Advanced Java : AJ67

Dt : 26/9/2024(Day-1)

Java Module:

1.CoreJava

2.AdvJava

3.Spring(SpringCore,SpringMVC,SpringJDBC,SpringSecurity)

4.SpringBoot

5.MicroServices

Job Role :

(i)Full Stack Java Developer

(ii)Java Developer

Summary of CoreJava:

1.Java Programming Components(Java Alphabets)

2.Java Programming Concepts

3.Object Oriented Programming features

1.Java Programming Components(Java Alphabets)

(a)Variables

(b)Methods

(c)Constructors

(d)Blocks

(e)Classes

(f)Interfaces

(g)AbstractClasses

2.Java Programming Concepts

(a)Object Oriented Programming

(b)Exception Handling Process.

(c)Java Collection Framework(JCF)

(Data Structure Components)

(d)Multi Threading process

(e)IO Streams and Files

(f)Networking in Java

3.Object Oriented Programming features

(a)Class

(b)Object

(c)Abstraction

(d)Encapsulation

(e)PolyMorphism

(f)Inheritance

Note:

=>Using CoreJava Components and Concepts,we can develop Stand-Alone Applications.

faq:

define Stand-Alone Applications?

=>The application which is installed in one computer and performs actions in the same computer is

known as Stand-Alone Application or DeskTop Application or Windows Application.

=>Non-Server Applications are known as Stand-Alone Applications

==========

**imp*

AdvJava:

=>AdvJava will provide the following technologies to develop Server based Applications,which means Web Applications

1.JDBC

2.Servlet

3.JSP

1.JDBC:

=>JDBC stands for 'Java DataBase Connectivity' and which is used to interact with Database product.

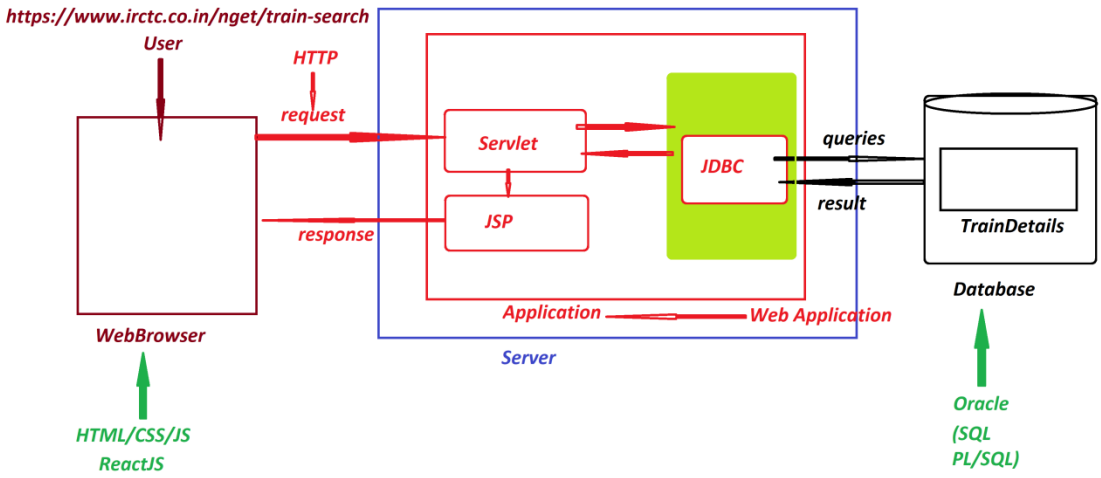
2.Servlet:

=>Servlet means Server-program and which accepts the request from User(WebBrowser) and provide the response.

3.JSP:

=>JSP stands for 'Java Server Page' and which is response from Web Application.

Diagram:



==========

faq:

define Storage?

=>The memory location where the data is available for access is known Storage.

Types of Storages:

=>According to Java Application development,the Storages are categorized into four types:

1.Field Storage

2.Object Storage

3.File Storage

4.Database Storage

1.Field Storage:

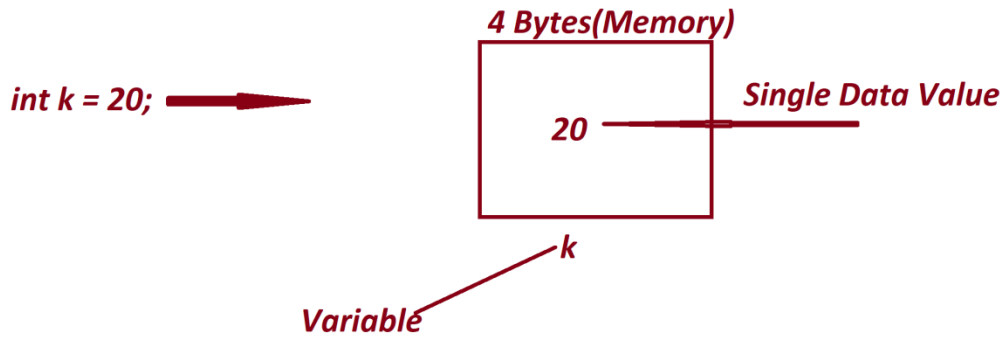
=>The memory generated to hold 'single data value' is known as Field Storage.

=>when we use Primitive datatypes(byte,short,int,long,float,double,char,boolean) in the program,will

generate Field Storages.

Ex:

int k = 20;



**imp*

2.Object Storage:

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

=>when we use NonPrimitive datatypes(Class,Interface,Array,Enum) in the program,will generate Object Storage.

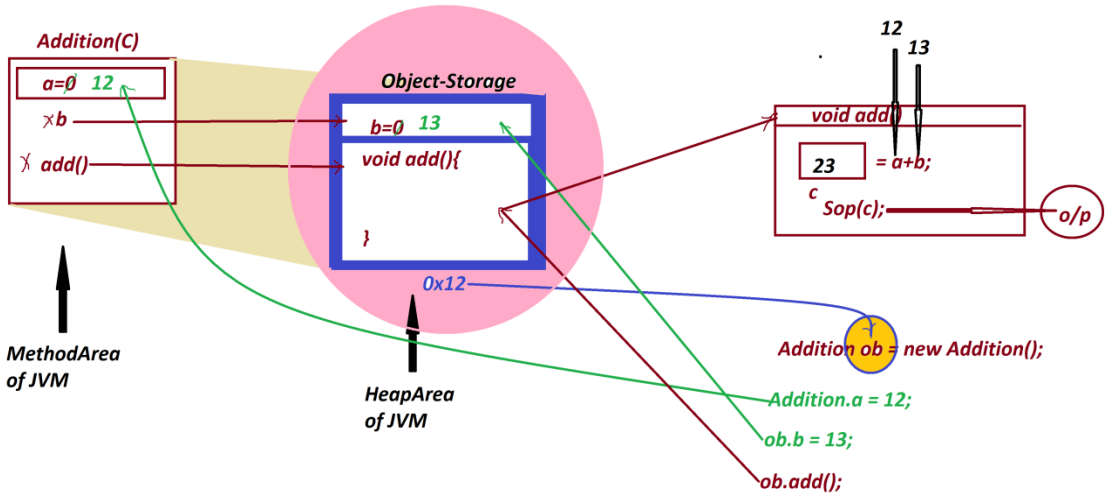
Ex:

```
class Addition
{
  static int a;
       int b;
  void add()
```

```
  {
     int c = a+b;
     System.out.println("Sum:"+c);
  }
}
```

Addition ob = new Addition();

Diagram:



****imp***

List of Objects Generated from CoreJava:

1.User defined Class Objects

2.String-Objects

(a)String Class Objects

(b)StringBuffer Class Objects

(c)StringBuilder Class Objects

3.WrapperClass Objects

(a)Byte Object

(b)Short Object

(c)Integer Object

(d)Long Object

(e)Float Object

(f)Double Object

(g)Character Object

(h)Boolean Object

4.Array Objects

(a)Array holding User defined class Objects

(b)Array holding String Objects

(c)Array holding WrapperClass Objects

(d)Array holding DisSimiler Objects(Object Array)

(e)Array holding Array Objects(Jagged Array)

5.Collection<E> Objects

1.List<E> Objects

(a)ArrayList<E> Object

(b)LinkedList<E> Object

(c)Vector<E> Object

=>Stack<E> Object

2.Queue<E> Objects

(a)PriorityQueue<E> Object

=>Deque<E>

(b)ArrayDeque<E> Object

(c)LinkedList<E> Object

3.Set<E> Objects

(a)HashSet<E> Object

(b)LinkedHashSet<E> Object

(c)TreeSet<E> Object

6.Map<K,V> Objects

(a)HashMap<K,V> Object

(b)LinkedHashMap<K,V> Object

(c)TreeMap<K,V> Objects

7.Enum<E> Objects

==========

Dt : 27/9/2024(Day-2)

Note:

=>The 'Field Storages' and 'Object Storages',which are generated part of JVN while program execution

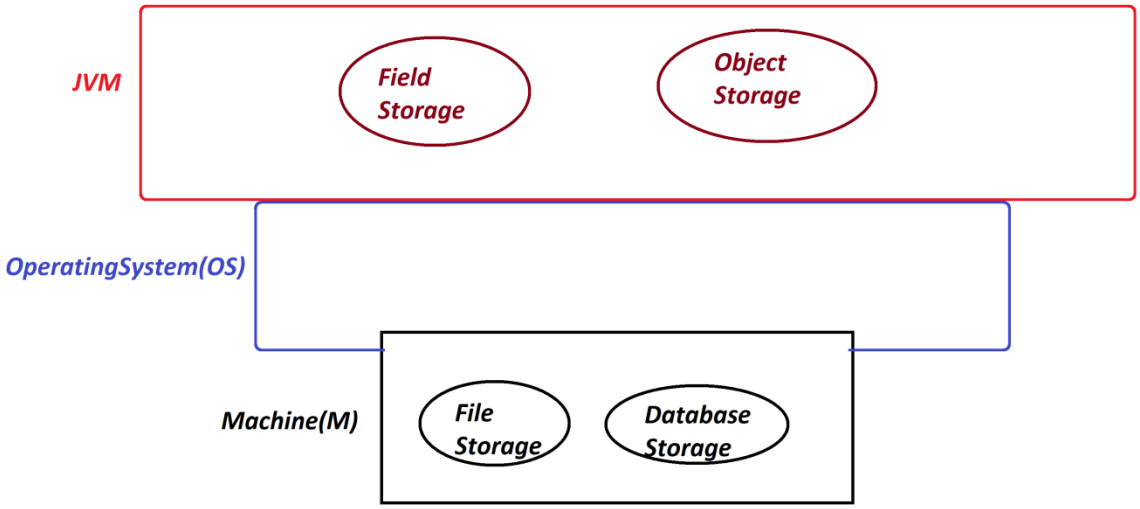
will be destroyed automatically when JVM shutdowns.

=>when we want to have permanent storage for Application,then we have to take the support of any one

of the following:

3.File Storage

4.Database Storage



3.File Storage:

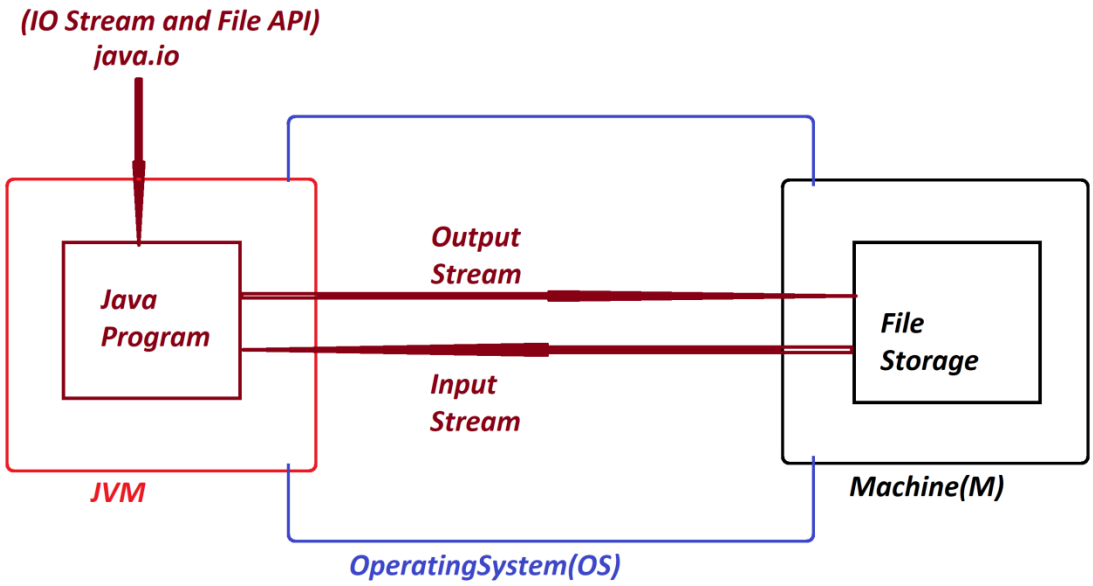
=>The smallest permanent storage of computer-system which is 'controlled and managed' by the

OperatingSystem is known as File Storage.

=>when we want to establish communication b/w Java-Program and File Storage,the Java-Program must

be constructed using 'Classes and Interfaces' available from 'java.io' package,known as 'IO Stream and File API'

Diagram:



DisAdvantages of File Storage:

(i)File Storage is not perferable for Large-Scale applications,because which cannot organize huge data.

(ii)File Storage will lead to Data Redundancy,which means lead to replication of data.

(iii)Data Sharing problems in File Storage.

(iv)Data is not secure in File Storage.

Note:

=>DisAdvantages of File Storage can be Overcomed using Database Storage.

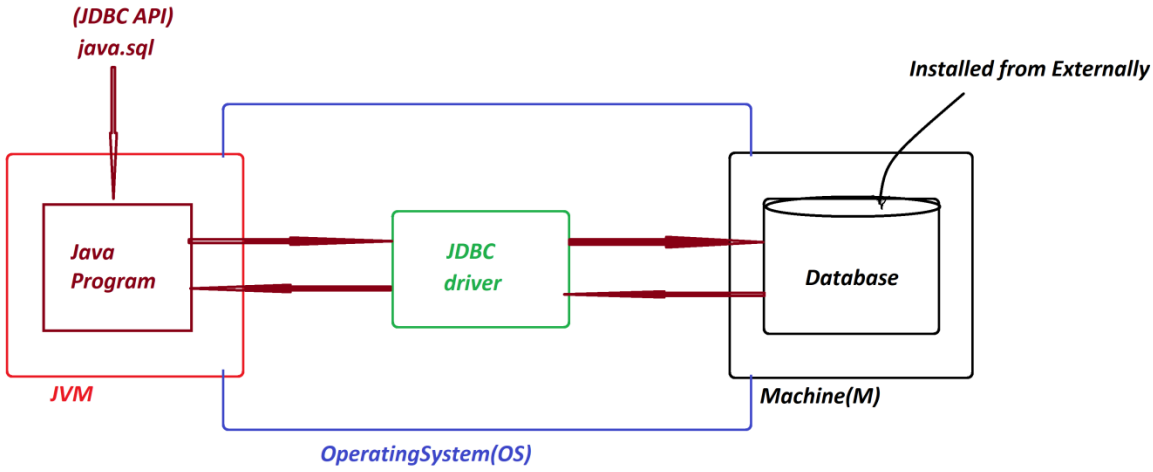
**imp*

4.Database Storage:

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

=>In the process of establishing communication b/w Java-Program and Database product,the Java Program must be constructed using 'classes and Interfaces' available from 'java.sql' package and must take the support of JDBC-Driver.

Diagram:



---

faq:

define 'driver'?

=>The small s/w program part of OperatingSystem,which establish connection b/w two end-points for communication is known as 'driver'

Ex:

Audio driver

Video driver

Network driver

faq:

define JDBC driver?

=>The driver which establish connection b/w Java-Program and Database product for communication is

known as JDBC driver.(Java DataBase Connectivity Driver)

Types of JDBC drivers:

=>These JDBC drivers are categorized into four types:

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

2.Native API driver(Type-2 driver)

3.Network Protocol driver(Type-3 driver)

4.Thin driver(Type-4 driver)

Note:

=>In realtime application development,we use 'Thin driver'(Type-4 driver)

==========

faq:

define API?

=>API stands for 'Application Programming Interface' and which is a PlatForm to develop applications using Language or Technology or Framework.

=>Accoding to Java-Language,each package is one API.

=>The following are some important APIs:

java.lang - Language API

java.util - Utility API(Collection Framework API)

java.io - IO Stream and File API

java.net - Networking API

java.sql - JDBC API

==========
=

Dt : 28/9/2024(Day-3)

**imp*

Making ComputerSystem ready to development JDBC Application:

step-1 : Download and Install Database Product(Oracle)

step-2 : Perform Login Process to Database

step-3 : Create table with name Customer67

(phno,cname,ccity,mid)

primary key : phno

```
create table Customer67(phno number(15),cname varchar2(15),
ccity varchar2(15),mid varchar2(25),primary key(phno));
```

step-4 : Insert min 5 Customer details using SQLCommandLine

```
insert into Customer67 values(9898981234,'Alex','Hyd','a@gmail.com');

insert into Customer67 values(7878781234,'Ram','Hyd','rm@gmail.com');
insert into Customer67 values(6565651234,'Raj','Hyd','rj@gmail.com');
```

step-5 : Copy DB-Jar file into User defined folder on DeskTop or

any drive

Note:

=>DB Jar file is available from "lib" folder of "jdbc"

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

ojdbc6.jar - Oracle11

step-6 : Find the PortNo and ServiceName of Oarcle Product

Note:

=>PortNo and ServiceName is available from "tnsnames.ora" file of "ADMIN" folder of "network"

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

PortNo : 1521

ServiceName : XE

==========

**imp*

JDBC API:

=>"java.sql" package is known as JDBC-API and which provide 'Classes and Interfaces' to construct JDBC Applications.

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

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

1.createStatement()

2.prepareStatement()

3.prepareCall()

4.getAutoCommit()

5.setAutoCommit()

6.setSavepoint()

7.releaseSavepoint()

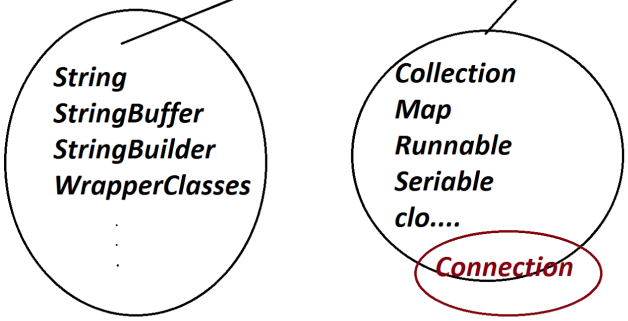
8.commit()

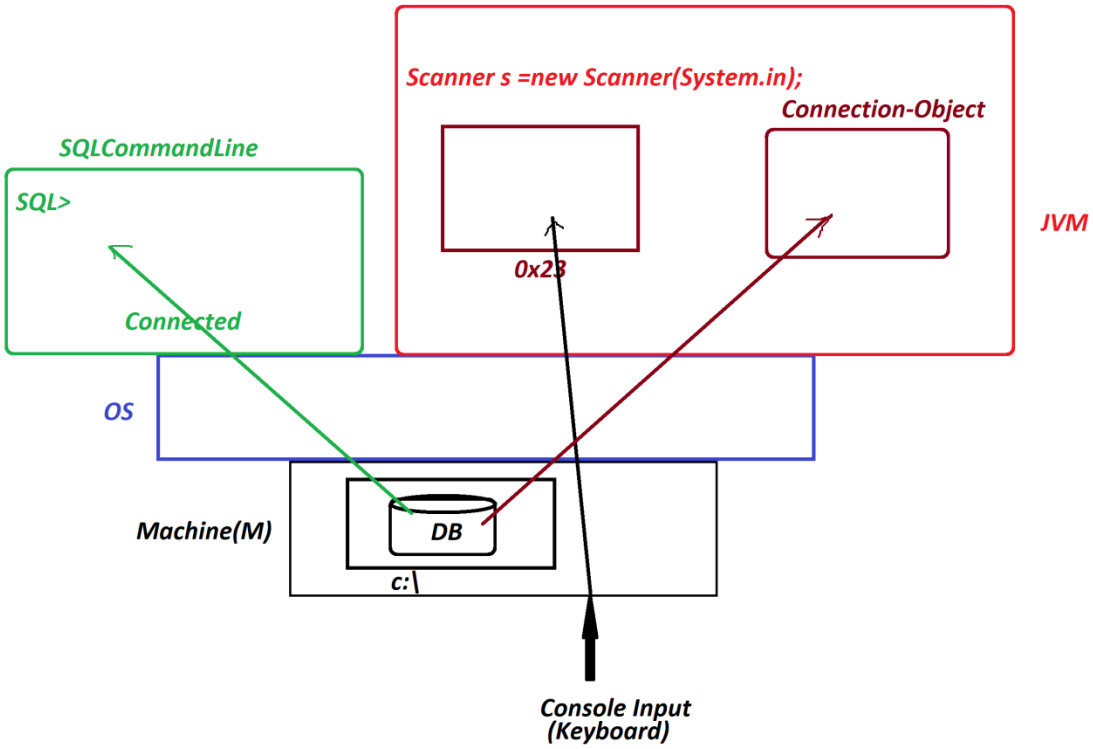
9.rollback()

10.close()

==========

Variables Methods Constructors Blocks Classes Interfaces AbstractClasses




Scanner s =new Scanner(System.in);
Connection-Object
SQLCommandLine
SQL>
JVM
0x23
Connected
OS
Machine(M)
DB
c:\
Console Input
(Keyboard)

Dt : 29/9/2024(Day-4)

Note:

=>we use getConnection()-method from 'java.sql.DriverManger' class to create implementation object for 'Connection-Interface' and this 'Connection-Object' internally connected to Database product.

=>This getConnection()-method internally holding 'Anonymous Local InnerClass as implementation class of Connection interface'.

Method Signature of getConnection():

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

syntax:

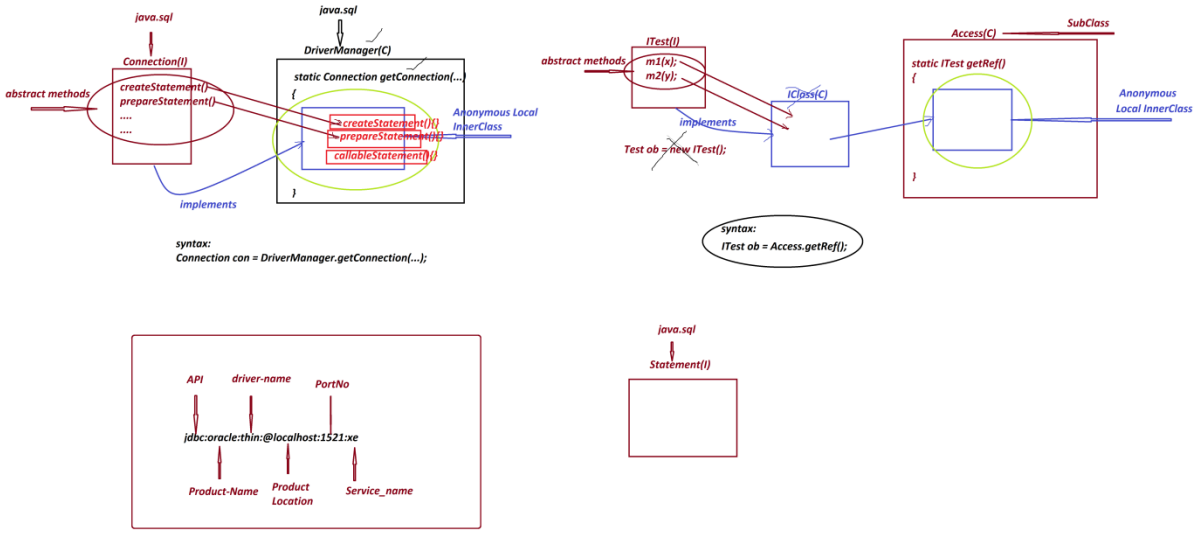
Connection con = DriverManager.getConnection("DB-URL","DB-UName","DB-PWord");

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

DB-UName => system

DB-PWord => tiger

Diagrams:



**imp*

JDBC statements:

=>The statements which specify the actions to be performed on database product are known as JDBC-statements.

=>These JDBC-statements are categorized into three types:

1.Statement

2.PreparedStatement

3.CallableStatement

1.Statement:

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

(Normal queries means Create,Insert,Retrieve(Select),Update,Delete)

=>we use createStatement()-method from 'Connection-Interface' to create implementation object for 'Statement-Interface'.

=>This createStatement()-method internally holding 'Anonymous Local InnerClass as

implementation class 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 two important methods of Statement:

(i)executeQuery()

(ii)executeUpdate()

(i)executeQuery():

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

Method Signature of executeQuery():

public abstract java.sql.ResultSet executeQuery(java.lang.String)

throws java.sql.SQLException;

syntax:

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

(ii)executeUpdate():

=>executeUpdate()-method is used to execute Non-Select queries like Create,Insert,Update and delete.

Method Signature of executeUpdate():

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

syntax:

int k = stm.executeUpdate();

Dt : 30/9/2024(Day-5)

**imp*

Steps used to establish communication to Database Product:

step-1 : Loading driver

step-2 : Creating Connection

step-3 : Preparing JDBC-statement

step-4 : Executing query

step-5 : Closing connection

**imp*

Construct JDBC Application using IDE Eclipse:

step-1 : Open IDE Eclipse,while opening name the WorkSpace and Click "Launch"

step-2 : Create Java Project

step-3 : Add DB-Jar file to Java Project through Build-path

RightClick on Java Project->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 in "src"

step-5 : Create class in Package and write JDBC-Code to display all details of Customer.

Program : DBCon1.java

```
package test;
import java.sql.*;
public class DBCon1 {
      public static void main(String[] args) {
       try {
         //step-1 : Loading driver
         Class.forName("oracle.jdbc.driver.OracleDriver");
         //step-2 : Creating Connection
         Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
         //step-3 : Preparing JDBC-statement
         Statement stm = con.createStatement();
         //step-4 : Executing query
         ResultSet rs = stm.executeQuery("select * from Customer67");
         System.out.println("-------CustomerDetails-------");
         while(rs.next())
         {
              System.out.println(rs.getLong(1)+"\t"+rs.getString(2)+"\t"+
               rs.getString(3)+"\t"+rs.getString(4));
         }//end of loop
         //step-5 : Closing connection
         con.close();
       }catch(Exception e) {
         System.out.println(e.toString());
       }
      }
}
```

o/p:

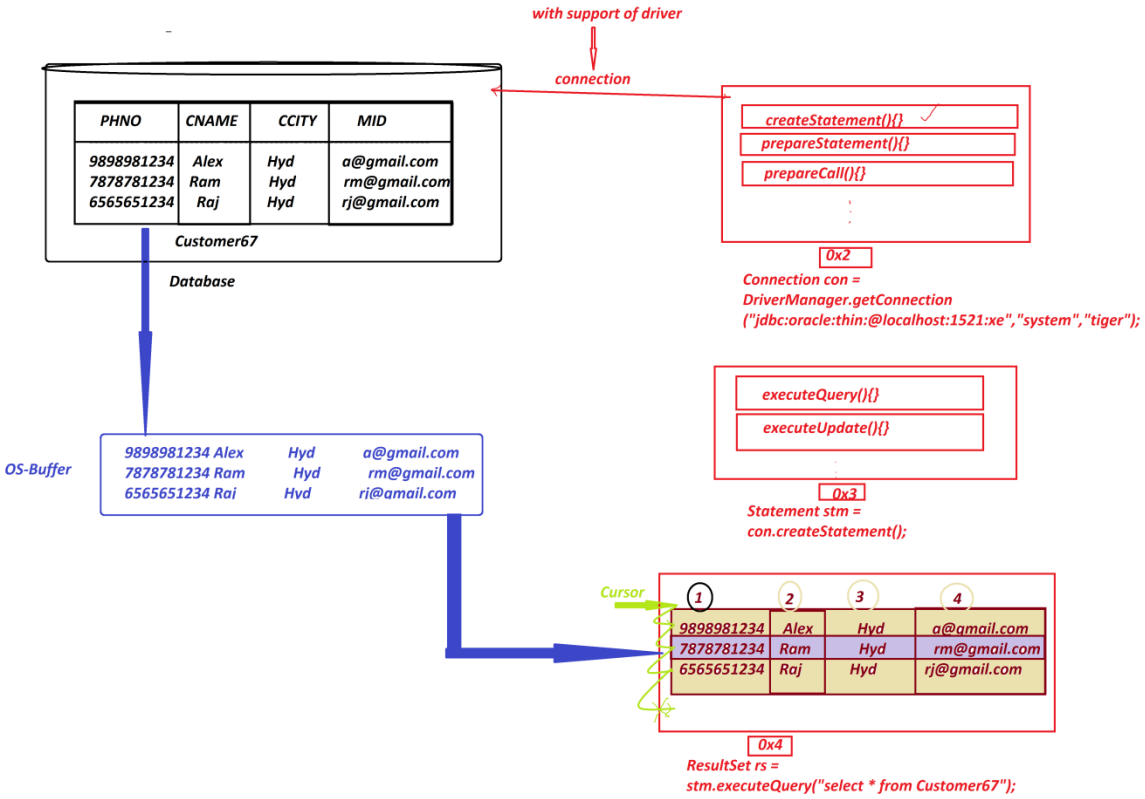
-CustomerDetails-------

9898981234 Alex Hyd a@gmail.com

7878781234 Ram Hyd rm@gmail.com

6565651234 Raj Hyd rj@gmail.com

Layout:



==========

Assignment-1:

step-1 : Create DB Table with name BookDetails67 from SQLCommandLine

(bcode,bname,bauthor,bprice,bqty)

Primary Key : bcode

step-2 : Insert min 5 BookDetails from SQLCommandLine

step-3 : Construct JDBC Application to display all BookDetails.

==========

Dt : 1/10/2024(Day-6)

Ex:

Construct JDBC Application to read Customer-details from Console input and insert into DB Table(Customer67)

Program : DBCon2.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon2 {
        public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    try(s;){
        System.out.println("Enter the Cust-phNo:");
        long phNo = Long.parseLong(s.nextLine());
        System.out.println("Enter the Cust-Name:");
        String cName = s.nextLine();
        System.out.println("Enter the Cust-City:");
        String cCity = s.nextLine();
        System.out.println("Enter the Cust-MailId:");
        String mId = s.nextLine();

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

```
        Statement stm = con.createStatement();
        int k = stm.executeUpdate
        ("insert into Customer67 values("+phNo+",'"+cName+"','"+cCity+"','"+mId+"')");
        System.out.println("The value in k : "+k);
        if(k>0) {
                System.out.println("Customer details inserted Successfully....");
        }
        con.close();
    }
    catch(SQLIntegrityConstraintViolationException sicve) {
        System.out.println("Customer details already available....");
    }catch(Exception e) {
         System.out.println(e.toString());
    }
        }
}
```

o/p:

Enter the Cust-phNo:

3434341234

Enter the Cust-Name:

TYU

Enter the Cust-City:

Hyd

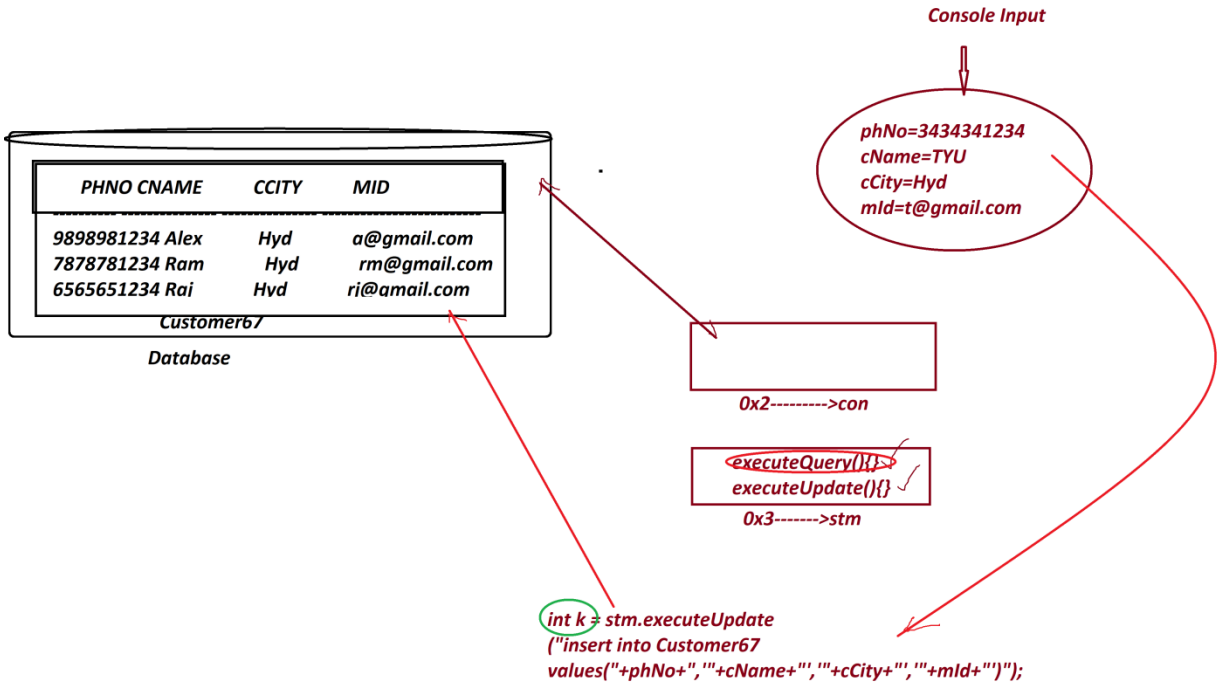
Enter the Cust-MailId:

T@gmail.com

The value in k : 1

Customer details inserted Successfully....

Diagram:



==========

Ex:

Construct JDBC Application to display Customer details based on PhoneNo.

Program : DBCon3.java

```
package test;

import java.util.*;

import java.sql.*;
```

```
public class DBCon3 {

        public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    try(s;){
        System.out.println("Enter the Cust-PhNo to display details:");
        long phNo = s.nextLong();


        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection
                        ("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
        Statement stm = con.createStatement();
        ResultSet rs = stm.executeQuery
                        ("select * from Customer67 where phno="+phNo+"");
        if(rs.next()) {
            System.out.println(rs.getLong(1)+"\t"+rs.getString(2)+"\t"+
       rs.getString(3)+"\t"+rs.getString(4));
        }else {
            System.out.println("Invalid Customer Phone No....");
        }
        con.close();
    }catch(Exception e) {
        System.out.println(e.toString());
    }
        }
}
```

o/p:

Enter the Cust-PhNo to display details:

9898981234

9898981234 Alex Hyd a@gmail.com

==========

Assignment-2:

Construct JDBC Application read BookDetails from Console Input and insert into DB Table BookDetails67.

Assignment-3:

Construct JDBC Application to display BookDetails based on bookCode.

==========

**imp*

2.PreparedStatement:

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

=>we use prepareStatement()-method from 'Connection-Interface' to create implementation object for 'PreparedStatement-Interface'

=>This prepareStatement()-method internally holding 'Anonymous Local InnerClass as implementation class 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 of executeQuery():

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

syntax:

ResultSet rs = ps.executeQuery();

(ii)executeUpdate():

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

Method Signature of executeUpdate():

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

syntax:

int k = ps.executeUpdate();

Dt : 2/10/2024(Day-7)

Ex:

step-1 : Create table with name Product67

(code,name,price,qty)

Primary Key : code

create table Product67(code varchar2(10),name varchar2(15),price number(10,2),

qty number(10),primary key(code));

step-2 : Construct JDBC application to perform the following operations based on user-choice:

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

Program : DBCon4.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon4 {
      public static void main(String[] args) {
       Scanner s = new Scanner(System.in);
       try(s;){
         Class.forName("oracle.jdbc.driver.OracleDriver");
         Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
         PreparedStatement ps1 = con.prepareStatement
                    ("insert into Product67 values(?,?,?,?)");//Compilation
process
         PreparedStatement ps2 = con.prepareStatement
                    ("select * from Product67");//Compilation Process
```

```
		PreparedStatement ps3 = con.prepareStatement
				("select * from Product67 where code=?");//Compilation
process
		while(true) {
			System.out.println("=====Choice=====");
			System.out.println("\t1.AddProduct"
					+ "\n\t2.ViewAllProducts"
					+ "\n\t3.ViewProductByCode"
					+ "\n\t4.UpdateProductByCode(price-qty)"
					+ "\n\t5.DeleteProductByCode"
					+ "\n\t6.Exit");
			System.out.println("Enter your Choice:");
			switch(Integer.parseInt(s.nextLine())) {
			case 1:
				System.out.println("-----Enter Product details------");
				System.out.println("Enter the Prod-Code:");
				String pC = s.nextLine();
				System.out.println("Enter the Prod-Name:");
				String pN = s.nextLine();
				System.out.println("Enter the Prod-Price:");
				float pP = Float.parseFloat(s.nextLine());
				System.out.println("Enter the Prod-Qty:");
				int pQ = Integer.parseInt(s.nextLine());

				//Loading data to PreparedStatement Object
				ps1.setString(1, pC);
				ps1.setString(2, pN);
				ps1.setFloat(3, pP);
				ps1.setInt(4, pQ);

				int k1 = ps1.executeUpdate();//Execution process
				if(k1>0) {
					System.out.println("Product Added
Successfully....");
				}
				break;
			case 2:
				ResultSet rs1 = ps2.executeQuery();
				System.out.println("-------Product Details-------");
				while(rs1.next()) {

System.out.println(rs1.getString(1)+"\t"+rs1.getString(2)+

"\t"+rs1.getFloat(3)+"\t"+rs1.getInt(4));
				}//end of loop
				break;
			case 3:
				System.out.println("Enter the Prod-Code to display
Product details:");
				String pC2 = s.nextLine();
```

```
                        //Loading data to PreparedStatement Object
                        ps3.setString(1, pC2);
                        ResultSet rs2 = ps3.executeQuery();
                        if(rs2.next()) {

System.out.println(rs2.getString(1)+"\t"+rs2.getString(2)+

"\t"+rs2.getFloat(3)+"\t"+rs2.getInt(4));
                        }else {
                              System.out.println("Invalid Product Code....");
                        }
                        break;
                  case 4:
                        break;
                  case 5:
                        break;
                  case 6:
                        System.out.println("Operations Stopped....");
                        System.exit(0);
                  default:
                        System.out.println("Invalid choice....");
                  }//end of switch
            }//end of loop
           }catch(Exception e) {
            System.out.println(e.toString());
           }
          }
}
```

o/p:

=====Choice=====

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

6.Exit

Enter your Choice:

3

Enter the Prod-Code to display Product details:

A121

A121 Mouse 789.23 10

=====Choice=====

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

6.Exit

Enter your Choice:

2

-------Product Details-------

A121 Mouse 789.23 10

A123 CDR 678.12 12

=====Choice=====

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

6.Exit

Enter your Choice:

3

Enter the Prod-Code to display Product details:

D111

Invalid Productb Code....

=====Choice=====

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

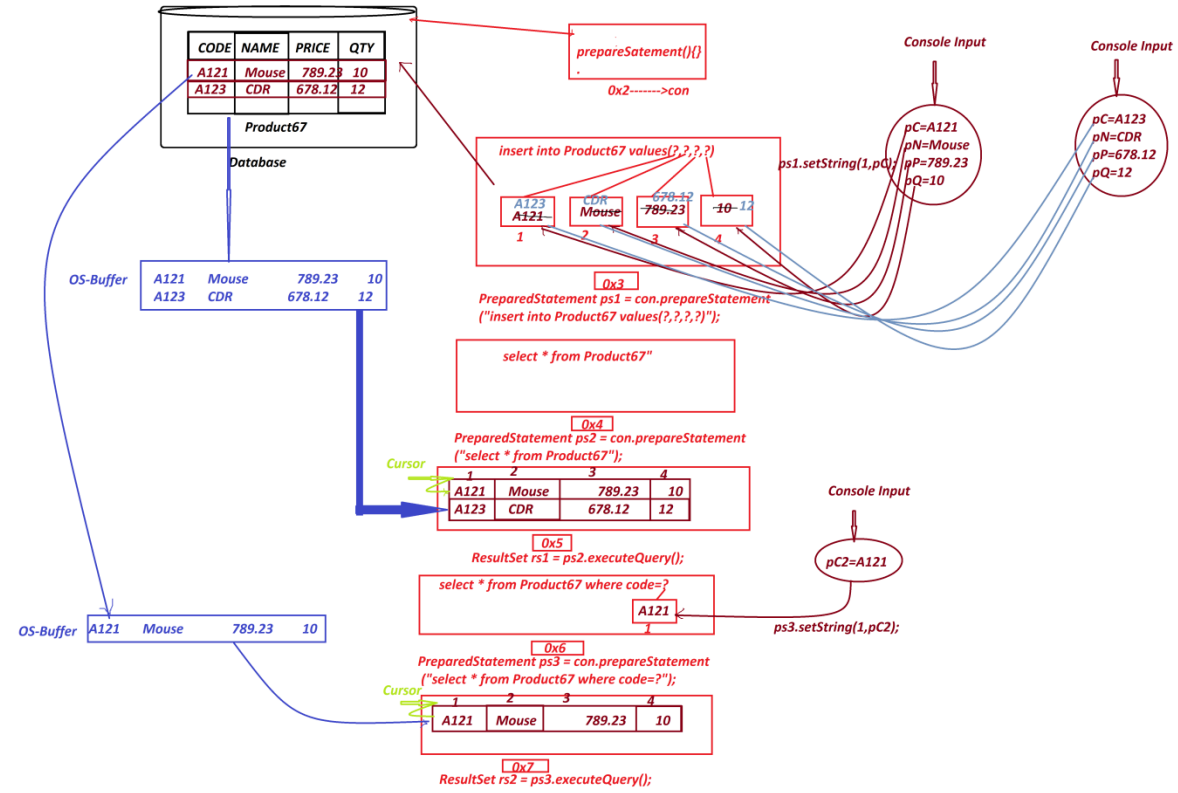
6.Exit

Enter your Choice:

6

Operations Stopped....

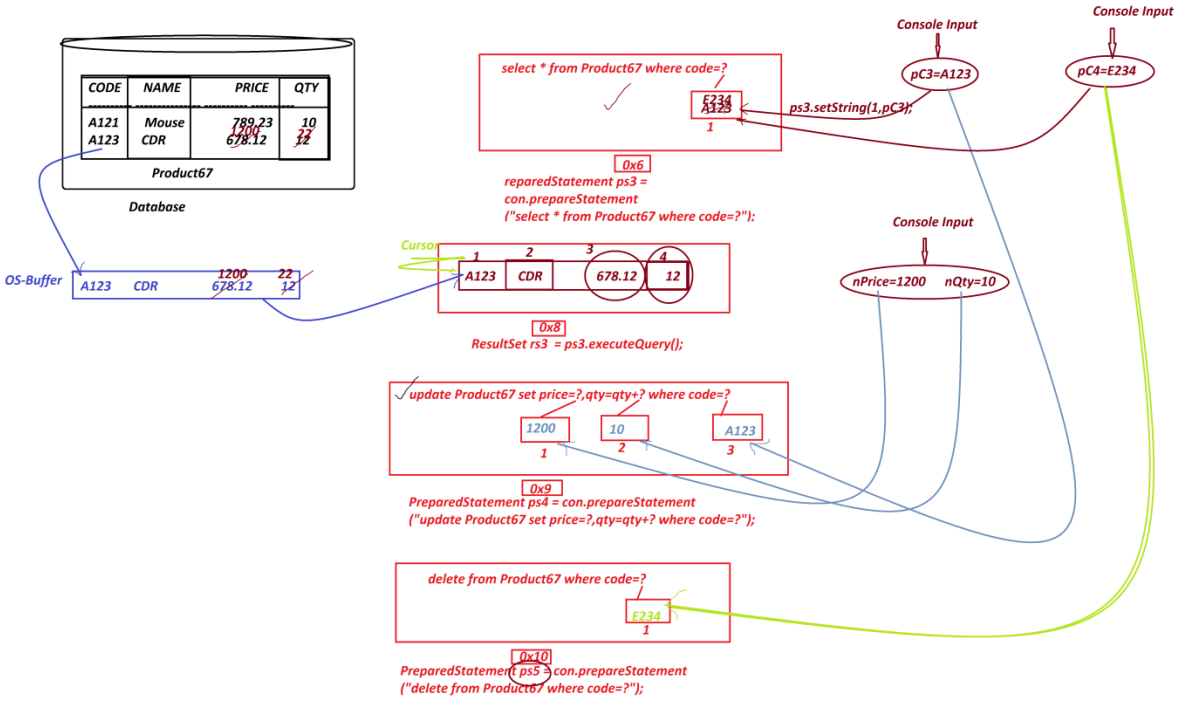
Diagram:


CODE NAME PRICE QTY
A121 Mouse 789.23 10
A123 CDR 678.12 12
Product67
Database
prepareSatement(){}
0x2------>con
Console Input
Console Input
pC=A121
pN=Mouse
pP=789.23
pQ=10
pC=A123
pN=CDR
pP=678.12
pQ=12
insert into Product67 values(?,?,?,?)
ps1.setString(1,pC);
A123
A121
CDR
Mouse
678.12
789.23
10 12
1 2 3 4
0x3
PreparedStatement ps1 = con.prepareStatement
("insert into Product67 values(?,?,?,?)");
OS-Buffer
A121 Mouse 789.23 10
A123 CDR 678.12 12
select * from Product67"
0x4
PreparedStatement ps2 = con.prepareStatement
("select * from Product67");
Cursor
1 2 3 4
A121 Mouse 789.23 10
A123 CDR 678.12 12
0x5
ResultSet rs1 = ps2.executeQuery();
Console Input
pC2=A121
select * from Product67 where code=?
A121
1
ps3.setString(1,pC2);
OS-Buffer
A121 Mouse 789.23 10
0x6
PreparedStatement ps3 = con.prepareStatement
("select * from Product67 where code=?");
Cursor
1 2 3 4
A121 Mouse 789.23 10
0x7
ResultSet rs2 = ps3.executeQuery();

==========

Dt : 3/10/2024(Day-8)

Layout:



Program : DBCon4.java(Code Updated)

```
package test;

import java.util.*;

import java.sql.*;

public class DBCon4 {

        public static void main(String[] args) {

    Scanner s = new Scanner(System.in);

    try(s;){

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

        Connection con = DriverManager.getConnection

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

        PreparedStatement ps1 = con.prepareStatement
```

```
                ("insert into Product67 values(?,?,?,?)");//Compilation process
        PreparedStatement ps2 = con.prepareStatement
                ("select * from Product67");//Compilation Process
        PreparedStatement ps3 = con.prepareStatement
                ("select * from Product67 where code=?");//Compilation process
        PreparedStatement ps4 = con.prepareStatement
                ("update Product67 set price=?,qty=qty+? where code=?");//Compilation
process
        PreparedStatement ps5 = con.prepareStatement
                ("delete from Product67 where code=?");//Compilation Process
        while(true) {
            System.out.println("=====Choice=====");
            System.out.println("\t1.AddProduct"
                    + "\n\t2.ViewAllProducts"
                    + "\n\t3.ViewProductByCode"
                    + "\n\t4.UpdateProductByCode(price-qty)"
                    + "\n\t5.DeleteProductByCode"
                    + "\n\t6.Exit");
            System.out.println("Enter your Choice:");
            switch(Integer.parseInt(s.nextLine())) {
            case 1:
                System.out.println("-----Enter Product details------");
                System.out.println("Enter the Prod-Code:");
                String pC = s.nextLine();
                System.out.println("Enter the Prod-Name:");
                String pN = s.nextLine();
```

```
            System.out.println("Enter the Prod-Price:");
            float pP = Float.parseFloat(s.nextLine());
            System.out.println("Enter the Prod-Qty:");
            int pQ = Integer.parseInt(s.nextLine());

            //Loading data to PreparedStatement Object
            ps1.setString(1, pC);
            ps1.setString(2, pN);
            ps1.setFloat(3, pP);
            ps1.setInt(4, pQ);

            int k1 = ps1.executeUpdate();//Execution process
            if(k1>0) {
                  System.out.println("Product Added Successfully....");
            }
            break;
      case 2:
            ResultSet rs1 = ps2.executeQuery();
            System.out.println("-------Product Details-------");
            while(rs1.next()) {
                  System.out.println(rs1.getString(1)+"\t"+rs1.getString(2)+
                              "\t"+rs1.getFloat(3)+"\t"+rs1.getInt(4));
            }//end of loop
            break;
      case 3:
```

```
System.out.println("Enter the Prod-Code to display Product details:");
String pC2 = s.nextLine();
//Loading data to PreparedStatement Object
ps3.setString(1, pC2);
ResultSet rs2 = ps3.executeQuery();
if(rs2.next()) {
        System.out.println(rs2.getString(1)+"\t"+rs2.getString(2)+
                        "\t"+rs2.getFloat(3)+"\t"+rs2.getInt(4));
}else {
        System.out.println("Invalid Product Code....");
}
break;
case 4:
System.out.println("Enter the Prod-Code to update price and qty:");
String pC3 = s.nextLine();
//Loading data PreparedStatement Object
ps3.setString(1, pC3);
ResultSet rs3  = ps3.executeQuery();
if(rs3.next()) {
        System.out.println("Old price of product:"+rs3.getFloat(3));
        System.out.println("Enter the new price for product:");
        float nPrice = Float.parseFloat(s.nextLine());
        System.out.println("Existing Product qty:"+rs3.getInt(4));
        System.out.println("Enter the new qty(new stock) for product:");
        int nQty = Integer.parseInt(s.nextLine());
```

```
                //Loading data to PreparedStatement Object
                ps4.setFloat(1, nPrice);
                ps4.setInt(2, nQty);
                ps4.setString(3, pC3);

                int k2 = ps4.executeUpdate();//Execution process
                if(k2>0) {
                        System.out.println("Product Updated Successfully...");
                }
        }else {
                System.out.println("Invalid Prod-Code...");
        }
        break;
case 5:
        System.out.println("Enter the Prod-Code to delete Product details:");
        String pC4 = s.nextLine();
        //Loading the data to PreparedStatement Object
        ps3.setString(1, pC4);
        ResultSet rs4 = ps3.executeQuery();
        if(rs4.next()) {
                //Loading the data to PreparedStatement Object
                ps5.setString(1, pC4);
                int k3 = ps5.executeUpdate();//Execution process
                if(k3>0) {
```

```
                                        System.out.println("Product deleted successfully...");
                                }
                        }else {
                                System.out.println("Invalid Prod-code.....");
                        }
                        break;
                case 6:
                        System.out.println("Operations Stopped....");
                        System.exit(0);
                default:
                        System.out.println("Invalid choice....");
                }//end of switch
        }//end of loop
    }catch(Exception e) {
        System.out.println(e.toString());
    }
        }
}
```

o/p:

```
=====Choice=====
        1.AddProduct
        2.ViewAllProducts
        3.ViewProductByCode
        4.UpdateProductByCode(price-qty)
        5.DeleteProductByCode
```

6.Exit

Enter your Choice:

1

-----Enter Product details------

Enter the Prod-Code:

D345

Enter the Prod-Name:

MDD

Enter the Prod-Price:

781.23

Enter the Prod-Qty:

10

Product Added Successfully....

=====Choice=====

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

6.Exit

Enter your Choice:

2

-------Product Details-------

A121 Mouse 789.23 10

A123 CDR 1200.0 22

E234 FDD 456.23 11

D345 MDD 781.23 10

=====Choice=====

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

6.Exit

Enter your Choice:

3

Enter the Prod-Code to display Product details:

D345

D345 MDD 781.23 10

=====Choice=====

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

6.Exit

Enter your Choice:

4

Enter the Prod-Code to update price and qty:

A121

Old price of product:789.23

Enter the new price for product:

892.34

Existing Product qty:10

Enter the new qty(new stock) for product:

5

Product Updated Successfully...

=====Choice=====

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

6.Exit

Enter your Choice:

2

-------Product Details-------

A121 Mouse 892.34 15

A123 CDR 1200.0 22

E234 FDD 456.23 11

D345 MDD 781.23 10

=====Choice=====

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

6.Exit

Enter your Choice:

5

Enter the Prod-Code to delete Product details:

E234

Product deleted successfully...

=====Choice=====

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

6.Exit

Enter your Choice:

2

-------Product Details-------

A121 Mouse 892.34 15

A123 CDR 1200.0 22

D345 MDD 781.23 10

=====Choice=====

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

6.Exit

Enter your Choice:

6

Operations Stopped....

==========

Assignment:

step-1 : Create table with name Employee67 from SQLCommandLine

(eid,ename,edesg,bsal,hra,da,totsal)

Primary Key : eid

step-2 : Construct JDBC Application to perform the following Operations based on User choice:

1.AddEmployee

2.ViewAllEmployees

3.ViewEmployeeById

4.UpdateEmployeeById(bsal-hra-da-totsal)

5.DeleteEmployeeById

Note::

Read(Input) : eId,eName,eDesg,bSal

Calculate : hra,da,totSal

hra = 93% of bSal

da = 61% of bSal

totSal = bSal+hra+da

==========

Dt : 4/10/2024(Day-9)

**imp*

'ResultSet' in JDBC:

=>'ResultSet' is an interface from java.sql package and which is instantiated (implementation Object) using executeQuery()-method.

=>This 'ResultSet' object will hold the data retrieved from select-queries.

=>Based on the cursor movement the 'ResultSet-Objects' are categorized into two types:

(a)NonScrollable ResultSet Objects

(b)Scrollable ResultSet Objects

(a)NonScrollable ResultSet Objects:

=>In NonScrollable ResultSet Objects the cursor can be moved only in one direction,which means the cursor can be moved from top-of-table data to bottom-of-table data.

=>we use the following syntax to create NonScrollable ResultSet Objects:

syntax : Using 'Statement'

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

syntax : Using 'PreparedStatement'

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

**imp*

(b)Scrollable ResultSet Objects:

=>In Scrollable ResultSet Objects the cursor can be moved in both directions,which means we can move the cursor in forward and backward direction.

=>we use the following syntax to create Scrollable ResultSet Objects:

syntax : Using 'Statement'

ResultSet rs = stm.executeQuery("select-query",type,mode);

syntax : Using 'PreparedStatement'

ResultSet rs = ps.executeQuery(type,mode);

Note::

(i)"type" specify the direction of cursor on ResultSet-Objects.

=>The following fields(Variables) from ResultSet Interface will specify the "type"

public static final int TYPE_FORWARD_ONLY;

public static final int TYPE_SCROLL_INSENSITIVE;

public static final int TYPE_SCROLL_SENSITIVE;

(ii)"mode" specify the action to be performed on ResultSet-Objects.

=>The following fields(Variables) from ResultSet Interface will specify the 'mode"

public static final int CONCUR_READ_ONLY;

public static final int CONCUR_UPDATABLE;

syntax : Using 'Statement'

```
Statement stm = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                    ResultSet.CONCUR_READ_ONLY);
```

syntax : Using 'PreparedStatement'

```
PreparedStatement ps = con.prepareStatement("select * from Product67",
                                    ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
```

=>we use the following some important methods to control cursor on ResultSet Objects:

1.afterLast()

2.beforeFirst()

3.next()

4.previous()

5.first()

6.last()

7.absolute(int)

8.relative(int)

1.afterLast():

=>afterLast() method will make the cursor point after the last row.

syntax:

```
rs2.afterLast();
```

2.beforeFirst():

=>beforeFirst() method will make the cursor point before the first row.

syntax:

```
rs2.beforeFirst();
```

3.next():

=>next() method will move the cursor in forward direction.

syntax:

```
rs2.next();
```

4.previous():

=>previous() method will move the cursor in backward direction.

syntax:

rs2.previous();

5.first():

=>first() method will make the cursor point to the first row of ResultSet Object.

syntax:

rs1.first();

6.last():

=>last() method will make the cursor point to the last row of ResultSet Object.

syntax:

rs1.last();

7.absolute(int):

=>absolute(int) method is used to move the cursor to specified row-no.

syntax:

rs1.absolute(4);

8.relative(int):

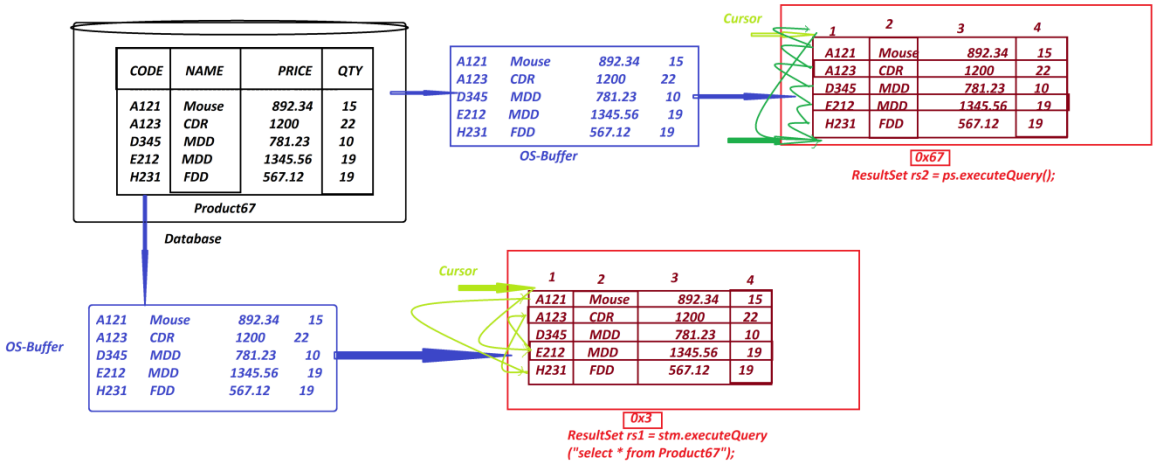
=>relative(int) method is used to move the cursor forward or backward from the current position of Cursor.

syntax:

rs1.relative(-2);

Ex:

Layout:



Program : DBCon5.java

```
package test;
import java.sql.*;
public class DBCon5 {
      public static void main(String[] args) {
       try {
         Class.forName("oracle.jdbc.driver.OracleDriver");
         Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
         System.out.println("********Statement**********");
         Statement stm =
con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                     ResultSet.CONCUR_READ_ONLY);
         ResultSet rs1 = stm.executeQuery("select * from Product67");
         System.out.println("=====Row-4(absolute(4)=======");
         rs1.absolute(4);
         System.out.println(rs1.getString(1)+"\t"+rs1.getString(2)+
                     "\t"+rs1.getFloat(3)+"\t"+rs1.getInt(4));
         System.out.println("=====Row-2(relative(-2)------");
         rs1.relative(-2);
         System.out.println(rs1.getString(1)+"\t"+rs1.getString(2)+
                     "\t"+rs1.getFloat(3)+"\t"+rs1.getInt(4));
         System.out.println("=====Last Row=======");
         rs1.last();
```

```
            System.out.println(rs1.getString(1)+"\t"+rs1.getString(2)+
                        "\t"+rs1.getFloat(3)+"\t"+rs1.getInt(4));
            System.out.println("=====First Row=====");
            rs1.first();
            System.out.println(rs1.getString(1)+"\t"+rs1.getString(2)+
                        "\t"+rs1.getFloat(3)+"\t"+rs1.getInt(4));
            System.out.println("*********PreparedStatement*********");
            PreparedStatement ps = con.prepareStatement("select * from
Product67",
                        ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
            ResultSet rs2 = ps.executeQuery();
            rs2.afterLast();
            while(rs2.previous()) {
                  System.out.println(rs2.getString(1)+"\t"+rs2.getString(2)+
                              "\t"+rs2.getFloat(3)+"\t"+rs2.getInt(4));
            }
          }catch(Exception e) {
            System.out.println(e.toString());
          }
        }
}
```

o/p:

```
********Statement**********

=====Row-4(absolute(4)=======

E212    MDD    1345.56         19

=====Row-2(relative(-2)------

A123    CDR     1200.0  22

=====Last Row=======

H231    FDD     567.12  19

=====First Row=====

A121    Mouse  892.34  15

*********PreparedStatement*********

H231    FDD     567.12  19

E212    MDD    1345.56         19
```

D345 MDD 781.23 10

A123 CDR 1200.0 22

A121 Mouse 892.34 15

==========

Dt : 5/10/2024(Day-10)

**imp*

Streams with Database product:

define stream?(Normal definition)

=>The contineous flow of data is known as stream.

Types of streams:

=>streams in java are categorized into two types:

1.Byte Stream(Binary Stream)

2.Character Stream

1.Byte Stream(Binary Stream):

=>The contineous flow of data in the form of 8-bits is known as Byte Stream or Binary Stream.

=>Through Byte Stream we can send all multi-media data files, which means Text,Audio,Video,Image and Animation.

2.Character Stream:

=>The contineous flow of data in the form of 16-bits is known as Character Stream.

=>Through Character Stream it is preferable to send only Text data and not preferable for Audio,Video,Image and Animation.

Note::

=>In realtime application development,we use only Byte Stream

---

=>we use the following sql-types to store stream data:

(a)BLOB

(b)CLOB

(a)BLOB:

=>BLOB stands for 'Binary Large OBject' and which is used to

store Byte Stream data or Binary Stream data.

(b)CLOB:

=>CLOB stands for 'Character Large OBject' and which is used to

store Character Stream data.

==========

**imp*

Storing Stream data to Database Product:

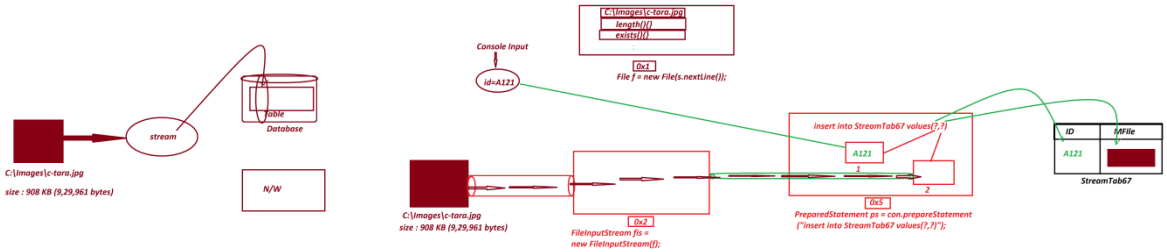
step-1 : Create table with name StreamTab67(id,mfile)

Primary Key : id

```
create table StreamTab67(id varchar2(10),mfile BLOB,
primary key(id));
```

step-2 : Construct JDBC Application to store Image to DB Product

Layout:



Program : DBCon6.java

```
package test;
import java.util.*;
import java.io.*;
import java.sql.*;
public class DBCon6
{

	public static void main(String[] args)
	{
	   Scanner s = new Scanner(System.in);
	   try(s;)//Java9
	   {
	      Class.forName("oracle.jdbc.driver.OracleDriver");
	      Connection con = DriverManager.getConnection
	                  ("jdbc:oracle:thin:@localhost:1521:xe",
	                              "system","tiger");
	      PreparedStatement ps = con.prepareStatement
	                  ("insert into StreamTab67 values(?,?)");
	      System.out.println("Enter the id to store image:");
	      String id = s.nextLine();
	      System.out.println("Enter the fPath&fName(Source-path)");
	      File f = new File(s.nextLine());
	      if(f.exists()) {
	            FileInputStream fis = new FileInputStream(f);
	            ps.setString(1, id);
	            ps.setBinaryStream(2, fis, f.length());

	            int k = ps.executeUpdate();
	            if(k>0) {
	                  System.out.println("Image stored Successfully....");
	            }
	      }else {
	            System.out.println("Invalid fPath or fName...");
	      }
	      con.close();
	   }//end of try
```

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

}
```

o/p:

Enter the id to store image:

A121

Enter the fPath&fName(Source-path)

C:\Images\c-tara.jpg

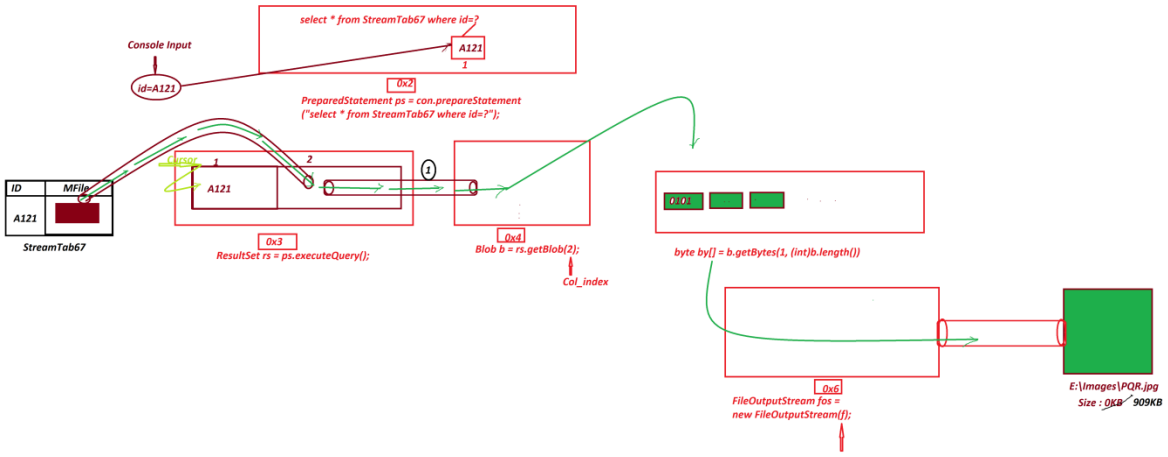
Image stored Successfully....

==========

Dt : 6/10/2024(Day-11)

Retrieving Stream(Image) from database product:

Layout:



Program : DBCon7.java

```
package test;
import java.util.*;
import java.io.*;
import java.sql.*;
public class DBCon7 {
	public static void main(String[] args) {
	  Scanner s = new Scanner(System.in);
	  try(s;){
	      Class.forName("oracle.jdbc.driver.OracleDriver");
	      Connection con = DriverManager.getConnection
	                  ("jdbc:oracle:thin:@localhost:1521:xe",
	                              "system","tiger");
	      System.out.println("Enter the id to retrieve Image:");
	      String id = s.nextLine();
	      PreparedStatement ps = con.prepareStatement
	                  ("select * from StreamTab67 where id=?");
	      ps.setString(1, id);
	      ResultSet rs = ps.executeQuery();
	      if(rs.next()) {
	            Blob b = rs.getBlob(2);
	            byte by[] = b.getBytes(1, (int)b.length());
	            System.out.println("Enter the fPath&fName to store
image(Destination):");
	            File f = new File(s.nextLine());
	            FileOutputStream fos = new FileOutputStream(f);
	            fos.write(by);
```

```
                System.out.println("Image retrieved successfully...");

                fos.close();
            }else {
                System.out.println("Invalid id...");
            }
        }catch(Exception e) {
            System.out.println(e.toString());
        }
    }
}
```

o/p:

Enter the id to retrieve Image:

A121

Enter the fPath&fName to store image(Destination):

E:\Images\PQR.jpg

Image retrieved successfully...

==========

faq:

define FileInputStream?

=>'FileInputStream' is a class from java.io package and which is

used to connect the file to read Byte Stream data.

syntax:

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

faq:

define FileOutputStream?

=>'FileOutputStream' is a class from java.io package and which is

used to create a new file with 0KB and connect the file to

write Byte Stream data.

syntax:

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

faq:

define getBlob() method?

=>getBlob()-method is from 'ResultSet-Interface' and which is used to create implementation object for 'java.sql.Blob' interface,this Blob-Object internally connected to stream-column of "ResultSet"

syntax:

Blob b = rs.getBlob(2);

faq:

define getBytes() method?

=>getBytes()-method is from 'java.sql.Blob' interface and which is used to convert stream into Byte-Array.

syntax:

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

==========

Dt : 7/10/2024(Day-12)

**imp*

3.CallableStatement:

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

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

=>This prepareCall()-method internally holding 'Anonymous Local InnerClass as implementation class 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("{call Proc/Func}");

---

faq:

define 'Procedure'?

=>Procedure is a set-of-queries executed at-a-time on Database product and after execution procedure will not return any value,which means procedure is NonReturn_type.

structure of Procedure:

create or replace procedure Procedure_name

(para_list) is

begin

```
 query-1
 query-2
 ...
end;
/
```

Types of Procedures:

=>According to JDBC,Procedures are categorized into three types:

(a)IN-Parameter Procedures

(b)OUT-Parameter Procedures

(c)IN-OUT Prameter Procedures

(a)IN-Parameter Procedures:

=>The Procedures which take the data from Java-Program and send to Database product are known as IN-Parameter Procedures.

(b)OUT-Parameter Procedures:

=>The Procedures which take the data from Database product and send to Java-Program are known as OUT-Parameter Procedures.

(c)IN-OUT Prameter Procedures:

=>The Procedures which take and give data are known as IN-OUT Parameter Procedures.

Creating and executing Procedure:(In-Parameter Procedure)

step-1 : Create the following tables related to Bank-Customer

BankCustomer67(accno,name,bal,acctype)

CustAddress67(accno,city,state,pincode)

CustContact67(accno,mid,phno)

```
create table BankCustomer67(accno number(15),name varchar2(15),bal number(10,2),
acctype varchar2(15),primary key(accno));

create table CustAddress67(accno number(15),city varchar2(15),state varchar2(15),
pincode number(10),primary key(accno));

create table CustContact67(accno number(15),mid varchar2(25),phno number(15),
primary key(accno));
```

step-2 : Construct Procedure to insert details into Cust-tables

```
create or replace procedure CreateAccount67
(acno number,nm varchar2,bl number,atype varchar2,cty varchar2,st varchar2,
pcode number,md varchar2,pno number) is
begin
 insert into BankCustomer67 values(acno,nm,bl,atype);
 insert into CustAddress67 values(acno,cty,st,pcode);
 insert into CustContact67 values(acno,md,pno);
end;
/
```

step-3 : Construct JDBC Application to execute Procedure to insert data to

Customer tables

Program : DBCon8.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon8 {
      public static void main(String[] args) {
         Scanner s = new Scanner(System.in);
         try(s;){
             Class.forName("oracle.jdbc.driver.OracleDriver");
             Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
             CallableStatement cs = con.prepareCall
                         ("{call CreateAccount67(?,?,?,?,?,?,?,?,?)}");

             System.out.println("Enter Cust-AccNo:");
             long accNo = Long.parseLong(s.nextLine());
             System.out.println("Enter Cust-Name:");
             String name = s.nextLine();
             System.out.println("Enter Cust-Balance:");
             float bal = Float.parseFloat(s.nextLine());
             System.out.println("Enter Cust-AccType:");
             String accType = s.nextLine();
             System.out.println("Enter Cust-City:");
             String city = s.nextLine();
             System.out.println("Enter Cust-State:");
             String state = s.nextLine();
             System.out.println("Enter Cust-PinCode:");
             int pinCode = Integer.parseInt(s.nextLine());
             System.out.println("Enter Cust-MailId:");
             String mId = s.nextLine();
             System.out.println("Enter Cust-PhoneNo:");
             long phNo = Long.parseLong(s.nextLine());

             //Load data to CallableStatement Object
             cs.setLong(1, accNo);
             cs.setString(2, name);
             cs.setFloat(3, bal);
             cs.setString(4, accType);
             cs.setString(5, city);
             cs.setString(6, state);
             cs.setInt(7, pinCode);
             cs.setString(8, mId);
```

```
            cs.setFloat(9, phNo);

            cs.execute();//Procedure executed
            System.out.println("Customer accout created successfully...");
            con.close();
        }catch(Exception e) {
            System.out.println(e.toString());
        }
    }
}
```

o/p:

Enter Cust-AccNo:

6123456

Enter Cust-Name:

Alex

Enter Cust-Balance:

5000

Enter Cust-AccType:

Savings

Enter Cust-City:

Hyd

Enter Cust-State:

TG

Enter Cust-PinCode:

506112

Enter Cust-MailId:

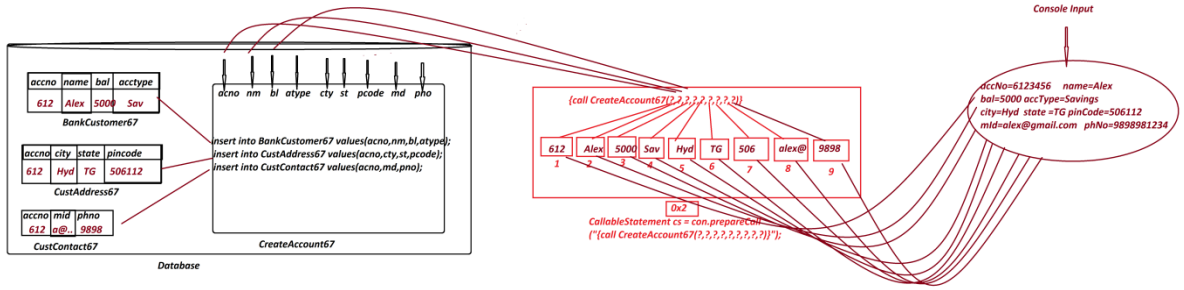
alex@gmail.com

Enter Cust-PhoneNo:

9898981234

Customer accout created successfully...

Layout:



==========

Dt : 8/10/2024(Day-13)

Construct and Execute IN-OUT Parameter Procedure:

step-1 : Create Procedure to display BankCustomer details based on accNo.

```
create or replace procedure RetrieveDetails67
(acno number,nm OUT varchar2,bl OUT number,atype OUT varchar2,cty OUT varchar2,
st OUT varchar2,pcode OUT number,md OUT varchar2,pno OUT number) is
begin
 select name,bal,acctype into nm,bl,atype from BankCustomer67 where accno=acno;
 select city,state,pincode into cty,st,pcode from CustAddress67 where accno=acno;
 select mid,phno into md,pno from CustContact67 where accno=acno;
end;
/
```

step-2 : Construct JDBC Application to execute procedure.

Program : DBCon9.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon9 {
      public static void main(String[] args) {
       Scanner s = new Scanner(System.in);
       try(s;){
         Class.forName("oracle.jdbc.driver.OracleDriver");
         Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
         CallableStatement cs = con.prepareCall
                     ("{call RetrieveDetails67(?,?,?,?,?,?,?,?,?)}");
         System.out.println("Enter the Cust-AccNo to display details:");
```

```
            long accNo = Long.parseLong(s.nextLine());
            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.INTEGER);
            cs.registerOutParameter(8, Types.VARCHAR);
            cs.registerOutParameter(9, Types.BIGINT);
            cs.execute();
            System.out.println("-------BankCustomer Details------");
            System.out.println("Cust-AccNo:"+accNo);
            System.out.println("Cust-Name:"+cs.getString(2));
            System.out.println("Cust-Bal:"+cs.getFloat(3));
            System.out.println("Cust-AccType:"+cs.getString(4));
            System.out.println("Cust-City:"+cs.getString(5));
            System.out.println("Cust-State:"+cs.getString(6));
            System.out.println("Cust-PinCode:"+cs.getInt(7));
            System.out.println("Cust-MailId:"+cs.getString(8));
            System.out.println("Cust-PhoneNo:"+cs.getLong(9));

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

o/p:

Enter the Cust-AccNo to display details:

6123456

-------BankCustomer Details------

Cust-AccNo:6123456

Cust-Name:Alex

Cust-Bal:5000.0

Cust-AccType:Savings

Cust-City:Hyd

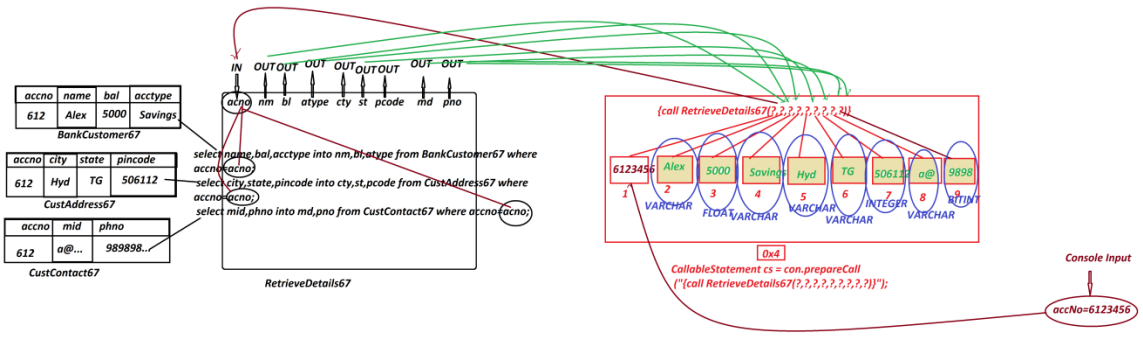
Cust-State:TG

Cust-PinCode:506112

Cust-MailId:alex@gmail.com

Cust-PhoneNo:9898981400

Diagram:



Assignment-1:

step-1 : Construct the following tables related to Student

StuDetails67(rollno,name,branch)

StuAddress67(rollno,city,state,pincode)

StuContact67(rollno,mid,phno)

StuMarks67(rollno,sub1,sub2,sub3,sub4,sub5,sub6)

StuResult67(rollno,totmarks,per,result)

Step-2 : Construct Procedure to insert Student-data

step-3 : Construct JDBC application to execute IN-Parameter Procedure

Note:

Read(Input) : rollNo,name,branch,city,state,pincode,mid,phno,6-sub Marks

Calculate : totMarks,per,result

Conditions:

=>All Subject marks must be in b/w 0 to 100,then only perform calculations and insert data to database table.

=>If any subject marks are in b/w 0 to 34 then result must be "Fail"

Assignment-2:

step-1 : Construct Procedure to display Student data based on rollNo.

step-2 : Construct JDBC Application to execute IN-OUT Parameter Procedure

==========

faq:

define 'Function'?

=>'Function' is also a set-of-queries executed at-a-time on database product and after execution Function will return the value.

=>we use 'return' statement to return the value after function execution.

Structure of Function:

```
create or replace function Function_name
(para_list) return data_type as var data_type;
begin
 //query-1
 //query-2
```

```
   ...
  //query-n
    return var;
end;
/
 ================================================================
```

Dt : 10/10/2024(Day-14)

**imp*

Creating and Executing Function on Database Product:

step-1 : Construct function to display balance of Customer based on accNo

```
create or replace function RetrieveBalance67
(acno number) return number as bl number;
begin
   select bal into bl from BankCustomer67 where accno=acno;
   return bl;
end;
/
```

step-2 : Construct JDBC Application to execute function.

Program : DBCon10.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon10 {
      public static void main(String[] args) {
       Scanner s = new Scanner(System.in);
       try(s;){
         Class.forName("oracle.jdbc.driver.OracleDriver");
         Connection con = DriverManager.getConnection
                   ("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
         CallableStatement cs = con.prepareCall
                     ("{call ?:=RetrieveBalance67(?)}");
         System.out.println("Enter the Cust-AccNo to retrieve balance:");
         long accNo = s.nextLong();

         cs.registerOutParameter(1, Types.FLOAT);
```

```
            cs.setLong(2, accNo);

            cs.execute();
            System.out.println("Cust-AccNo:"+accNo);
            System.out.println("Cust-Bal:"+cs.getFloat(1));

            con.close();
          }catch(Exception e) {
            System.out.println(e.toString());
          }
        }
}
```

o/p:

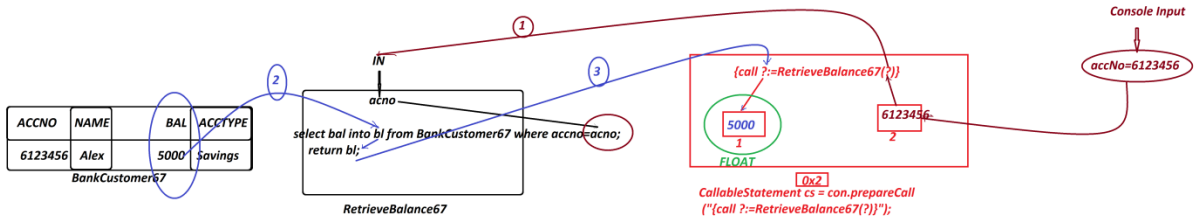
Enter the Cust-AccNo to retrieve balance:

6123456

Cust-AccNo:6123456

Cust-Bal:5000.0

Layout:



==========

Assignment:

Construct and execute Function to display percentage of student based on rollNo.

==========

****imp***

Transaction Management in JDBC:

define Transaction?

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

A - Atomicity

C - Consistency

I - Isolation

D - Durability

A - Atomicity

=>Atomicity means executing all statements in transaction at-a-time or not-at-all.

C - Consistency

=>The process in which the selected state of resources must remain same until the transaction is completed,is known as Consistency.

I - Isolation

=>The process in which multiple users are executed independently is known as Isolation.

D - Durability

=>The process in which the transcation details are stored and available,is known as Durability.

Transaction : Book Tickets using BookMyShow

1.Login process
2.Region
3.Select Movie
4.Select data
5.Select Show time
6.No of tickets
7.Select the tickets(Block the tickets)
8.Payment
(i)...
(ii)...
9.If Payment Successful,then tickets confirmed
10.Message(Mobile,Mail)
11.Logout

---

define Transaction Management?

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

=>we use the following methods in Transaction Management:

(a)getAutoCommit()

(b)setAutoCommit()

(c)setSavepoint()

(d)releaseSavepoint()

(e)commit()

(f)rollback()

(a)getAutoCommit():

=>getAutoCommit()-method is used get commit-status

syntax:

boolean k = con.getAutoCommit();

(b)setAutoCommit():

=>setAutoCommit()-method is used to set the commit-status to "true" or "false".

syntax:

con.setAutoCommit(false);

(c)setSavepoint():

=>setSavepoint()-method is used set a save-point for transaction rollback.

syntax:

Savepoint sp = con.setSavepoint();

(d)releaseSavepoint():

=>releaseSavepoint()-method is used to delete the save-point.

syntax:

con.releaseSavepoint();

(e)commit():

=>commit()-method is used to save permanently to database.

syntax:

con.commit();

(f)rollback():

=>rollback()-method is used to re-set the transaction when transaction failed.

syntax:

con.rollback(sp);

Dt : 14/10/2024(Day-15)

Ex:

DB-Table : Bank67(accno,name,bal,acctype)

Primary Key : accno

create table Bank67(accno number(15),name varchar2(15),bal number(10,2),

acctype varchar2(15),primary key(accno));

Insert two Customer details:

insert into Bank67 values(6123456,'Alex',12000,'savings');

insert into Bank67 values(313131,'Ram',500,'Savings');

*SQL> select * from Bank67;*

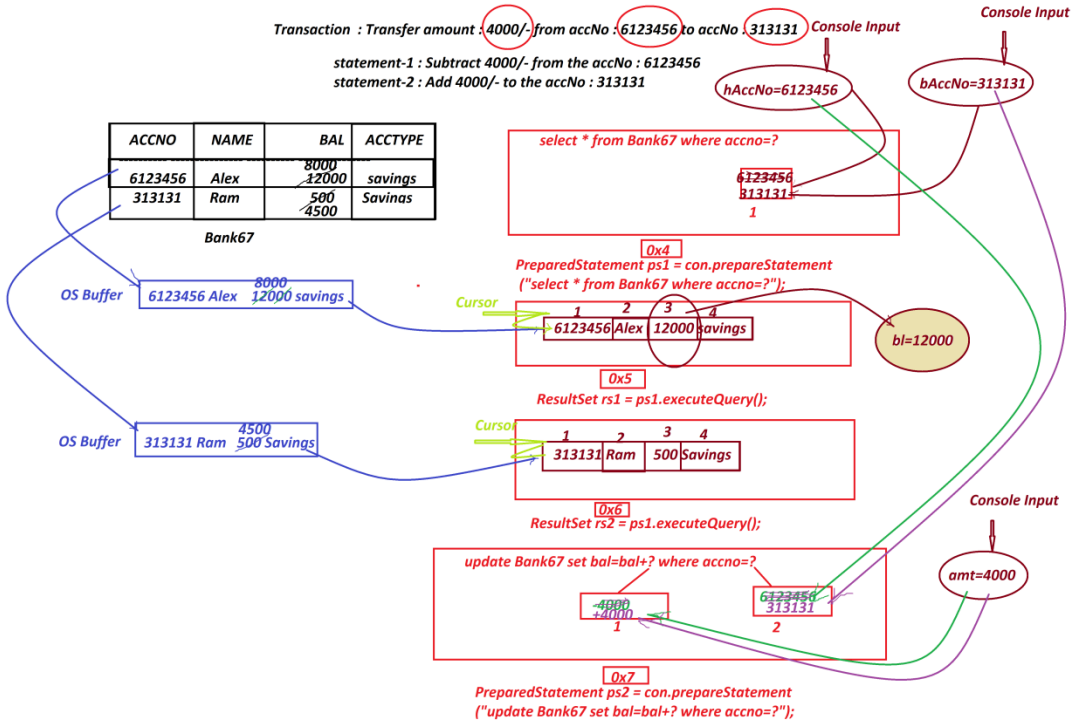
```
    ACCNO NAME            BAL ACCTYPE
---------- --------------- ---------- ---------------
   6123456 Alex            12000 savings
    313131 Ram             500 Savings
SQL>
```

Transaction : Transfer amount : 4000/- from accNo : 6123456 to accNo : 313131

statement-1 : Subtract 4000/- from the accNo : 6123456

statement-2 : Add 4000/- to the accNo : 313131

Layout:



Program : DBCon11.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon11 {
      public static void main(String[] args) {
       Scanner s = new Scanner(System.in);
       try(s;){
         Class.forName("oracle.jdbc.driver.OracleDriver");
         Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
         System.out.println("Commit-status : "+con.getAutoCommit());
         con.setAutoCommit(false);
         System.out.println("Commit-status : "+con.getAutoCommit());
           PreparedStatement ps1 = con.prepareStatement
                    ("select * from Bank67 where accno=?");
           PreparedStatement ps2 = con.prepareStatement
                    ("update Bank67 set bal=bal+? where accno=?");
         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 bl = rs1.getFloat(3);
               System.out.println("Available bal : "+bl);
               System.out.println("Enter the benifiecieryAccNo:");
               long bAccNo = s.nextLong();
               ps1.setLong(1, bAccNo);
               ResultSet rs2 = ps1.executeQuery();
               if(rs2.next()) {
                     System.out.println("Enter the amount to be
transferred:");
                     float amt = s.nextFloat();
                     if(amt<=bl) {
                          //statement-1 : Subtract 4000/- from the accNo :
6123456
                           ps2.setFloat(1,-amt);
                           ps2.setLong(2, hAccNo);
                           int i = ps2.executeUpdate();//Buffer Updated

                    //statement-2 : Add 4000/- to the accNo : 313131
                           ps2.setFloat(1, +amt);
                           ps2.setLong(2, bAccNo);
                           int j = ps2.executeUpdate();//Buffer Updated

                           if(i==1 && j==1) {
                                 System.out.println("Transaction is
Successfull....");
                                 con.commit();//Database updated permanently
                           }else {
                                System.out.println("Transaction failed....");
                                con.rollback(sp);
                           }
                     }else {
                           System.out.println("Insufficient fund...");
                     }
               }else {
                     System.out.println("Invalid benifiecieryAccNo....");
               }
          }else {
                System.out.println("Invalid homeAccNo....");
          }
          con.close();
        }catch(Exception e) {
          System.out.println(e.toString());
        }
       }
}
```

o/p:

Commit-status : true

Commit-status : false

Enter the HomeAccNo:

6123456

Available bal : 12000.0

Enter the benifiecieryAccNo:

313131

Enter the amount to be transferred:

4000

Transaction is Successfull....

==========

Assignment:

Modify the above program by displaying the following messages from catch-block

"Invalid hAccNo..."

"Invalid bAccNo..."

"Insufficient Fund..."

==========

Dt : 15/10/2024(Day-16)

**imp*

Batch Processing in JDBC:

faq:

define Batch Processing?

=>The process of collecting multiple queries as a batch and executed at-a-time on database product is known as Batch Processing.

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

(a)addBatch()

(b)executeBatch()

(c)clearBatch()

(a)addBatch():

=>addBatch()-method is used to add query to a batch.

Method Signature:

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

syntax:

stm.addBatch("NonSelect-Query");

(b)executeBatch():

=>executeBatch()-method is used to execute all queries from the batch at-a-time.

Method Signature:

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

syntax:

int k[] = stm.executeBatch();

(c)clearBatch():

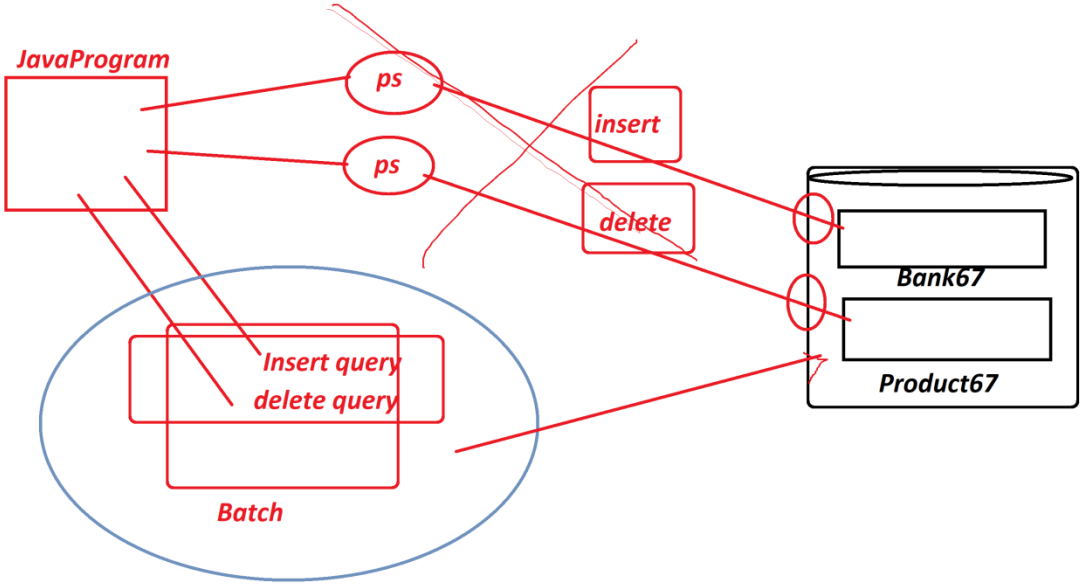
=>clearBatch()-method is used to delete all queries from the batch and destroy the batch.

Method Signature:

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

syntax:

stm.clearBatch();



Ex:

Program : DBCon11.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon12 {
      public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        try(s;){
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection
```

```
("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
            Statement stm = con.createStatement();
            System.out.println("------Add Customer to Bank67-----");
            System.out.println("Enter the Cust-accNo:");
            long accNo = Long.parseLong(s.nextLine());
            System.out.println("Enter the Cust-Name:");
            String name = s.nextLine();
            System.out.println("Enter the Cust-Bal:");
            float bal = Float.parseFloat(s.nextLine());
            System.out.println("Enter the Cust-accType:");
            String accType = s.nextLine();
            stm.addBatch
            ("insert into Bank67
values("+accNo+",'"+name+"',"+bal+",'"+accType+"')");

            System.out.println("-------Delete product details based on
code(Product67)----");
            System.out.println("Enter the Product code to delete product
details:");
            String pCode = s.nextLine();
            stm.addBatch
            ("delete from Product67 where code='"+pCode+"'");

            int k[] = stm.executeBatch();
            for(int i : k)
            {
                  System.out.println("query executed : "+i);
            }//end of loop
            stm.clearBatch();
            con.close();
        }catch(Exception e) {
            //System.out.println(e.toString());
            e.printStackTrace();
        }
      }
}
```

o/p:

------Add Customer to Bank67-----

Enter the Cust-accNo:

232323

Enter the Cust-Name:

Rajj

Enter the Cust-Bal:

8000

Enter the Cust-accType:

Savings

-------Delete product details based on code(Product67)----

Enter the Product code to delete product details:

A123

query executed : 1

query executed : 1

----

faq:

wt is the advantage of Batch Processing?

=>In Batch Processing the execution-control reaches database product only once and execute all queries at-a-time,which saves the execution time and generate high performance of an application.

Note:

(i)Through Batch Processing,we can execute only NonSelect queries and which is also known as Batch Update processing.

(ii)'Statement' is preferable to perform batch processing,because we can perform operations on multiple database tables.

==========

faq:

define Meta Data?

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

(Meta Data means data about data)

=>According to JDBC,Meta data means one object holding information about other

Object known as Meta data object(Meta data component)

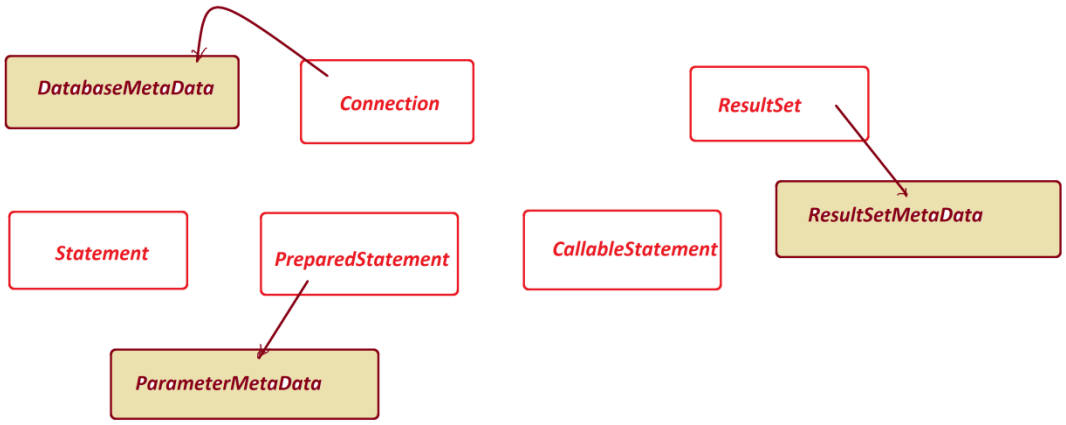
=>The following are some important Meta data components in JDBC:

1.DatabaseMetaData

2.ParameterMetaData

3.ResultSetMetaData

Diagram:



1.DatabaseMetaData:

=>DatabaseMetaData is an interface from java.sql package and which is instantiated

using getMetaData()-method from 'Connection' interface.

=>This DatabaseMetaData-Object will hold information about Connection-Object.

syntax:

DatabaseMetaData dmd = con.getMetaData();

2.ParameterMetaData:

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

using getParameterMetaData()-method from 'PreparedStatement' interface.

=>This ParameterMetaData-Object will hold information about PreparedStatement

Object.

syntax:

ParameterMetaData pmd = ps.getParameterMetaData();

3.ResultSetMetaData:

=>ResultSetMetaData is an interface from java.sql package and which is instantiated

using getMetaData()-method from ResultSet-Interface.

=>This ResultSetMetaData-object will hold information about ResultSet-Object.

syntax:

ResultSetMetaData rsmd = rs.getMetaData();

Dt:16/10/2024(Day-17)

Ex:

Program : DBCon13.java

```
package test;
import java.sql.*;
public class DBCon13
{
	public static void main(String[] args)
	{
	try {
	  Class.forName("oracle.jdbc.driver.OracleDriver");
	  Connection con = DriverManager.getConnection
	              ("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
	  System.out.println("******DatabaseMetaData******");
	  DatabaseMetaData dmd = con.getMetaData();
	  System.out.println("Product-Name:"+dmd.getDatabaseProductName());
	  System.out.println("Driver-Name:"+dmd.getDriverName());
	  PreparedStatement ps1 = con.prepareStatement
	              ("insert into Bank67 values(?,?,?,?)");
	  System.out.println("*****ParameterMetaData*****");
	  ParameterMetaData pmd = ps1.getParameterMetaData();
	  System.out.println("Parameter-Count:"+pmd.getParameterCount());
	  PreparedStatement ps2 = con.prepareStatement
	              ("select code,price,qty from Product67");
	  ResultSet rs = ps2.executeQuery();
	  System.out.println("*****ResultSetMetaData*****");
	  ResultSetMetaData rsmd = rs.getMetaData();
	  System.out.println("Column-Count:"+rsmd.getColumnCount());
	  System.out.println("2nd Column-Name:"+rsmd.getColumnName(2));
	  con.close();
	}catch(Exception e) {
	  e.printStackTrace();
	}
	}
}
```

o/p:

*******DatabaseMetaData******

Product-Name:Oracle

Driver-Name:Oracle JDBC driver

*****ParameterMetaData*****

Parameter-Count:4

********ResultSetMetaData********

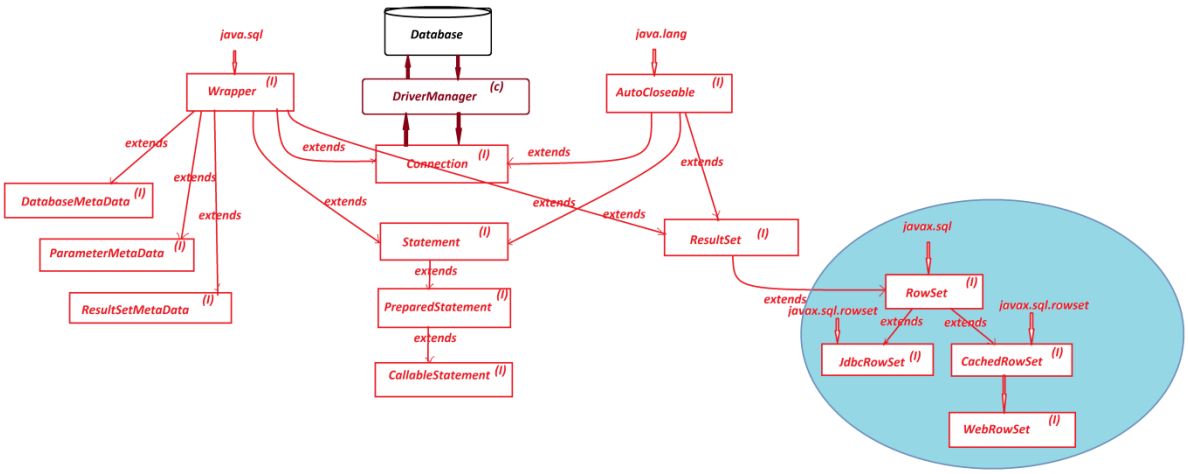
Column-Count:3

2nd Column-Name:PRICE

==========

****imp***

Hierarchy of JDBC Components:



==========

faq:

define 'Wrapper' in JDBC?

=>'Wrapper' is an interface from java.sql package and,which specify the JDBC components binded to database product or not.

=>The following are two impotant methods from 'Wrapper':

(i)unwrap()

(ii)isWrapperFor()

(i)unwrap():

=>unwrap()-method will dis-connect the component from Database product.

(ii)isWrapperFor():

=>isWrapperFor()-method will check the componet is binded to database product or not.

---

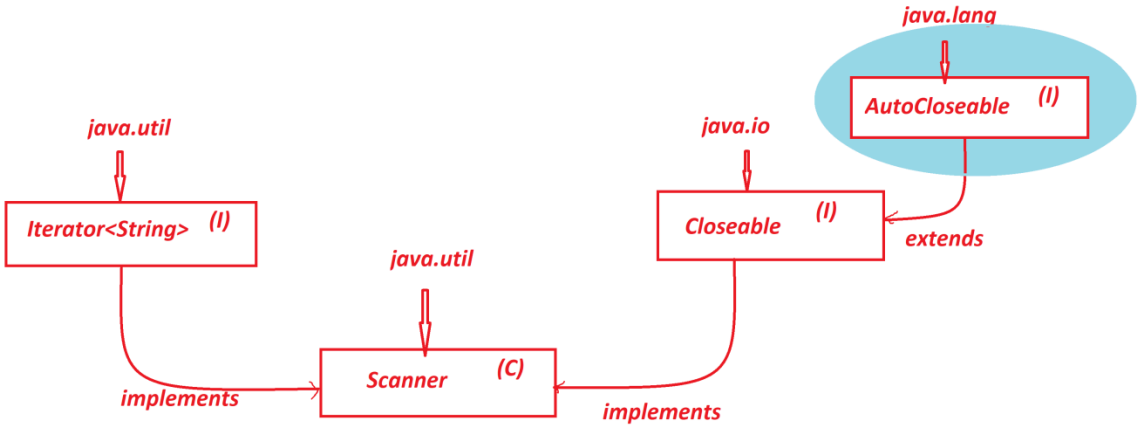
faq:

define 'AutoCloseable'?

=>'AutoCloseable' is an interface from 'java.lang' package and which support Auto-closing operation in try-with-resourse-statement of Exception handling process.

=>The resource-components which are used in try-with-resource-statement must be implemented from 'AutoCloseable' interface.

Hierarchy of Scanner-class:



==========

Dt : 17/10/2024(Day-18)

Program : DBCon14.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon14 {
	public static void main(String[] args) {
	   Scanner s = new Scanner(System.in);
	 try(s;)
	 {
	   Class.forName("oracle.jdbc.driver.OracleDriver");
	   Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
	   try(con;){
		Statement stm = con.createStatement();
		try(stm;){
			System.out.println("Enter the
NonSelect-query(create,insert,update,delete)");
			String qry = s.nextLine();
			int k = stm.executeUpdate(qry);
			System.out.println("The value in k : "+k);
			if(k>=0) {
				System.out.println("query executed
Successfully....");
			}
		}//end of try
	   }//end of try
	 }//end of try
	 catch(Exception e)
	 {
	   e.printStackTrace();
	 }
	}
}
```

o/p:

Enter the NonSelect-query(create,insert,update,delete)

create table Emp67(eid varchar2(10),ename varchar2(15),edesg varchar2(15),primary key(eid))

The value in k : 0

query executed Successfully....

Enter the NonSelect-query(create,insert,update,delete)

insert into Emp67 values('A121','Alex','SE')

The value in k : 1

query executed Successfully....

Enter the NonSelect-query(create,insert,update,delete)

update Emp67 set edesg='SE' where eid='E231'

The value in k : 1

query executed Successfully....

Enter the NonSelect-query(create,insert,update,delete)

delete from Emp67 where eid='D231'

The value in k : 1

query executed Successfully....

==========

faq:

define 'RowSet'?

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

'java.sql.ResultSet' interface.

(RowSet is a Child-Interface of ResultSet)

=>RowSet-Object also will hold the result generated from Select-queries.

=>RowSet-Objects are automatically Scrollable-Objects.

=>This 'RowSet' is categorized into two types:

1.JdbcRowSet

2.CachedRowSet

1.JdbcRowSet:

=>After getting the data from database product,JdbcRowSet-object still connected to database product.

2.CachedRowSet:

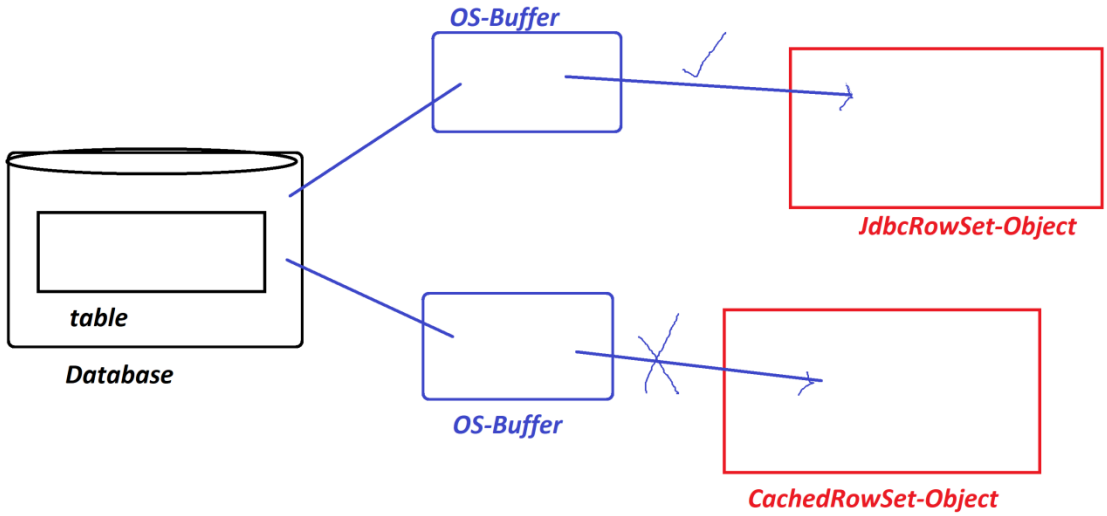
=>After getting the data from database product,CachedRowSet-object is disConnected from database product automatically.

Note:

=>In realtime,we use CachedRowset in the form of WebRowSet for getting the data from Database product.

java - represent Java-Library

javax - represent Extended-Java-Library



==========

*imp

Types of JDBC drivers:

=>These JDBC drivers are categorized into four types:

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

2.Native API driver(Type-2 driver)

3.Network Protocol driver(Type-3 driver)

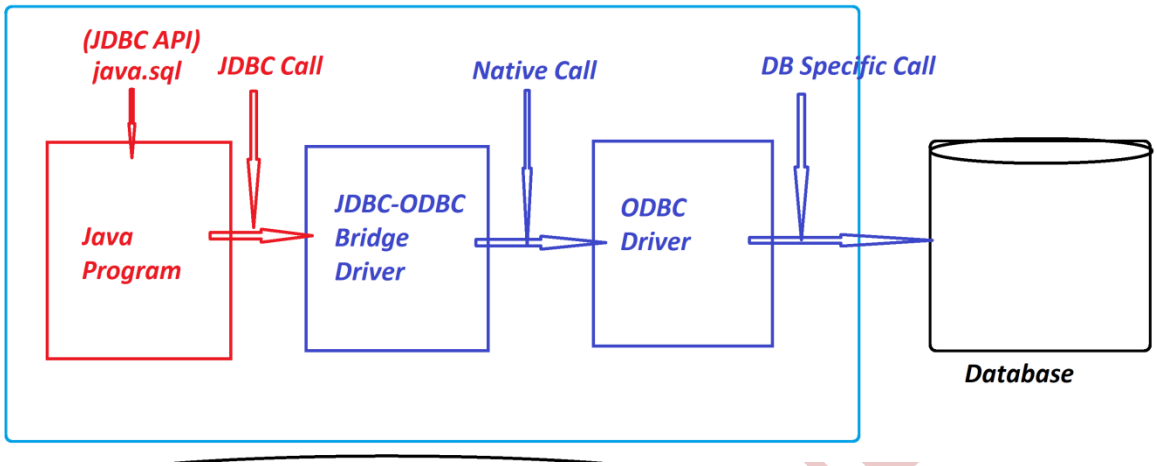
4.Thin driver(Type-4 driver)

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

=>Type-1 diver will take the support of ODBC-driver to establish connection to Database product.

=>In Type-1 driver,internally JDBC-Call converted into Native call,and this Native call is converted into DB Specific call for connection.

Diagram:



DisAdvantage:

=>Type-1 driver internally uses more conersions,and which waste the execution time

and degrades the performance of an application.

Note:

=>From Java8 version(2014) onwards Type-1 driver support is not available.

faq:

define ODBC driver?

=>ODBC stands for 'Open DataBase Connectivity' and which is used to communicate

with any type database.

=>ODBC-driver internally uses c/c++ code and which is PlatForm dependent driver.

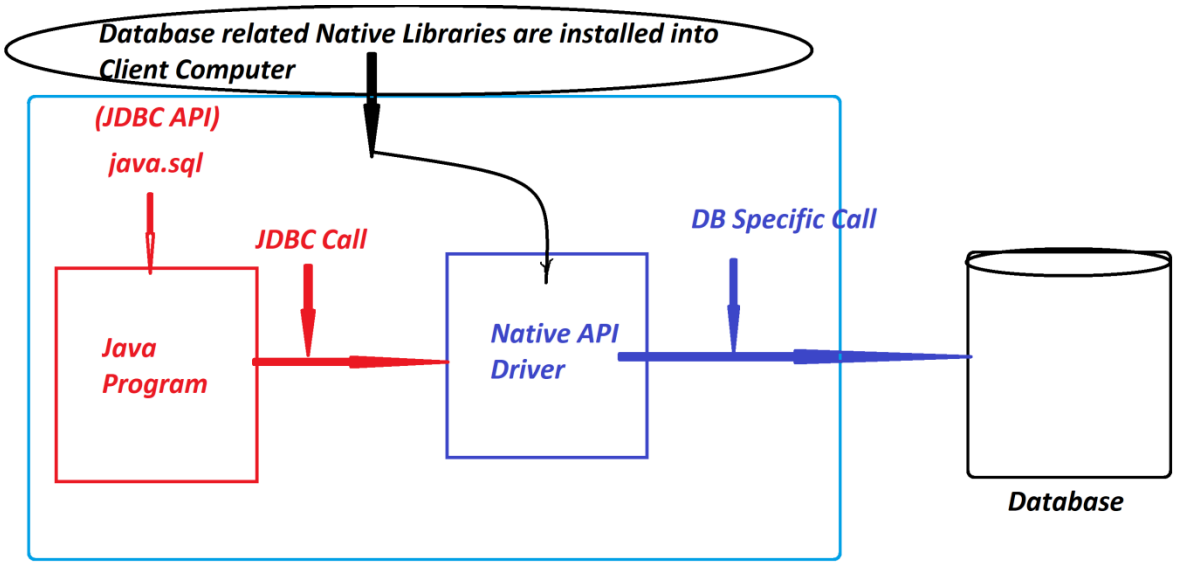
2.Native API driver(Type-2 driver):

=>Type-2 driver will take the support of 'Database related Native Libraries' to

establish connection to Database product.

=>when we want to use Type-2 driver,the Client Computer must be installed with 'Database related Native libraries'

Diagram:



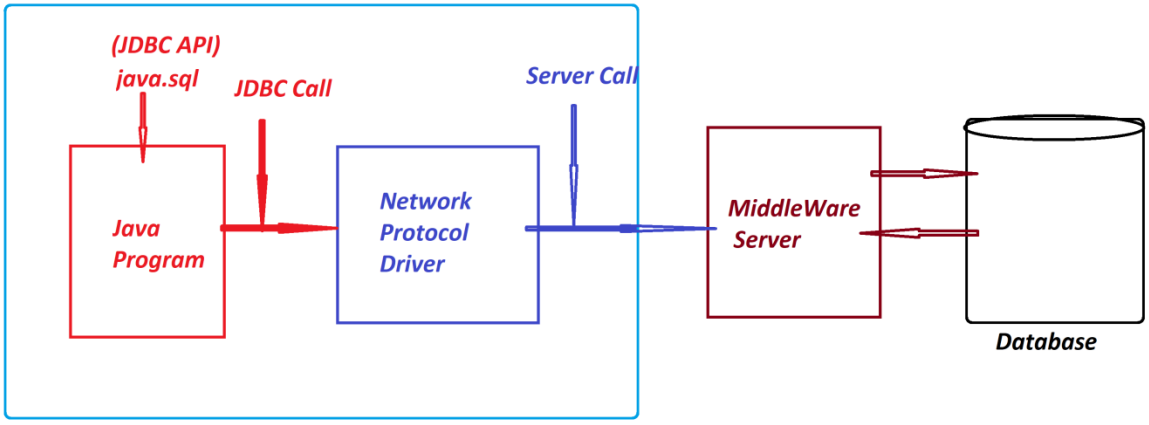
DisAdvantage:

=>when we use Type-2 driver,the application will become Database dependent and which is not preferable in realtime.

3.Network Protocol driver(Type-3 driver):

=>Type-3 driver will take the support of intermediate Middle-Ware servers to communicate with database product.

Diagram:



DisAdvantage:

=>In Type-3 driver Network related code and components are involved,and execution time increases and degrades the performance of an application.

4.Thin driver(Type-4 driver):

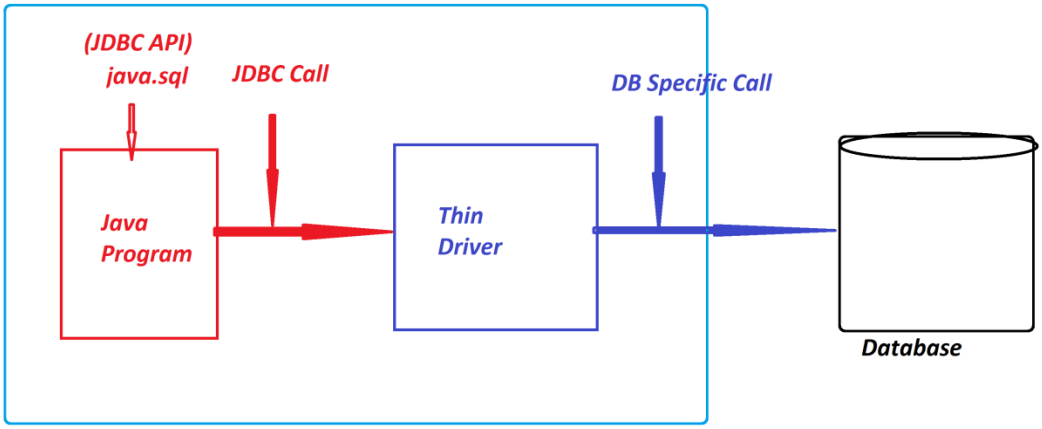
=>Type-4 driver will take the support of JDBC-Network-Protocol to establish Connection to Database product

=>In Type-4 driver(Thin driver) the JDBC-call converted directly into DB Specific call for connection.

=>Type-4 driver is pure Java driver and PlatForm independent driver.

=>This Type-4 driver is HighPerformance driver.

Diagram:



==========

Dt : 18/10/2024(Day-19)

Summary of Objects:

CoreJava:

1.User defined class Objects

2.String-Objects

3.WrapperClass-Objects

4.Array-Objects

5.Collection<E>-Objects

6.Map<K,V>-Objects

7.Enum<E>-Objects

JDBC:

1.Connection Object

2.Statement Object

3.PreparedStatement Object

4.CallableStatement Object

5.ResultSet Objects

(i)Scrollable ResultSet Objects

(ii)NonScrollable ResultSet Objects

6.Metadata-Objects

(i)DatabaseMetaData Object

(ii)ParameterMetaData Object

(iii)ResultSetMetaData Object

7.RowSet Objects

(i)JdbcRowSet Object

(ii)CachedRowSet Object

==========

faq:

wt is the diff b/w

(a)JavaSE

(b)JavaEE

(c)JavaME

(a)JavaSE:

=>JavaSE means 'Java Standard Edition' and which is used for basic application development,which means used to development NonServer Applications.

Ex:

CoreJava + JDBC

(b)JavaEE:

=>JavaEE means 'Java Enterprise Edition' and which is used to develop Server Applications like Web Applications and Enterprise Applications.

Ex:

Servlet,JSP,WebServices,...

(c)JavaME:

=>JavaME means 'Java Micro Edition' and which is used to development Embeded applications and Mobile Applications.

(Machine Related Applications and Mobile Application)

==========

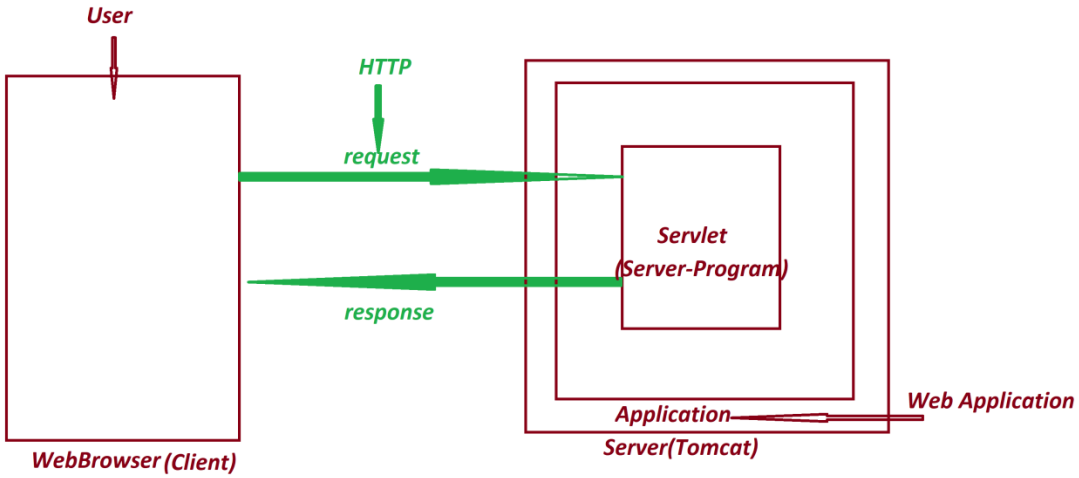
*imp

Servlet Programming:

define Servlet?

=>Servlet is a Server-program,which accepts the request from user through WebBrowser and provides the response

Diagram:



faq:

define Client?

=>The user who raises request to server is known as Client.

faq:

define Server?

=>Server means service provider,which means accepts the request and provide the response.

=>These Servers internally provide 'WebContainer' to execute WebApplications.

Types of Servers:

=>Servers are categorized into two types:

1.Web Servers

2.Application Servers

1.Web Servers:

=>Web Servers will provide only 'WebContainer' to execute WebApplications.

=>WebServers are preferable to execute static content.

=>Web Servers will accepts requests from HTTP protocol.

EX:

Tomcat

2.Application Servers:

=>Application Servers will provide both 'Web Container' and 'EJB container'.

(EJB - Enterprise Java Bean)

=>Application Servers will execute both 'Web Applications' and 'Enterprise Applications'(Enterprise Distributed Applications)

=>Application Servers are preferable for dynamic content.

=>Application Servers will accept the request from HTTP,RPC and RMI protocols.

(RPC - Remote Procedure Call)

(RMI - Remote Method Invocation)

Ex:

WebLogic

WebSphere

JBoss

...

==========

Dt : 19/10/2024(Day-20)

**imp*

Installing Tomcat Server:

step-1 : Download Tomcat10 version from 'APHACHE Organization'

Link=>

https://tomcat.apache.org/download-10.cgi

step-2 : Install Tomcat Server

=>While Installation process,

Select the type of Install : Full (click Next)

Server Shutdown port : 8089

HTTP/1.1 Connector port : 8081 or 8082 or 8083 ....

User Name : V

Password : nit

(Click Next)

C:\Tomcat 10.1

step-3 : Start the Tomcat Server

Click 'startup' or 'Tomcat10w' file from 'bin' folder of Tomcat to

start the server.

C:\Tomcat 10.1\bin

step-4 : Open WebBrowser(Client) check the Tomcat-server is

responding or not,using the following URL

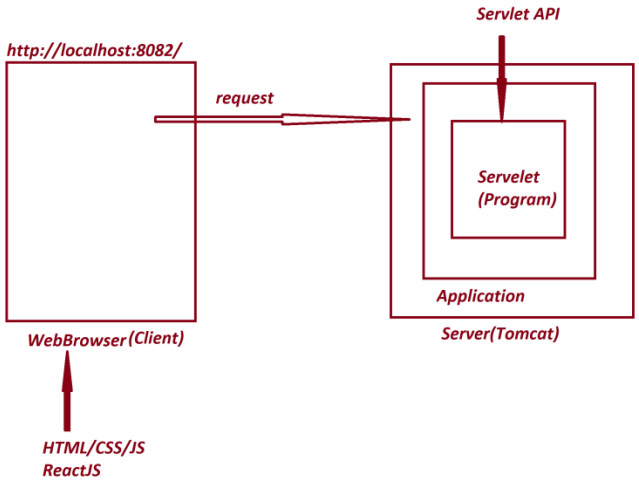
http://localhost:8082/

step-5 : Stop the Tomcat Server

Click 'shutdown' or 'Tomcat10w' file from 'bin' folder of

Tomcat to stop the server

C:\Tomcat 10.1\bin



==========

****imp***

Servlet API:

=>'jakarta.servlet' package is known as Servlet-API from Tomcat10

version onwards.(up to Tomcat9 version we use 'javax.servlet')

=>This 'jakarta.servlet' will provide 'classes and interfaces'

used in Servlet application development.

=>'jakarta.servlet.Servlet' interface is the root of Servlet-API

=>The following are some important methods from 'Servlet'

interface:

1.init()

2.service()

3.destroy()

4.getServletInfo()

5.getServletConfig()

1.init():

=>init()-method is used to perform initialization process,which

means making the programming components ready for execution.

Method Signature:

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

2.service():

=>service()-method is used to take the request from user and

provide the response.

Method Signature:

```
public abstract void service(jakarta.servlet.ServletRequest,
```

jakarta.servlet.ServletResponse) throws

jakarta.servlet.ServletException, java.io.IOException;

3.destroy():

=>destroy()-method is used to perform service-closing operations.

Method Signature:

public abstract void destroy();

4.getServletInfo():

=getSevletInfo()-method will hold servlet information.

(purpose of Servlet program)

Method Signature:

public abstract java.lang.String getServletInfo();

5.getServletConfig():

=>getServletConfig()-method will hold configuration details.

Method Signature:

public abstract jakarta.servlet.ServletConfig

getServletConfig();

Dt : 20/10/2024(Day-21)

Note:

=>In the process of Constructing Servlet-program,the program must be implemented from 'jakarta.servlet.Servlet' interface.

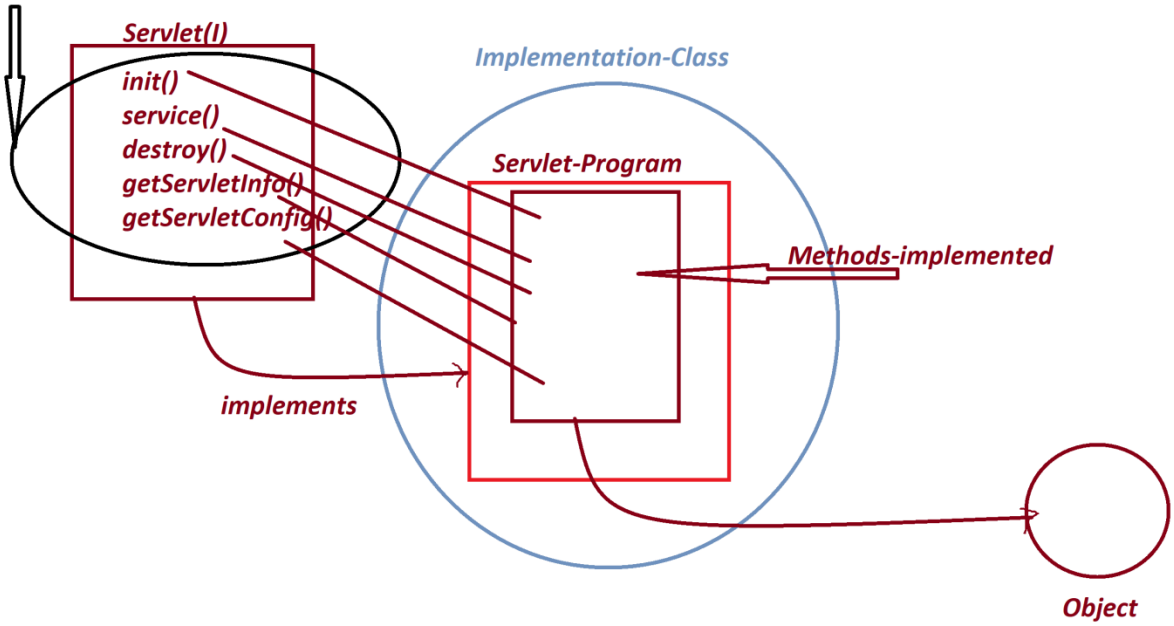
=>This implemented servlet-program must implement(construct-body) all abstract methods of Servlet-Interface.

=>Server will instantiate(creating Object) Servlet-program automatically when loaded for execution.

=>Server will execute the following Life-Cycle methods automatically:

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

AbstractMethoda



**imp*

Construct Servlet Application(Web Application) using IDE Eclipse:

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

click 'Launch'

step-2 : Create 'Dynamic Web Project'

Click on File->New->Project->Web->select 'Dynamic Web Project' and

click 'Next'->name the project and click 'Finish'

step-3 : Add 'servlet-api.jar' file to project

RightClick on Project->Build path->Configure Build path->Libraries->

select 'Classpath' and click 'Add External Jars'->Browse and select

'servlet-api.jar' from "lib" folder of "Tomcat"->Open->Apply->

'Apply and Close'

step-4 : Add server(Tomcat) to IDE Eclipse(One time process)

Click on 'servers'->click 'Click this link to create New Server'->

selcet the server(tomcat-server) and click 'Next'->

Browse and select 'Tomcat Installation Directory'->select Folder->

click 'Finish'

step-5 : Construct HTML file to read input to Servlet-program

RightClick on webapp->New->HTML file->name the file and click

'Finish'

employee.html

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

step-6 : Create package in 'src/main/java' of Java Resources

step-7 : Create Servlet-Program in package

DisplayServlet.java

```
package test;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
@WebServlet("/dis")
public class DisplayServlet implements Servlet
{
        public void init(ServletConfig scf) throws ServletException
  {
          //NoCode
  }
        public void service(ServletRequest req,ServletResponse res) throws
                                ServletException,IOException
        {
                String eId = req.getParameter("eid");
                String eName = req.getParameter("ename");
                String eDesg = req.getParameter("edesg");
           PrintWriter pw = res.getWriter();
           res.setContentType("text/html");
           pw.println("=======EmployeeDetails======");
           pw.println("<br>Emp-Id:"+eId);
           pw.println("<br>Emp-Name:"+eName);
```

```
            pw.println("<br>Emp-Desg:"+eDesg);

        }

        public void destroy()

        {

                //NoCode

        }

         public String getServletInfo()

         {

                return "Servlet displaying Employee details";

         }

         public ServletConfig getServletConfig()

         {

                return this.getServletConfig();

         }


}
```

step-8 : Create 'web.xml' in 'WEB-INF'(Mapping file)

RightClick on WEB-INF->New->Other->XML->XML file->name the file as

'web.xml' and click 'Finish'

web.xml

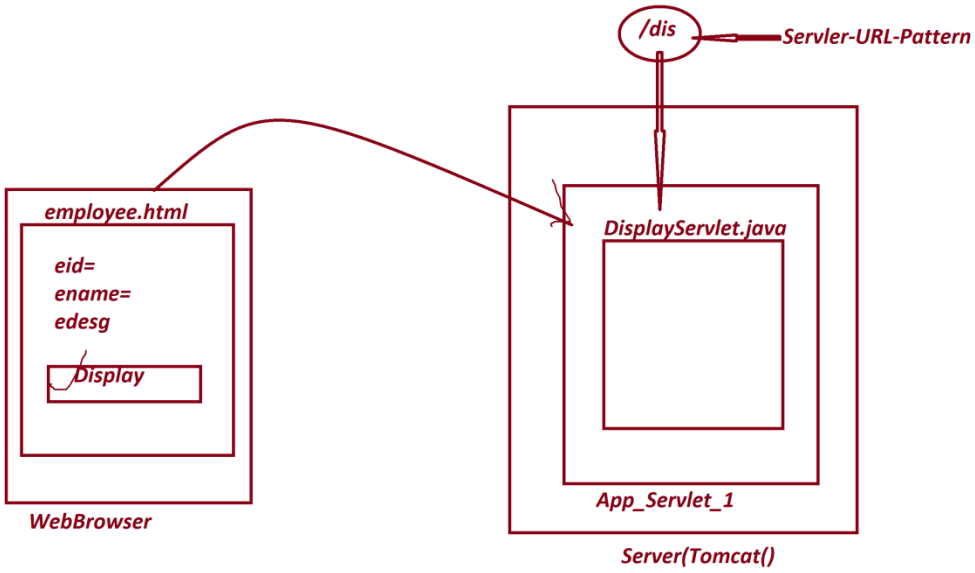
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <welcome-file-list>
      <welcome-file>employee.html</welcome-file>
  </welcome-file-list>
</web-app>
```

step-9 : Execute the Application

RightClick on Project->Run AS->Run On Server->select the Server and

click 'Finish'

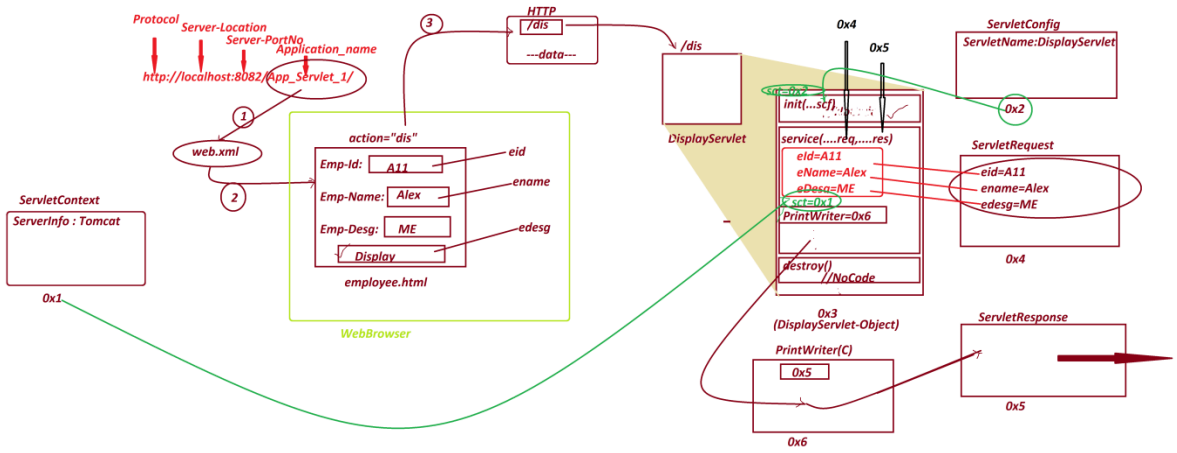
http://localhost:8082/App_Servlet_1/



Dt : 21/10/202(Day-22)

Execution flow of above application:

http://localhost:8082/App_Servlet_1/



ServletContext:

=>'ServletContext' is an interface from jakarta.servlet package and which is instantiated automatically when WebApplication(Project) is deployed into Server.

=>This ServletContext-object is loaded with Server-Information.

=>we use getServletContext()-method from 'ServletRequest' to access the reference of ServletContext-Object

Method Signature of getServletContext();

public abstract jakarta.servlet.ServletContext getServletContext();

syntax:

ServletContext sct = req.getServletContext();

ServletConfig:

=>'ServletConfig' is an interface from jakarta.servlet package and which is instantiated automatically when servlet-program loaded for execution.

=>ServletConfig-Object is loaded with Servlet_name

ServletRequest:

=>'ServletRequest' is an interface from jakara.servlet package and which is instantiated Automatically while service()-method execution.

=>ServletRequest-Object is loaded with HTML form data.

ServletResponse:

=>'ServletResponse' is an inerface from jakarta.servlet package and which is instantiated automaticallu while service()-method execution.

=>'ServletResponse-Object' is loaded with the data which we are sending as respose/output.

faq:

define getWriter()-method?

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

=>This 'PrintWriter' object will hold thr reference of ServletResponse Object. (HAS-A relation)

Method Signaturre of getWriter():

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

syntax:

PrintWriter pw = res.getWriter();

faq:

define setContentType() method?

=>setContentType()-method will specify the type of data we are sending as response.

=>This setContentType()-method is from 'ServletResponse'

Method Signature:

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

syntax:

res.setContentType("text/html");

faq:

define getParameter()-method?

=>getParameter()-method is from 'ServletRequest' and which is used to get parameter values from ServletRequest Object.

Method Signature:

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

syntax:

String var = req.getParameter("para-name");

Ex:

String eId = req.getParameter("eid");

String eName = req.getParameter("ename");

String eDesg = req.getParameter("edesg");

Program:DisplayServlet.java(Modified Code)

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.annotation.*;

@WebServlet("/dis")

public class DisplayServlet implements Servlet

{

        public ServletConfig scf = null;//Instance Variable(Object Variable)

        public void init(ServletConfig scf) throws ServletException

  {

          this.scf=scf;

  }

        public void service(ServletRequest req,ServletResponse res) throws

                                ServletException,IOException

        {

                String eId = req.getParameter("eid");

                String eName = req.getParameter("ename");

                String eDesg = req.getParameter("edesg");

                ServletContext sct = req.getServletContext();

          PrintWriter pw = res.getWriter();

          res.setContentType("text/html");

          pw.println("=======EmployeeDetails======");

          pw.println("<br>Emp-Id:"+eId);

          pw.println("<br>Emp-Name:"+eName);

          pw.println("<br>Emp-Desg:"+eDesg);
```

```
        pw.println("<br>======ServletContext======");
        pw.println("<br>Server-Info:"+sct.getServerInfo());
        pw.println("<br>======ServletConfig======");
        pw.println("<br>Servlet-name:"+scf.getServletName());;
    }
    public void destroy()
    {
        //NoCode
    }
    public String getServletInfo()
    {
        return "Servlet displaying Employee details";
    }
    public ServletConfig getServletConfig()
    {
        return this.getServletConfig();
    }

}
```

o/p:

=======EmployeeDetails======

Emp-Id:A11

Emp-Name:Alex

Emp-Desg:ME

======ServletContext======

Server-Info:Apache Tomcat/10.1.20

======ServletConfig======

Servlet-name:test.DisplayServlet

==========

Assignment-1:

Construct Servlet Application to read and display Product details.

(code,name,price,qty)

Assignment-2:

Construct Servlet Application to read and display UserDetails.

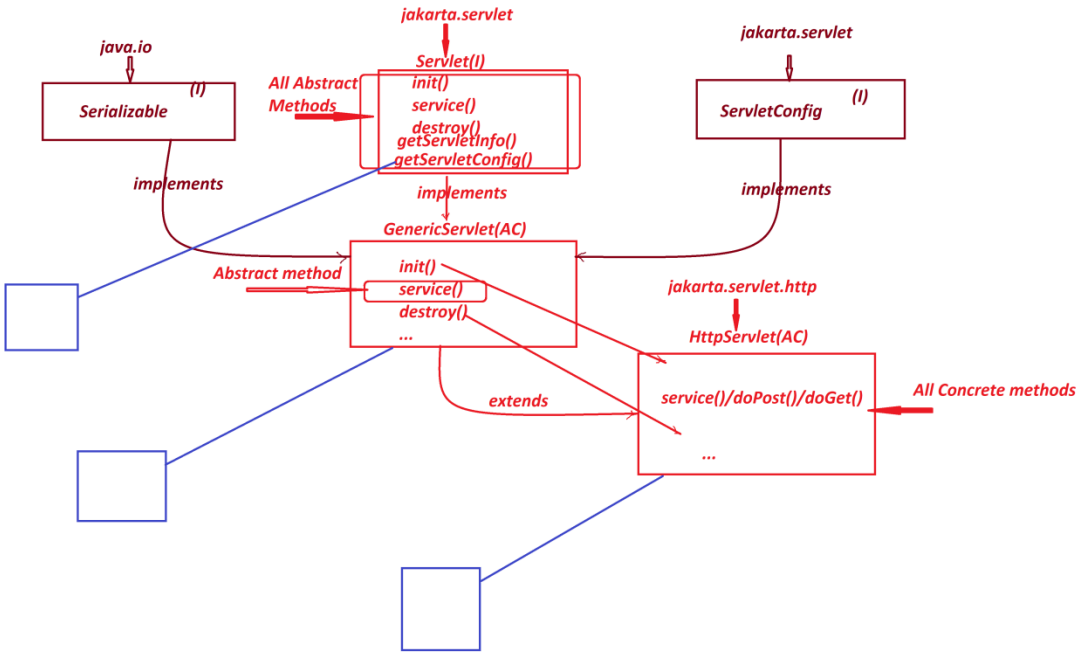
(name,city,mid,phno)

==========

Dt : 22/10/2024(Day-23)

**imp*

Hierarchy of Servlet-API:



==========

Note::

=>In the process of constructing Servlet-program,we can use any one of the following design model:

1.Using 'Servlet' interface

2.Using 'GenericServlet' AbstractClass

3.Using 'HttpServlet' AbstractClass

1.Using 'Servlet' interface:

=>In the process of construcing Servlet-program,the program must be implemented

from Servlet-Interface.

=>In this design model,we must implement all abstract methods of Servlet-Interface.

2.Using 'GenericServlet' AbstractClass:

=>In the process of Constructing Servlet-Program,the program must be extended from 'GenericServlet' AbstractClass.

=>In this design model,we must implement service()-method and remaing two Life-cycle methods are optional methods.

3.Using 'HttpServlet' AbstractClass:

=>In the process of Constructing Servlet-Program,the program must be extended from 'HttpServlet' AbstractClass.

=>In this design model,all methods are optional methods.

=>Life-Cycle methods:

init()

service()/doPost()/doGet()

destroy()

Note::

service() : will accept both requests(GET and POST)

doPost() : will accept only POST request

doGet() : will accept only GET request

==========

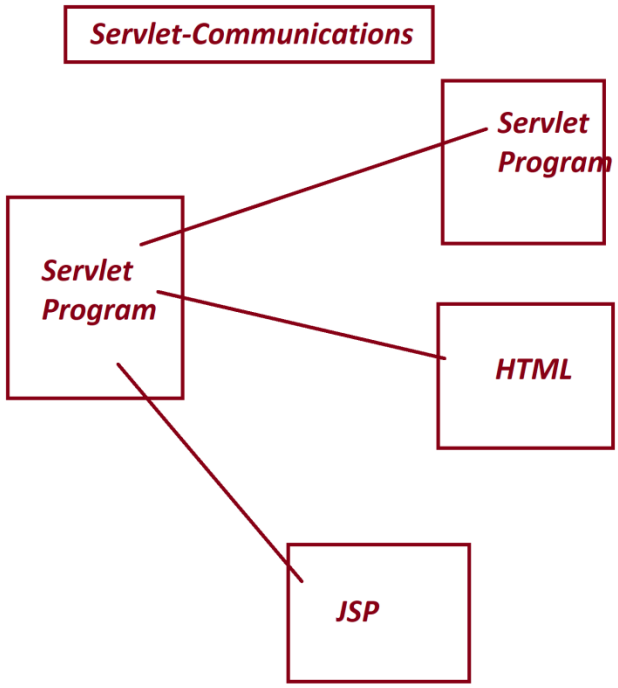
**imp*

'RequestDispacher' in Servlet Programming:

=>'RequestDispatcher' is an interface from jakarta.servlet package and which is

used to perform Servlet-Communcations like Servlet to Servlet Communication,

Servlet to HTML Communication and Servlet to JSP Communication.



=>Servlet-Communications are categorized into two types:

1.Forward Communication process

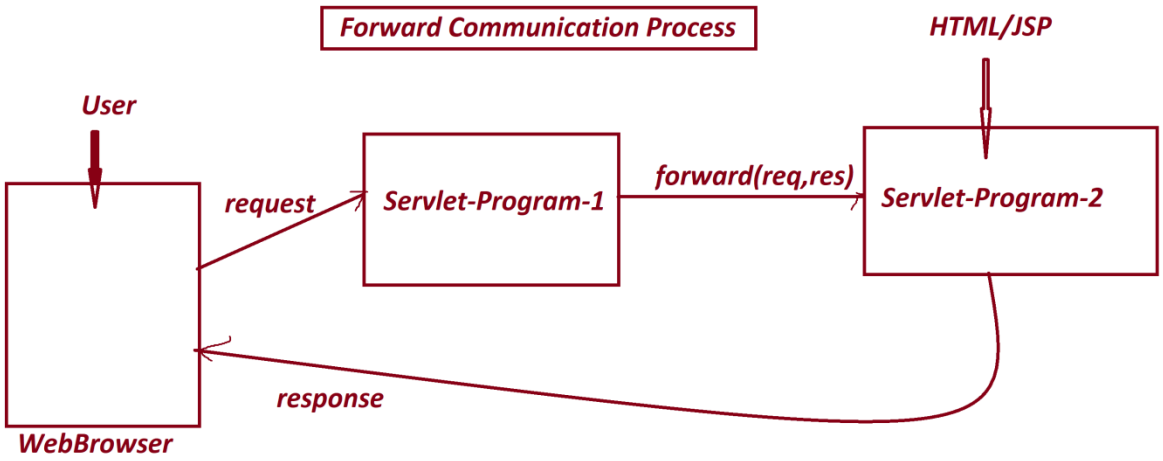
2.Include Communication process

1.Forward Communication process:

=>In Forward Communication process,Servlet-program-1 will take the request and forwards the request to Servlet-program-2,in this process Servlet-program-2 will provide the response.

=>This Servlet-Program-2 can be replaced with HTML/JSP

Diagram:



=>we use forward()-method from 'RequestDispatcher' to perform 'Forward Communication process'

=>Method Signature of forward():

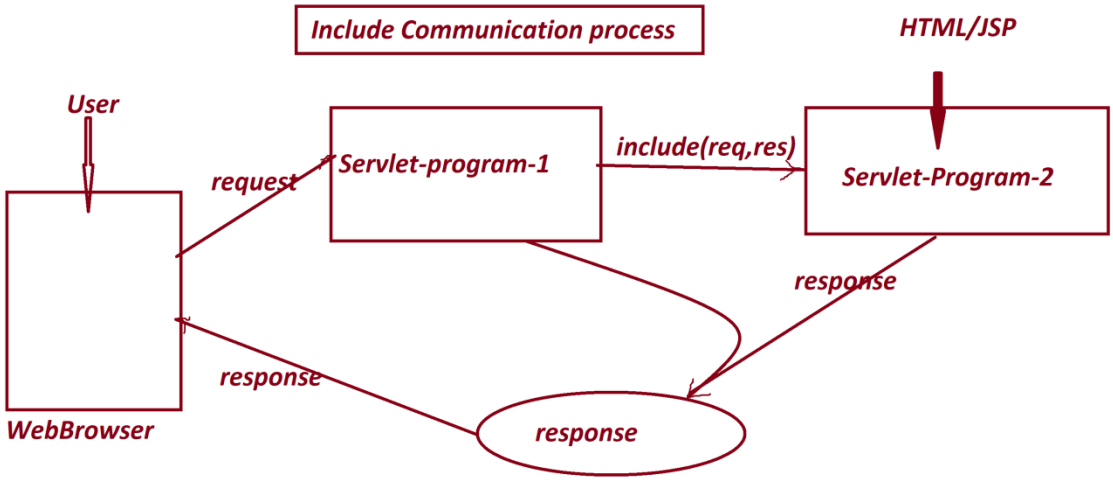
```
public abstract void forward(jakarta.servlet.ServletRequest,
 jakarta.servlet.ServletResponse) throws jakarta.servlet.ServletException,
                                        java.io.IOException;
```

2.Include Communication process:

=>In Include Communication Process,Servlet-Program-1 will take the request and generate the response,but the response is included with the response of Servlet-Program-2.

=>This Servlet-Program-2 can be replaced with HTML/JSP.

Diagram:



=>we use include()-method from 'RequestDispatcher' to perform 'Include Communication process'.

=>Method Signature of include():

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

Note::

=>we use getRequestDispatcher()-method from 'ServletRequest' to create implementation object for 'RequestDispatcher'-Interface.

Method Signature of getRequestDispatcher():

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

syntax:

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

rd.forward(req,res);

(or)

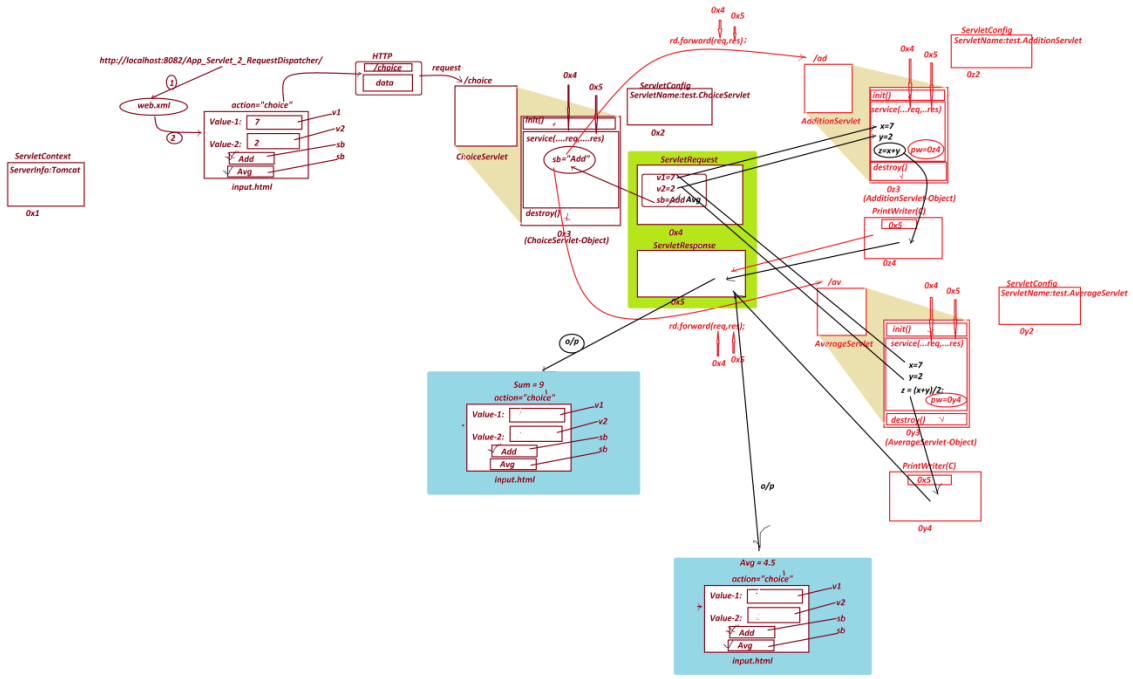
rd.include(req,res);

==========

Dt : 23/10/2024(Day-24)

Ex-Application:(Demonstrating 'RequestDispatcher')

Layout:



ProjectName : App_Servlet_2

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 value-1:<input type="text" name="v1"><br>
Enter the value-2:<input type="text" name="v2"><br>
<input type="submit" name="sb" value="Add">
<input type="submit" name="sb" value="Avg">
</form>
</body>
</html>
```

web.xml

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

ChoiceServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/choice")

public class ChoiceServlet extends GenericServlet

{

        @Override

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

  IOException

  {

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

          if(sb.equals("Add")) {

                  RequestDispatcher rd = req.getRequestDispatcher("ad");

                  rd.forward(req, res);

          }else {

                  RequestDispatcher rd = req.getRequestDispatcher("av");

                  rd.forward(req, res);

          }
```

```
  }
}
```

AdditionServlet.java

```
package test;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/ad")
public class AdditionServlet extends GenericServlet
{
        @Override
 public void service(ServletRequest req,ServletResponse res)throws ServletException,
 IOException
 {
            int x = Integer.parseInt(req.getParameter("v1"));
            int y = Integer.parseInt(req.getParameter("v2"));
            int z =x+y;
            PrintWriter pw = res.getWriter();
            res.setContentType("text/html");
            pw.println("Sum="+z+"<br>");
            RequestDispatcher rd = req.getRequestDispatcher("input.html");
            rd.include(req, res);
 }
}
```

AverageServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/av")

public class AverageServlet extends GenericServlet

{

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

 IOException

 {

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

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

        float z = (float)(x+y)/2;

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

        pw.println("Avg="+z+"<br>");

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

        rd.include(req, res);

 }

}
```

==========

Assignment:

Update above application with the following choices:

=>Sub

=>Mul

=>Div

=>ModDiv

=>Greater

=>Smaller

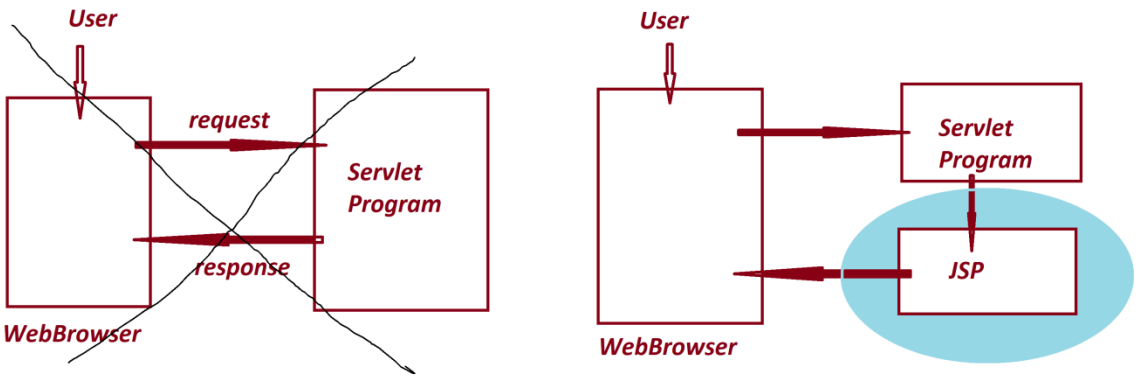
==========

Dt : 24/10/2024(Day-25)

Note:

(i)'ServletRequest' and 'ServletResponse' objects of Servlet-Program-1 can be reused by Servlet-program-2 in forward communication process.

(ii)Every Servlet-Program in application will have its own ServletConfig-Object, but total WebApplication will have only one ServletContext-Object.



**imp*

JSP Programming:

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

=>JSP is tag-based-programming language and which is more simple when compared to Servlet programming.

=>In JSP the programs are saved with (.jsp) as an extension.

Ex:

View.jsp

=>JSP programs are combination of both HTML code and Java Code.

=>We use the following JSP-tags to write Java-Code in JSP programs:

1.Scripting tags

2.Directive tags

3.Action tags

1.Scripting tags:

=>Scriting tags means basic tags used to write normal code in JSP programs.

Types:

(a)Scriptlet tag

(b)Expression tag

(c)Declarative tag

(a)Scriptlet tag:

=>Scriptlet tag is used to write Normal JavaCode in JSP programs.

syntax:

```
<%
  ----JavaCode----
%>
```

(b)Expression tag:

=>Expression tag is used to assign the value to variable or which is used to send data directly to WebBrowser.

syntax:

```
<%=
  -----Value/Expression----
%>
```

(c)Declarative tag:

=>Declarative tags are used to declare variable and methods in JSP-Programs.

syntax:

<%!

-----Variables;methods------

%>

2.Directive tags:

=>Directive tags will specify the informations of current JSP program.

=>Types:

(a)@page

(b)@include

(c)@taglib

(a)@page:

=>'@page' will provide specifications of current running JSP page.

=>This tag will specify Language used,setContentType,include,...

syntax:

<%@page ....%>

(b)@include:

=>'@include' used to include the file in current JSP program

syntax:

<%@include file="file_name"%>

(c)@taglib:

=>'@taglib' is used to link external libraries to current running JSP program.

syntax:

<%@taglib url=....%>

3.Action tags:

=>Actions tags are used to perform some action at execution process.

=>Types:

(a)<jsp:include>

(b)<jsp:forward>

(c)<jsp:param>

(d)<jsp:useBean>

(e)<jsp:setProperty>

(f)<jsp:getProperty>

(a)<jsp:include>:

=><jsp:include> will perform include communication in JSP programs.

syntax:

<jsp:include page="name"...>

(b)<jsp:forward>:

=><jsp:forward> will perform forward communication in JSP programs.

syntax:

<jsp:forward page="name"...>

(c)<jsp:param>:

=><jsp:param> is subtag of <jsp:forward> and which is used to send parameter-value in forward communication process.

syntax:

```
<jsp:forward>
   <jsp:param name="para-name" value="value"/>
</jsp:forward>
```

(d)<jsp:useBean>:

=><jsp:useBean> is used to instantiate bean object or access the existing bean object.

syntax:

```
<jsp:useBean id="var"...>
```

(e)<jsp:setProperty>:

=><jsp:setProperty> is used to load the data to bean Object.

syntax:

```
<jsp:useBean property="pro-name" param="para-name" name="bean-name">
```

(f)<jsp:getProperty>:

=><jsp:getProperty> is used to get the data from bean Object.

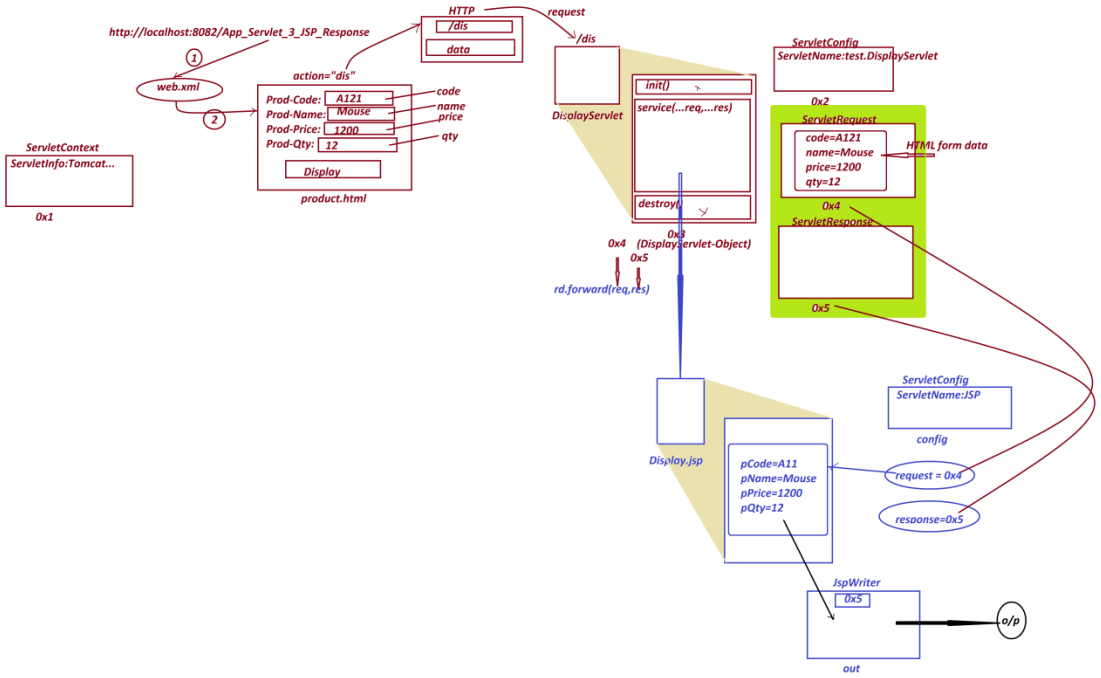
syntax:

=><jsp:getProperty property="pro-name" name="bean-name">

==========

Ex-Application:(Demonstrating JSP-Response)

Layout:



ProjectName : App_Servlet_3_JSP_Response

product.html

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

web.xml

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

DisplayServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/dis")

public class DisplayServlet extends GenericServlet

{

        @Override

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

  IOException

  {

                RequestDispatcher rd = req.getRequestDispatcher("Display.jsp");

     rd.forward(req, res);

  }

}
```

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>
<%
String pCode = request.getParameter("code");
String pName = request.getParameter("name");
float pPrice = Float.parseFloat(request.getParameter("price"));
int pQty = Integer.parseInt(request.getParameter("qty"));
out.println("=====ProductDetails=====<br>");
out.println("ProductCode:"+pCode+"<br>");
out.println("ProductName:"+pName+"<br>");
out.println("ProductPrice:"+pPrice+"<br>");
out.println("ProductQty:"+pQty+"<br>");

%>
</body>
</html>
```

==========

Dt : 25/10/2024(Day-26)

ProjectName : App_Servlet_3_JSP_Response(Modified Code)

product.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
<style>
table, th, td {
  border: 1px solid black;
}
</style>
</head>
<body>
<form action="dis" method="post" >
      <table>
      <tr>
            <td>ProductCode</td>
            <td><input type="text" name="code"><br></td>
      </tr>
      <tr>
            <td>ProductName</td>
            <td><input type="text" name="name"><br></td>
      </tr>
      <tr>
            <td>ProductPrice</td>
            <td><input type="text" name="price"><br></td>
      </tr>
      <tr>
            <td>ProductQty</td>
            <td><input type="text" name="qty"><br></td>
      </tr>
      <tr>
            <td><input type="submit" value="Display"></td>
            <td></td>
      </tr>
      </table>
  </form>
</body>
</html>
```

web.xml

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

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

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

DisplayServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/dis")

public class DisplayServlet extends GenericServlet

{

        @Override

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

  IOException

  {

                RequestDispatcher rd = req.getRequestDispatcher("Display.jsp");

     rd.forward(req, res);

  }

}
```

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>
<style>
table, th, td {
```

```
  border: 1px solid black;
}
</style>
</head>
<body>
<%
String pCode = request.getParameter("code");
String pName = request.getParameter("name");
float pPrice = Float.parseFloat(request.getParameter("price"));
int pQty = Integer.parseInt(request.getParameter("qty"));
out.println("=====ProductDetails=====<br>");
%>
<table>
      <tr>
            <td>ProductCode:</td>
            <td><%=pCode %></td>
      </tr>
      <tr>
            <td>ProductName:</td>
            <td><%=pName %></td>
      </tr>
      <tr>
            <td>ProductPrice:</td>
            <td><%=pPrice %></td>
      </tr>
      <tr>
            <td>ProductQty:</td>
            <td><%=pQty %></td>
      </tr>
      </table>
</body>
</html>
```

==========

Assignment:

Construct Application to read and display UserRegistrationDetails.

UserName

PassWord

FirstName

LastName

City

MailId

PhoneNo

Note::

=>Use JSP response

=>Use HTML/CSS/JS

(Perform Form Validations)

==========

**imp*

DAO Layer:

=>DAO stands for 'Data Access Object' and which is separate layer in MVC holding database related codes or holding persistent logics.

(MVC - Model View Controller)

=>Database related codes means the logics related to create,Insert,Retrieve,update and delete operations on database.

Note::

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

==========

**imp*

Bean classes:

=>The classes which are constructed with the following rules are known as Bean Classes:

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

Rule-2 : The variables declared in the class must be 'private' variables.

Rule-3 : The class must be declated with 0-argument Constructor

(0-parameter Constructor)

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

=>These Bean-Classes will generate bean-objects.

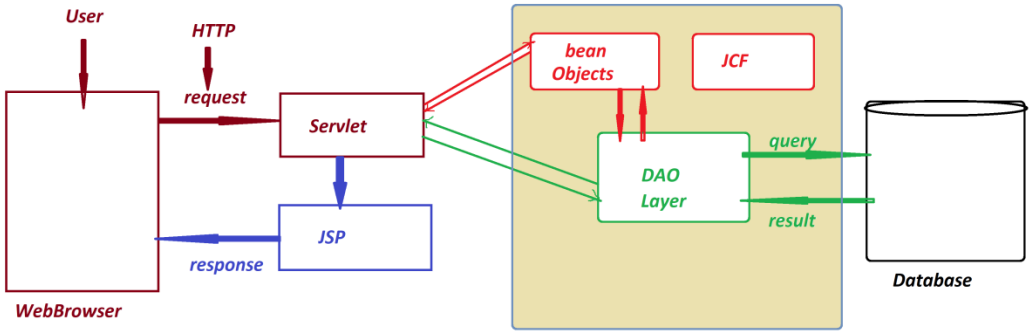
=>These bean-objects are intermediate storages b/w Servlet-program and Database product

Note::

=>we use JCF-Objects to organize multiple bean-objects.

(JCF - Java Collection<E> Framework)

Diagram:



==========

**imp*

"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 the methods related to 'attribute':

(a)setAttribute()

(b)getAttribute()

(c)removeAttribute()

(d)getAttributeNames()

(a)setAttribute():

=>setAttribute()-method is used to set attribute to objects.

Method Signature:

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

(b)getAttribute():

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

Method Signature:

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

(c)removeAttribute():

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

Method Signature:

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

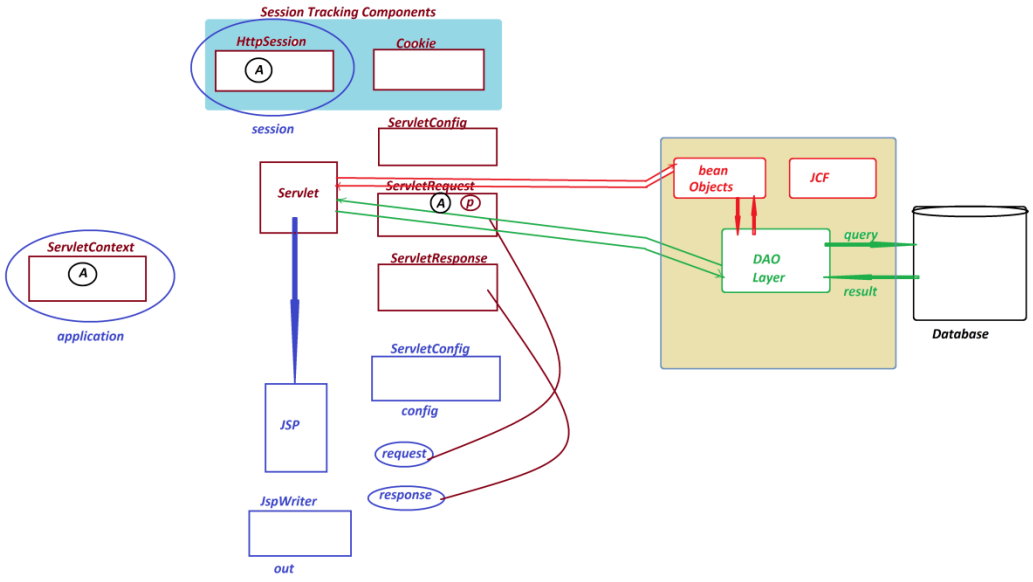
(d)getAttributeNames():

=>getAttributeNames()-method is used to get multiple attribute names from the Objects.

Method Signature:

public abstract java.util.Enumeration<java.lang.String> getAttributeNames();

Diagram:



==========

Dt : 26/10/2024(Day-27)

faq:

define 'request'?

=>The query which is generated by the user through WebBrowser is known as 'request' or 'Http request'

=>This request is categorized into two types:

1.POST request

2.GET request

1.POST request:

=>The request which is generated to send data to server is known as POST request.

=>Through POST request we can send any type of data,which means we can send all types of MultiMedia data(text,audio,video, Image,Animation)

=>Through POST request we can send UnLimited data.

=>The data which we are sending through POST request is more secure,because the data is encapsulated(binded) into the body part of HTTP protocol.

=>we use the following syntax to generate POST request:

```
<form action="" method="POST">
  ....
</form>
```

=>we use doPost()-method from 'HttpServlet' to accept POST request.

Method signature of doPost():

protected void doPost(jakarta.servlet.http.HttpServletRequest,

jakarta.servlet.http.HttpServletResponse) throws

jakarta.servlet.ServletException, java.io.IOException;

2.GET request:

=>The request which is generated to get data from server is known

as GET request.

=>Through GET request we can send only text data.

=>Through GET request we can send limited data.

(data upto 4KB or 8KB)

=>The data in GET request is not secure,because the data is

added to query-string and displayed in AddressBar.

=>To generate GET request,we use the following syntaxes:

syntax-1: declaring method="GET" in <form> tag

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

syntax-2: Submit <form> tag which out writing 'method'

specification.

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

syntax-3:The request generated through 'hyperlinks' is GET request.

syntax-4: GET request is generated when we use servlet-url-pattern in AddressBar

=>we use doGet()-method from 'HttpServlet' to accept GET request

Method Signature of doGet():

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

==========

**imp*

Session Management:

define 'session'?

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

define Session Mangement?

=>The process of recording the state-of-user and tracking the user in session is known as Session Tracking process or Session Management.

=>we use the following Session Tracking techniques in Servlet programming:

1.Cookie

2.HttpSession

3.URL re-write

4.Hidden Form Fields

Dt : 28/10/2024(Day-28)

**imp*

1.Cookie:

=>The piece of information persisted(stored and available) b/w multiple requests is knonw as ' cookie'

=>'Cookie' is generated by Server-Program,but stored and available in WebBrowser to track the user.

=>Cookies are categorized into two types:

(a)Persistent cookies

(b)NonPersistent cookies

(a)Persistent cookies:

=>The cookies which are stored and available in WebBrowser until user-logout are known as Persistent cookies.

(b)NonPersistent cookies:

=>Cookies which are destroyed automatically,when WebBrowser is closed are known as NonPersistent cookies

Note:

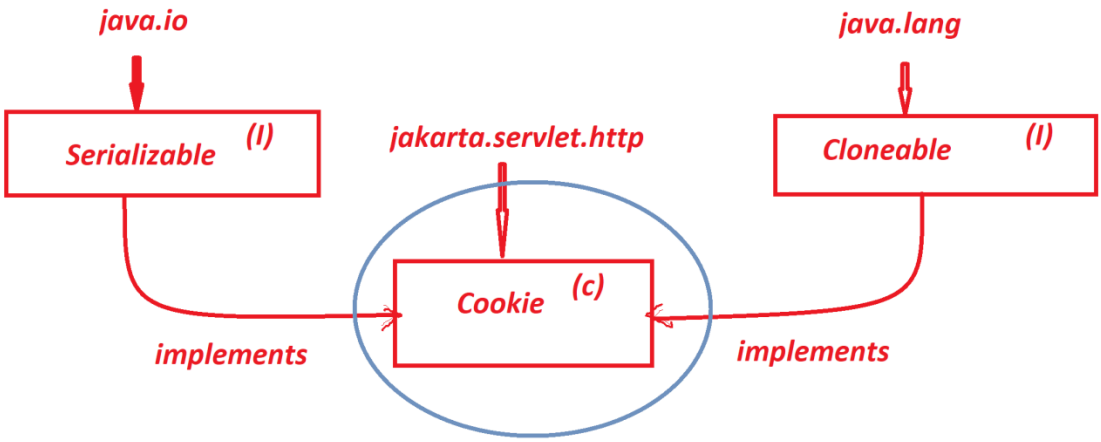
=>we use 'Cookie' class from 'jakarta.servlet.http' package to

construct 'cookie session tracking process'

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

```
public jakarta.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();
public void setAttribute(java.lang.String, java.lang.String);
public java.lang.String getAttribute(java.lang.String);
public java.util.Map<java.lang.String, java.lang.String>
                                   getAttributes();
```

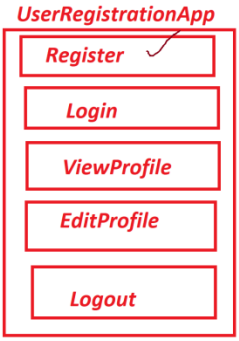
Hierarchy of 'Cookie' class:



Note:
=>Cookie-Objects are Serializable and Cloneable

Ex:(Demonstrating 'Cookie Session Tracking process')

ProjectName : UserRegistrationApp



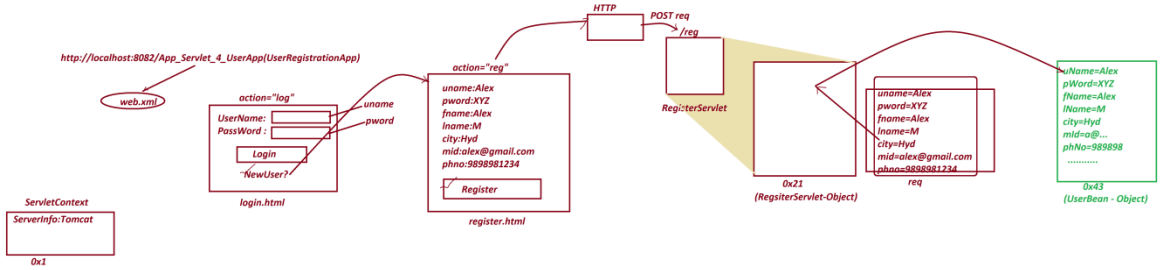
DBTable : UserReg67(

create table UserReg67(uname varchar2(15),pword varchar2(15),

fname varchar2(15),lname varchar2(15),city varchar2(15),

mid varchar2(25),phno number(15),primary key(uname));

Layout:



login.html

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

register.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="reg" method="post">
UserName:<input type="text" name="uname"><br>
Password:<input type="text" name="pword"><br>
FirstName:<input type="text" name="fname"><br>
LastName:<input type="text" name="lname"><br>
```

```
City:<input type="text" name="city"><br>
MailId:<input type="text" name="mid"><br>
PhoneNo:<input type="text" name="phno"><br>
<input type="submit" value="Register">
</form>
</body>
</html>
```

web.xml

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

UserBean.java

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class UserBean implements Serializable
{
   private String uName,pWord,fName,lName,city,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 getCity() {
	return city;
}
public void setCity(String city) {
	this.city = city;
}
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;
}

}
```

RegisterServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.http.*;

import jakarta.servlet.annotation.*;

@WebServlet("/reg")

public class RegisterServlet extends HttpServlet

{

	@Override

  protected void doPost(HttpServletRequest req,HttpServletResponse res)

  throws ServletException,IOException

  {
```

```
        UserBean ub = new UserBean();//Bean Object


        ub.setuName(req.getParameter("uname"));

        ub.setpWord(req.getParameter("pword"));

        ub.setfName(req.getParameter("fname"));

        ub.setlName(req.getParameter("lname"));

        ub.setCity(req.getParameter("city"));

        ub.setmId(req.getParameter("mid"));

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



    }
}
```

Dt : 29/10/2024(Day-29)

RegisterServlet.java(Updated)

```
package test;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/reg")
public class RegisterServlet extends HttpServlet
{
        @Override
  protected void doPost(HttpServletRequest req,HttpServletResponse res)
  throws ServletException,IOException
  {
        UserBean ub = new UserBean();//Bean Object


        ub.setuName(req.getParameter("uname"));
        ub.setpWord(req.getParameter("pword"));
        ub.setfName(req.getParameter("fname"));
        ub.setlName(req.getParameter("lname"));
        ub.setCity(req.getParameter("city"));
        ub.setmId(req.getParameter("mid"));
        ub.setPhNo(Long.parseLong(req.getParameter("phno")));
```

```
        int k = new RegisterDAO().insert(ub);

        if(k>0) {

                req.setAttribute("msg","User registered Successfully...<br>");

                req.getRequestDispatcher("RegSuccess.jsp").forward(req, res);

        }

  }

}
```

DBInfo.java

```
package test;
public interface DBInfo
{
   public static final String driver="oracle.jdbc.driver.OracleDriver";
   public static final String dbUrl="jdbc:oracle:thin:@localhost:1521:xe";
   public static final String uName="system";
   public static final String pWord="tiger";
}
```

DBConnection.java

```
package test;
import java.sql.*;;
public class DBConnection
{
    private static Connection con=null;
    private DBConnection() {}
    static
    {
      try {
            Class.forName(DBInfo.driver);
            con = DriverManager.getConnection
                        (DBInfo.dbUrl,DBInfo.uName,DBInfo.pWord);
      }catch(Exception e) {
            e.printStackTrace();
      }
    }//end of block
    public static Connection getCon()
    {
      return con;
    }
}
```

RegisterDAO.java

```
package test;
import java.sql.*;
public class RegisterDAO
{
   public int k = 0;
   public int insert(UserBean ub)
   {
         try {
               Connection con = DBConnection.getCon();
               PreparedStatement ps = con.prepareStatement
                           ("insert into UserReg67 values(?,?,?,?,?,?,?)");
               ps.setString(1, ub.getuName());
               ps.setString(2, ub.getpWord());
               ps.setString(3, ub.getfName());
               ps.setString(4, ub.getlName());
               ps.setString(5, ub.getCity());
               ps.setString(6, ub.getmId());
               ps.setLong(7, ub.getPhNo());
               k = ps.executeUpdate();
         }catch(Exception e) {
               e.printStackTrace();
         }
         return k;
   }
}
```

RegSuccess.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);
%>
<%@include file="login.html" %>
</body>
</html>
```

Dt : 30/10/2024(Day-30)

LoginDAO.java

```
package test;
import java.sql.*;
public class LoginDAO
{
    public UserBean ub = null;
    public UserBean login(String uN,String pW)
    {
      try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                        ("select * from UserReg67 where uname=? and
pword=?");
            ps.setString(1, uN);
            ps.setString(2, pW);
            ResultSet rs = ps.executeQuery();
            if(rs.next()) {
                  ub = new UserBean();
                  ub.setuName(rs.getString(1));
                  ub.setpWord(rs.getString(2));
                  ub.setfName(rs.getString(3));
                  ub.setlName(rs.getString(4));
                  ub.setCity(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 jakarta.servlet.*;

import jakarta.servlet.http.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")
```

```
@WebServlet("/log")
public class LoginServlet extends HttpServlet
{
        @Override
 protected void doPost(HttpServletRequest req,HttpServletResponse res)throws ServletException,
 IOException
 {
        String uN = req.getParameter("uname");
        String pW = req.getParameter("pword");
        UserBean ub = new LoginDAO().login(uN,pW);
        if(ub==null) {
                req.setAttribute("msg","Invalid Login process...<br>");
                req.getRequestDispatcher("Msg.jsp").forward(req, res);
        }else {
                Cookie ck = new Cookie("fname",ub.getfName());
                ServletContext sct = req.getServletContext();
                        //Accessing SrvletContext Object reference
                sct.setAttribute("ubean", ub);
                res.addCookie(ck);//Adding Cookie is response
                req.getRequestDispatcher("LogSuccess.jsp").forward(req, res);
        }
 }
}
```

LogSuccess.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)application.getAttribute("ubean");
out.println("Welcome User : "+ub.getuName()+"<br>");
%>
<a href="view">ViewProfile</a>
<a href="logout">Logout</a>
</body>
</html>
```

Msg.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);
%>
<%@include file="login.html"%>
</body>
</html>
```

Dt : 1/11/2024(Day-31)

ViewProfileServlet.java

```
package test;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/view")
public class ViewProfileServlet extends HttpServlet
{
	@Override
  protected void doGet(HttpServletRequest req,HttpServletResponse res)throws ServletException,
  IOException
  {
		Cookie c[] = req.getCookies();
		if(c==null) {
			req.setAttribute("msg", "Sorry ! Session Expired...<br>");
			req.getRequestDispatcher("Msg.jsp").forward(req, res);
		}else {
			String fN = c[0].getValue();
			req.setAttribute("fname", fN);
			req.getRequestDispatcher("ViewProfile.jsp").forward(req, res);
		}
  }
```

```
}
```

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>
<%
String fName = (String)request.getAttribute("fname");
UserBean ub = (UserBean)application.getAttribute("ubean");
out.println("Page belongs to User : "+fName+"<br>");
out.println(ub.getfName()+"  "+ub.getlName()+"  "+
ub.getCity()+"  "+ub.getmId()+"  "+ub.getPhNo()+"  "+
            "<a href='edit'>Edit</a>"+"<br>");
%>
<a href="logout">Logout</a>
</body>
</html>
```

EditProfileServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.http.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/edit")

public class EditProfileServlet extends HttpServlet

{

        @Override

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

```
IOException

{

        Cookie c[] = req.getCookies();

        if(c==null) {

                req.setAttribute("msg", "Sorry! Session Expired...<br>");

                req.getRequestDispatcher("Msg.jsp").forward(req, res);

        }else {

                String fN = c[0].getValue();

                req.setAttribute("fname", fN);

                req.getRequestDispatcher("EditProfile.jsp").forward(req, res);

        }

 }

}
```

EditProfile.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>
<%
String fName = (String)request.getAttribute("fname");
UserBean ub = (UserBean)application.getAttribute("ubean");
out.println("Page belongs to User : "+fName+"<br>");
%>
<form action="update" method="post">
City:<input type="text" name="city" value=<%=ub.getCity() %>><br>
MailId:<input type="text" name="mid" value=<%=ub.getmId() %>><br>
PhoneNo:<input type="text" name="phno" value=<%=ub.getPhNo() %>><br>
<input type="submit" value="UpdateProfile">
</form>
</body>
```

```
</html>
```

Dt : 2/11/2024(Day-32)

UpdateProfileDAO.java

```
package test;
import java.sql.*;
public class UpdateProfileDAO
{
   public int k = 0;
   public int update(UserBean ub)
   {
         try {
               Connection con = DBConnection.getCon();
               PreparedStatement ps = con.prepareStatement
                           ("update UserReg67 set city=?,mid=?,phno=? where
uname=? and pword=?");
               ps.setString(1,ub.getCity());
               ps.setString(2,ub.getmId());
               ps.setLong(3,ub.getPhNo());
               ps.setString(4,ub.getuName());
               ps.setString(5,ub.getpWord());
               k = ps.executeUpdate();
         }catch(Exception e) {
               e.printStackTrace();
         }
         return k;
   }
}
```

UpdateProfileServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.http.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/update")

public class UpdateProfileServlet extends HttpServlet

{
```

```
        @Override
 protected void doPost(HttpServletRequest req,HttpServletResponse res)throws ServletException,
 IOException
 {
        Cookie c[] = req.getCookies();
        if(c==null) {
                req.setAttribute("msg", "Sorry ! Session Expired...<br>");
                req.getRequestDispatcher("Msg.jsp").forward(req, res);
        }else {
                String fN = c[0].getValue();
                req.setAttribute("fname", fN);
                ServletContext sct = req.getServletContext();//Accessing ServletContext Object
reference
                UserBean ub = (UserBean)sct.getAttribute("ubean");
                ub.setCity(req.getParameter("city"));
                ub.setmId(req.getParameter("mid"));
                ub.setPhNo(Long.parseLong(req.getParameter("phno")));
                int k = new UpdateProfileDAO().update(ub);
                if(k>0) {
                        req.setAttribute("msg","Profile Updated Successfully...<br>");
                        req.getRequestDispatcher("UpdateProfile.jsp").forward(req, res);
                }
        }
 }
}
```

UpdateProfile.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)request.getAttribute("fname");
String msg = (String)request.getAttribute("msg");
out.println("Page belongs to User : "+fName+"<br>");
out.println(msg);
%>
<a href="view">ViewProfile</a>
<a href="logout">Logout</a>
</body>
</html>
```

LogoutServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.http.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/logout")

public class LogoutServlet extends HttpServlet

{

        @Override

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

  IOException

  {

            Cookie c[] = req.getCookies();

            if(c==null) {
```

```
                req.setAttribute("msg", "Sorry ! Session Expired...<br>");
                req.getRequestDispatcher("Msg.jsp").forward(req, res);
        }else {
                ServletContext sct = req.getServletContext();
                sct.removeAttribute("ubean");
                c[0].setMaxAge(0);
                res.addCookie(c[0]);
                req.getRequestDispatcher("Logout.jsp").forward(req, res);
        }
  }
}
```

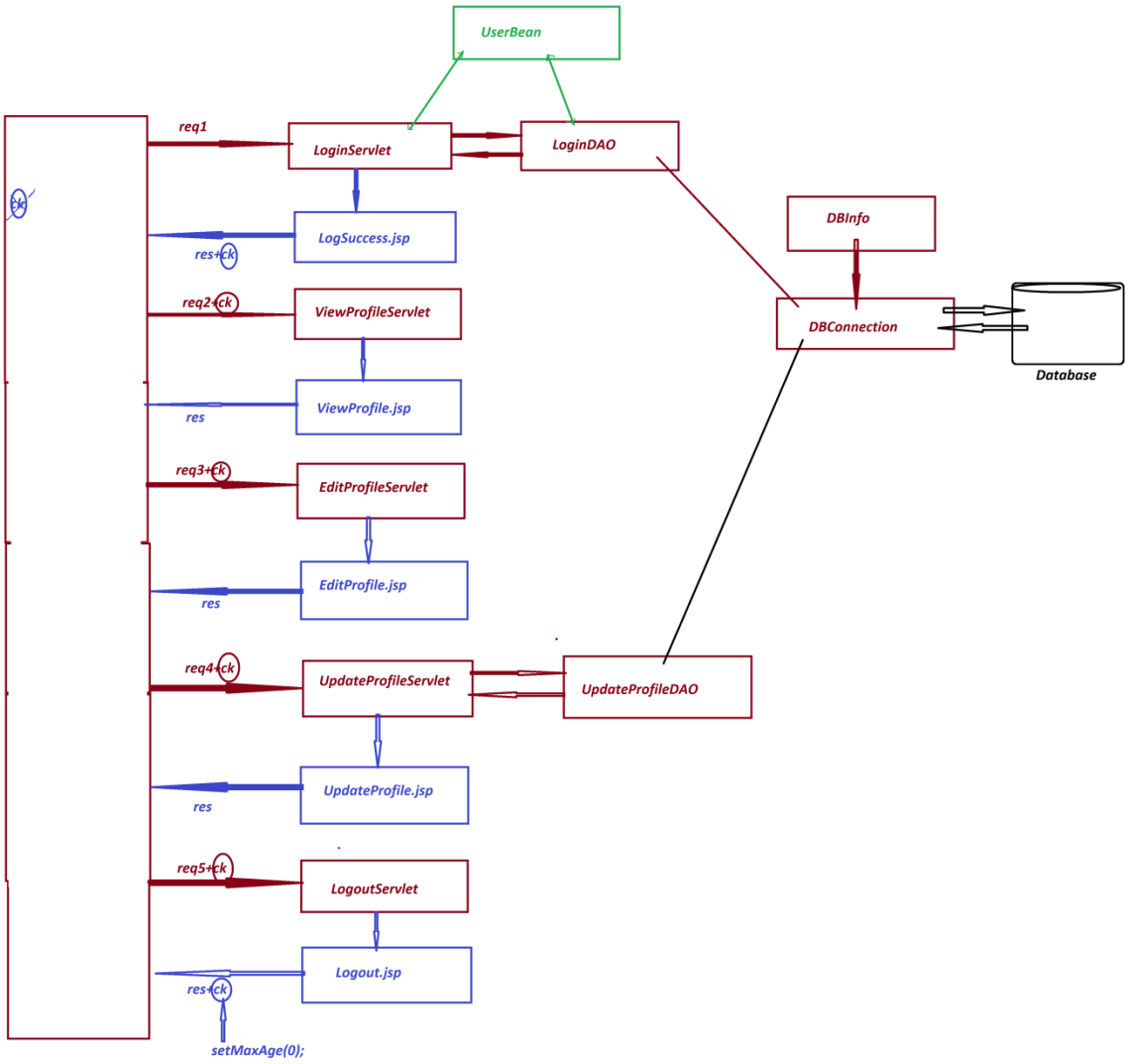
Logout.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
out.println("User Logged Out Successfully...<br>");
%>
<%@include file="login.html"%>
</body>
</html>
```

==========

Dt : 4/11/2024(Day-33)

Summary of Cookie Session Tracking Application:



==========

faq:

wt is the diff b/w

(a)JAR

(b)WAR

(c)EAR

(a)JAR:

=>JAR stands for 'Java Archive' and which is compressed format of more number of Class-files,which means more number of byte-code-files.

=>Stand-Alone-Applications or NonServer-Applications are converted into Jar-file.

(b)WAR:

=>WAR stands for 'Web Archive' and which is compressed format of HTML files,XML files,Servlet,JSP,externals JARs and other UI files.

=>AdvJava WebApplication Converted into WAR-file.

(c)EAR:

=>EAR stands for 'Enterprise Archive' and which is compressed format of Jar-files,WAR-files and other Services(WebServices)

=>Enterprise Applications are converted int EAR-files.

Note:

=>Server Applications are categorized into two types:

1.Web Applications

2.Enterprise Applications

1.Web Applications:

=>The application which is developed using Servlet,JSP,DAO layer and Beans is known as Web Application.

2.Enterprise Applications:

=>The application which is executing in distributed environment and depending on the features like 'Security','Load balancing' and 'Clustering' is known as Enterprise Application or Enterprise distributed application.

==========

**imp*

Generating WAR file from IDE Eclipse:

step-1 : Generate WAR file

RightClick on Project(WebApplication)->Export->WAR File->

Browse and select folder(user defined folder) to store WAR-file->

name the WAR-file(UApp.war)->click 'Save'->click 'Finish'

step-2 : Start the Tomcat Server and deploy WAR-file for execution

start the Tomcat-server->access the Tomcat Server from WebBrowser->

click 'Manager App'->perform Login-process->

click 'Choose File' from 'WAR file to deploy'->

Browse and Select WAR-file->click 'Open'->click 'Deploy'

Same Computer:

http://localhost:8082/UApp/

Remote Computer:

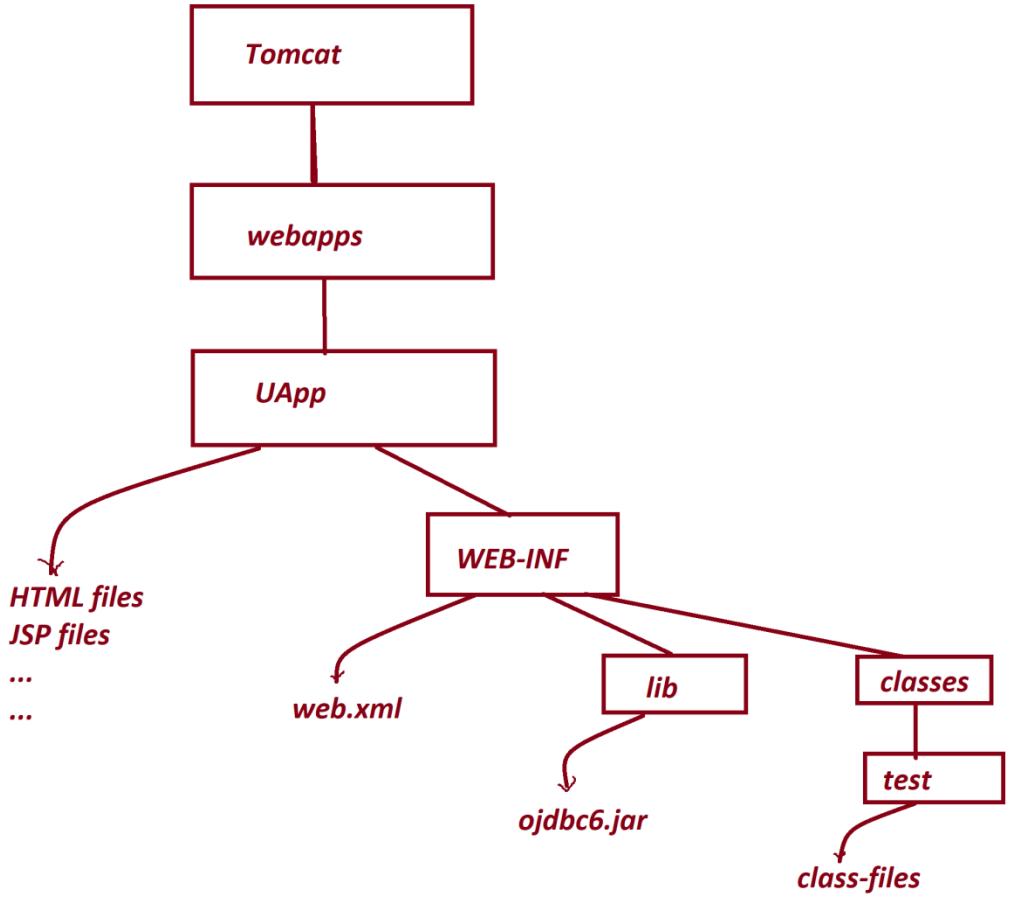
http://172.20.10.2:8082/UApp/

==========

Dt : 5/11/2024(Day-34)

**imp*

Deployment Directory Structure of Tomcat:



==========

**imp*

2.HttpSession:

=>'HttpSession' is an interface from 'jakarta.servlet.http' package and which is used in Session tracking process.

=>The following are some important methods of HttpSession:

public abstract void setAttribute

(java.lang.String, java.lang.Object);

public abstract java.lang.Object getAttribute

(java.lang.String);

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

public abstract java.util.Enumeration<java.lang.String>

getAttributeNames();

public abstract void invalidate();

public abstract jakarta.servlet.ServletContext

getServletContext();

=>we use getSession()-method from 'HttpServletRequest' to

create implementation object for 'HttpSession-Interface',

because this getSession()-method internally holding

'Anonymous Local InnerClass as implementation class of

HttpSession-Interface'.

Method Signature of getSession():

public abstract jakarta.servlet.http.HttpSession getSession

(boolean);

public abstract jakarta.servlet.http.HttpSession getSession();

****imp***

3.URL re-write:

=>The process of adding parameter-value to Servlet-url-pattern is

known as 'URL re-write' process.

=>This URL re-write process is used to transfer some information

from one servlet to another servlet in Session Tracking process.

syntax:

Servlet_url_pattern?para1=value&para2=value$para3=value&...

Note::

"?" : represent separator b/w Servlet_url_pattern and parameters.

"&" : represent separator b/w parameters

**imp*

4.Hidden Form Fields:

=>The process of declaring <input type="hidden"...> in <form> tag

is known as Hidden Form Field.

=>The data in Hidden Form Fields are not displayed to the End-User,

bute which are used to transfer the information from one Servlet

to another servlet in Session Tracking process.

syntax:

```
<form action="" methpd="">
  <input type="hidden" name="nm" value="val">Br>
   ....
</form>
```

Note::

=>'Url re-write' and 'Hidden Form fields' are used as SubTechniques

in 'Cookie' and 'HttpSession' Applications.

==========

Application : Online Product Store

DB tables : Customer67(phno,name,city,mid)

Primary Key : phno

Admin67(uname,pword,fname,lname,city,mid,phno)

primary Key : uname

Product67(code,name,price,qty)

```
create table Customer67(phno number(15),name varchar2(15),
city varchar2(15),mid varchar2(25),primary key(phno));

create table Admin67(uname varchar2(15),pword varchar2(15),
fname varchar2(15),lname varchar2(15),city varchar2(15),
mid varchar2(25),phno number(15),primary key(uname));

create table Product67(code varchar2(10),name varchar2(15),
price number(10,2),qty number(10),primary key(code));
```

Dt : 6/11/2024(Day-35)

Layout:

home.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<a href="CustomerLogin.html">CustomerLogin</a><br>
<a href="AdminLogin.html">AdminLogin</a>
</body>
</html>
```

web.xml

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

AdminLogin.html

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

DBInfo.java

```
package test;
```

```
public interface DBInfo
{
   public static final String driver="oracle.jdbc.driver.OracleDriver";
   public static final String dbUrl="jdbc:oracle:thin:@localhost:1521:xe";
   public static final String uName="system";
   public static final String pWord="tiger";
}
```

DBConnection.java

```
package test;
import java.sql.*;;
public class DBConnection
{
    private static Connection con=null;
    private DBConnection() {}
    static
    {
      try {
            Class.forName(DBInfo.driver);
            con = DriverManager.getConnection
                        (DBInfo.dbUrl,DBInfo.uName,DBInfo.pWord);
      }catch(Exception e) {
            e.printStackTrace();
      }
    }//end of block
    public static Connection getCon()
    {
      return con;
    }
}
```

AdminBean.java

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

```
	return pWord;
}
public void setpWord(String pWord) {
	this.pWord = pWord;
}
public String getfName() {
	return fName;
}
public void setfName(String fName) {
	this.fName = fName;
}
public String getlName() {
	return lName;
}
public void setlName(String lName) {
	this.lName = lName;
}
public String getCity() {
	return city;
}
public void setCity(String city) {
	this.city = city;
}
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;
}

}
```

AdminLoginDAO.java

```
package test;
import java.sql.*;
public class AdminLoginDAO {
   public AdminBean ab = null;
   public AdminBean login(String uN,String pW) {
         try {
               Connection con = DBConnection.getCon();
               PreparedStatement ps = con.prepareStatement
```

```
                        ("select * from Admin67 where uname=? and
pword=?");
                ps.setString(1, uN);
                ps.setString(2, pW);
                ResultSet rs = ps.executeQuery();
                if(rs.next()) {
                        ab = new AdminBean();
                        ab.setuName(rs.getString(1));
                        ab.setpWord(rs.getString(2));
                        ab.setfName(rs.getString(3));
                        ab.setlName(rs.getString(4));
                        ab.setCity(rs.getString(5));
                        ab.setmId(rs.getString(6));
                        ab.setPhNo(rs.getLong(7));
                }
            }catch(Exception e) {
                        e.printStackTrace();
                }
                return ab;
    }
}
```

AdminLoginServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.http.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/admin")

public class AdminLoginServlet extends HttpServlet{

        @Override

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

 IOException

 {

          String uN = req.getParameter("uname");
```

```
		String pW = req.getParameter("pword");

		AdminBean ab = new AdminLoginDAO().login(uN, pW);

		if(ab==null) {

			req.setAttribute("msg", "Invalid Login Process...<br>");

			req.getRequestDispatcher("Msg.jsp").forward(req, res);

		}else {

			HttpSession hs = req.getSession();//New Session created

			hs.setAttribute("abean", ab);

			req.getRequestDispatcher("LogSuccess.jsp").forward(req, res);

		}

	}

}
```

LogSuccess.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.AdminBean"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
AdminBean ab = (AdminBean)session.getAttribute("abean");
out.println("Welcome Admin:"+ab.getfName()+"<br>");
%>
<a href="Product.html">AddProduct</a>
<a href="view">ViewAllProducts</a>
<a href="logout">Logout</a>
</body>
</html>
```

Msg.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);
%>
<%@include file="home.html" %>
</body>
</html>
```

Assignment:

CustomerLogin.html

CustomerRegister.html

CustomerBean.java

CustomerRegisterDAO.java

CustomerRegisterServlet.java

RegSuccess.jsp

==========

Dt : 8/11/2024(Day-37)

ViewAllProducts.jsp(Edited Code)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.*,java.util.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
AdminBean ab = (AdminBean)session.getAttribute("abean");
ArrayList<ProductBean> al
=(ArrayList<ProductBean>)session.getAttribute("alist");
out.println("Page belongs to Admin:"+ab.getfName()+"<br>");
if(al.size()==0){
      out.println("Products not Available...<br>");
}else{
      Iterator<ProductBean> it = al.iterator();
      while(it.hasNext()){
            ProductBean pb = (ProductBean)it.next();
            out.println(pb.getCode()+"  "+pb.getName()+"  "+
            pb.getPrice()+"  "+pb.getQty()+"  "+
                        "<a
href='edit?pcode="+pb.getCode()+"'>Edit</a>"+"  "+
            "<a href='delete?pcode="+pb.getCode()+"'>Delete</a>"+"<br>");
      }
}
%>
<a href="Product.html">AddProduct</a>
<a href="logout">Logout</a>
</body>
</html>
```

EditProductServlet.java

```
package test;

import java.io.*;

import java.util.*;

import jakarta.servlet.*;

import jakarta.servlet.http.*;
```

```
import jakarta.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/edit")
public class EditProductServlet extends HttpServlet{
	@SuppressWarnings("unchecked")
	@Override
 protected void doGet(HttpServletRequest req,HttpServletResponse res)throws ServletException,
 IOException
 {
	 HttpSession hs = req.getSession(false);//Accessing Existing Session
	 if(hs==null) {
		req.setAttribute("msg","Session Expired...<br>");
		req.getRequestDispatcher("Msg.jsp").forward(req, res);
	 }else {
		ArrayList<ProductBean> al = (ArrayList<ProductBean>)hs.getAttribute("alist");
		String pCode = req.getParameter("pcode");
		Iterator<ProductBean> it = al.iterator();
		while(it.hasNext()) {
			ProductBean pb = (ProductBean)it.next();
			if(pCode.equals(pb.getCode())) {
				req.setAttribute("pbean", pb);
				req.getRequestDispatcher("EditProduct.jsp").forward(req, res);
				break;
			}//end of if
		}//end of loop
```

```
        }

 }

}
```

EditProduct.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>
<%
AdminBean ab = (AdminBean)session.getAttribute("abean");
ProductBean pb = (ProductBean)request.getAttribute("pbean");
out.println("Page belongs to Admin:"+ab.getfName()+"<br>");
%>
<form action="update" method="post">
ProductPrice:<input type="text" name="price" value=<%=pb.getPrice() %>><br>
ProductQty:<input type="text" name="qty" value=<%=pb.getQty() %>><br>
<input type="submit" value="UpdateProduct">
</form>
</body>
</html>
```

Dt : 9/11/2024(Day-38)

EditProduct.jsp(Edited Code)

```
<%@ 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>
<%
AdminBean ab = (AdminBean)session.getAttribute("abean");
ProductBean pb = (ProductBean)request.getAttribute("pbean");
out.println("Page belongs to Admin:"+ab.getfName()+"<br>");
%>
<form action="update" method="post">
<input type="hidden" name="pcode" value=<%=pb.getCode() %>>
ProductPrice:<input type="text" name="price" value=<%=pb.getPrice() %>><br>
ProductQty:<input type="text" name="qty" value=<%=pb.getQty() %>><br>
<input type="submit" value="UpdateProduct">
</form>
</body>
</html>
```

UpdateProductDAO.java

```
package test;
import java.sql.*;
public class UpdateProductDAO {
    public int k=0;
    public int updateProduct(ProductBean pb) {
      try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                        ("update Product67 set price=?,qty=? where code=?");
            ps.setFloat(1, pb.getPrice());
            ps.setInt(2, pb.getQty());
            ps.setString(3, pb.getCode());
            k = ps.executeUpdate();
      }catch(Exception e) {
            e.printStackTrace();
      }
      return k;
    }
}
```

UpdateProductServlet.java

```
package test;
import java.io.*;
import java.util.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/update")
public class UpdateProductServlet extends HttpServlet
{
        @SuppressWarnings("unchecked")
        @Override
  protected void doPost(HttpServletRequest req,HttpServletResponse res)throws ServletException,
  IOException
  {
          HttpSession hs = req.getSession(false);//Accessing Existing Session
          if(hs==null) {
                  req.setAttribute("msg","Session Expired...<br>");
                  req.getRequestDispatcher("Msg.jsp").forward(req, res);
          }else {
                  ArrayList<ProductBean> al = (ArrayList<ProductBean>)hs.getAttribute("alist");
                  String pCode = req.getParameter("pcode");
                  Iterator<ProductBean> it = al.iterator();
```

```
                while(it.hasNext()) {
                        ProductBean pb = (ProductBean)it.next();
                        if(pCode.equals(pb.getCode())) {
                                pb.setPrice(Float.parseFloat(req.getParameter("price")));
                                pb.setQty(Integer.parseInt(req.getParameter("qty")));
                                int k = new UpdateProductDAO().updateProduct(pb);
                                if(k>0) {
                                        req.setAttribute("msg","Product Updated
Successfully...<br>");
                                        req.getRequestDispatcher("UpdateProduct.jsp").forward(req,
res);
                                }//end of if
                                break;
                        }//end of if
                }//end of loop
        }
  }
}
```

UpdateProduct.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>
<%
AdminBean ab = (AdminBean)session.getAttribute("abean");
String msg = (String)request.getAttribute("msg");
out.println("Page belongs to Admin:"+ab.getfName()+"<br>");
```

```
out.println(msg);
%>
<a href="Product.html">AddProduct</a>
<a href="view">ViewAllProducts</a>
<a href="logout">Logout</a>
</body>
</html>
```

LogoutServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.http.*;

import jakarta.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);//Accessing Existing Session

        if(hs==null) {

                req.setAttribute("msg","Session Expired...<br>");

                req.getRequestDispatcher("Msg.jsp").forward(req, res);

        }else {

                hs.removeAttribute("abean");

                hs.removeAttribute("alist");

                hs.invalidate();
```

```
                req.getRequestDispatcher("Logout.jsp").forward(req, res);

        }

 }

}
```

Logout.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
out.println("Logged out Successfully...<br>");
%>
<%@include file="home.html"%>
</body>
</html>
```

==========

Dt : 13/11/2024(Day-39)

Assignment:

CustomerLogin.html

CustomerRegister.html

CustomerBean.java

CustomerRegisterDAO.java

CustomerRegisterServlet.java

RegSuccess.jsp

Layout:

Assignment Layout:

==========

**imp*

Filters in Servlet Programming:

=>The pre-processing component which is executed before Servlet-Program is known as 'Filter'.

=>Filters are Security gate-ways in applications.

=>we use the following components in constructing filter-applications:

1.Filter

2.FilterChain

3.FilterConfig

1.Filter:

=>'Filter' is an interface from 'jakarta.servlet' package and which is implemeted to construct Filter-programs.

=>The following are the life-cycle methods of Filter:

public void init(jakarta.servlet.FilterConfig)

throws jakarta.servlet.ServletException;

public abstract void doFilter(jakarta.servlet.ServletRequest,

jakarta.servlet.ServletResponse, jakarta.servlet.FilterChain)

throws java.io.IOException, jakarta.servlet.ServletException;

public void destroy();

2.FilterChain:

=>'FilterChain' is an interface from jakarta.servlet package and which is used on interlink Servlet-program running on Same-url-pattern.

=>The following is one important method from FilterChain:

public abstract void doFilter(jakarta.servlet.ServletRequest,

jakarta.servlet.ServletResponse) throws java.io.IOException,

jakarta.servlet.ServletException;

3.FilterConfig:

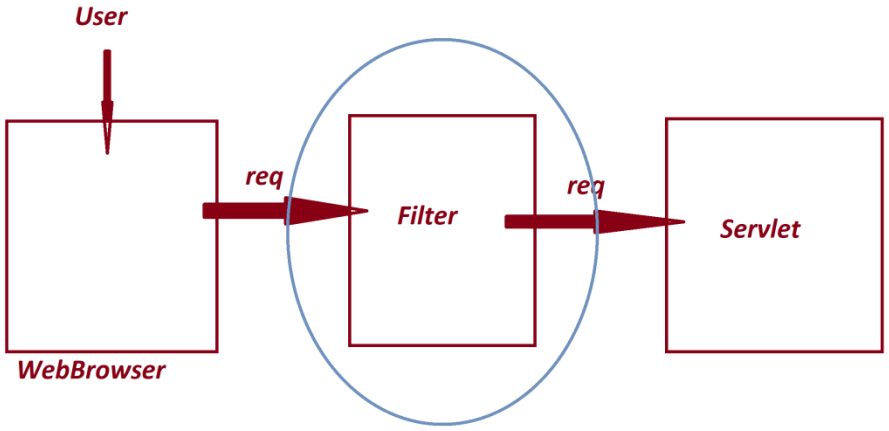
=>'FilterConfig' is an interface from jakarta.servlet package and which is instantiated when Filter-program loaded for execution.

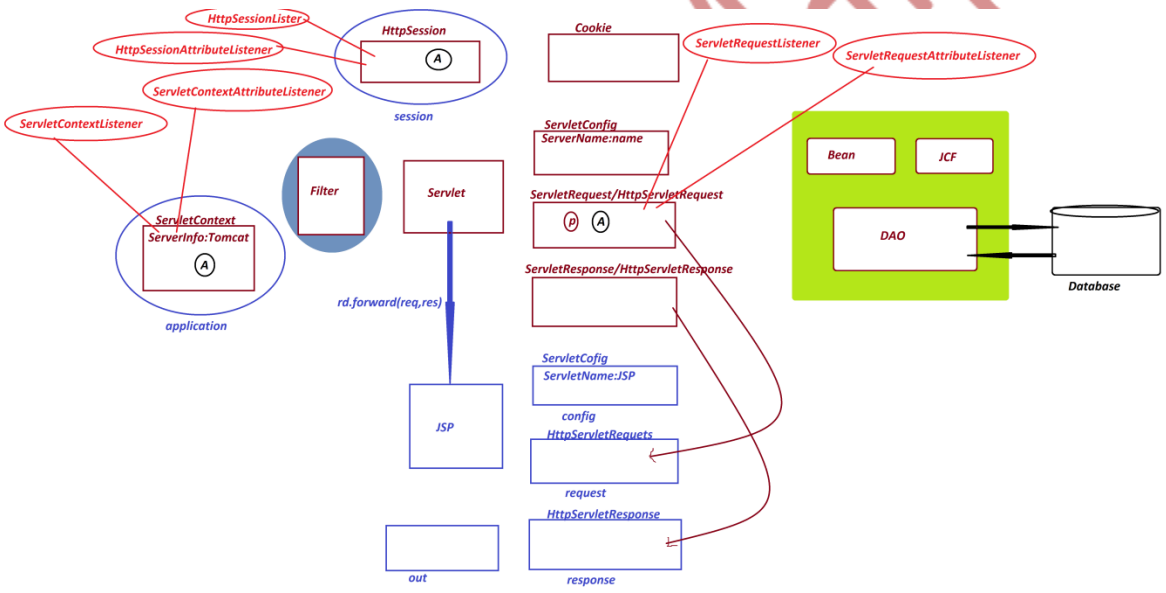
=>This 'FilterConfig' object is loaded with FilterName.

Note:

="Filter" and 'Servlet" programs are executed based on same url-pattern,in this process Filter-Program will have HighestPriority in execution than Servlet-program.

Diagram:


User
req
Filter
req
Servlet
WebBrowser

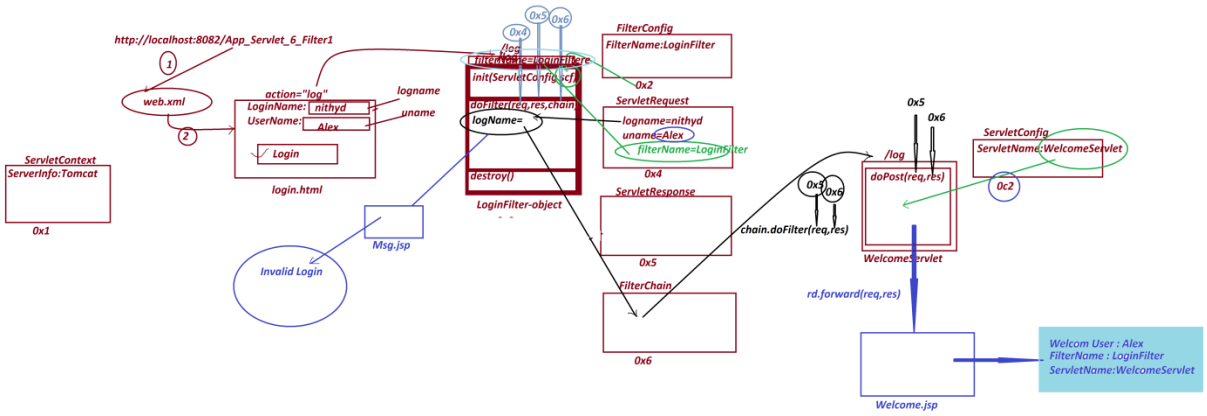

HttpSessionLister
HttpSessionAttributeListener
ServletContextAttributeListener
ServletContextListener
HttpSession
A
session
ServletContext
ServerInfo:Tomcat
A
application
Filter
Servlet
rd.forward(req,res)
JSP
Cookie
ServletRequestListener
ServletRequestAttributeListener
ServletConfig
ServerName:name
ServletRequest/HttpServletRequest
P
A
ServletResponse/HttpServletResponse
ServletCofig
ServletName:JSP
config
HttpServletRequets
request
HttpServletResponse
response
out
Bean
JCF
DAO
Database

==========

Dt : 14/11/2024(Day-40)

Ex:(Demonstrating Filter)

Layout:



login.html

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

web.xml

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

LoginFilter.java

```
package test;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
@WebFilter("/log")
public class LoginFilter implements Filter
{
        public String filterName = null;
        @Override
   public void init(FilterConfig fcf)throws ServletException
   {
        filterName = fcf.getFilterName();
   }
        @Override
   public void doFilter(ServletRequest req,ServletResponse res,FilterChain chain)throws
   ServletException,IOException
   {
        String logName = req.getParameter("logname");
        if(logName.equals("nithyd")) {
                req.setAttribute("filterName", filterName);
                chain.doFilter(req, res);
        }else {
                req.setAttribute("msg","Invalid login process...<br>");
                req.getRequestDispatcher("Msg.jsp").forward(req, res);
```

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

WelcomeServlet.java

```
package test;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/log")
public class WelcomeServlet extends HttpServlet
{
        @Override
  protected void doPost(HttpServletRequest req,HttpServletResponse res)throws ServletException,
  IOException
  {
            ServletConfig scf = this.getServletConfig();
            String servletName = scf.getServletName();
            req.setAttribute("servletName", servletName);
```

```
            req.getRequestDispatcher("Welcome.jsp").forward(req, res);

  }

}
```

Welcome.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 uName = request.getParameter("uname");
String filterName = (String)request.getAttribute("filterName");
String servletName = (String)request.getAttribute("servletName");
out.println("Welcome User:"+uName+"<br>");
out.println("FilterName:"+filterName+"<br>");
out.println("ServletName:"+servletName+"<br>");
%>
</body>
</html>
```

Msg.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);
%>
<%@include file="login.html" %>
</body>
</html>
```

==========

*imp

Listerners in Servlet programming:

=>The program which is executed background to servlet-objects is known as "listener"

=>In Servlet-Programming the listeners can be added to the following Objects:

1.ServletContext Object

2.ServletRequest Object

3.HttpSession Object

1.ServletContext Object:

=>The following are the listeners added to ServletContext-Object:

(a)ServletContextListener

Methods:

public void contextInitialized(jakarta.servlet.ServletContextEvent);

public void contextDestroyed(jakarta.servlet.ServletContextEvent);

(b)ServletContextAttributeListener

Methods:

public void attributeAdded(jakarta.servlet.ServletContextAttributeEvent);

public void attributeRemoved(jakarta.servlet.ServletContextAttributeEvent);

public void attributeReplaced(jakarta.servlet.ServletContextAttributeEvent);

2.ServletRequest Object:

=>The following are the listeners added to ServletRequest Object:

(a)ServletRequestListener

Methods:

public void requestDestroyed(jakarta.servlet.ServletRequestEvent);

public void requestInitialized(jakarta.servlet.ServletRequestEvent);

(b)ServletRequestAttributeListener

public void attributeAdded(jakarta.servlet.ServletRequestAttributeEvent);

public void attributeRemoved(jakarta.servlet.ServletRequestAttributeEvent);

public void attributeReplaced(jakarta.servlet.ServletRequestAttributeEvent);

3.HttpSession Object:

=>The following are the listeners added to HttpSession Object:

(a)HttpSessionListener

Methods:

public void sessionCreated(jakarta.servlet.http.HttpSessionEvent);

public void sessionDestroyed(jakarta.servlet.http.HttpSessionEvent);

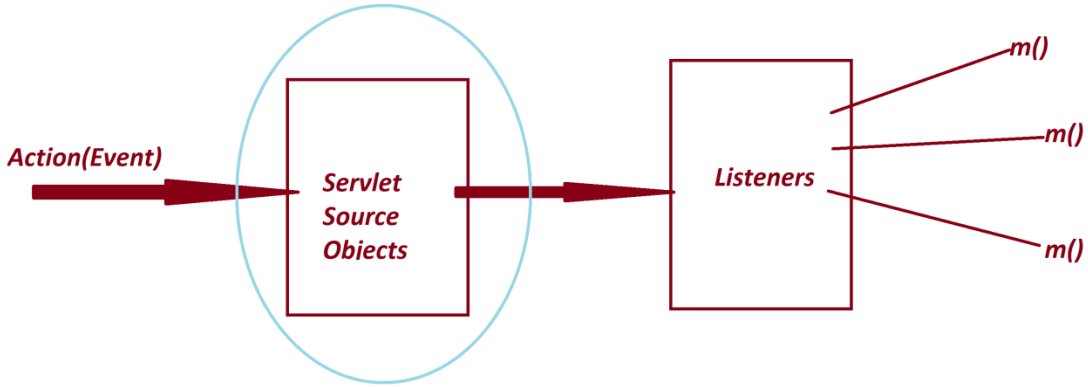
(b)HttpSessionAttributeListener

Methods:

public void attributeAdded(jakarta.servlet.http.HttpSessionBindingEvent);

public void attributeRemoved(jakarta.servlet.http.HttpSessionBindingEvent);

public void attributeReplaced(jakarta.servlet.http.HttpSessionBindingEvent);



Dt : 15/11/2024(Day-41)

Ex:

ContextListener.java

```
package test;
import jakarta.servlet.*;
import jakarta.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("ServletContexr destroyed...");
   }
}
```

RequestListener.java

```
package test;
import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
@WebListener
public class RequestListener implements ServletRequestListener,ServletRequestAttributeListener
{
```

```
        public void requestInitialized(ServletRequestEvent sre)
        {
                System.out.println("Request Initialized...");
        }
        public void requestDestroyed(ServletRequestEvent sre)
        {
      System.out.println("Request Destroyed....");
        }
        public void attributeAdded(ServletRequestAttributeEvent srae)
        {
                System.out.println("Attribute added to request....");
        }
   public void attributeRemoved(ServletRequestAttributeEvent srae)
   {
        System.out.println("Attribute removed from request....");
   }
   public void attributeReplaced(ServletRequestAttributeEvent srae)
   {
        System.out.println("Attribute replaced in request....");
   }
}
```

SessionListener.java

```
package test;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
```

```
@WebListener
public class SessionListener implements HttpSessionListener,HttpSessionAttributeListener
{
        public void sessionCreated(HttpSessionEvent hse)
        {
                System.out.println("Session created....");
        }
   public void sessionDestroyed(HttpSessionEvent hse)
   {
        System.out.println("Session destroyed....");
   }
   public void attributeAdded(HttpSessionBindingEvent hsbe)
   {
        System.out.println("Attribute added to Session...");
   }
   public void attributeRemoved(HttpSessionBindingEvent hsbe)
   {
        System.out.println("Attribute removed from Session...");
   }
   public void attributeReplaced(HttpSessionBindingEvent hsbe)
   {
        System.out.println("Attribute replaced in Session...");
   }
}
```

Note:

=>Add these listeners to HttpSession Application

==========

**imp*

Servlet Life-Cycle:

=>Servlet Life-cycle demonstrates diff stages(states) of Servlet Program from starting to ending.

=>The following are some important stages of Servlet Life-cycle:

1.Loading process

2.Instantiation process

3.Initialization process

4.Request handling process

5.Destroying process

1.Loading process:

=>Servlet programs are identified based on url-pattern and loaded for execution is known as 'Loading process'.

2.Instantiation process:

=>when Servlet program loaded for execution,then it is automatically instantiated known as 'Instantiation Process'

Note:

=>After Instantiation process,the execution-controls will find the following Life-Cycle methods:

GenericServlet:

init()

service()

destroy()

HttpServlet:

init()

service()/doPost()/doGet()

destroy()

Filter:

init()

doFinter()

destroy()

3.Initialization process:

=>The process of making the programming components ready for execution is known as Initialization process,which means creating DAO object,Bean Objects and JCF object.

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

4.Request handling process:

=>The process of accepting request and providing the response is known as 'Request handling process'

=>we use service()/doPost()/doGet() method to perform Request Handling process.

5.Destroying process:

=>The process of closing the resources and services opened part of execution is known as destroying process.

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

==========

Dt : 16/11/2024(day-42)

**imp*

Web Application Architecture Models:

=>In the process of constructing applications,the realtime market will use some development architecture known as 'Web Application Architecture Models'

=>These Web Application Architecture models are categorized into two types:

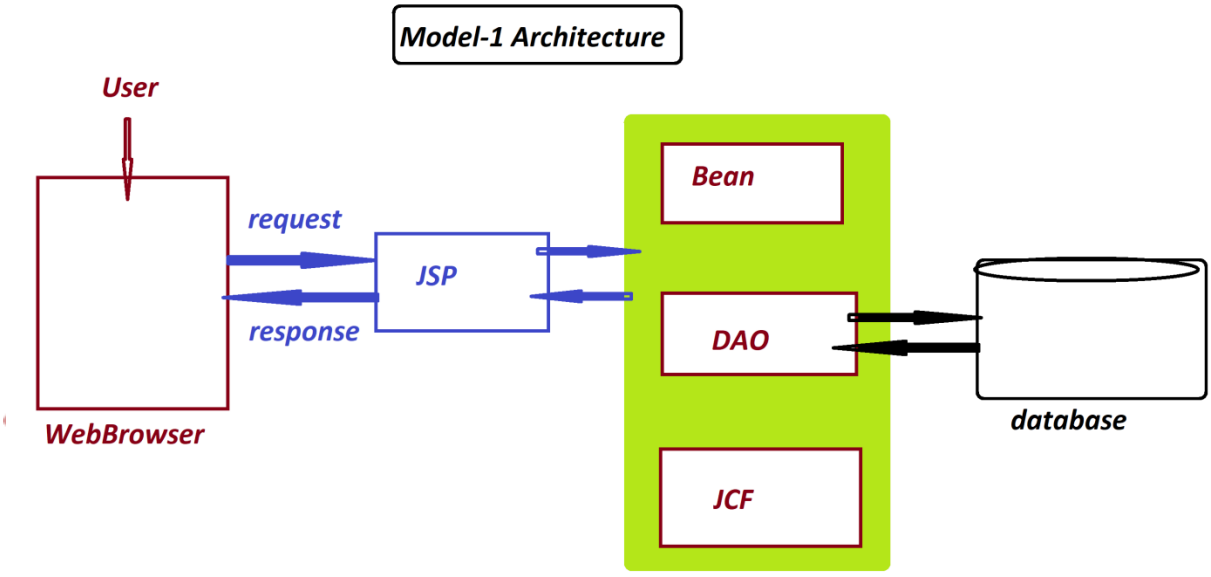
1.Model-1 Architecture

2.Model-2 Architecture

1.Model-1 Architecture:

=>In Model-1 Architecture JSP is the controller of application,which means JSP will control beans,DAO and JCFs.

Diagram:



*imp

2.Model-2 Architecture:

=>Model-2 Architecture introduced to overcome some dis-advantages of Model-1 Architecture.

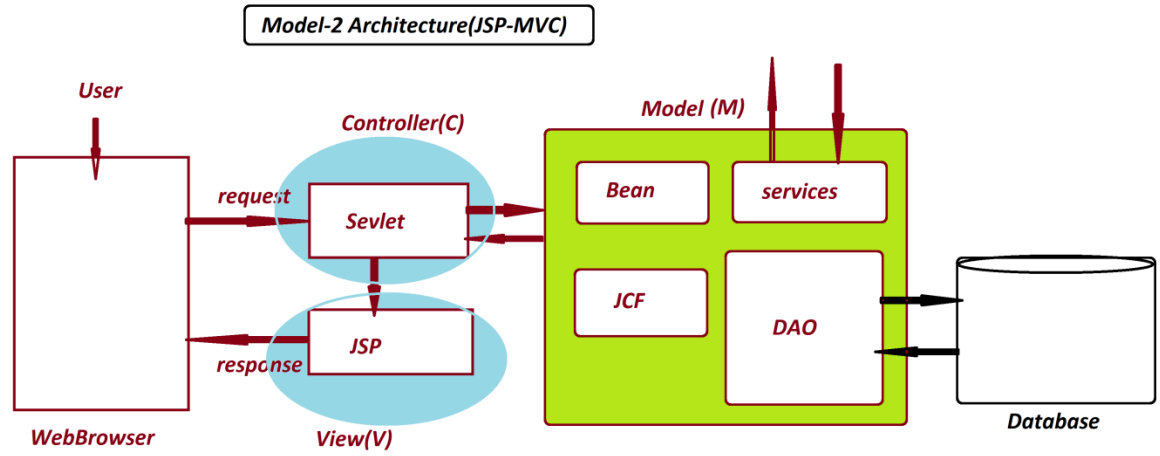
=>This Model-2 Architecture depends on MVC structure.

M - Model

V - View

C - Controller

Diagram:



M - Model

=>Model-layer represents beans where enterprise data is available.

=>Model-layer also includes JCF,DAO and services.

V - View

=>View-layer represents JSP,which is presentation of WebApp(response of WebApp)

C - Controller

=>Control-Layer represent servlet,which accepts the request and provide the response.

=>Servlet will control JSP and Model layers

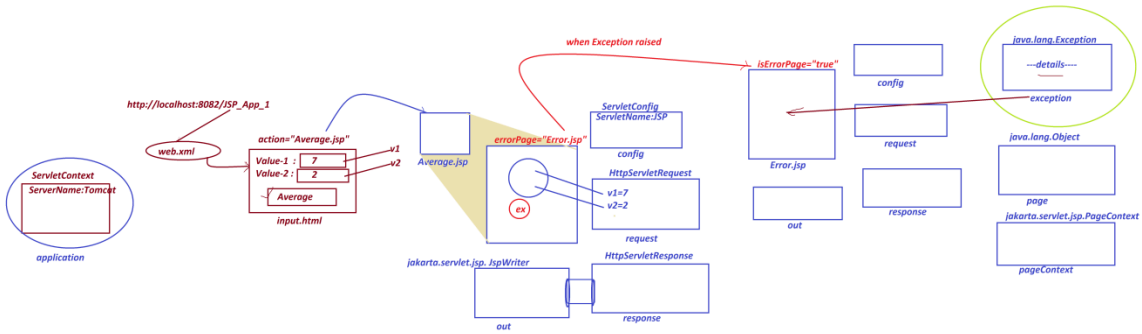
==========

Examples Demonstrating Model-1 Architecture:

Ex-1:

Construct JSP Application to demonstratte 'declarative tag'

Layout:



input.jsp

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="Average.jsp" method="post">
Enter the value-1:<input type="text" name="v1"><br>
Enter the value-2:<input type="text" name="v2"><br>
<input type="submit" value="Average">
</form>
</body>
</html>
```

web.xml

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

Average.jsp

```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>
<%!
float avg(int x,int y)
{
      float z = (float)(x+y)/2;
      return z;
}
%>
<%
int v1 = Integer.parseInt(request.getParameter("v1"));
int v2 = Integer.parseInt(request.getParameter("v2"));
float res = avg(v1,v2);
out.println("Average:"+res+"<br>");
%>
<%@include file="input.html" %>
</body>
</html>
```

Error.jsp

```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>
```

==========

Dt : 18/11/2024(Day-43)

**imp*

=>The following are the implicit Objects generated in JSP:

1.application - jakarta.servlet.ServletContext

2.config - jakarta.servlet.ServletConfig

3.request - jakarta.servlet.http.HttpServletRequest

4.response - jakarta.servlet.http.HttpServletResponse

5.out - jakarta.servlet.jsp.JspWriter

6.session - jakarta.servlet.http.HttpSession

7.exception - java.lang.Exception

8.page - java.lang.Object

9.pageContext - jakarta.servlet.jsp.PageContext

Ex-2:

(Construct JSP Application to demonstrate <jsp:include>,

<jsp:forward> and <jsp:param>)

Layout:

ProjectName : JSP_App_2

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 value-1:<input type="text" name="v1"><br>
Enter the value-2:<input type="text" name="v2"><br>
<input type="submit" value="Add" name="s1">
<input type="submit" value="Sub" name="s1">

</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>
```

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">
   <jsp:param value="<%=100 %>" name="k"/>
</jsp:forward>
<%
}else{
 %>
   <jsp:forward page="Subtraction.jsp">
       <jsp:param value="<%=200 %>" name="z"/>
   </jsp:forward>
 <%
}
%>
```

```
</body>
</html>
```

Addition.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 v1 = Integer.parseInt(request.getParameter("v1"));
int v2 = Integer.parseInt(request.getParameter("v2"));
int k = Integer.parseInt(request.getParameter("k"));
int v3 = v1+v2;
out.println("Sum:"+v3+"<br>");
out.println("Parameter Value:"+k+"<br>");
%>
<jsp:include page="input.html"/>
</body>
</html>
```

Subtraction.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 v1 = Integer.parseInt(request.getParameter("v1"));
int v2 = Integer.parseInt(request.getParameter("v2"));
int z = Integer.parseInt(request.getParameter("z"));
int v3 = v1-v2;
out.println("Sum:"+v3+"<br>");
out.println("Parameter value:"+z+"<br>");
%>
<jsp:include page="input.html"/>
</body>
</html>
```

==========

Assignment:

Update above application to handle exception for NonInteger values.

==========

Dt : 19/11/2024(Day-44)

Ex-3:

(Construct JSP Application to demonstrate <jsp:useBean>,

<jsp:setProperty> and <jsp:getProperty>)

Layout:

user.html

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

web.xml

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

UserBean.java

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class UserBean implements Serializable{
  private String uName,mId;
  private long phNo;
  public UserBean() {}
```

```
public String getuName() {
	return uName;
}
public void setuName(String uName) {
	this.uName = uName;
}
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;
}

}
```

Load.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.UserBean"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<jsp:useBean id="ub" class="test.UserBean" scope="session"/>
<jsp:setProperty property="uName" param="uname" name="ub"/>
<jsp:setProperty property="mId" param="mid" name="ub"/>
<jsp:setProperty property="phNo" param="phno" name="ub"/>
<%
out.println("data loaded to bean object Successfully...");
%>
<a href="View.jsp">ViewDetails</a>
</body>
</html>
```

View.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
```

```
    import="test.UserBean"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<jsp:useBean id="ub" type="test.UserBean" scope="session"/>
UserName:<jsp:getProperty property="uName" name="ub"/><br>
MailId:<jsp:getProperty property="mId" name="ub"/><br>
PhoneNO:<jsp:getProperty property="phNo" name="ub"/><br>
</body>
</html>
```

==========

*imp

Expression Language(EL):

=>Expression Language(EL) introduced into JSP to make the code more simple and can access all the implicit objects of JSP.

syntax:

$(Expression)

=>The folowing are the implicit objects of EL:

1.applicationScope

2.sessionScope

3.requestScope

4.pageScope

5.param

6.paramValues

7.header

8.headerValues

9.cookie

10.initParam

11.pageContext

1.applicationScope:

=>applicationScope is used to access the attribute from ServletContext.

2.sessionScope:

=>sessionScope is used to access the attribute from HttpSession.

3.requestScope:

=>requestScope is used to access the attribute from request-object.

4.pageScope:

=>pageScope is used to access the attribute from PageContext Object.

5.param:

=>param is used to access single parameter-value from the request-Object

6.paramValues:

=>paramValues is used to access multiple parameter-values from the request object

7.header:

=>header is used retrive HTTP protocol header name.

8.headerValues:

=>headerValues is used retrieve HTTP protocol multiple header names.

9.cookie:

=>cookie is used to get the cookies from request-object.

10.initParam:

=>initParam is used to access the initilization parameter-values from config object.

11.pageContext:

=>pageContext is implicit name used by JSP-EL to access PageContext-object

---

****imp***

Summary of Objects generated:

CoreJava:

1.User defined Class Objects

2.String-Objects

3.WrapperClass-Objects

4.Array-Objects

5.Collection<E> Objects

6.Map<K,V> Objects

7.Enum<E> Objects

JDBC:

1.Connection Object

2.Statement Object

3.PreparedStatement Object

4.CallableStatement Object

5.ResultSet Object

6.DatabaseMetaData Object

7.ParameterMetaData Object

8.ResultSetMetaData Object

Servlet:

1.ServletContext Object

2.ServletConfig Object

3.ServletRequest/HttpServletRequest Object

4.ServletResponse/HttpServletResponse Object

5.PrintWriter Object(for Servlet response)

6.HttpSession Object

7.Cookie Object

8.Bean Object

9.DAO Object

10.JCF Object

JSP:(Implicit Objects)

1.application

2.session

3.config

4.request

5.response

6.out

7.exception

8.page

9.pageContext

JSP-EL:(Implicit Objects)

1.applicationScope

2.sessionScope

3.requestScope

4.pageScope

5.param

6.paramValues

7.header

8.headerValues

9.cookie

10.initParam

11.pageContext

Diagram:

==========

Dt : 20/11/2024(Day-45)

define JSTL?

=>JSTL stands for 'JSP Standard Tag Librarie' and,which provide some tags

to make code more simple.

=>The following are some important JSTL tags:

1.Core Tags

2.Function Tags

3.Formatting Tags

4.XML Tags

5.SQL Tags

1.Core Tags:

=>The fundamental tags used for writing basic codes are known as CoreTags.

2.Function Tags:

=>The tags which support String functions are known as Function Tags.

3.Formatting Tags:

=>The tags which are used to format the data for persentation are known as Formatting tags.

4.XML Tags:

=>The tags which are used to write XML code in JSP are known as XML tags.

5.SQL Tags:

=>The tags which are used to write DB related Code in JSP are known as SQL Tags.

Note:

=>According to realtime application development using MVC,'SQL-Tags' and 'XML-Tags'

are deprecated.

=>"@taglib" directive tag is used to include JSTL libraries into Current Running

Program.

==========

Ex:(EL-JSTL Application)

step-1 : Download "jstl-1.2.jar" file from Internet Source.

step-2 : Copy the JSTL jar file into "lib" folder of JSP.

step-3 : Type and execute the program

Ex:

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form method="post" action="CoreTags.jsp">
 Enter the String:<input type="text" name="str"><br>
 <input type="submit" value="Display">
 </form>
</body>
</html>
```

CoreTags.jsp

```
<%@ 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>
```

==========

faq:

define 'page'?

=>'page' is an implicit object of JSP and which is generated from

'java.lang.Object' class

=>'page' is used in JSP to perform Cloning-process and Thread-Communication

process.

==========

faq:

define 'pageContext'?

=>'pageContext' is an implicit object of JSP and which is generated from

'PageContext' abstractClass.

=>This 'pageContext' object can access all other implicit JSP objects directly.

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

```
public abstract jakarta.servlet.ServletContext getServletContext();
public abstract jakarta.servlet.ServletConfig getServletConfig();
public abstract jakarta.servlet.http.HttpSession getSession();
public abstract jakarta.servlet.ServletRequest getRequest();
public abstract jakarta.servlet.ServletResponse getResponse();
public abstract jakarta.servlet.jsp.JspWriter getOut();
public abstract java.lang.Exception getException();
public abstract java.lang.Object getPage();
```

==========

faq:

JSP Life-Cycle:

=>JSP Life-Cycle demonstrates different states(stages) of JSP program from starting to ending.

=>The following are the stages of JSP program:

1.Translation process

2.Compilation process

3.Loading process

4.Instantiation process

5.Initialization process

6.Request Handling Process

7.Destroying process

1.Translation process:

=>The process of separing java-code from JSP-Program is known as Translation process.

2.Compilation process:

=>The process of compiling the Java-code and generating CompiledCode is known as Compilation process.

(After Compilation Process the generated SourceCode will be destroyed automatically)

3.Loading process:

=>The process of loading JSP-Program for execution is known as Loading process.

4.Instantiation process:

=>when the JSP program loaded for execution it is automatically instantiated known as Instantiation process.

Note:

=>After Instantiation process,the execution controls will identify the following LifeCycle methods:

(i)_jspInit()

(ii)_jspService()

(iii)_jspDestroy()

5.Initialization process:

=>The process of making the programming components ready for execution is known as Initialization process.

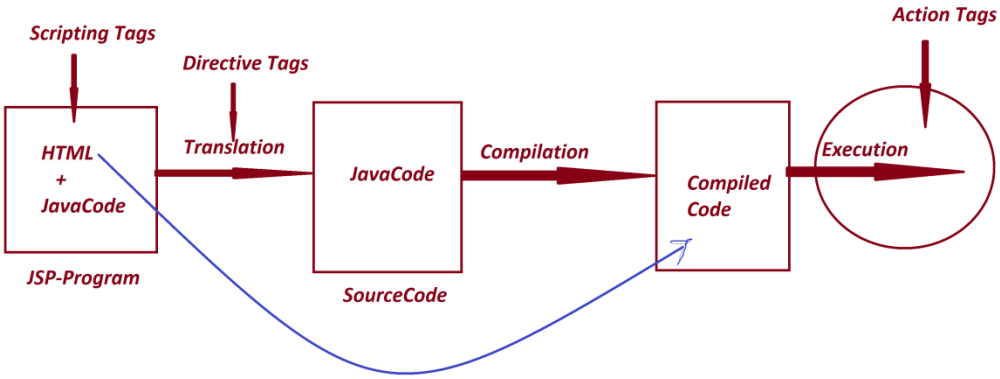
6.Request Handling Process:

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

7.Destroying process:

=>The process of closing the opened resources and services is known as Destroying Process.

Diagram:



==========

Code	Message	Description
100	Continue	Only a part of the request has been received by the server, but as long as it has not been rejected, the client should continue with the request
101	Switching Protocols	The server switches protocol.
200	OK	The request is OK
201	Created	The request is complete, and a new resource is created
202	Accepted	The request is accepted for processing, but the processing is not complete.
203	Non-authoritative Information	
204	No Content	
205	Reset Content	
206	Partial Content	
300	Multiple Choices	A link list. The user can select a link and go to that location. Maximum five addresses
301	Moved Permanently	The requested page has moved to a new url
302	Found	The requested page has moved temporarily to a new url
303	See Other	The requested page can be found under a different url
304	Not Modified	

305	*Use Proxy*	
306	*Unused*	*This code was used in a previous version. It is no longer used, but the code is reserved*
307	*Temporary Redirect*	*The requested page has moved temporarily to a new url.*
400	*Bad Request*	*The server did not understand the request*
401	*Unauthorized*	*The requested page needs a username and a password*
402	*Payment Required*	*You cannot use this code yet*
403	*Forbidden*	*Access is forbidden to the requested page*
404	*Not Found*	*The server cannot find the requested page.*
405	*Method Not Allowed*	*The method specified in the request is not allowed.*
406	*Not Acceptable*	*The server can only generate a response that is not accepted by the client.*
407	*Proxy Authentication Required*	*You must authenticate with a proxy server before this request can be served.*
408	*Request Timeout*	*The request took longer than the server was prepared to wait.*
409	*Conflict*	*The request could not be completed because of a conflict.*
410	*Gone*	*The requested page is no longer available.*

411	*Length Required*	*The "Content-Length" is not defined. The server will not accept the request without it.*
412	*Precondition Failed*	*The precondition given in the request evaluated to false by the server.*
413	*Request Entity Too Large*	*The server will not accept the request, because the request entity is too large.*
414	*Request-url Too Long*	*The server will not accept the request, because the url is too long. Occurs when you convert a "post" request to a "get" request with a long query information.*
415	*Unsupported Media Type*	*The server will not accept the request, because the media type is not supported.*
417	*Expectation Failed*	
500	*Internal Server Error*	*The request was not completed. The server met an unexpected condition.*
501	*Not Implemented*	*The request was not completed. The server did not support the functionality required.*
502	*Bad Gateway*	*The request was not completed. The server received an invalid response from the upstream server.*
503	*Service Unavailable*	*The request was not completed. The server is temporarily overloading or down.*

504	*Gateway Timeout*	*The gateway has timed out.*
505	*HTTP Version Not Supported*	*The server does not support the "http protocol" version.*