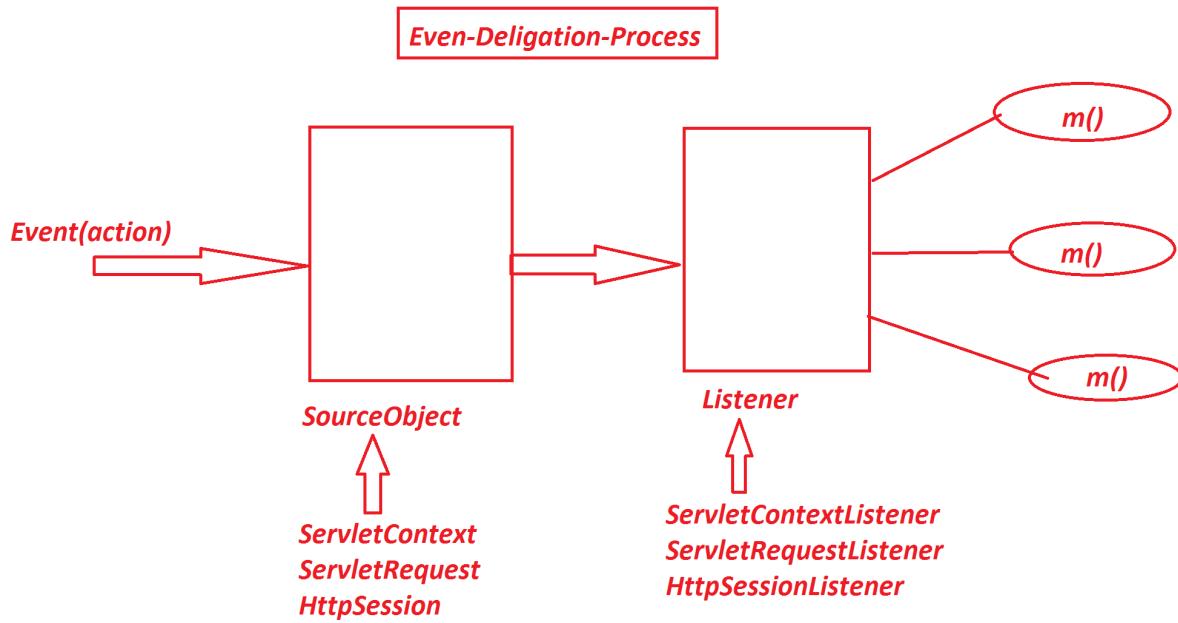
=>In Servlet programming we can add listeners to the following objects:

- (i)ServletContext
- (ii)ServletRequest
- (iii)HttpSession

define Event-Deligation-process?

=>The process in which,when the event is performed on Source object will be delegated to Listener and in this process the methods of Listener will handle the event,is known as 'Event-Deligation-Process'.

Diagram:



=>we consider the following while adding Listeners:

- (a)Source Object
- (b)Event Class(Event Category)
- (c)Listener
- (d)methods

Source Object : ServletContext

Event Class : ServletContextEvent,ServletContextAttributeEvent

Listener : ServletContextListener,ServletContextAttributeListener

methods :

```
public void contextInitialized(javax.servlet.ServletContextEvent);
```

```
public void contextDestroyed(javax.servlet.ServletContextEvent);

public void attributeAdded(javax.servlet.ServletContextAttributeEvent);
public void attributeRemoved(javax.servlet.ServletContextAttributeEvent);
public void attributeReplaced(javax.servlet.ServletContextAttributeEvent);
```

Source Object : ServletRequest

Event Class : ServletRequestEvent,ServletRequestAttributeEvent  
Listener : ServletRequestListener,ServletRequestAttributeListener  
methods :  
public void requestDestroyed(javax.servlet.ServletRequestEvent);  
public void requestInitialized(javax.servlet.ServletRequestEvent);

```
public void attributeAdded(javax.servlet.ServletRequestAttributeEvent);
public void attributeRemoved(javax.servlet.ServletRequestAttributeEvent);
public void attributeReplaced(javax.servlet.ServletRequestAttributeEvent);
```

Source Object : HttpSession

Event Class : HttpSessionEvent,HttpSessionBindingEvent  
Listener : HttpSessionListener,HttpSessionAttributeListener  
methods :  
public void sessionCreated(javax.servlet.http.HttpSessionEvent);
public void sessionDestroyed(javax.servlet.http.HttpSessionEvent);

```
public void attributeAdded(javax.servlet.http.HttpSessionBindingEvent);  
public void attributeRemoved(javax.servlet.http.HttpSessionBindingEvent);  
public void attributeReplaced(javax.servlet.http.HttpSessionBindingEvent);
```

---

Dt : 4/3/2022

Ex\_Program:

Update HttpSession project with Listeners.

SessionListener.java

```
package test;  
  
import javax.servlet.http.*;  
  
import javax.servlet.annotation.*;  
  
@WebListener  
  
public class SessionListener implements HttpSessionListener,HttpSessionAttributeListener{  
  
    @Override  
  
    public void sessionCreated(HttpSessionEvent hse) {  
  
        System.out.println("Session created...");  
  
    }  
  
    @Override  
  
    public void sessionDestroyed(HttpSessionEvent hse) {  
  
        System.out.println("Session destroyed...");  
  
    }  
  
    @Override  
  
    public void attributeAdded(HttpSessionBindingEvent hsbe) {  
  
        System.out.println("Attribute Added to Session Object...");  
  
    }
```

```
    }

    @Override

    public void attributeRemoved(HttpSessionBindingEvent hsbe) {

        System.out.println("Attribute Removed from Session Object...");

    }

}
```

### RequestListener.java

```
package test;

import javax.servlet.*;

import javax.servlet.annotation.*;

@WebListener

public class RequestListener implements ServletRequestListener{

    @Override

    public void requestInitialized(ServletRequestEvent sre) {

        System.out.println("Request Initialized...");

    }

    @Override

    public void requestDestroyed(ServletRequestEvent sre) {

        System.out.println("Request Destroyed...");

    }

}
```

### ContextListener.java

```
package test;

import javax.servlet.*;

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

```
@WebListener

public class ContextListener implements ServletContextListener{

    public void contextInitialized(ServletContextEvent sce) {

        System.out.println("ServletContext Initialized....");

    }

    public void contextDestroyed(ServletContextEvent sce) {

        System.out.println("ServletContext Destroyed...");

    }

}
```

---

Note:

=>we use the following tag related to listener in web.xml

```
<listener>

<listener-class>package_name.Listener_Class_name</listener-class>

</listener>
```

---

faq:

Types of Listeners:

=>Listeners are categorized into three types:

- 1.Application Listener
- 2.Request Listener
- 3.Session Listener

### 1.Application Listener:

=>The Listener which is added to the ServletContext is known as Application Listener.

### 2.Request Listener:

=>The Listener which is added to the ServletRequest is known as Request Listener.

### 3.Session Listener:

=>The Listener which is added to the HttpSession is known as Session Listener

=====

faq:

define Servlet Collaboration?

=>The process of establishing communication b/w servlets is known as Servlet Collaboration process.

=>This ServletCollaboration process can be done in two ways:

(a)using 'RequestDispatcher'

(b)using sendRedirect() method

(a)using 'RequestDispatcher':

=>RequestDispatcher provide the following two methods to perform Servlet Communication process

(i)forward() - used for forward communication process

(ii)include() - used for include communication process

\*imp

(b)using sendRedirect() method:

=>sendRedirect() method is used to establish communication b/w Servlet programs executing in different WebApplications.

=>sendRedirect() method is available HttpServletResponse.

Method Signature:

```
public abstract void sendRedirect(java.lang.String) throws java.io.IOException;
```

syntax:

```
res.sendRedirect("http://localhost:8082/TestApp2/...");
```

http://localhost:8082/TestApp2/second?uname=raj&mid=raj@gmail.com

WebApp : 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
    protected void doPost(HttpServletRequest req,HttpServletResponse res) throws
    ServletException,IOException{
        String uName = req.getParameter("uname");
        String mId = req.getParameter("mid");
        res.sendRedirect("http://localhost:8082/TestApp2/second?uname="+uName
                    +"&mid="+mId);
    }
}
```

```
}
```

---

```
WebApp : 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

    protected void doGet(HttpServletRequest req,HttpServletResponse res) throws

    ServletException,IOException{

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

        pw.println("====SecondServlet====");

        pw.println("<br>UserName:"+req.getParameter("uname"));

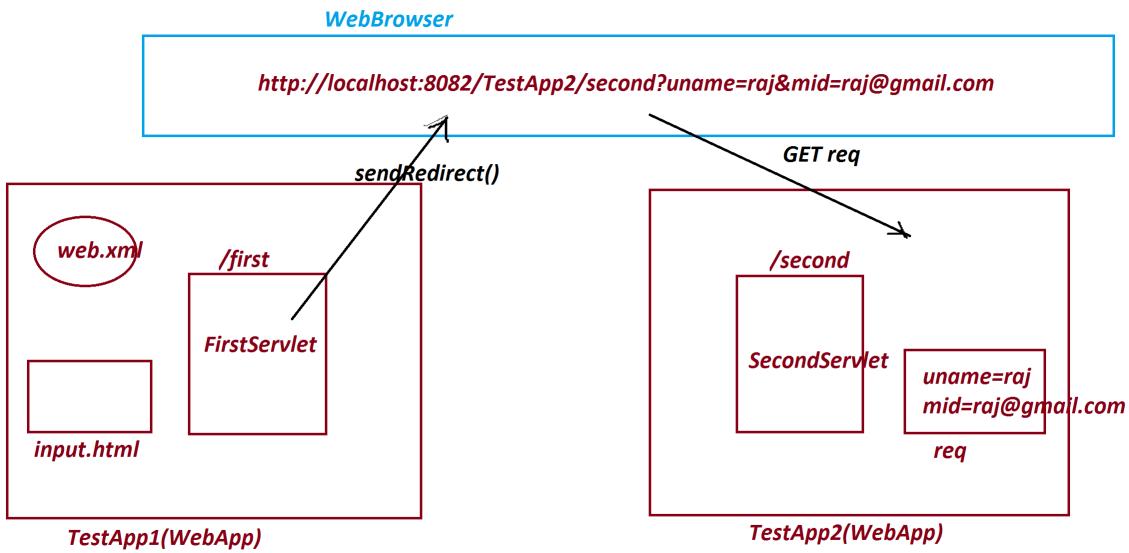
        pw.println("<br>MailId:"+req.getParameter("mid"));

    }

}
```

---

```
Diagram:
```



faq:

wt is the diff b/w

(i)RequestDispatcher

(ii)sendRedirect()

(i)RequestDispatcher:

=>In RequestDispatcher same request is forwarded from one Servlet program to another Servlet progra.

=>RequestDispatcher is used to establish communication b/w Servlet programs of Same WebApp.

(ii)sendRedirect():

=>In `sendRedirect()` method separate new request is generated for Second Servlet program

=>`sendRedirect()` method is used to establish communication b/w Servlets programs

of different WebApplications.

=====

Assignment:

FirstServlet will perform login process,

=>If Login is Successfull display complete user details using SecondServlet.

=>else,Invalid login process.

=====

Dt : 7/3/2022

**define Annotation?**

=>The tag based information which is added to the programming components like Class, Interface, method and Variable is known as 'Annotation'.

=>we use '@' symbol to represent annotations.

=>Annotations will provide information to Compiler at compilation stage or information to execution control at execution stage.

=>The following are some important annotations in CoreJava:

(a)@Override

(b)@SuppressWarnings

**(a)@Override:**

=>'@Override' annotation will give information to compiler at compilation stage to check the method is Overriding method or not.

**(b)@SuppressWarnings:**

=>'@SuppressWarnings' annotation will give information to compiler at compilation stage to close the raised warnings.

---

=>The following are some important annotations in AdvJava:

(a)@WebServlet

(b)@WebFilter

(c)@WebListener

(a)@WebServlet:

=>'@WebServlet' annotation will hold 'servlet-url-pattern' and supports Execution control to identify the Servlet program for execution

(b)@WebFilter:

=>'@WebFilter' annotation will hold 'filter-url-pattern' and supports Execution control to identify the Filter program for execution

(c)@WebListener:

=>'@WebListener' annotation will support Execution control to execute Listeners.

---

---

\*imp

JSP Programming:(Part-3)

=>JSP Stands for 'Java Server Page' and which is response from WebApplication.

=>JSP is tag based programming language and which is more easy when compared to Servlet programming.

=>Programs in JSP are saved with (.jsp) as an extention.

=>JSP programs are combination of both HTML code and Java Code.

=>JSP provides the following tags to write JavaCode part of JSP programs:

1.Scripting tags

=>Scriptlet tag

=>Expression tag

=>Declarative tag

## 2.Directive tags

=>page

=>include

=>taglib

## 3.Action tags

=>jsp:include

=>jsp:forward

=>jsp:param

=>jsp:useBean

=>jsp:setProperty

=>jsp:getProperty

## 1.Scripting tags:

=>Scripting tags are used to write JavaCode part of JSP programs.

=>Scripting tags are categorized into the following:

(a)Scriptlet tag

(b)Expression tag

(c)Declarative tag

### (a)Scriptlet tag:

=>Scriptlet tag is used to write normal JavaCode part of JSP programs

**syntax:**

`<% ---JavaCode--- %>`

**(b)Expression tag:**

=>Expression tag is used to assign the value to variable or which is used to display the data to the WebBrowser.

**syntax:**

`<%= expression %>`

**(c)Declarative tag:**

=>Declarative tag is used to declare variables and methods in JSP programs.

**syntax:**

`<%! variables;methods %>`

---

**2.Directive tags:**

=>The tags which are used to specify the directions in translation process are known as Directive Tags.

The following are the types of Directive tags:

- (a)page
- (b)include
- (c>taglib

**(a)page:**

=>'page' directive tag specifies the translator to add the related attribute to current JSP page.

**syntax:**

`<%@ page attribute="value" %>`

**exp:**

`<%@ page import="java.util.*"%>`

**List of attributes:**

- 1.import
- 2.contentType
- 3.extends
- 4.info
- 5.buffer
- 6.language
- 7.isELIgnored
- 8.isThreadSafe

**9.autoFlush**

**10.session**

**11.pageEncoding**

**12.errorPage**

**13.isErrorPage**

---

**(b)include:**

=>'include' directive tag specifies the file to be included to current JSP page.

**syntax:**

<%@ include file="file-name"%>

---

**Exp:**

<%@ include file="input.html" %>

---

**(c>taglib:**

=>'taglib' directive tag specifies to add specified url to current JSP page and which is used part of EL(Expression Lang) and JSTL (JSP Standard Tag Lib).

**syntax:**

<%@ taglib url="urloftaglib" prefix="prefixoftaglib"%>

---

### Ex\_Application:

JSP application to read number and display the factorial of given number?

#### 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="v"><br>
<input type="submit" value="Factorial">
</form>
</body>
</html>
```

#### Display.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
int fact;
int factorial(int n)
{
    fact=1;
    for(int i=n;i>=1;i--)
    {
        fact=fact*i;
    }
    return fact;
}
%>
<%
```

```
int val = Integer.parseInt(request.getParameter("v"));
int result = factorial(val);
out.println("Factorial : "+result+"<br>");

%>
<%@ include file="input.html"%>
</body>
</html>
```

web.xml

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

---

Dt : 8/3/2022

**Ex\_Application-2:(With Exception Handling process)**

**JSP application to read number and display the factorial of given number?**

*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="v"><br>
<input type="submit" value="Factorial">
</form>
</body>
</html>
```

---

*Display.jsp*

```
<%@ page language="java"
contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"
errorPage="Error.jsp"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%!
int fact;
int factorial(int n)
{
    fact=1;
    for(int i=n;i>=1;i--)
    {
        fact=fact*i;
    }
    return fact;
}
```

```
%>
<%
int val = Integer.parseInt(request.getParameter("v"));
int result = factorial(val);
out.println("Factorial : "+result+"<br>");
%>
<%@ include file="input.html"%>
</body>
</html>
```

---

#### *web.xml*

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

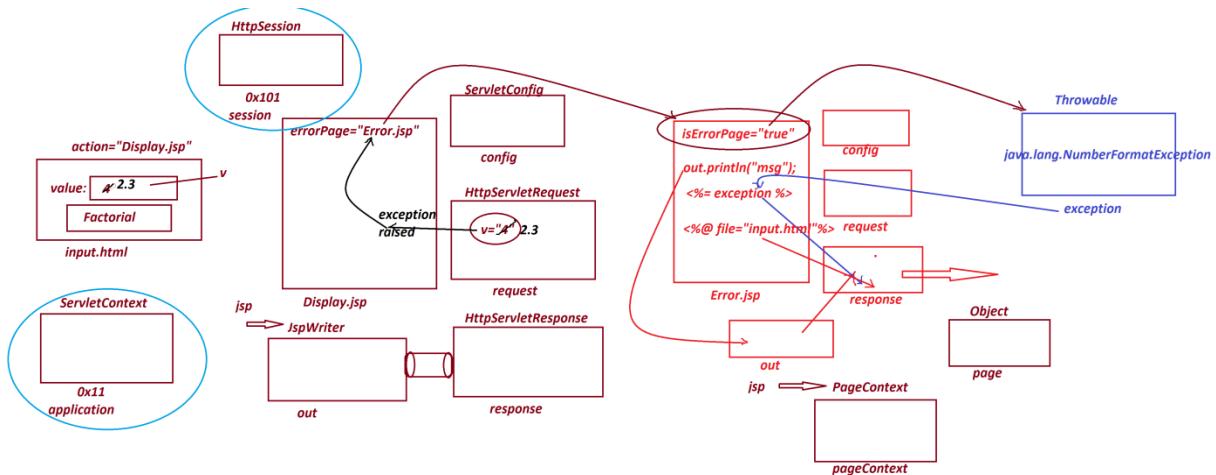
---

#### *Error.jsp*

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    isErrorPage="true"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <%
    out.println("Enter Only Integer values...<br>"); 
    %>
    <%= exception%>
    <br>
    <%@include file="input.html"%>
</body>
</html>
```

---

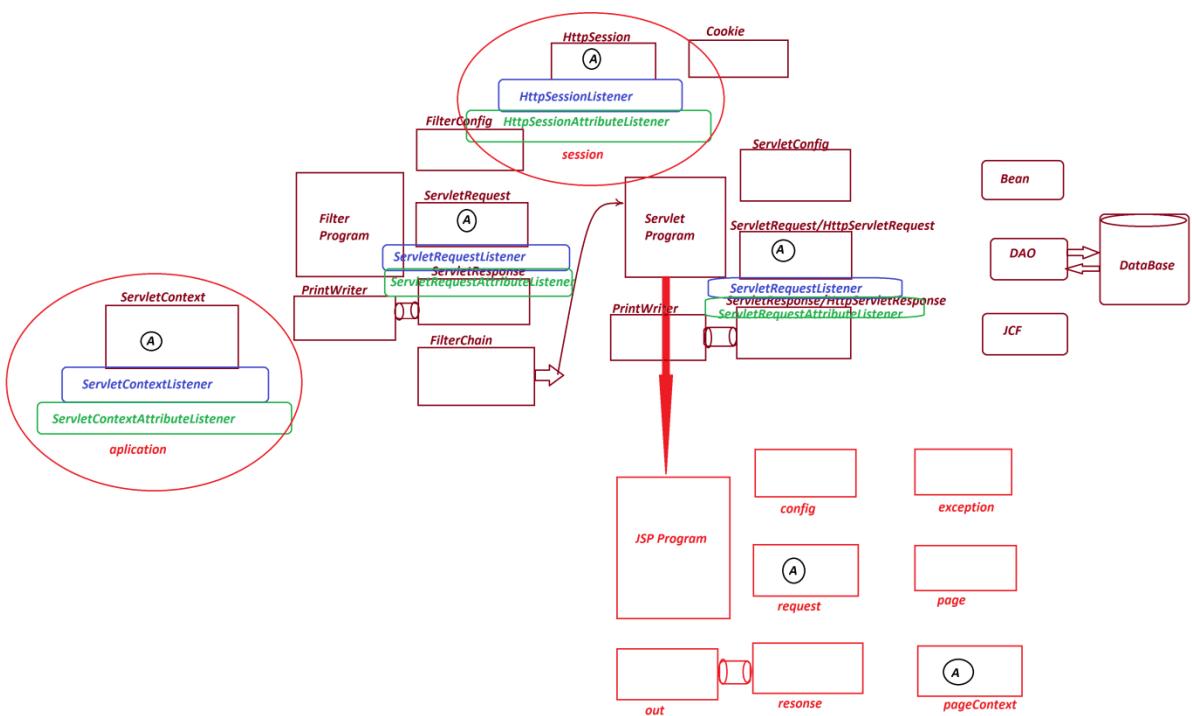
*Diagram:*



=>The following are the implicit objects generated in JSP programming:

- 1.application** - `javax.servlet.ServletContext`
- 2.config** - `javax.servlet.ServletConfig`
- 3.request** - `javax.servlet.http.HttpServletRequest`
- 4.response** - `javax.servlet.http.HttpServletResponse`
- 5.out** - `javax.servlet.jsp.JspWriter`
- 6.session** - `javax.servlet.http.HttpSession`
- 7.exception** - `java.lang.Throwable`
- 8.page** - `java.lang.Object`
- 9.pageContext** - `javax.servlet.jsp.PageContext`

*Diagram of Objects Layout:*



### 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" flush="true/false"/>
```

```
<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"/>
```

=====

*Ex\_Program : Demonstarting <jsp:forward>,<jsp:include> and <jsp:param>*  
*input.html*

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="Choice.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>
```

*Choice.jsp*

```
<%@ page language="java"
contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String s1 = request.getParameter("s1");
```

```
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"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
int v1 = Integer.parseInt(request.getParameter("v1"));
int v2 = Integer.parseInt(request.getParameter("v2"));
int v3 = v1+v2;
out.println("Sum: "+v3+"<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"%>
<!DOCTYPE html>
<html>
```

```
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
int v1 = Integer.parseInt(request.getParameter("v1"));
int v2 = Integer.parseInt(request.getParameter("v2"));
int v3 = v1-v2;
out.println("Sub: "+v3+"<br>");
%>
<jsp:include page="input.html"/>
</body>
</html>
```

---

#### Error.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    isErrorPage="true"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
out.println("Enter Only Integer Values...<br>");
%>
<%= exception%>
<br>
<jsp:include page="input.html"/>
</body>
</html>
```

---

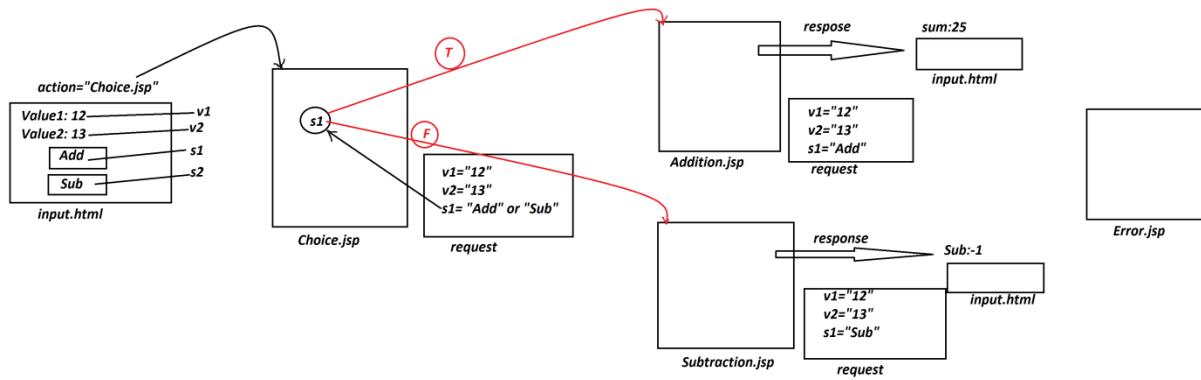
#### web.xml

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

</web-app>

---

**Diagram:**



===== Venkatesh Ma' =====

Dt : 9/3/2022

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 from the bean by using the bean's getter method and writes the value to the current JspWriter.

**Types of attributes:**

(a)name

(b)property

**(a)name:**

=>The name attribute takes the reference variable name on which we want to invoke the getter method.

**(b)property:**

=>which gets the value of a bean property and invokes the getter method of the bean property.

---

**The following are some rare used Actions tags:**

7.<jsp:plugin>:

The <jsp:plugin> action tag provide easy support for including a java applet in the client Web browser, using a built-in or downloaded java plug-in.

**Syntax:**

<jsp:plugin attributes>

```
<!-optionally one jsp:params or jsp:fallback tag-  
</jsp:plugin>
```

#### 8. <jsp:fallBack >

*The <jsp:fallback> action tag allows us to specify a text message to be displayed if the required plug-in cannot run and this action tag must be used as a child tag with the <jsp:plugin> action tag.*

*Syntax:*

```
<jsp:fallback>  
Test message that has to be displayed if the plugin cannot be started  
</jsp:fallback>
```

#### 9. <jsp:params >

*The <jsp:params> action tag sends the parameters that we want to pass to an applet.*

*Syntax:*

```
<jsp:params>  
  <!-one or more jsp:param tags---  
</jsp:params>
```

---

*Ex\_Application:*

*input.html*

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="Retrieve.jsp" method="post">
Enter the ProdCode:<input type="text" name="PCODE"><br>
<input type="submit" value="Retrieve">
</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:1521:xe",
                 "system", "manager");
        }catch(Exception e) {e.printStackTrace();}
    }
    public static Connection getCon() {
        return con;
    }
}
```

---

#### *RetrieveDAO.java*

```
package test;

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

public class RetrieveDAO {
```

```
public ResultSet rs=null;

public ResultSet retrieve(HttpServletRequest req) {

    try {

        Connection con = DBConnection.getCon();

        PreparedStatement ps = con.prepareStatement

            ("select * from Product41 where pcode=?");

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

        rs = ps.executeQuery();

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

    return rs;
}

}
```

---

#### ProductBean.java

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class ProductBean implements Serializable{
    private String pCode,pName;
    private float pPrice;
    private int pQty;
    public ProductBean() {}
    public String getpCode() {
        return pCode;
    }
    public void setpCode(String pCode) {
        this.pCode = pCode;
    }
    public String getpName() {
        return pName;
    }
    public void setpName(String pName) {
        this.pName = pName;
    }
}
```

```

public float getpPrice() {
    return pPrice;
}
public void setpPrice(float pPrice) {
    this.pPrice = pPrice;
}
public int getpQty() {
    return pQty;
}
public void setpQty(int pQty) {
    this.pQty = pQty;
}
}

```

---

### Retrieve.jsp

```

<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.* , java.sql.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
try{
    ResultSet rs = new RetrieveDAO().retrieve(request);
    if(rs.next()){
        %>
        <jsp:useBean id="pb" class="test.ProductBean"
scope="session"/>
        <jsp:setProperty property="pCode"
value="<%=">rs.getString(1) %>" name="pb"/>
        <jsp:setProperty property="pName"
value="<%=">rs.getString(2) %>" name="pb"/>
        <jsp:setProperty property="pPrice"
value="<%=">rs.getFloat(3) %>" name="pb"/>
        <jsp:setProperty property="pQty"
value="<%=">rs.getInt(4) %>" name="pb"/>

        <a href="Display.jsp">ViewProductDetails</a>
    <%

```

```
        }else{
            out.println("Invalid ProdCode...<br>");
        %>
        <jsp:include page="input.html"/>
        <%
    }
}catch(Exception e){e.printStackTrace();}

%>
</body>
</html>
```

---

### Display.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<jsp:useBean id="pb" type="test.ProductBean" scope="session"/>
ProductCode:<jsp:getProperty property="pCode" name="pb"/><br>
ProductName:<jsp:getProperty property="pName" name="pb"/><br>
ProductPrice:<jsp:getProperty property="pPrice"
name="pb"/><br>
ProductQty:<jsp:getProperty property="pQty" name="pb"/><br>
</body>
</html>
```

---

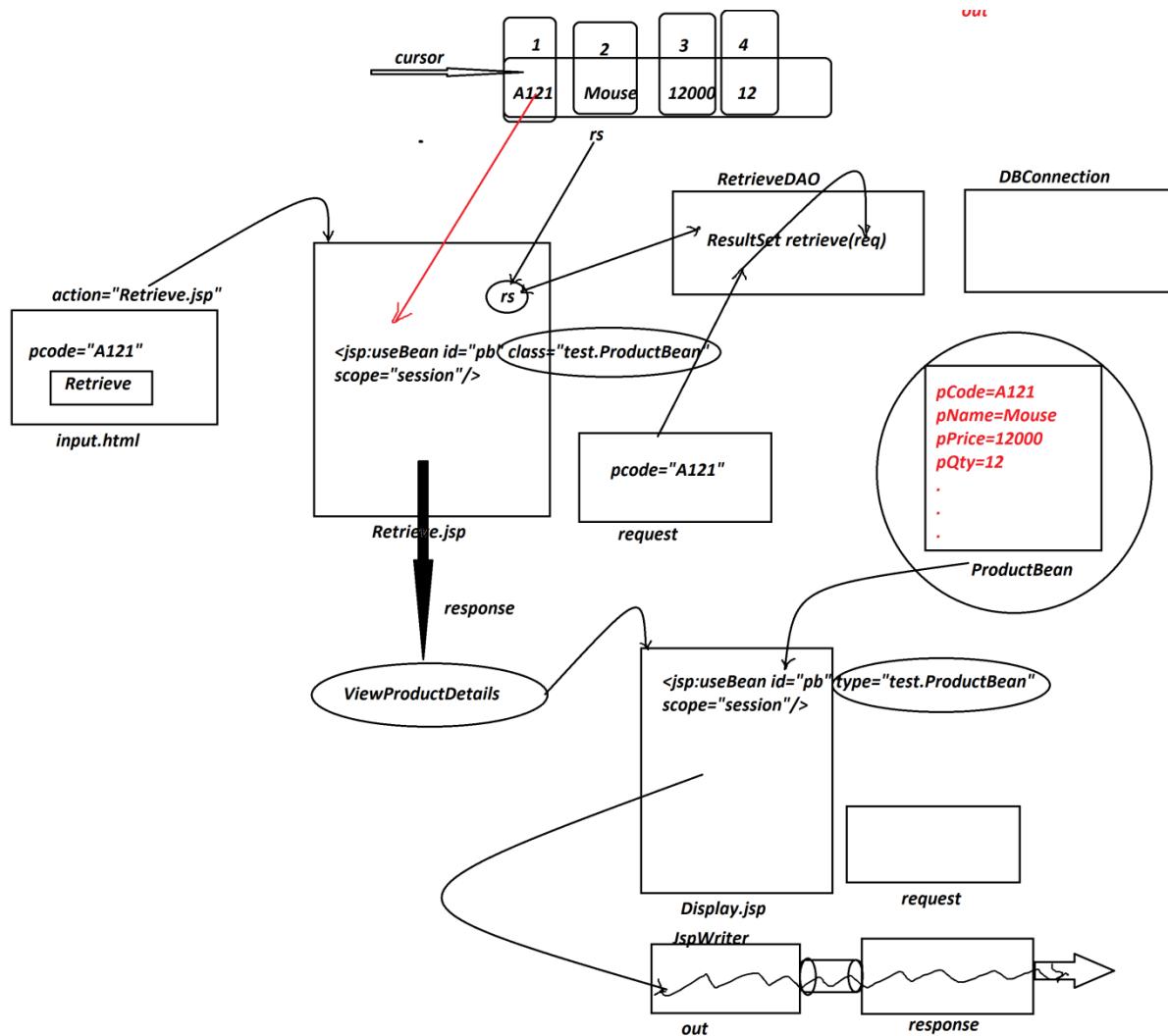
### web.xml

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

---

Dt : 10/3/2022

**Diagram:**



faq:

**define JSP Life-Cycle?**

=>JSP Life-Cycle demonstrates different states of JSP program from starting to ending of JSP program.

=>The following are the stages of JSP Life-Cycle:

(a) Translation process

*(b) Compilation Process*

*(c) Loading process*

*(d) Instantiation process*

*(e) Initialization process*

*(f) Request Handling process*

*(g) Destroying process*

*(a) Translation process:*

=>*The process of separating the JavaCode(ServletCode) from JSP programs is known as*

*Translation process.*

*(b) Compilation Process:*

=>*The process of compiling the Source code generated from Translation process is known as Compilation process.*

*Note:*

=>*After Compilation process the source code is destroyed.*

*(c) Loading process:*

=>*The process of loading JSP program for execution is known as Loading process.*

*(d) Instantiation process:*

=>*When the JSP program is loaded then it is automatically instantiated known as Instantiation process.*

=>After instantiation process we can find the following Life-Cycle methods:

- (i) `jspInit()`
- (ii) `jspService()`
- (iii) `jspDestroy()`

(e) Initialization process:

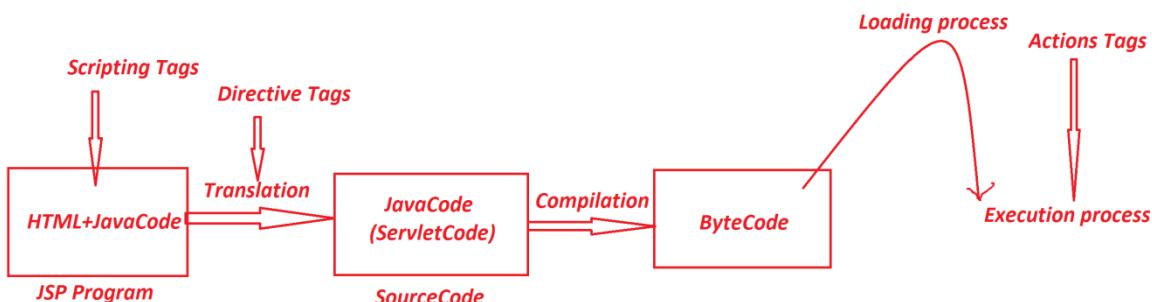
=>The process of making the programming components ready for execution is known as **Initialization process**.

(f) Request Handling process:

=>The process of accepting the request and providing the response is known as '**Request Handling Process**'.

(g) Destroying process:

=>The process of destroying the JSP instances is known as **destroying process**.



=====

=====

*\*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.***

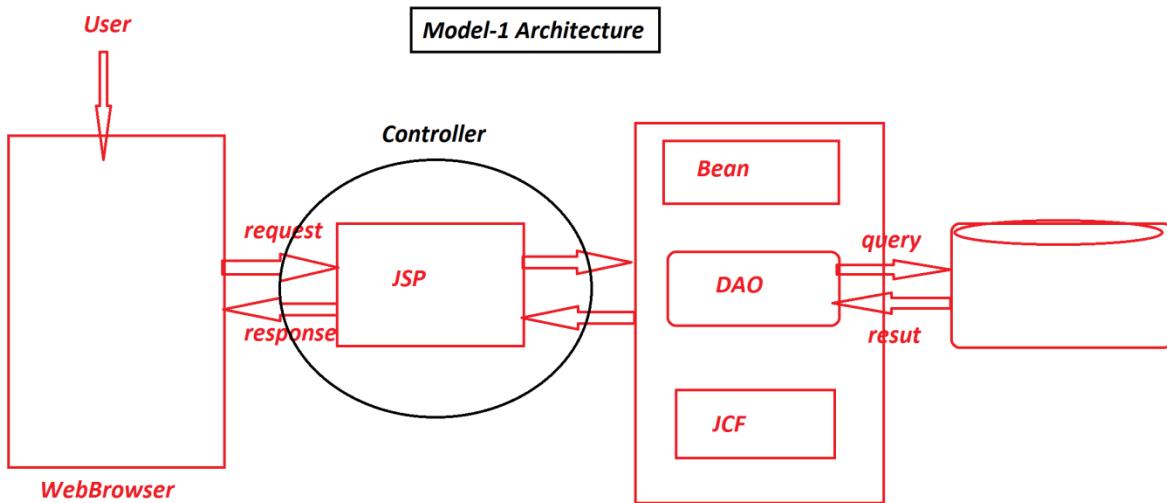
***These models are:***

- 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.***

***Diagram:***



#### ***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.***

*Exp:*

*JSP\_App1*

*JSP\_App2*

*JSP\_App3*

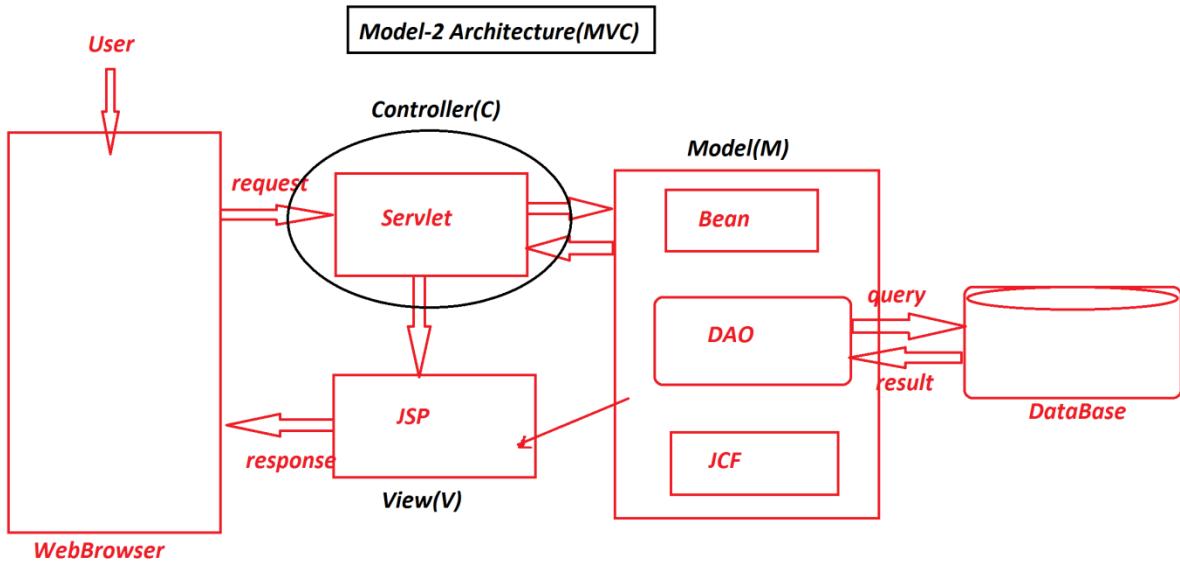
---

**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.*

*Diagram:*



=>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.*

---

-----

Venkatesh Maiopathiji

Dt : 11/3/2022

**JSP\_MVC\_Application-1:**

**DBConnection.java**

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

---

**LoginDAO.java**

```
package test;
import java.sql.*;
import javax.servlet.http.*;
public class LoginDAO {
    public UserBean ub=null;
    public UserBean login(HttpServletRequest req) {
        try {
            Connection con = DBConnection.getCon();
            //Accessing Existing Connection
            PreparedStatement ps = con.prepareStatement
```

```
("select * from UserReg41 where uname=? and pword=?");  
ps.setString(1,req.getParameter("uname"));  
ps.setString(2,req.getParameter("pword"));  
ResultSet rs = ps.executeQuery();  
if(rs.next())  
{  
    ub = new UserBean();//Creating Bean Object  
    ub.setuName(rs.getString(1));  
    ub.setpWord(rs.getString(2));  
    ub.setfName(rs.getString(3));  
    ub.setlName(rs.getString(4));  
    ub.setAddr(rs.getString(5));  
    ub.setmId(rs.getString(6));  
    ub.setPhNo(rs.getLong(7));  
}  
}catch(Exception e) {e.printStackTrace();}  
return ub;  
}  
}
```

---

### *LoginServlet.java*

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

```
import javax.servlet.http.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/login")
public class LoginServlet extends HttpServlet{
    @Override
    protected void doPost(HttpServletRequest req,
                          HttpServletResponse res) throws ServletException, IOException{
        UserBean ub = new LoginDAO().login(req);
        if(ub==null) {
            req.setAttribute("msg", "Invalid Login Process..<br>");
            RequestDispatcher rd = req.getRequestDispatcher("LoginFail.jsp");
            rd.forward(req, res);
        }else {
            HttpSession hs = req.getSession();
            hs.setAttribute("fname", ub.getfName());
            hs.setAttribute("userbean", ub);
            RequestDispatcher rd = req.getRequestDispatcher("LoginSuccess.jsp");
            rd.forward(req, res);
        }
    }
}
```

---

*LogoutServlet.java*

```
package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/logout")

public class LogoutServlet extends HttpServlet{

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

        HttpSession hs = req.getSession(false);

        if(hs==null) {

            req.setAttribute("msg","Session Expired...<br>");

        }else {

            hs.invalidate();

            req.setAttribute("msg","User Loggedout Successfully...<br>");

        }

        RequestDispatcher rd = req.getRequestDispatcher("LoginFail.jsp");

        rd.forward(req, res);

    }

}
```

---

#### UserBean.java

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

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

```
}
```

### *ViewProfileServlet.java*

```
package test;

import java.io.*;

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

@SuppressWarnings("serial")

@WebServlet("/view")

public class ViewProfileServlet extends HttpServlet{

    protected void doGet(HttpServletRequest req,
                         HttpServletResponse res) throws ServletException, IOException{
        HttpSession hs = req.getSession(false);
        if(hs==null) {
            req.setAttribute("msg","Session Expired...<br>");
            RequestDispatcher rd = req.getRequestDispatcher("LoginFail.jsp");
            rd.forward(req, res);
        }else {
            RequestDispatcher rd = req.getRequestDispatcher("ViewProfile.jsp");
            rd.forward(req, res);
        }
    }
}
```

---

*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>
```

---

*LoginFail.jsp*

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
    String msg = (String)request.getAttribute("msg");
    out.println(msg);
%>
<jsp:include page="login.html"/>
</body>
</html>
```

---

*LoginSuccess.jsp*

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
```

```
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
    String fName =(String)session.getAttribute("fname");
    out.println("Welcome user : "+fName+"<br>");
%>
<a href="view">ViewProfile</a>
<a href="logout">Logout</a>
</body>
</html>
```

---

#### ViewProfile.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.UserBean"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
    UserBean ub = (UserBean)session.getAttribute("userbean");
    out.println("Page of User : "+ub.getfName()+"<br>");
    out.println("====UserDetails====<br>");
    out.println(ub.getfName()+"&nbsp&nbsp"+ub.getlName()+"&nbsp&nbsp"
    +
    ub.getAddr()+"&nbsp&nbsp"+ub.getId()+"&nbsp&nbsp"+ub.getPhNo()+
    "<br>");%>
<a href="view">ViewProfile</a>
<a href="logout">Logout</a>
</body>
</html>
```

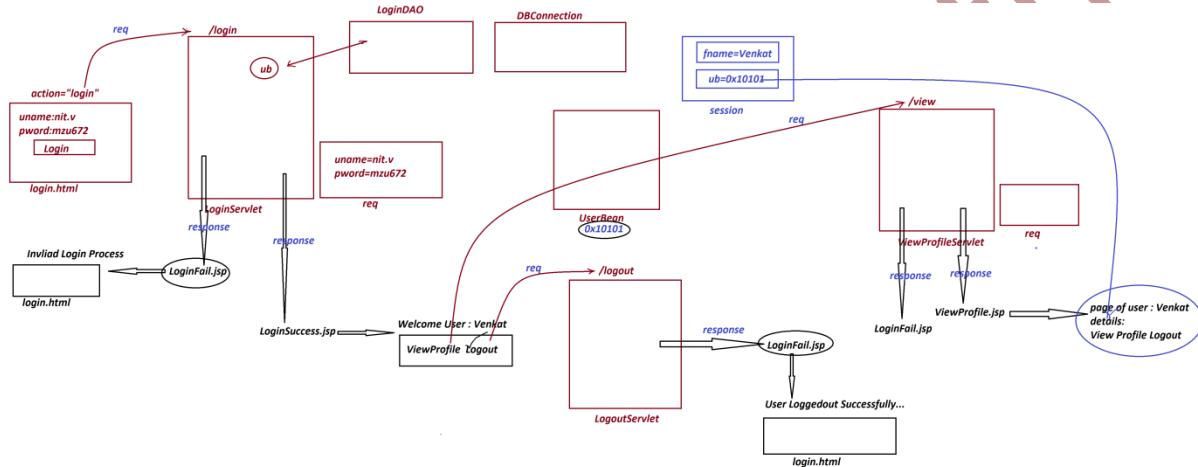
---

## *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>
```

---

## *Diagram:*



## *Assignment:*

*Update the application with Update profile*

*(address,mailId,phno)*

---

Dt : 12/3/2022

JSP\_MVC\_App-2:

DB Tables :

UserTab41(uname,pword,fname,lname,addr,mid,phno)

AdminTab41(uname,pword,fname,lname,addr,mid,phno)

Book41(bcode,bname,bauthor,bprice,bqty)

ServletPrograms :

AddBookServlet.java

package test;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@SuppressWarnings("serial")

@WebServlet("/add")

public class AddBookServlet extends HttpServlet{

    @Override

    protected void doPost(HttpServletRequest req,

                HttpServletResponse res) throws ServletException,

        IOException{

            HttpSession hs = req.getSession(false);

            if(hs==null) {

                req.setAttribute("msg","Session Expired...");

```
req.getRequestDispatcher("Fail.jsp").forward(req, res);

}else {

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

    if(k>0) {

        req.getRequestDispatcher("AddBook.jsp").forward(req, res);

    }

}

}

-----
```

#### *AdminLoginServlet.java*

```
package test;

import java.io.*;

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

@SuppressWarnings("serial")
@WebServlet("/log2")

public class AdminLoginServlet extends HttpServlet{

    @Override

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

    String fName = new LoginDAO().login(req);
```

```
if(fName==null) {  
    req.setAttribute("msg","Invalid Login Process...");  
    req.getRequestDispatcher("Fail.jsp").forward(req, res);  
}  
else {  
    HttpSession hs = req.getSession();  
    hs.setAttribute("fname",fName);  
    req.getRequestDispatcher("Admin.jsp").forward(req, res);  
}  
}  
}  
-----
```

#### *ChoiceServlet.java*

```
package test;  
  
import java.io.*;  
  
import javax.servlet.*;  
  
import javax.servlet.http.*;  
  
import javax.servlet.annotation.*;  
  
@SuppressWarnings("serial")  
@WebServlet("/choice")  
public class ChoiceServlet extends HttpServlet{  
  
    @Override  
  
    protected void doPost(HttpServletRequest req,  
                          HttpServletResponse res) throws ServletException,  
                                         IOException{
```

```
String s1 = req.getParameter("s1");

req.getServletContext().setAttribute("s1", s1);

req.getRequestDispatcher("Choice.jsp").forward(req, res);

}

}
```

---

#### *DeleteBookServlet.java*

```
package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.annotation.*;

import javax.servlet.http.*;

@SuppressWarnings("serial")

@WebServlet("/delete")

public class DeleteBookServlet extends HttpServlet{

    @Override

    protected void doPost(HttpServletRequest req,

                          HttpServletResponse res) throws ServletException,

IOException{

        HttpSession hs = req.getSession(false);

        if(hs==null) {

            req.setAttribute("msg", "Session Expired....");

            req.getRequestDispatcher("Fail.jsp").forward(req, res);

        }else {
```

```
        int k = new DeleteDAO().delete(req);

        if(k>0) {

            req.getRequestDispatcher("Delete.jsp").forward(req, res);

        }

    }

}

-----
```

#### *LogoutServlet.java*

```
package test;

import java.io.*;

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

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

public class LogoutServlet extends HttpServlet{

    @Override
    protected void doGet(HttpServletRequest req,
                         HttpServletResponse res) throws ServletException,
                                         IOException{
        HttpSession hs = req.getSession(false);
        if(hs==null) {
            req.setAttribute("msg","Session expired... ");
        }
    }
}
```

```
        }else {
            hs.invalidate();
            req.setAttribute("msg","LoggedOut Successfully..");
        }//End of else

        RequestDispatcher rd =
            req.getRequestDispatcher("Fail.jsp");
        rd.forward(req, res);
    }
}
```

---

```
UserLoginServlet.java

package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/log1")
public class UserLoginServlet extends HttpServlet{
    @Override
    protected void doPost(HttpServletRequest req,
        HttpServletResponse res) throws ServletException,
        IOException{
        String fName = new LoginDAO().login(req);
```

```
if(fName==null) {  
    req.setAttribute("msg","Invalid Login Process...");  
    req.getRequestDispatcher("Fail.jsp").forward(req, res);  
}  
else {  
    HttpSession hs = req.getSession();  
    hs.setAttribute("fname",fName);  
    req.getRequestDispatcher("User.jsp").forward(req, res);  
}  
}  
}  
-----
```

#### *ViewBooksServlet.java*

```
package test;  
  
import java.io.*;  
  
import java.util.*;  
  
import javax.servlet.*;  
  
import javax.servlet.http.*;  
  
import javax.servlet.annotation.*;  
  
@SuppressWarnings("serial")  
@WebServlet("/view")  
  
public class ViewBooksServlet extends HttpServlet{  
  
    protected void doGet(HttpServletRequest req,  
                         HttpServletResponse res) throws ServletException,  
                                         IOException{
```

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

if(hs==null) {

    req.setAttribute("msg","Session Expired...");

    req.getRequestDispatcher("Fail.jsp").forward(req, res);

} else {

    ArrayList<BookBean> al = new RetrieveDAO().retrieve();

    hs.setAttribute("al",al);

    req.getRequestDispatcher("View.jsp").forward(req, res);

}

}
```

---

DAO :

*DBConnection.java*

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

---

*DeleteDAO.java*

```
package test;

import java.sql.*;

import javax.servlet.http.*;

public class DeleteDAO {

    public int k=0;

    public int delete(HttpServletRequest req) {
        try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps=con.prepareStatement
                ("delete from Book38 where bcode=?");
            ps.setString(1,req.getParameter("bcode"));
            k = ps.executeUpdate();
        }catch(Exception e) {e.printStackTrace();}
        return k;
    }
}
```

---

*InsertDAO.java*

```
package test;

import javax.servlet.http.*;

import java.sql.*;

public class InsertDAO {

    public int k=0;
```

```
public int insert(HttpServletRequest req) {  
    try {  
        Connection con = DBConnection.getCon();  
        PreparedStatement ps = con.prepareStatement  
                ("insert into Book38 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;  
}  
}
```

---

```
-----  
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();

String s1= (String)req.getServletContext().getAttribute("s1");

PreparedStatement ps=null;

if(s1.equals("User")) {

    ps = con.prepareStatement ("select * from UserTab38 where uname=? and pword=?");

} else {

    ps = con.prepareStatement ("select * from AdminTab38 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;

}

-----
```

### RetrieveDAO.java

```
package test;
```

```
import java.util.*;
import java.sql.*;
public class RetrieveDAO {
    public ArrayList<BookBean> al = new ArrayList<BookBean>();
    public ArrayList<BookBean> retrieve(){
        try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                ("select * from Book38");
            ResultSet rs = ps.executeQuery();
            while(rs.next()) {
                BookBean bb = new BookBean();
                bb.setbCode(rs.getString(1));
                bb.setbName(rs.getString(2));
                bb.setbAuthor(rs.getString(3));
                bb.setbPrice(rs.getFloat(4));
                bb.setbQty(rs.getInt(5));
                al.add(bb); //Adding BookBean to ArrayList
            }
        } catch(Exception e) {e.printStackTrace();}
        return al;
    }
}
```

---

*Beans:*

*BookBean.java*

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class BookBean implements Serializable{
    private String bCode,bName,bAuthor;
    private float bPrice;
    private int bQty;
    public BookBean() {}
    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;
    }
}
```

---

*JSP :*

### AddBook.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String fName = (String)session.getAttribute("fname");
out.println("Page of "+fName+"<br>");
%>
<jsp:include page="link2.html"/>
<%
out.println("<br>Book Added Successfully... ");
%>
</body>
</html>
```

---

### Admin.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String fName = (String)session.getAttribute("fname");
out.println("Welcome Admin "+fName+"<br>");
%>
<jsp:include page="link2.html"/>
</body>
</html>
```

---

### Choice.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String s1 = (String) application.getAttribute("s1");
if(s1.equals("User")){
    %>
    <jsp:forward page="UserLogin.html"/>
    <%
} else{
    %>
    <jsp:forward page="AdminLogin.html"/>
    <%
}
%>
</body>
</html>
```

---

#### Delete.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String fName = (String) session.getAttribute("fname");
out.println("Page of "+fName+"<br>");
%>
<jsp:include page="link2.html"/>
<%
out.println("<br>Book Details deleted Successfully... ");
%>
```

```
</body>
</html>
```

---

### *Fail.jsp*

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String msg=(String)request.getAttribute("msg");
out.println(msg+"<br>");
String s1 = (String)application.getAttribute("s1");
if(s1.equals("User")){
    %>
    <jsp:include page="UserLogin.html"/>
    <%
}else{
    %>
    <jsp:include page="AdminLogin.html"></jsp:include>
    <%
}
%>
</body>
</html>
```

---

### *User.jsp*

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
```

```
<%
String fName = (String)session.getAttribute("fname");
out.println("Welcome User "+fName+"<br>");
%>
<jsp:include page="link1.html"/>
</body>
</html>
```

---

### View.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="java.util.* , test.BookBean"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String fName = (String)session.getAttribute("fname");
ArrayList<BookBean> al = (ArrayList<BookBean>)

session.getAttribute("al");
out.println("Page of "+fName+"<br>");
String s1 = (String)application.getAttribute("s1");
if(s1.equals("User")){
    %>
    <jsp:include page="link1.html"/>
    <%
} else{
    %>
    <jsp:include page="link2.html"/>
    <%
}
%>
<br>
<%
Iterator<BookBean> it = al.iterator();
while(it.hasNext()){
    BookBean bb = (BookBean)it.next();
    out.println(bb.getbCode()+"&nbsp&nbsp"+bb.getbName()+
        "&nbsp&nbsp"+bb.getbAuthor()+"&nbsp&nbsp"+
        "&nbsp&nbsp"+bb.getbPrice());
}
```

```
bb.getbPrice () + "nbspnbsp"+bb.getbQty ()) ;  
}  
%>  
</body>  
</html>
```

---

HTML :

*AdminLogin.html*

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="ISO-8859-1">  
<title>Insert title here</title>  
</head>  
<body>  
    <form action="log2" method="post">  
        UserName:<input type="text" name="uname"><br>  
        Password:<input type="password" name="pword"><br>  
        <input type="submit" value="Login">  
    </form>  
</body>  
</html>
```

---

*book.html*

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="ISO-8859-1">  
<title>Insert title here</title>  
</head>  
<body>  
    <form action="add" 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="AddBook">  
    </form>  
</body>  
</html>
```

---

*code.html*

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

---

*home.html*

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="choice" method="post">
<input type="submit" value="User" name="s1">
<input type="submit" value="Admin" name="s1">
</form>
</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="code.html">DeleteBook</a>
<a href="logout">Logout</a>
</body>
</html>
```

---

### *UserLogin.html*

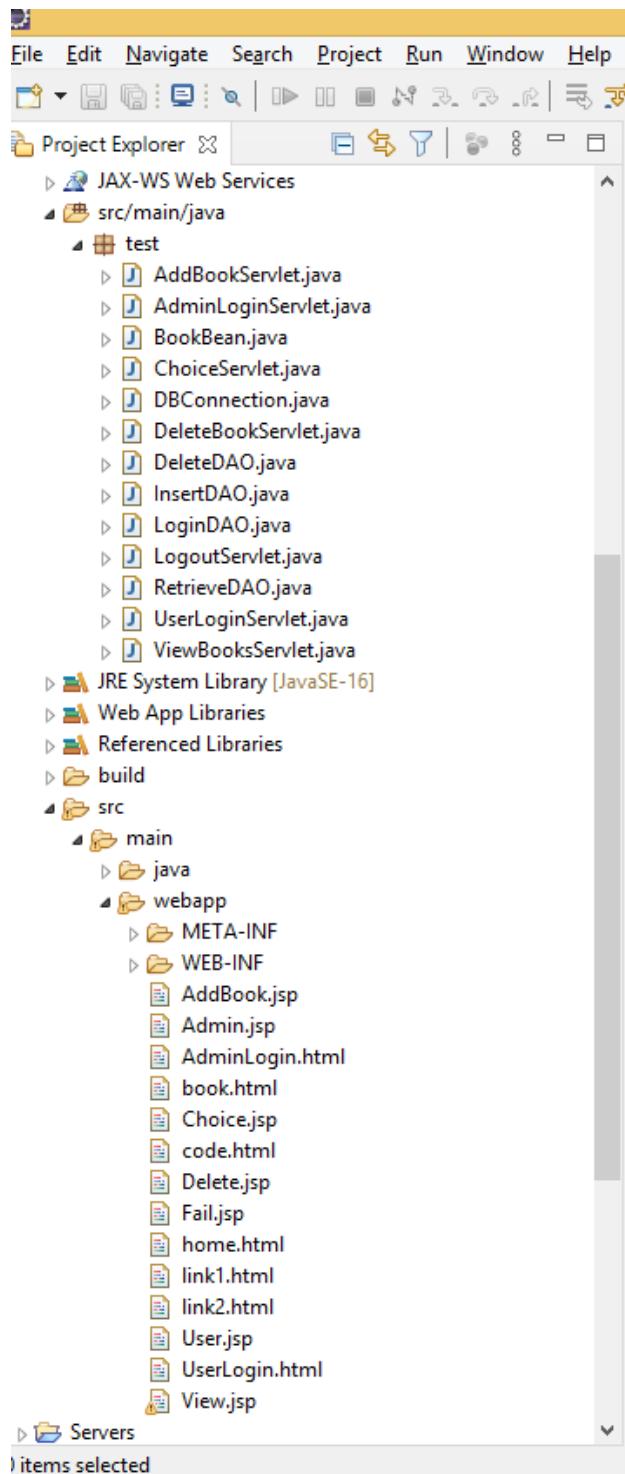
```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="log1" method="post">
UserName:<input type="text" name="uname"><br>
PassWord:<input type="password" name="pword"><br>
<input type="submit" value="Login">
<a href="">NewUser</a>
</form>
</body>
</html>
```

---

### *XML:*

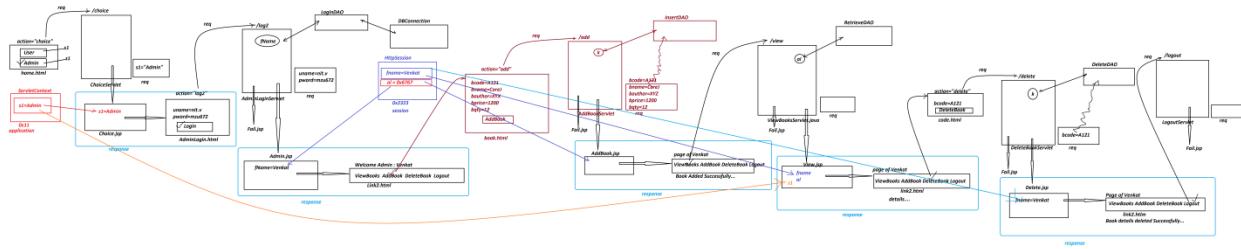
#### *web.xml*

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



Maipathii

Venkatesh Maipathiji



Dt : 14/3/2022

faq:

wt it the diff b/w

(i)JAR

(ii)WAR

(iii)EAR

(i)JAR:

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

(ii)WAR:

=>WAP means 'Web Archeive' and which is compressed format of HTML files,XML file,

UI-Files,Servlets,JSP and Jars.

=>WebApplications are converted into WAR files.

(iii)EAR:

=>EAR means 'Enterprise Archeive' and which compressed format of Enterprise Java Beans,  
WARs and JARs.

=>Enterprise Applications are converted into EAR files.

=====

\*imp

**Generating WAR file and executing from Tomcat Server:**

**step-1 : Generating WAR file using IDE Eclipse**

**RightClick on WebApplication->Export->WAR file->Browse the destination folder to store  
WAR file->name the file and click 'save'->click 'finish'**

**step-2 : start the Tomcat server**

**step-3 : Open WebBrowser and access the Server(Tomcat)**

**step-4 : Deploy the WAR file into Tomcat**

**Click on Manager App->perform login process->From 'WAR file to deploy' click 'choose file'->  
browse and select the file->click 'deploy'**

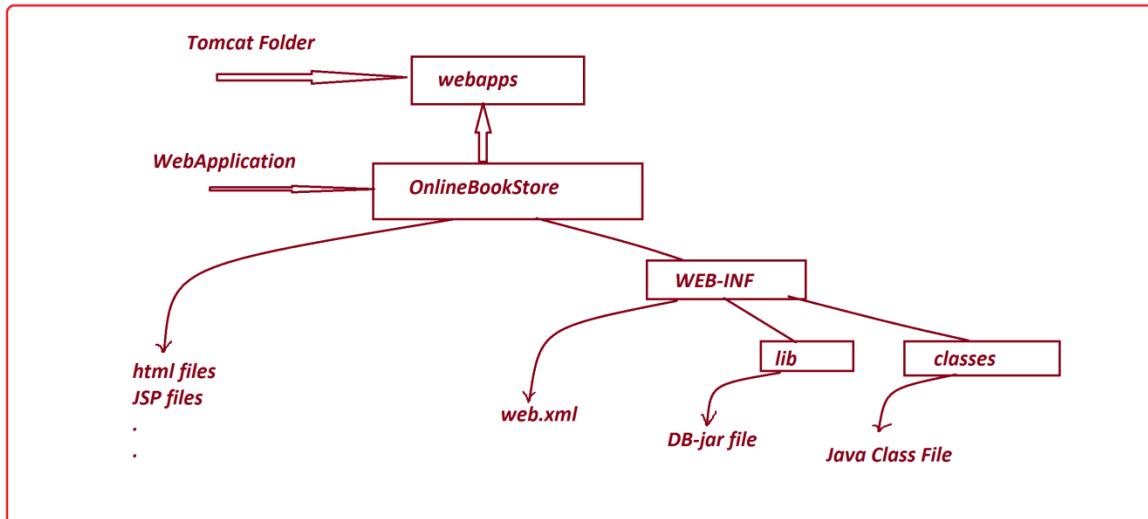
**step-5 : Execute the application as follows**

**<http://localhost:8082/OnlineBookStore>**

**Note:**

**=>The WebApplications are deployed into 'webapps' folder of 'Tomcat' server.**

**=>The following is the Deployment directory Structure of Tomcat:**



=

Dt : 15/3/2022

**Expression Language(EL) in JSP:**

=>This EL simplifies the accessibility of data stored in java Bean

component and other objects like request,session,application,etc..

**Note:**

*It is newly added feature in JSP technology.*

**syntax of EL:**

***\$(expression)***

***The following are the implicit objects of EL:***

- 1.pageScope**
- 2.requestScope**
- 3.sessionScope**
- 4.applicationScope**
- 5.param**
- 6.paramValues**
- 7.header**
- 8.headerValues**
- 9.cookie**
- 10.initParam**
- 11.pageContext**

**1.pageScope : It maps the given attribute name with the value set in the page scope**

**2.requestScope : It maps the given attribute name with the value set in the request scope**

**3.sessionScope : It maps the given attribute name with the value set in the session scope**

**4.applicationScope : It maps the given attribute name with the value set in the application scope**

**5.param** : It maps the request parameter to the single value

**6.paramValues** :

*It maps the request parameters to the array of values*

**7.header** : It maps the request header name to the single value

**8.headerValues** : It maps the request header names to the array of values

**9.cookie** : It maps the cookie name to the cookie value

**10.initParam** : It maps the Initialization parameter

**11.pageContext** : It provides access to many objects request,session,...

---

*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*

*(v)Jagged 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*

*(iii)LinkedList<E> Object*

**6. Map<K,V> Objects**

*(i)HashMap<K,V> Object*

*(ii)LinkedHashMap<K,V> Object*

*(iii)TreeMap<K,V> Object*

*(iv)Hashtable<K,V> Object*

**7. Enum<E> objects**

---

**Utility Classes:**

*(i)java.util.Scanner*

*(ii)java.util.StringTokenizer*

*(iii)java.util.StringJoiner*

---

**JDBC Objects:**

**1.Connection Object**

**2.Statement Object**

**3.PreparedStatement Object**

**4.CallableStatement object**

**5.Scrollable ResultSet Object**

**6.Non-Scrollable ResultSet Object**

**7.DatabaseMetaData object**

**8. ParameterMetaData object**

**9. ResultSetMetaData object**

**10. RowSet Object**

**11. RowSetMetaData**

-----

**12. Connection pooling object (Vector<E> object)**

=====

**Servlet Objects:**

**1. ServletContext object**

**2. ServletConfig Object**

**3. ServletRequest object / HttpServletRequest object**

**4. ServletResponse object / HttpServletResponse object**

**5. PrintWriter object**

**6. HttpSession object**

**7. Cookie object**

**8. DAO Layer object**

**9. JavaBean object**

**10. JCF Object**

**JSP objects:**

**1. application**

**2. config**

**3. request**

**4. response**

- 5.out*
  - 6.session*
  - 7.exception*
  - 8.page*
  - 9.pageContext*
- 

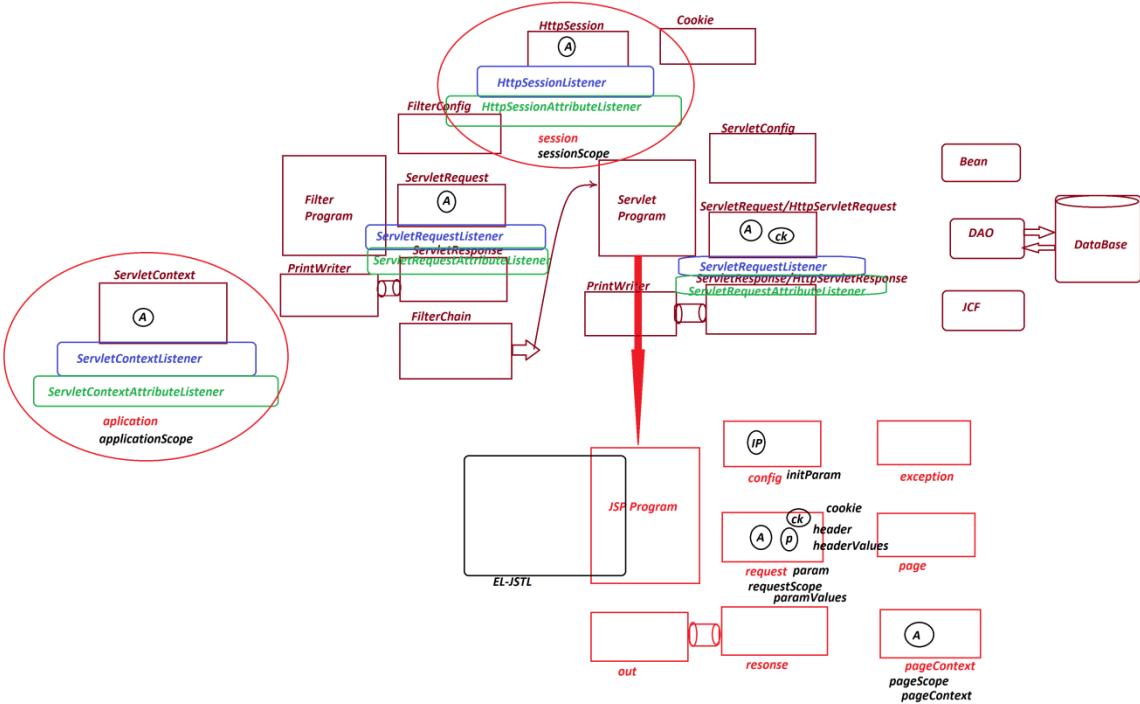
*JSP EL objects:*

- 1.pageScope*
- 2.sessionScope*
- 3.requestScope*
- 4.applicationScope*
- 5.param*
- 6.paramValues*
- 7.cookie*
- 8.pageContext*
- 9.header*
- 10.headerValues*
- 11.initParam*

*Note:*

=>In realtime JSPEL replaces with SpEL.

*Objects Layout diagram:*



### ***JSTL(JSP Standard Tag Lib):***

=>This JSTL represents a set of tags to simplify the JSP development.

#### ***Advantages of JSTL:***

- (i)Fast Development**
- (ii)Code Reusability**
- (iii)No need to use scriptlet tag**

***The following are the JSTL tags:***

- (1)Core Tags**
- (2)Function Tags**
- (3)Formatting Tags**
- (4)XML Tags**

## (5)SQL Tags

**Note:**

=>we use "taglib" directive tag to declare JSTL Tags.

=>To Execute JSTL Tags we use the following steps:

(i)Download the following Jar files to execute JSTL tags

*javax.servlet.jsp.jstl-1.2.1.jar*

*javax.servlet.jsp.jstl-api-1.2.1.jar*

(ii)These jars must be available within 'lib' folder of

'WEB-INF' in deployment directory

---

(1)Core Tags:

=>These JSTL core tags provides variable support,URL management, flow control etc. (Basic code writing)

**syntax:**

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>
```

The following are the List of JSTL Core Tags:

c:out ->It displays the result of an expression, similar to

`<%=...%> tag.`

**c:import->It Retrieves relative or an absolute URL.**

**c:set-> It sets the value to variable.**

**c:remove->It is used for removing the variable .**

**c:catch-> It is used for Catching any Throwabe exception that occurs  
in the body.**

**c:if-> It is an conditional tag**

**c:choose, c:when, c:otherwise->**

**It is the simple conditional tag that includes its body content  
if the evaluated condition is true.**

**\*imp**

**c:forEach-> It is the basic iteration tag.**

**c:forTokens-> It iterates over tokens which is separated by the  
supplied delimeters.**

**c:param-> It adds a parameter in a containing 'import' tag's URL.**

**c:redirect-> It redirects the browser to a new URL and supports the  
context-relative URLs.**

**c:url-> It creates a URL with optional query parameters.**

---

**(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.***

---

### **(3)Formatting Tags:**

**=>These formatting tags provide support for message formatting, number formating and date formatting etc.**

**=>These formatting tags are also used for internationalized web sites to display and format text,time,date and numbers.**

**syntax:**

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt"
    prefix="fmt" %>
```

**List of some Formatting tags:**

***fmt:parseNumber : It is used to Parse the string representation of a currency, percentage or number.***

***fmt:formatNumber : It is used to format the numerical value with specific format or precision.***

***fmt:parseDate : It parses the string representation of a time and date.***

***fmt:setTimeZone : It stores the time zone inside a time zone configuration variable.***

**\*imp**

***fmt:formatDate : It formats the time and date using the supplied***

*pattern and styles.*

---

#### **(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.*

---

*Exp\_Program:*

*input.html*

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

```
<title>Insert title here</title>
</head>
<body>
    <form action="CoreTags.jsp" method="post">
        Enter the String<input type="text" name="str"><br>
        <input type="submit" value="Display">
    </form>
</body>
</html>
```

#### CoreTags.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<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>
</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>
```

---