JDBC BY RAGHU SIR

API:-API is a set of classes & Interfaces used to perform any task.

API→Application Programming Interface

Creating API:-

Step 1:-Write classes & Interfaces and save as _____.java file

Step 2:-Compile above java files using Javac command which generates .class files.

Step 3:-Pack all .class files into .Jar (Final executable) using command Java/Java, .Jar is called API.

Sun/Oracle has provided two API's Listed below

API Name	Usage
Java Mail	To send Emails
JDBC	To perform Database operations (insert, update, delete, select)
Servlets & JSP	To develop dynamic web applications.
JDBC → Java Database Connection. JAXB →Java Architecture for XML Binding.	

Web Applications:-

It is a collection of web pages, running in web server, can be accessed by web-browser by using request and response process.

Web Applications are 2 types

1.>Static Web Application.

2.>Dynamic Web Application.

1.>Static Web Application:-

On every access by web-browser, it will show same output (fixed content) such application are called [non-change] web Application.

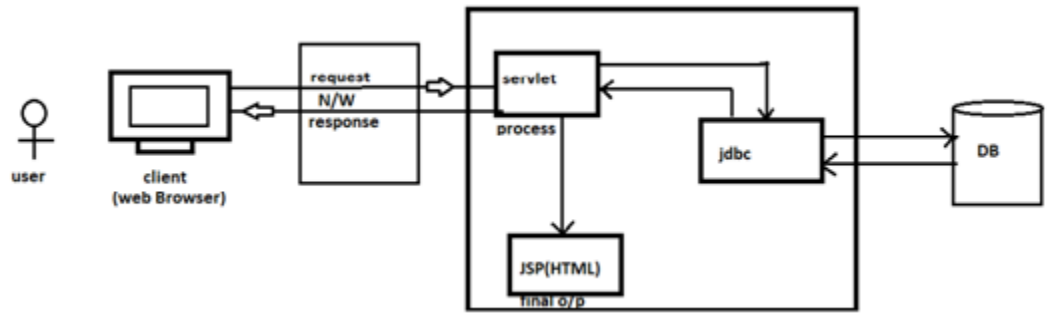
HTML (Hyper Text Markup Language] is a base concept to develop web pages.

2.>Dynamic Web Application:-

Content of web page changes based on requirement, such applications are called as Dynamic (Change content) Web Applications.

=>Advance Java, ASP.net, PHP are used to Develop Dynamic web Application.
=>To Develop Web Applications using Advance Java concepts used are:
Servlets, JSP (Java Server Pages) and JDBC (Java DataBase Connectivity).
It can also use concept like HTML, SQL.

Dynamic Web Applications Architecture using Advance Java:-



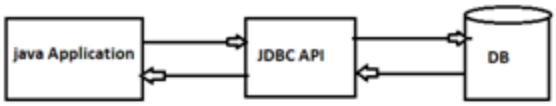
=>User should operate client machine which has one software installed i.e. web browser (ex. Firefox, Chrome, …..)
=>By using web browser user can make request, which will be passed through network. Request will be given to provider which has one software installed ie.web server (ex: Apache Tomcat, ApacheHttp, IIS, Glassfish, Jetty…..).
=>In web server request will be taken by Servlet and it will process it.
If Servlets needs data or wants to store data in DB, it uses JDBC.
=>JDBC makes communication link between any java Application and Database.
=>Servlet will dispatch data to JSP (Java Server Pages) for final output.
=>This is given as response back to client machine. (ex: Response: Inbox Page, Login Page, Login Fail Page etc……)

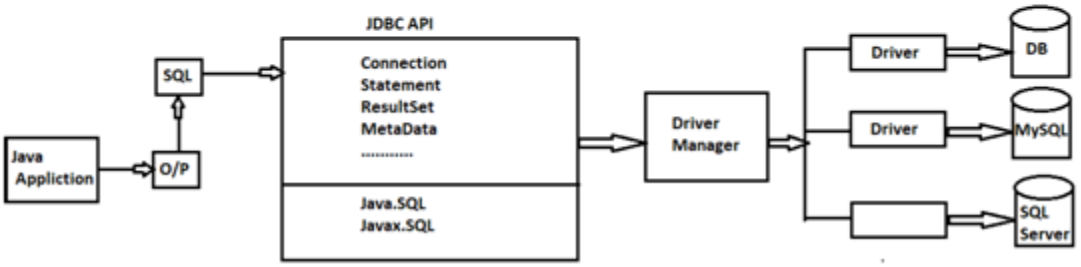
JDBC API:-

JDBC stands for java database connectivity. It is an API (Application Programming Interface) giver by Sun Microsystems to perform “CURD” operations with Database using Java.
C=Create (Create Database/Tables/Rows…)
U=Update (modify tables/rows)

R=Read (Selecting data from Database)

D=Delete (remove rows/tables/DBs)





JDBC Architecture:-

Driver:-It is a pre-defined class (in simple java software) which creates a communication link between JDBC & Database.

=>Without Driver no JDBC concept works.

=>For Every Database Driver Software (class) is required.

=>DriverManager is a class which takes care of Driver, like

->Is this correct driver or not?

->Is this working or not?

->Is this loaded or not?

Example Driver Names:-

Oracle Driver = For oracle DataBase

MySQLDriver = For MySQL DataBase

SybaseDriver = For Sybase DataBase

>> JDBC API means classes and interfaces defined in two packages. Those are:

1>Java Sql

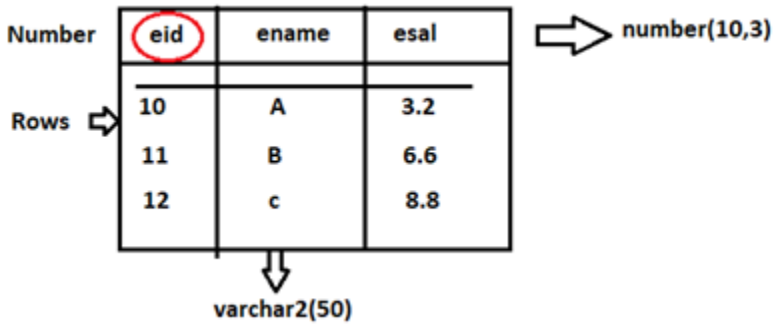
2>Javax.Sql (Extended)

=>Few classes and interfaces in above packages are given as connection, Statement, PreparedStatement, ResultSet, DatabaseMetadata, RowSet etc.

=> To work with JDBC, our output should be converted to SQL which can be understandable by Database.

Metadata:-

=>It provides information of Data (Data about Data) consider below table for data and metadata.



Ex:- Data is 10 its Metadata is

Type:- Number

Column – eid

Is Primary key – yes

Accept null = no

Is unique = yes

Writing JDBC Application:-

Here Programmer has to follow below steps, to do any operation (Select or non-select) with Database.

Those are:

1.>Load Driver class:-It will make a link between java Application and Database software >> It is a class in JDBC which implements Driver interface.

2.>Creating connection:-To create connection we should provide location of Database (URL) and username with password of Database for Access Permission to java.

3.>Create Statement:-It indicates one SQL query which should be given to DB. For one operation one SQL= one statement.

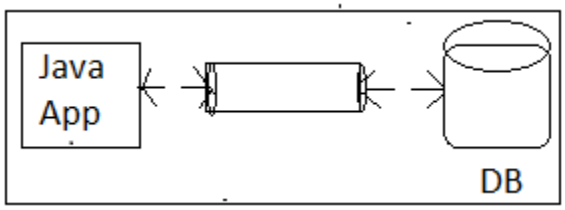
4.>Execute Statement:-Give above SQL to DB and execute it.

5.>Get result:-Final result of above SQL Query execution is given as either ResultSet obj [Select Operation] (or)

int/Boolean [Non – select operation]

6.>Print result:-Print above result using Sysout statement.

7.>Close connection:-Finally close connection between Java Application & Driver (class), url, username, password.



Types of Operations in Database:-

Select operations : Select query

Non – Select operations : insert, update, delete query

ResultSet is applicable for select query

int or Boolean is applicable for non-select query commonly used SQL queries in Database.

COMMONLY USED SQL QUERIES IN DATABASE:-

Database user:-

=>create user <username> identified by <password>;

grant dba to <username>;

=>alter user <username> identified by <password>;

=>conn username/password;

=>drop user <username> cascade;

Database Table:-

=>Create table <tablename> (eid number, ename varchar2 (20), esal number(5,2));

desc <tablename>;

=>alter table <tablename> dropcolumn columnname;

=>alter table <tablename> addcolumn datatype;

=>alter table <tablename> modifycolumn datatype;

Rename Table and column Names:-

=>alter table <tablename> rename to <newtablename>

alter table <tablename>renamecolumn <oldcolumnname>to newcolumnname.

DML queries:-

=>insert into <tablename> values (?, ?, ?);

=>update <tablename>set columnname= value (?)

=>truncate table <tablename>;

=>dalete from <tablename> where eid=1;

Data Select from Table:-

=>select * from emptab;

=>select ename from emptab:

=>select esal, ename from emptab;

=>select count(a) from emptab;

Edit Buffer:-

ed/edit with /.

JDBC Testing Setup Application:-Write one java application after installing software's, like JDK, Eclipse, Oracle/MySQL Databases.

Step 1:-Load Driver class using below statement

```
Class.forName(driverClass);
```

=>Here forName(String cls) is a state method defined in java.lang.class. It is used to load any class into memory (JVM) even driver class changes.

Step 2:-Creating connection between Java Application and Database using below statement.

```
Connection con=DriverManager.getConnection(url,username,password);
```

=>Here DriverManager is a class from java.sql.packae and having a static method getConnection type object. Here connection is interface given from java.sql.package.

Coding Steps:-

1>Open Eclipse Software.

2>Provide a folder name (workspace) or Just click OK.

3>Change to Java Mode.

Window-> perspective -> open perspective ->other -> choose java option -> finish.

4>Create java Project file -> new -> Java Project.

Enter any project name ex: jdbcFirstApp ->Finish.

5>Add jar (ojdbc/MySQL-connector) to Application,

->Right Click on project -> Build Path -> configure build path libraries -> Add external Jars -> choose jar -> apply ->Apply & close.

6) Write one class with code.

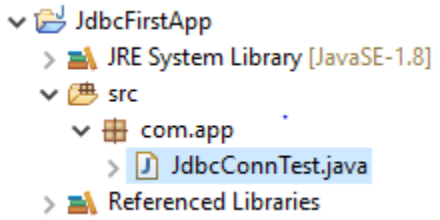
Right click on src -> new -> class -> enter package & class name

Ex: Package: com.app

Name: JdbcConnTest -> finish

(ctrl +(plus): zoom In , ctrl –(minus) : zoom out to increase display in eclipse/STS)

Folder System:-



Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;

public class JdbcConnTest {
public static void main(String[] args) throws Exception {
        //1.jdbc properties-oracle DB
         String driverclass="oracle.jdbc.driver.OracleDriver";
         String url="jdbc:oracle:thin:@localhost:1521:xe";
         String username="system";
```

```
        String password="manager";
        //2.jdbc code
        Class.forName(driverclass);
Connectioncon=DriverManager.getConnection(url,username,password;
        System.out.println(con);
        }
}
```

Output : Run As/ctrl+F11

Oracle Jar Location:

c:\oracleexe\app\oracle\product\10.2.0\server\jdbc\lib

First Application in JDBC:-

Step#1 open Eclipse and create Java Project

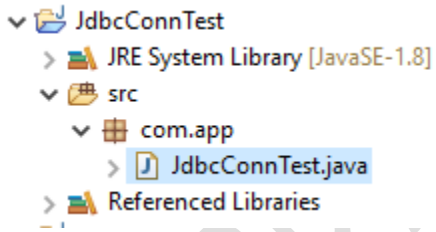
Step#2 Add ojdbc jar to project (using Build path)

Step#3 create a class under src folder

Package : com.app

Name : JdbcConnTest

Folder System:-

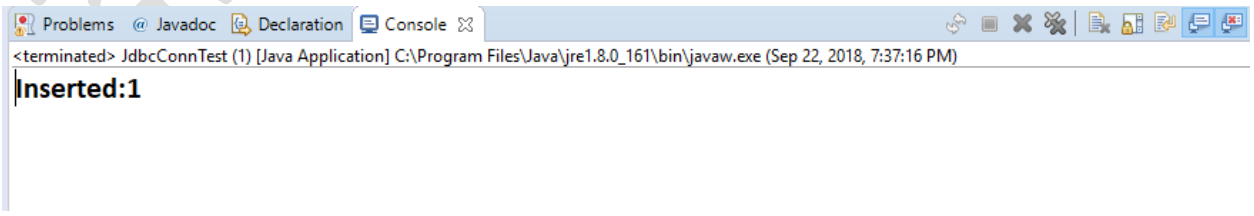


Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class JdbcConnTest
{
```

```
public static void main(String[] args) throws Exception
{
        //1.jdbc properties
        String driverclass="oracle.jdbc.driver.OracleDriver";
        String url="jdbc:oracle:thin:@localhost:1521:xe";
        String username="system";
        String password="manager";
        String sql="insert into student values(1,'vijay',5.5)";

        //jdbc code
        //load DriverClass
        Class.forName(driverclass);
        //2.create Statement connection
        Connectioncon=DriverManager.getConnection(url,username,password;
        //3.create Statement
        Statement st=con.createStatement();
        //4.Execute Statement
        //5.Get Result
        int count=st.executeUpdate(sql);
        //6.Print result
        System.out.println("Inserted:"+count);
        //7.close connection
        con.close();
        }
}
```

Output:--



Before Running this Goto Oracle command line and create below Queries.

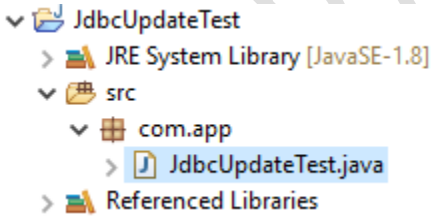
#1.open SQL cmd prompt

Window=>oracleDatabase=>Run SQl commandline

#2.login to Oracle
=>conn system/manager
#3.create table for student
=>create table student (sid number, sname varchar2(20), sfee number(5,2));
#4.Type select Query to check result
=>select * from student;

NOTE:-If table is not created or created and deleted (now not available) then if we run above application.
Output is : Exception
Java.sql.SQL Exception : Table () or view does not exist.
#1.open Eclipse/STS software and change to java mode
Windows=>Perspective=>Open Perspective=>other=>choose java option
#2.create new Java Project
#3.Add ojdbc jar for Oracle DB support
Right click on project=>Build path=>configure build path=>Library path=>click on Add External Jars=>choose Ojdbc jar
#4.Creta a class under src folder
Right click on src folder=>new =>class=>Enter Details
Package : com.app
Name : JdbcUpdateTest=>Finish

Folder System:-

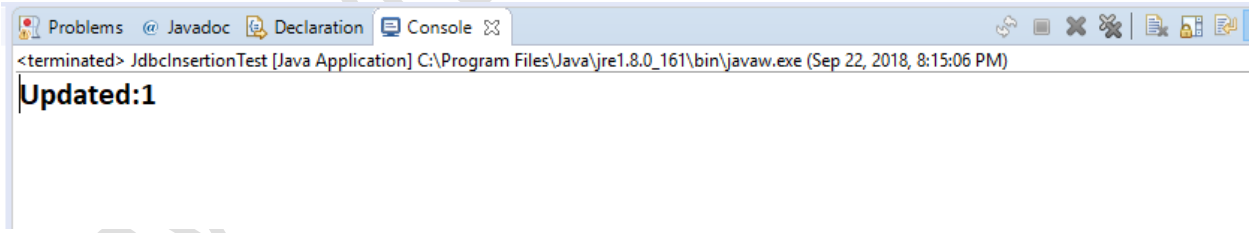


Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;
```

```
import java.sql.Statement;

public class JdbcUpdateTest {
    public static void main(String[] args) throws Exception {
        //jdbc properties
        String driverclass="oracle.jdbc.driver.OracleDriver";
        String url="jdbc:oracle:thin:@localhost:1521:xe";
        String username="system";
        String password="manager";
        String sql="update student set sname='venky',sfee=5.5,where    sid=1";

        //jdbc code
        Class.forName(driverclass);
        Connection con=DriverManager.getConnection(url, username, password);
        Statement st=con.createStatement();
        int count=st.executeUpdate(sql);
        System.out.println("Updated:"+count);
        con.close();
        }
}
```

Output:--



Update SQL:-This is used to modify the existed data in the table

Syntax is:-

update <table name> set <column name>=<value>,…….where <condition> .

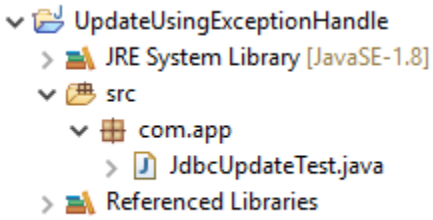
>> If where clause is not provided the query update all rows in database table.

Exceptions in JDBC:-

Exception:-It is a runtime error which terminates applications execution.
Every Exception is a class.
Exceptions are two types based on handling
a) Checked Exception
b) UnChecked Exception
a)Checked Exception:-A class that extends java.lang.Exception class is known as checked exception which must be handled by Programmer before compiling. Either use try-catch blocks (or) use throws keyword at method.
Ex: ClassNotFoundException, SQLException….etc.

b)UnChecked Exception:-A class that extends java.lang.RuntimeException class is know as unchecked Exception. It is not required to be handled by programmer.
Ex: NullPointerException, ArithmeticException……..etc.

Folder System:-

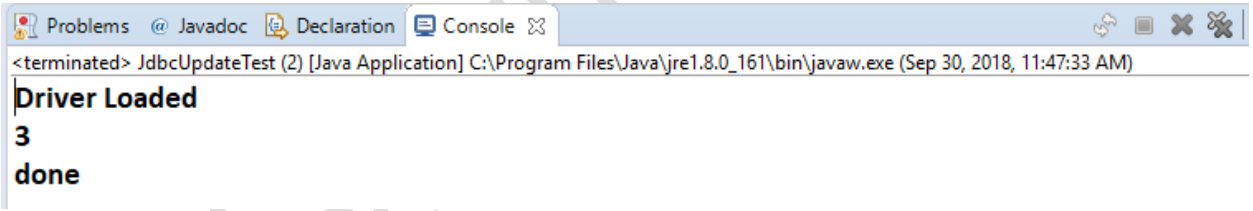


Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class JdbcUpdateTest {
    public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
            System.out.println("Driver Loaded");
             } catch (ClassNotFoundException cnfe) {
            System.out.println("driver class is invalid");
             }
        try {
Connection con=DriverManager.getConnection("jdbc:oracle:thin:
@localhost:1521:xe","system","manager");
             Statement st=con.createStatement();
             int count=st.executeUpdate("update student set sfee=10.5");
            System.out.println(count);
            con.close();
            System.out.println("done");
        } catch (SQLException sqle) {
            System.out.println("JDBC code problem:"+sqle);
                }
        }
}
```

Output:-



Fetching Data from Database using JDBC:-

=>Write “select SQL” query to get from Database to java Application.

=>Here Data will be stored in ResultSet object format (it is memory type inJDBC).

=>java.sql.ResultSet(I) is a interface it’s implementation class object is created by using method : ExecuteQuer(String sql).

=>ResultSet memory is created using column index, column label, column data type and Data in Rows format. Index number start from 1 (one).

=>ResultSet contains methods like:

next(): Boolean => this method is used to check next row exist or not in ResultSet.

get <DataType> (int columnIndex) (or) get<DataType> (String columnLable)

=>These methods are used to read data from ResulSet.

Ex: String s=rs.getString(2); (or) String s=rs.getString("ename");

f>Index number may change based on query select column order. But column labels will never get changed.

Ex#1: Select eid, ename, esal from emptab

Colum name	Index number
Eid	1
ename	2
esal	3

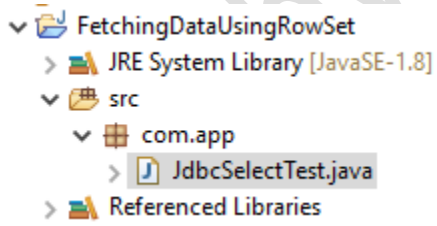
Ex #2: Select ename, esal, eid from emptab

Column name	Index number
Ename	1
esal	2
eid	3

g>ResultSet creates default pointer with position to zero(0^{th}) row also called as before to FirstRow.

(###@)To change pointer position use rs.next();

Folder System:-

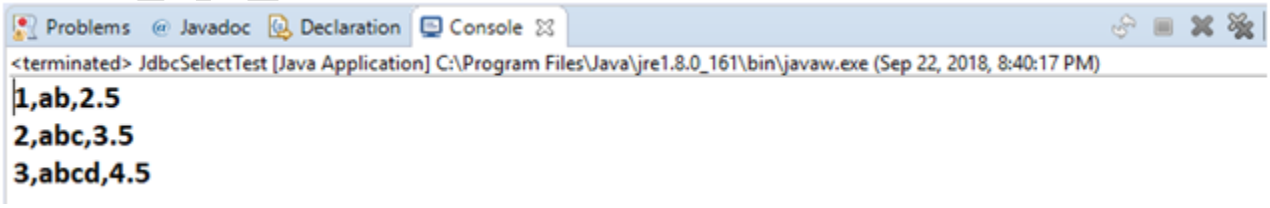


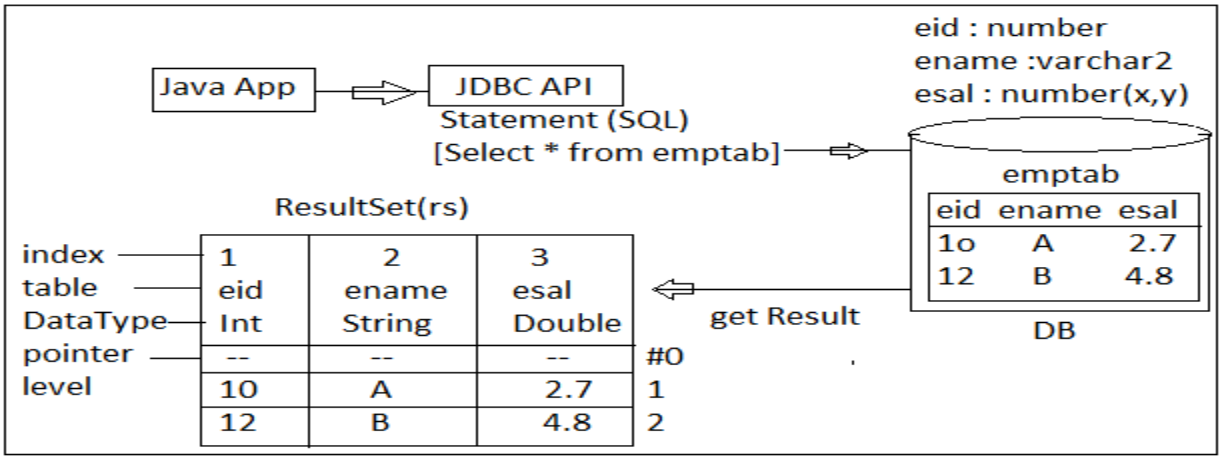
Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
```

```
public class JdbcSelectTest
{
    public static void main(String[] args) throws Exception
    {
        //jdbc properties
        String driverclass="oracle.jdbc.driver.OracleDriver";
        String url="jdbc:oracle:thin:@localhost:1521:xe";
        String username="system";
        String password="manager";
        String sql="select * from employee1";

        //jdbc code
        Class.forName(driverclass);
        Connection con = DriverManager.getConnection(url,username,password);
        Statement st=con.createStatement();
        ResultSet rs=st.executeQuery(sql);
        while(rs.next())
        {
        System.out.println(rs.getInt("eid")+","+rs.getString("ename")+","+
         rs.getDouble("esal"));
        }
        con.close();
   }
}
```

Output:-





DATABASE SETUP:-

1)Open SQL CommandLine and connect

=>conn system/manager

2)Create emptab

=>create table emptab(eid number,ename varchar(25),esal number(10,3));

3)Insert data

insert into emptab values(1,'ab',2.5);

insert into emptab values(2,'abc',3.5);

insert into emptab values(3,'abcd',4.5);

4)Finally save data

commit(press Enter)

DRIVERS IN JDBC:-

A driver is a class which implements an interface"java.sql.Driver" given by sun micro systems(Oracle corp.) which is part of JDK.

Ex: class MyLink implements java.sql.Driver {

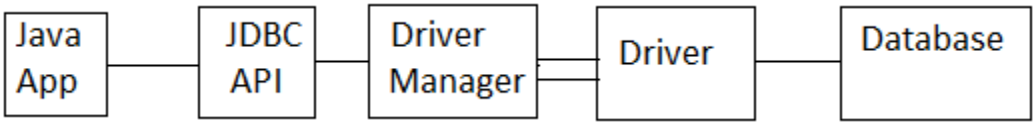
……..

}

=>Here MyLink can be called as Driver class(or) Driver software(or)Driver

=>Driver class is handled by DriverManager which is part of JDBC API.With out no JDBC program works.

Generic Architecture of JDBC Driver:-



Types of Drivers in JDBC:-

JDBC supports 4(Four) types of Driver. Given list as:-

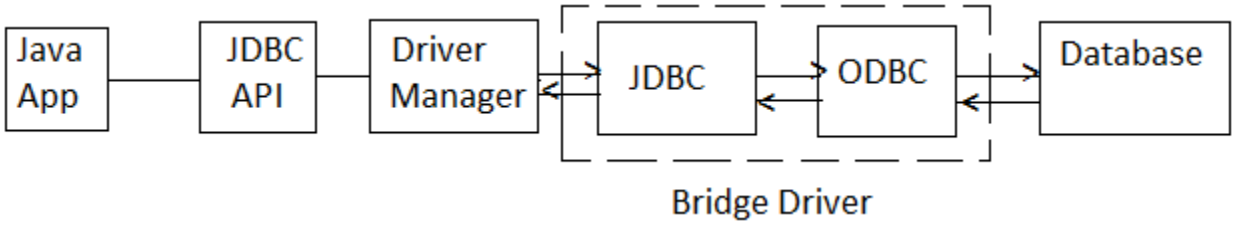
Type-1➔JDBC-ODBC Bridge Driver

Type-2➔Native API Driver (Partial Java Driver)

Type-3➔Net-Protocol Driver/Network Based Driver (Middle Ware Server)

Type-4➔Thin Driver/Native Protocol (100% Java Driver)

Type-1 JDBC-ODBC Bridge Driver:-



=>ODBC stands for “Object Data Base Connectivity”. This is software used by platforms (Operating system=OS) for Database connections.

=>This is very old process to link with database

For this sun Micro system has given class named as JdbcOdbcdriver(sun.jdbc.odbc package).

=>It will convert JDBC calls to ODBC calls and communicate with DB.

JDK8 onwards it is not supported.

Advantages:-

i>It supports almost major databases.

Disadvantages:-

i>In OS, we should install ODBC drivers (Client machine-OS).

ii>It is machine (OS) Dependent.

iii>It works very slow compared with other types.

iv>JDK 8 onwards it is not supported.

Coding Steps:-

#1. Download JDK7 or lesser version and install

#2. Create DSN (DataSourceName)

DSN means a name given to Database,to identify DB by ODBC(by os)

Mycomputer➔c:driver➔eindows folder➔system32

Ex:- c:\windows\system32 (or) c:\windows\system64

Click on option “odbcad32.exe”

Click on system DSN➔click Add

Choose oracle11G or oracleXE(one)

Next➔provide details like

DataSource Name : oraclemydsn

Description : none(optional)

TNS ServiceName : ORCL/XE

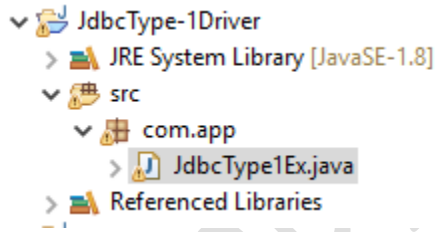
userId : system(username)

click on ok

driver class name is : sun.jdbc.odbc.JdbcOdbcDriver

url is >jdbc:odbc : <dsnName>

Folder System:-



Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;

public class JdbcType1Ex {
	public static void main(String[] args) throws Exception {
		//jdbc properties
		String driverclass="sun.jdbc.odbc.JdbcOdbcDriver";
```

```
            String url="jdbc:odbc:oraclemydsn";
            String user="system";
            String password="admin";
            //jdbc code
            Class.forName(driverclass);
            Connection con=DriverManager.getConnection(url, user, password);
            System.out.println("done");
      } }
```

Output:-

Type-2 Driver NATIVE API DRIVER:-Native is a keyword in java. It is used to link java language with c-language (non-java) code.

Native can be provided to method also called as "native methods".

A method which declared in java (non-java) is known as native method.

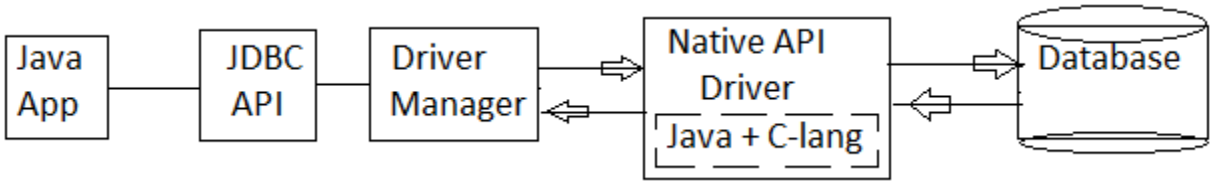
Example to native method is hashcode() from java.lang.object class.

Code look like:-

```
Public native int hashcode();
Ex:    java:-                          c-lang:-
   Native void show();                      void show() {
    (no-body)                                ----code----
    (declared)                             } (defined)
```

=>Native methods are used to execute code which cannot be done in java

Ex:-pointer (memory management operation)



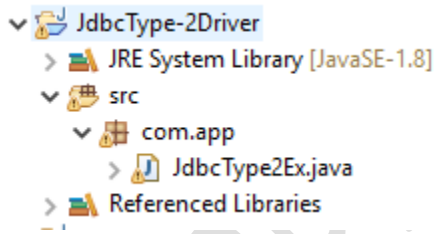
=>This is type-2 Driver in java which uses both java and c-language concept (native keyword).

=>Oracle has provided this type-2 driver implementation name as “OCI” (oracle Call Interface (or) Open Call Interface).
=>OCI is process of connecting DB using C-Language(Memory Operation using Pointers) This OCI is also called as OCI8 I Oracle 10g/11g version(here 8 means OCI version number).

CODING STEPS:-

Oracle10G➔ojdbc14.jar
Location of Jar: C:\oracleexe\app\oracle\product\10.2.dserver\jdbc\lib
Oracle11G➔ojdbc6.jar
Location of jar: D:\app\sathya\product\11.2.0\dbhome-1\jdbc\lib
URL Syntax:
Jdbc:oracle:oci:@<SID> (or)
Jdbc:oracle:oci8:@<SID>
SID=Service Identifier of Database
SQL Query to find SID name in DB
Select * from global_name; oracle11g➔ORCL

Folder System:-

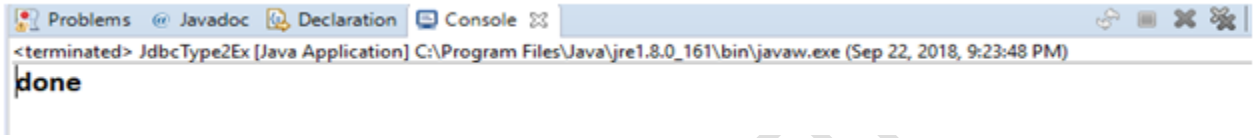


Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;

public class JdbcType2Ex {
public static void main(String[] args) throws Exception {
        //jdbc properties-oracle DB
        String driver="oracle.jdbc.driver.OracleDriver";
```

```
        String url="jdbc:oracle:oci8:@XE";
        String username="system";
        String password="manager";
        //jdbc code
        Class.forName(driver);
        Connection con = DriverManager.getConnection(url,username,password);
        System.out.println("done");
        }
}
```

Output:-



Advantages:-

=>It is faster than Type-1 driver

Disadvantages:-

=>DB vendor(database people) has to implement this using native library and install in client machine.

=>It is not available for all DBs(Given mainly for oracle DB).

=>It will not support Applets.

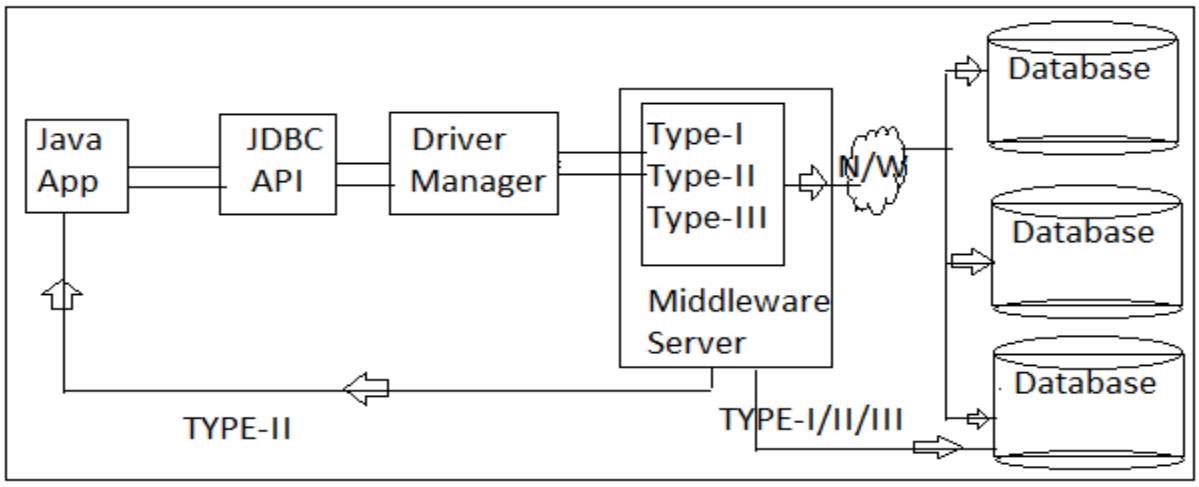
Type-3 driver : NET-PROTOCOL DRIVER/NETWORK BASED DRIVER (MIDDLE WARE SERVER):-

=>A server which behaves as a mediator between two Applications (source and Destination).It is used to “Switch to new destination without modifying source application”

=>In JDBC type-3 driver is used to connect Java App with server. This server will connect to any DB without doing code changes to java application.

=>Here Driverclass, URL in JDBC app is fixed .No changes required to link another DB from current DB.

=>But it internally uses Type-1/2/4 to link with DB.



=>IDS server is an open source server given by Ids solution to handle Type-3 driver in JDBC.

IDS stands for➔Internet Database access Server

LINK : http://www.idssoftware.com/download.html

=>Download & install.

=>It supports only oracle XE 10G not 11G DB, using only Type-1

Checking IDS server status:-

➔Windows key + R➔Enter ”services.msc”

➔Search for IDS server➔right click and run/start

Open below location to find Jar

C:\IDSServer\classes

JAR name is: jdk13drv.jar(add this to build path in Eclipse)

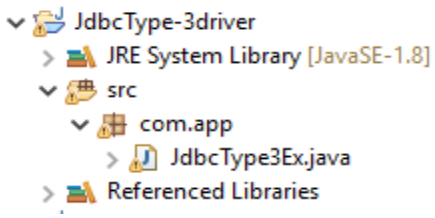
Driver class: ids.sql.IDSDriver

URL:jdbc:ids://<IPADDRESS>:12:/conn?dsn=<DSN>

=>To create DSN follow Type-1 driver steps

(IDS support Type-1 Driver)

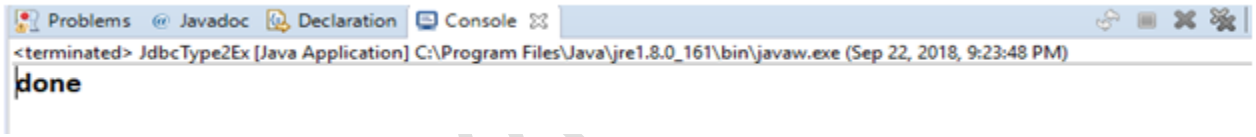
Folder System:-



Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;

public class JdbcType3Ex {
	public static void main(String[] args) throws Exception {
		String driverclass="ids.sql.IDSDriver";
		String url="jdbc:ids://localhost:12/conn?dsn=oracleab";
		String user="system";
		String password="admin";
		Class.forName(driverclass);
		Connection con=DriverManager.getConnection(url, user, password);
		System.out.println("done");
	} }
```

Output:-



Advantages:-

=>One driver used to connect with any Database.

=>No database related libraries required [(ojdbc.6)/ojdbc.7/14]

Disadvantages:-

=>It is very slow compared with all other drivers. We must install extra works as middleware server (IDS Server)

=>It needs network support.

=>Middleware server should have implemented Type-1/2/4 driver already

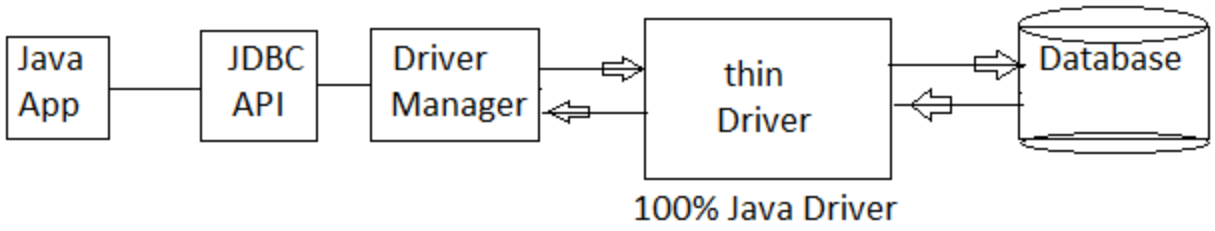
Type-4 driver (Native Protocol Driver/Thin Driver/100% Pure Java Driver):-

=>This is fastest Driver compared with others. It is implements in complete JavaCode.

=>It is light Weight (less memory) so, it is called as thin driver.

=>It never depends on OS/Native Language or any middle ware servers.

=>It makes simple java calls to db, to fetch or send data.



For Oracle DB:-

Driverclass = oracle.jdbc.driver.OracleDriver(or)
oracle.jdbc.OracleDriver

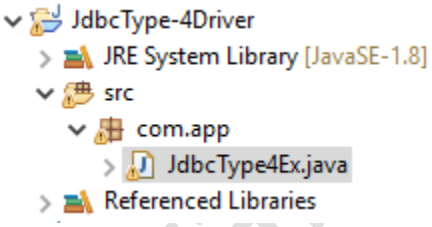
url = jdbc:oracle:thin:@<IPADDRESS>:<PORT>:<SID>

1)finding sid : login to DB
Select * from global_name;
O/P : SID (Ex:ORCL/XE)

2)Finding port no : tnsping➔open in cmd prompt➔tnsping SID(Ex:tnsping ORCL)
(observe……(PORT=1521)

3)finding IP Address
➔open cmd prompt➔ipconfig(or)use localhost
IPV4 Address…….:192.168.3.112
url = jdbc:oracle:thin:@localhost:1521:xe;

Folder System:-

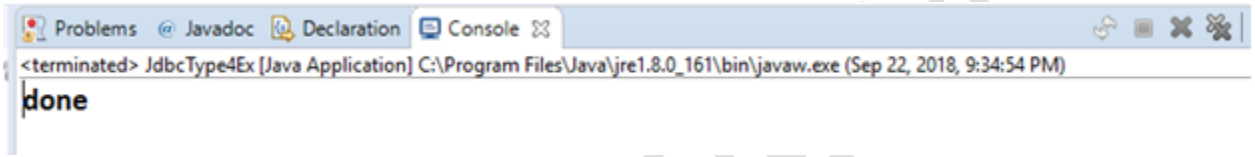


Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;
public class JdbcType4Ex  {
    public static void main(String[] args) throws Exception {
        //jdbc properties-oracle DB
```

```
        String driverclass="oracle.jdbc.driver.OracleDriver";
        String url="jdbc:oracle:thin:@localhost:1521:xe";
        String username="system";
        String password="manager";
        //jdbc code
        Class.forName(driverclass);
        Connection con = DriverManager.getConnection(url,username,password);
        System.out.println("done");
    } }
```

Output:-



Advantages:-

It is pure java driver, no other dependencies

It is fastest driver compared with all.

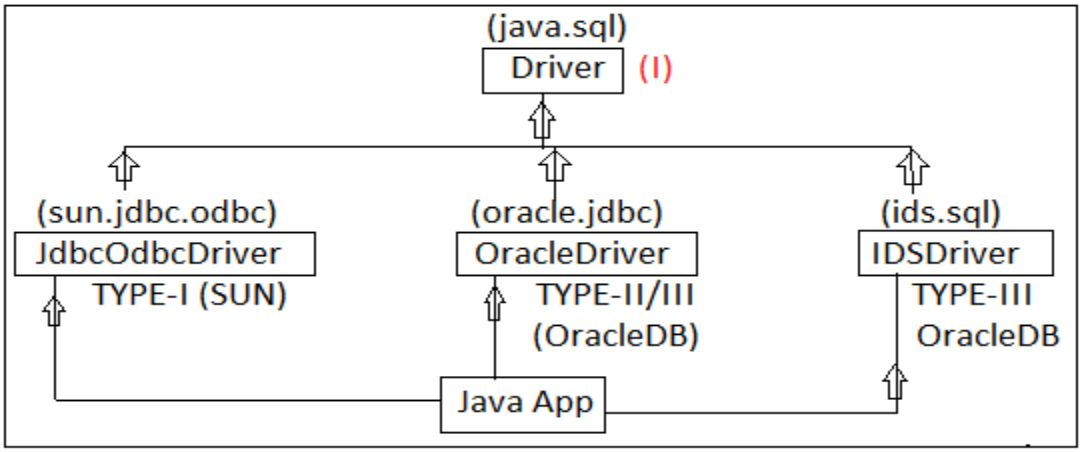
Supports for almost all high level databases

Disadvantages:-

For every database need Jar to get connect with DB.

Db to DB Driver class changes

Driver details for all types in JDBC:-



Connecting with MYSQL database:-

Softwares:

MYSQL community server 5.5.version

JAR : mysql-connector jar

JAR LOCATION : https://mvnrepository.com/artifact/
mysql/mysql-connector- java/5.1.6

Click on jar(.....) option to download

MYSQL JDBC Properties (for Type#4)

DriverClass= com.mysql.jdbc.Driver

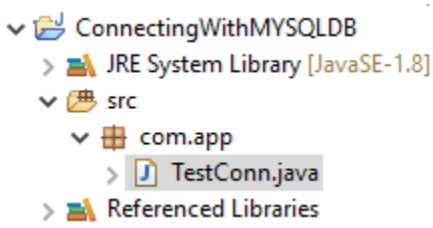
URL syntax : jdbc:mysql://<IP ADDRESS>:<PORT>/<dbname>

Ex: jdbc:mysql://localHost:3306/test

Username=root;

Password=root;

Folder Structure:-

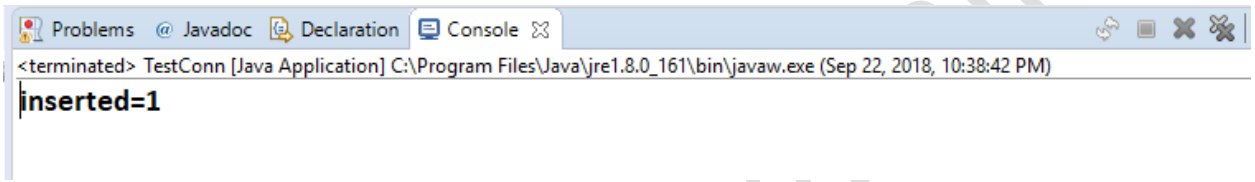


Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class TestConn {
    public static void main(String[] args) throws Exception
    {  //jdbc properties-MYSQL DB
       String driverclass="com.mysql.jdbc.Driver";
       String url="jdbc:mysql://localhost:3306/test";
       String username="root";
       String password="root";
```

```
        String sql="insert into student values(4,'d',8.6)";
        //jdbc code
        Class.forName(driverclass);
        Connection con = DriverManager.getConnection(url,username,password);
        Statement st=con.createStatement();
        int count=st.executeUpdate(sql);
        System.out.println("inserted="+count);
    }
}
```

Output:-



MYSQL Commands:-

1)see all databases

2)connect to one DB

mysql>use test;

3)see all tables

mysql>showtables;

4)create table

mysql>create table student(sid int, sname varchar(25), sfee(double);

5)insert row

mysql>insert into student(1,'a', 3.3);

6)save data

mysql>commit

Q)How many ways we can load/register Driver class?

ANS:-In official way only 2 formats

using Class.forName

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

b)using registerDriver() method

Ex: Driver driver =new OracleDriver();
DriverManager.registerDriver(driver);

Few other ways (write such code that will call static block)

Ex1: new OracleDriver();

Ex2: Driver d=new OracleDriver();

Ex3: Sysout(OracleDriver.BUILD-DATE);

➔In new JDBC API(4.x)It supports auto-loading of driver class i.e. without even step#1 in JDBC program works.

Ex : use jar ojdbc6.jar for oracle.

PostgreSQL Database:-Connection with JDBC

Software: postgreSQL-10.4-1-win64-bigsql.exe

JAR: postgresql-42.2.4.jar

JdbcProperties:-

Driverclass = org.postgresql.Driver

URL= jdbc:postgresql://<IP ADDRESS>:<PORT>/<DB>

Username= postgres password = admin

Step#>1 Go to postgreSQL

Start/windows button➔AllPrograms

postgreSQL➔click on PSQL

Step#2>create database and switch to it

➔Create database sathya

➔\c sathya

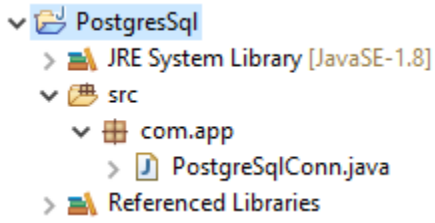
Step#3>create table

Create table student (sid int, sname char(30),sfee (float));

Step#4>insert one row

Insert into student values (1,'a', 3.3);

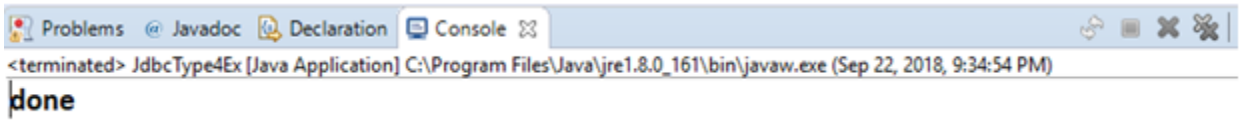
Folder Structure:-



Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class PostgreSqlConn {
public static void main(String[] args)throws Exception {
        //jdbc properties-postgreSQL DB
        String driver="org.postgresql.Driver";
        String url="jdbc:postgresql://localhost:5432/sathya";
        String username="postgres";
        String password="root";

         Class.forName(driver);
        Connection con = DriverManager.getConnection(url,username,password);
        System.out.println("done");
    }
}
```

Output:-



Prepared Statement:-It is also Interface which extends Statement Interface.

=>It will compile SQL query only one time (first time) 2nd time on wards. Only one RUN (no syntax check no compile).

=>It is faster compared to statement for multiple runs.

->Every SQL query re-write values with place holder (?) or positional parameter(?).

->Here ? Symbol indicates data comes at runtime or (later).

->This place holder is applicable for only value position not in column or table name Position

Ex (Valid Example):-

a>Insert into student values (?, ?, ?)

b>Update student set sname = ?, sfee = ? where sid = ?

c>delete from student where sid = ?

d>delete * from student where sname = ? or (sfee = ? and sid = ?)

Invalid ex:-

a>insert into ? values (1, 'a', 3.9)

b>update ? set ? = 10, ?=3.3 where ? = 55;

c>delete from ? where sid = 55;

d>select ?, ?, ? from ? where sid =55;

=>Every question mark symbol written in SQL query will be identified using one Number (position Number) start from one (1),(1,2,3…..)

Ex:-

a>insert into student values(?, ?, ?);

b>update student set sname =?, sfee = ? where sid !=?

c>select sid from student where sid > ? or sfee < > ?

=>< > means it is not equals to (:=) in DB.

Syntax:-

=>Overview of prepared statement code.

1>Load drivers

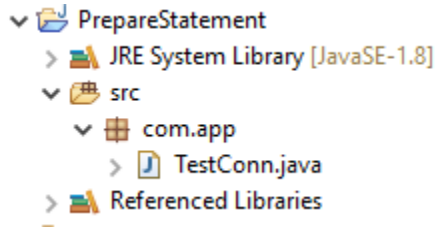
2>Create connection (con)

String sql = "……..?, ?, ?";
PreparedStatement ptmt con.prepareStatement(sql);
//Call set method to provide data
psmt.set<DataType> (int index, <DataType> value);
//execute update query..
//get and print result
//close connection

Insert Operation:-

Step#1:- Create student (sid, sname, sfee) table in DataBase.
Step#2:-Write below class in Eclipse

Folder structure:-

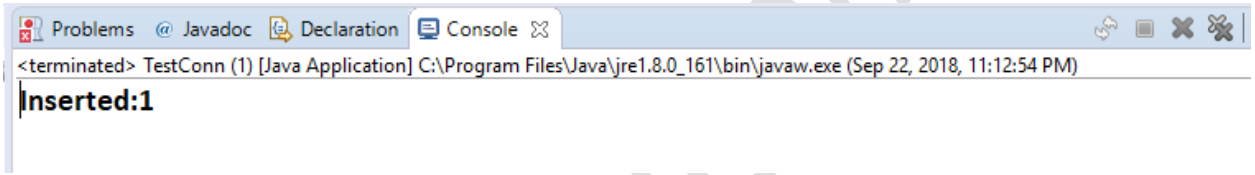


Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class TestConn {
public static void main(String[] args)throws Exception {
    //jdbc properties-MYSQL DB
    String driver="oracle.jdbc.driver.OracleDriver";
    String url="jdbc:oracle:thin:@localhost:1521:xe";
    String username="system";
    String password="manager";
    String sql="insert into student values(?,?,?)";
    //jdbc code
```

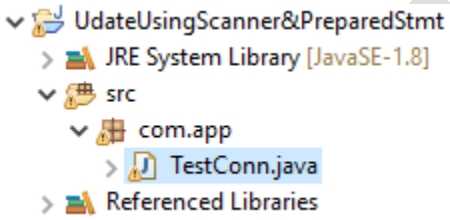
```
        Class.forName(driver);
        Connection con = DriverManager.getConnection(url,username,password);
        PreparedStatement pstmt=con.prepareStatement(sql);
        pstmt.setInt(1,10);
        pstmt.setString(2,"AJ");
        pstmt.setDouble(3,3.3);
        //execute
        int count=pstmt.executeUpdate();
        System.out.println("Inserted:"+count);
        con.close();
    }
}
```

Output:-



Update Row using Scanner& PreparedStatement:-

Folder Structure:-



Code:-

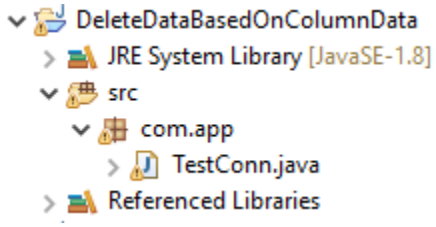
```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Scanner;

public class TestConn {
   public static void main(String[] args)throws Exception {
```

```
        //jdbc properties-MYSQL DB
        String driver="oracle.jdbc.driver.OracleDriver";
        String url="jdbc:oracle:thin:@localhost:1521:xe";
        String username="system";
        String password="manager";
        String sql="update student set sname=?,sfee=? where sid=?";
        //jdbc code
        Class.forName(driver);
        Connection con = DriverManager.getConnection(url,username,password);
        //create pstmt
        PreparedStatement pstmt=con.prepareStatement(sql);
        //use scanner to read data from console
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Student ID:");
        int stdId=sc.nextInt();
        System.out.println("Enter New Name:");
        String stdName=sc.next();
        System.out.println("Enter New Fee");
        double stdFee=sc.nextDouble();
        //setData
        pstmt.setString(1, stdName);
        pstmt.setDouble(2, stdFee);
        pstmt.setInt(3, stdId);
        //execute
        int count=pstmt.executeUpdate();
        if(count==0)
                System.out.println("No rows updated");
        else
                System.out.println("rows updated");
                System.out.println("Inserted:"+count);
        con.close();
        }
}
```

Output:-

Delete Student Based on column data using switch case option:-

Folder structure:-



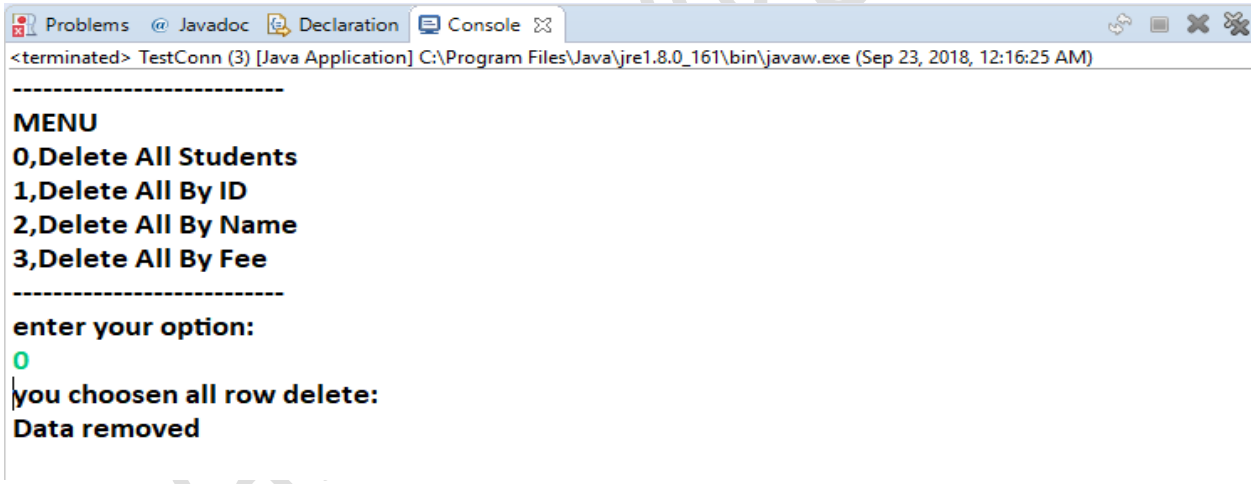
Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Scanner;

public class TestConn {
    public static void main(String[] args)throws Exception {
        //jdbc properties-MYSQL DB
        String driver="oracle.jdbc.driver.OracleDriver";
        String url="jdbc:oracle:thin:@localhost:1521:xe";
        String username="system";
        String password="manager";
        String sql="delete from student";
        //jdbc code
        Class.forName(driver);
        Connection con = DriverManager.getConnection(url,username,password);
```

```
PreparedStatement pstmt=null;
System.out.println("-------------------------");
System.out.println("MENU");
System.out.println("0,Delete All Students");
System.out.println("1,Delete All By ID");
System.out.println("2,Delete All By Name");
System.out.println("3,Delete All By Fee");
System.out.println("-------------------------");
Scanner sc=new Scanner(System.in);
System.out.println("enter your option:");
int option=sc.nextInt();

switch (option) {
case 1:
	sql+="where sid=?";
	System.out.println("enter student Id:");
	int stdId=sc.nextInt();
	pstmt=con.prepareStatement(sql);
	pstmt.setInt(1, stdId);
	break;
case 2:
	sql+="where sname=?";
	System.out.println("Enter student Name:");
	String stdName=sc.next();
	pstmt=con.prepareStatement(sql);
	pstmt.setString(1, stdName);
	break;
case 3:
	sql+="where sfee=?";
	System.out.println("Enter student fee:");
	Double stdFee=sc.nextDouble();
	pstmt=con.prepareStatement(sql);
	pstmt.setDouble(1, stdFee);
```

```
                break;
        default:
                pstmt=con.prepareStatement(sql);
                System.out.println("you choosen all row delete:");
                break;
                }
        int count=pstmt.executeUpdate();
        if(count==0)
                System.out.println("sorry!!Details not Found");
        else
                System.out.println("Data removed");
        con.close();
      }
}
```

Output:-



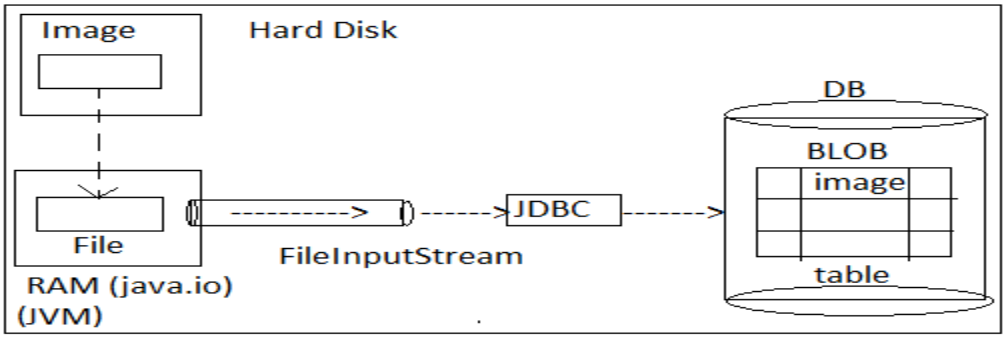
Storing Image in Database (ORACLE DB):-

=>An image can be stored in database using JDBC API, this can be implemented with Prepared Statement only.

=>For this JDBC program image should be loaded into java.io.File format and then read date using FIS(FileInputStream).

=>In database table, column datatype should be BLOB (BinaryLargeObject).It can store image,audio,video,document,excelfiles,PPTs etc..,

---



Step#1 create a database table shown as below

(number)	(varchar)	(blob)
fid	fname	fdata
100	uday.jpg	A1027...

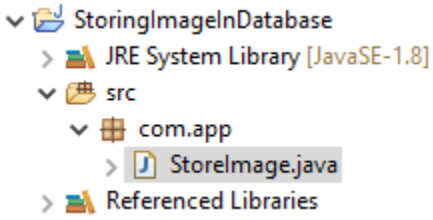
SQL:-create table imagetab(fid number , fname varchar2(25) , fdata blob);

Step#2 write JDBC Application and use PreparedStatement Object with SQL Calls.

Set BinaryStream(int index,InputStream is,int length):Void

Method to set BLOB data

Folder Structure:-

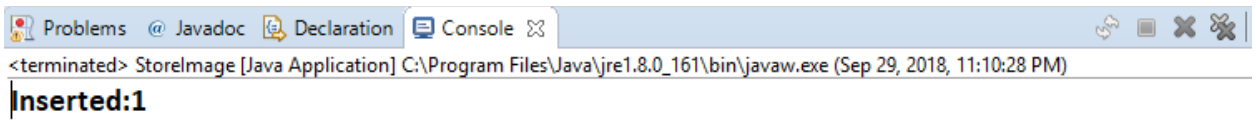


Code:-

```
package com.app;
import java.io.File;
import java.io.FileInputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
```

```
public class StoreImage {
	public static void main(String[] args) throws Exception {
	  //jdbc properties
	  String driverclass="oracle.jdbc.driver.OracleDriver";
	  String url="jdbc:oracle:thin:@localhost:1521:xe";
	  String username="system";
	  String password="manager";
	  String sql="insert into imagetab values (?,?,?)";
	  //jdbc code
	  Class.forName(driverclass);
	  Connection con = DriverManager.getConnection(url,username,password);
	  //1.load image into file object(RAM)
	  File f=new File("F:\\VENKY\\1.jpg");
	  //2.read data from file object
	  FileInputStream fis=new FileInputStream(f);
	  //3.create pstmt object
	  PreparedStatement pstmt=con.prepareStatement(sql);
	  //4.set basic data
	  pstmt.setInt(1,100);
	  pstmt.setString(2, "aa.jpg");
	  //5.set image data into BLOB
	  pstmt.setBinaryStream(3, fis,(int)f.length());
	  //6.execute pstmt
	  int count=pstmt.executeUpdate();
	  //7.print result
	  System.out.println("Inserted:"+count);
	  con.close();
	  } }
```

Output:-



Q>What are major difference between Statement and PreparedStatement?

Statement	PreparedStatement
1)It is a interface used to execute DDL Queries(also support DML). DDL: create/alter/drop	1) It is a interface used to execute DML Queries (insert, update, delete)
2)Every time it will be compiled by database	2)Only first time compiled. Also called as pre-compiled statements
3)It is slow Compared to PSTMT	3)It is faster second time onwards compared to STMT.
4)support only basic operations and basic data types	4)Supports even Large Objects(BLOB) along with data types.

COMMON THINGS:-

=>Both are Interfaces.

=>Both are from java.sql package.

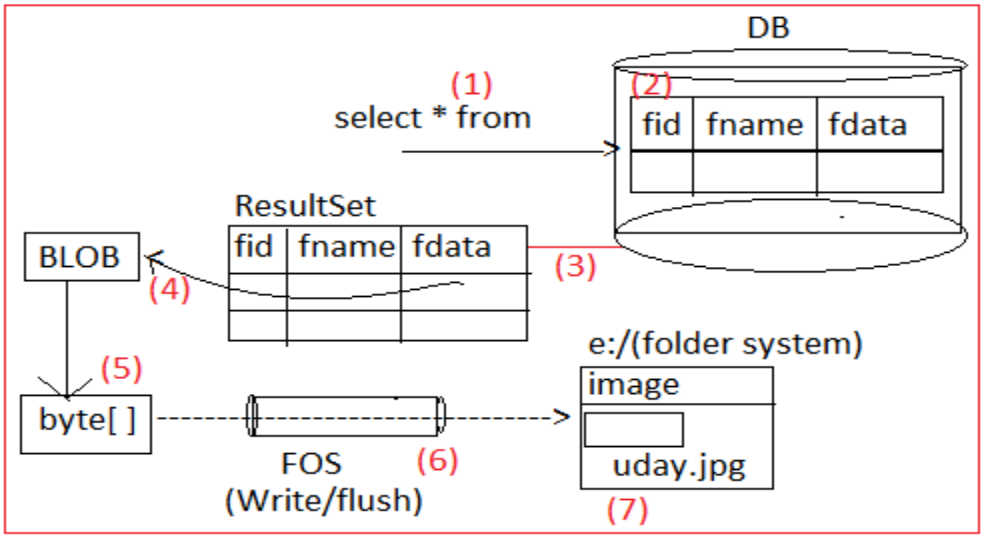
=>Both are used to represent SQL queries.

=>PreparedStatement extends Statement.

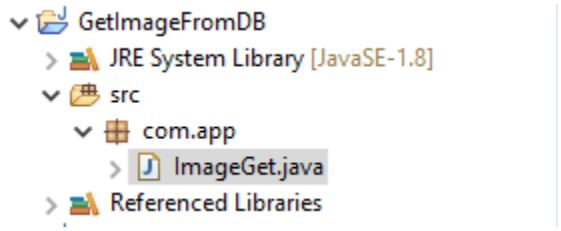
Get Image from Database to System Driver (Folder System):-

=>JDBC has provided one special class named as Blob(java.sql) used to store BLOB Data, selected from Database.

=>This java.sql.Blob data convert to byte[ ] and store in a file using FileOutputStream.



Folder Structure:-

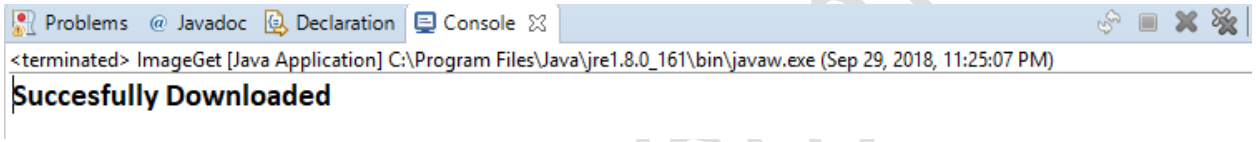


Code:-

```
package com.app;
import java.io.FileOutputStream;
import java.sql.Blob;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class ImageGet {
	public static void main(String[] args) throws Exception {
	  //jdbc properties
	  String driverclass="oracle.jdbc.driver.OracleDriver";
	  String url="jdbc:oracle:thin:@localhost:1521:xe";
	  String username="system";
	  String password="manager";
	  //SQL Query
	  String sql="select * from imagetab where id=?";
	  //jdbc code
	  Class.forName(driverclass);
	  Connection con = DriverManager.getConnection(url,username,password);
	  PreparedStatement pstmt=con.prepareStatement(sql);
	  pstmt.setInt(1, 100);
	  ResultSet rs=pstmt.executeQuery();
	  if(rs.next()) {
		Blob img=rs.getBlob(3);
		byte[] arr=img.getBytes(1,(int)img.length());
```

```
                FileOutputStream fos=new FileOutputStream("E:\\aa.jpg");
                fos.write(arr);
                fos.flush();
                fos.close();
                System.out.println("Done");
        }
        else {
                System.out.println("No image found with given Id");
        }
        con.close();
    } }
```

Output:-



Oracle Function:-

Functions are used to execute set of SQL queries at a time in an order to do a final work.

➔Every function contains

- Function name
- Inputs to function
- One output
- Return type
- Function body

SYNTAX:-

```
        Create or replace function<name>(inputs)
        Return<datatype>
        IS/AS var datatype;
BEGIN
SQL1, SQL2…..;
return var;
 END;
```

=>A function is used to execute a task.

JDBC CALLABLE STATEMENT - ORACLE FUNCTION CALL:-

Callable Statement:--

=> This is used to call functions or Procedures in JDBC.

(**only for call not for creating).

=>In general statement and Prepared Statements works on tables where as Callable Statements works on functions and Procedures.

Q>What is the difference between function and procedure in oracle?

Ans:-Function returns value (return type is applicable) but Procedure never returns value(no return type). We can get date using OUT parameter in Procedure if required.

Step#1 Create Table In oracle DB:-

CREATE TABLE EMPTAB (EID NUMBER, ENAME VARCHAR2(20), ESAL NUMBER(12,3));

Step#2 Insert Data:-

=>INSERT INTO EMPTAB VALUES (10,'AA', 23.5);

=>INSERT INTO EMPTAB VALUES (11,'BB', 24.5);

=>INSERT INTO EMPTAB VALUES (12,'CC', 65.5);

=>INSERT INTO EMPTAB VALUES (13,'DD', 89.5);

Step3# Create a Function In oracle Database:-

CREATE OR REPLACE FUNCTION GETEMPNAME (EMPID NUMBER) RETURN VARCHAR2
AS EMPNAME VARCHAR2(25);
BEGIN
SELECT ENAME INTO EMPNAME FROM EMPTAB WHERE EID=EMPID; RETURN EMPNAME;
END;
Step#4 Test Function in Oracle SELECT GETEMPNAME(13) FROM DUAL;
Step#5 Define JDBC Code

Code:-

```
package com.app;
import java.sql.CallableStatement;
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.Types;
public class TestFunction {
    public static void main(String[] args) throws Exception {
        String driverClass="oracle.jdbc.OracleDriver";
        String url="jdbc:oracle:thin:@localhost:1521:XE";
        String user="system"; String password="manager";
        String sql="{?=call GETEMPNAME(?)}";
        Class.forName(driverClass);
        Connection con=DriverManager.getConnection(url, user, password);
        CallableStatement cstmt=con.prepareCall(sql);
        cstmt.registerOutParameter(1, Types.VARCHAR);
        cstmt.setInt(2, 10);
        cstmt.execute();
        String empName=cstmt.getString(1);
        System.out.println(empName);
        con.close();
    }
}
```

ROWSET STEPS:-

STEP#1 create Rowset object using its implementation class

Ex: JdbcRowSet rs=new OracleJdbcRowSet();

STEP#2 provide Properties like

url, username, password, command(SQL)

Ex:-rs.setUrl("jdbc:oracle:thin:@localhost:1521:XE");

rs.setUsername("system");

rs.setCommand("select * from emptab");

STEP#3 Execute rowset object

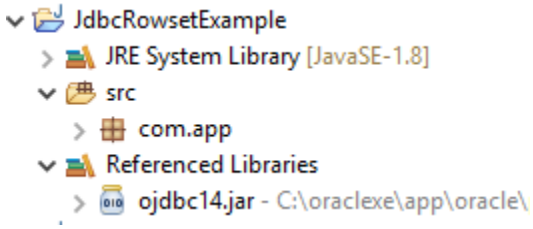
Ex: rs.execute();

STEP#4 Read result from rowset using rowset methods

Ex:-Methods are

rs.isBeforeFirst(); rs.relative();
rs.isAfterLast(); rs.first();
rs.next(); rs.last();
rs.previous(); rs.absolute();

Folder Structure:-



Code:-

```
package com.app;
import javax.sql.rowset.JdbcRowSet;
import oracle.jdbc.rowset.OracleJDBCRowSet;

public class JdbcRowsetEx {
    public static void main(String[] args) throws Exception{
        JdbcRowSet rs=new OracleJDBCRowSet();
        rs.setUrl("jdbc:oracle:thin:@localhost:1521:xe");
        rs.setUsername("system");
        rs.setPassword("manager");
        rs.setCommand("select * from student");
        rs.execute();
        System.out.println("CHECK CURSOR POSITION");
        boolean first=rs.isBeforeFirst();
        boolean last=rs.isAfterLast();
        System.out.println("BeforeFirst:"+first);
        System.out.println("AfterLast:"+last);
        System.out.println("MOVE FORWARD");
        while(rs.next()) {
        System.out.println(rs.getInt(1)+","+rs.getString(2)+","+rs.getDouble(3));
        }
```

```
        System.out.println("MOVE BACKWARD");
        while(rs.previous()) {
       System.out.println(rs.getInt(1)+","+rs.getString(2)+","+rs.getDouble(3));
        }
        System.out.println("MOVE TO 2ND ROW");
        rs.absolute(2);
        System.out.println(rs.getInt(1)+","+rs.getString(2)+","+rs.getDouble(3));
        System.out.println("MOVE ONE ROW BACK");
        rs.relative(-1);
        System.out.println(rs.getInt(1)+","+rs.getString(2)+","+rs.getDouble(3));
        System.out.println("GO TO FIRST ROW");
        rs.first();
        System.out.println(rs.getInt(1)+","+rs.getString(2)+","+rs.getDouble(3));
        System.out.println("GO TO LAST ROW");
        rs.last();
        System.out.println(rs.getInt(1)+","+rs.getString(2)+","+rs.getDouble(3));
        rs.close();
        }
}
```

Output:-

Note:-

isBeforeFirst() and isAfterLast() return true/false if cursor in that position or not.

rs.next(); is used to move cursor to next row(forward) in same way rs.presvious(); is used to move cursor back.

rs.absolute(position); is used to go to exact row number if exist.

rs.relative(+ve/-ve number) is used to go nearest row of those shifts forward & backward from current position.

rs.first(); moves curser to first row & rs.last(); move curser to last row.

Ex: rs.absolute(5); go to 5th row.

rs.relative(-2);go to backward 2nd row

rs.next(); go to next row.

rs.last(); go to last row.

rs.relative(3); go to forward 3rd rows.

CachedRowSet (I):-

CachedRowSet will execute command (SQL) and copies data into temporary memory "catch" and remove the connection. i.e. Dis-Connected Mode, where as JdbcRowSet will not close connection until our program ends i.e, Connected Mode.

Types of Temporary memory (2):-

1) Cache 2) Pool

1)Cache:-It is a group of different types (classes) of objects

Ex: Employee object and student objects and Admin object etc.

2)Pool:-It is a group of similar types of objects (All objects are related to one class).

Ex: Employee pool=only Employee object

Admin pool=only Admin objects

Connection pool=only Connection objects

Additional feature of CachedRowSet:-

1) Convert data to collection objects using toCollection() method

Code : Collection<?> cs=rs.toCollection();

2) Create another copy of cashedRowSet using create copy () methods.

Code : CachedRowSet rs2=rs.createCopy();

Note:-

1) Implementation class for CachedRowSet is: OracleCachedRowset (c).

2)CachedRowSet also supports all methods like JdbcRowSet methods.

Ex: next(),previous(),absolute(),first(),last() etc.

WebRowSet (I):-

It is a SerializedRowSet works in disconnected mode.

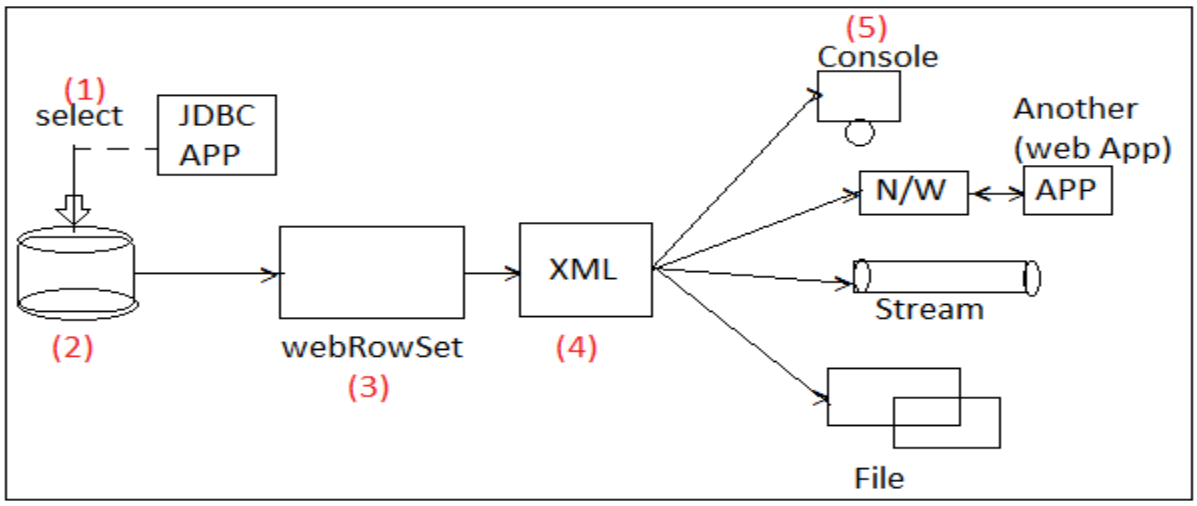
➔It supports all concepts supported by CachedRowSet and also this data can be sent over network, file, and stream in global (XML) format.

Code:- WebRowSet rs=new OracleWebRowSet();

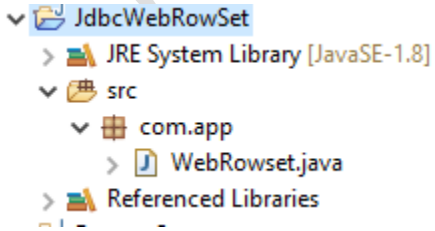
……..

rs.writexml(System.out);

➔WebRowSet converts data into global format like XML(tag based) so we can send to another application, even developed in other language(internally used web Services supports)



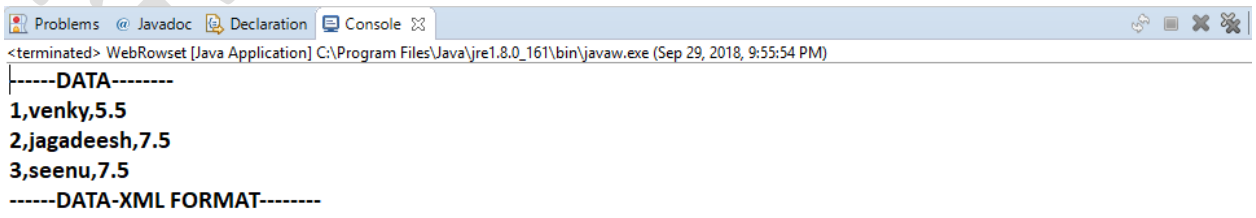
Folder Structure:-



Code:-

```
package com.app;
import javax.sql.rowset.WebRowSet;
import oracle.jdbc.rowset.OracleWebRowSet;

public class WebRowset
{
    public static void main(String[] args) throws Exception
    {
        WebRowSet rs=new OracleWebRowSet();
        rs.setUrl("jdbc:oracle:thin:@localhost:1521:xe");
        rs.setUsername("system");
        rs.setPassword("manager");
        rs.setCommand("select * from student");
        rs.execute();
        System.out.println("------DATA--------");
        while (rs.next()) {
        System.out.println(rs.getInt(1)+","+rs.getString(2)+","+rs.getDouble(3));
        }
        System.out.println("------DATA-XML FORMAT--------");
        //Web Rowset concepts
        rs.writeXml(System.out);
        rs.close();
     }
}
```

Output:-



```
<metadata>
 <column-count>3</column-count>
 <column-definition>
  <column-index>1</column-index>
  <auto-increment>false</auto-increment>
  <case-sensitive>false</case-sensitive>
  <currency>true</currency>
  <nullable>1</nullable>
  <signed>true</signed>
  <searchable>true</searchable>
  <column-display-size>22</column-display-size>
  <column-label>SID</column-label>
  <column-name>SID</column-name>
  <schema-name></schema-name>
  <column-precision>0</column-precision>
  <column-scale>-127</column-scale>
  <table-name></table-name>
  <catalog-name></catalog-name>
  <column-type>2</column-type>
  <column-type-name>NUMBER</column-type-name>
 </column-definition>
```

JoinRowSet:-

=>It can integrate multiple RowSets Data into one final RowSet based on one column link.

RowSet#1+RowSet#2…….= JoinRowSet

=>It will support all concepts supported by WebRowSet also and joining of multiple RowSet into one final RowSet

Simple Code:-

CachedRowSet rs1=…….

CachedRowSet rs2=…….

JoinRowSet jsr=new OracleJoinRowSet();

Jsr.addRowSet(rs1, ”columnName1”);

Jsr.addRowSet(rs2,”columnName2”);

FilteredRowSet (package: javax.sql.rowset)

=>This RowSet internally executes one Boolean condition which returns either true or false after comparing with Data Base table Row.

=>Writing this condition is called as predicate (javax.sql.rowset) implementation. Here predicate means returns true or false.

Step1:-Create predicate implementation class. Also override all 3 methods in eclipse Shortcut: keep cursor at class and press ctrl + l, then those "add unimplemented method" option.
Here: methods

Evaluate(RowSet rs):- It can have conditions over any/all column(s) [any name/any index]
Evaluate (Object value, int Column):- To have condition over Column Index based (like condition on: 1st column or 2nd column…) use this method.
Evaluate (Object value, String columnName):- this method used to write condition based on columnName.
Step#2:- Create FilteredRowSet object and set properties, execute
Step#3:- Link predicate object to RowSet

RowSetListener (Interface):-

Listener:-Listener is a concept which executes a task (method) automatically when it's related method is called. i.e., also called as"on every execution (Task#1 done) call automatically Listener method (Task#2 should also be done)".
=>RowSetListener is a interface given in a package: javax.sql, which has 3 methods rowSetChanged(), rowChanged() and cursorMoved().

=>rowSetChanged() : On RowSet object modified (loaded/closed) this method will be called automatically.
=>rowChanged() : If any row data is modified in RowSet this method will be called.
=>cursorMethod : If we use methods like next() ,previous() ,absolute() then this method will be called.

Transaction Management in JDBC:-

=>Based on database concept Transaction Management is required for only non-select operations [insert,update,delete]. Not required for select SQL.
Transaction Management has provided two special operations those are:

a)commit : Save all changes done so far
b)rollback : Cancel all modifications/changes done so far.

Q)Why Transaction Management Require?

ANS>To get consistency in Database operations Transaction Management required.

i.e, do all (or) do nothing

Example : Consider below Account table given in Database.

Aid	Aname	Bal
101	Ajay	1000
102	Vijay	500

Here we want to transfer money from Ajay account to Vijay account then two Sql queries required.

SQL#1 : To Deduct money from account #1 update account

set bal = bal-300 where aid = 101;

SQL#2 : To Credit money to account #2 update account

set bal = bal+300 where aid = 102;

If SQL#1 is executed and SQL#2 is not executed then consistency between data will not be maintained. So here execute SQL#1 and SQL#2 both “commit” else “rollback” both.

Example:-

Step#1 Disable auto-commit in JDBC

By default Auto Commit is true.

Step#2 write codes in try-catch block

Step#3 At the end of try-catch block write commit() method or some where

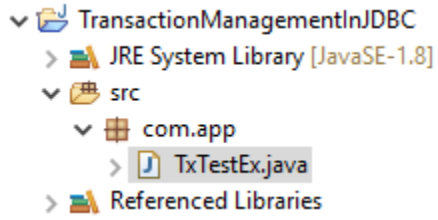
Step#4 In catch block write rollback method

Format:

```
Connection con=……..
con.setAutoCommit(false);
try  {
…….Statements&execute…….
con.commit();
}
catch(SQLException se)  {
con.rollback();
```

Sysout("Problem:"+se);
}
con.close();

Folder Structure:-

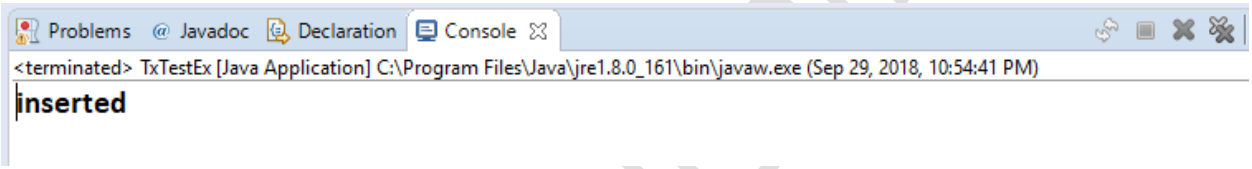


Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class TxTestEx
 {
      public static void main(String[] args) throws Exception
      {
        //jdbc properties
        String driverclass="oracle.jdbc.driver.OracleDriver";
        String url="jdbc:oracle:thin:@localhost:1521:xe";
        String username="system";
        String password="manager";
        //jdbc code
        Class.forName(driverclass);
        Connection con = DriverManager.getConnection(url,username,password);
        //1.disable auto commit
        con.setAutoCommit(false);
        Statement st=con.createStatement();
        try {           //2.execute code in try catch block
```

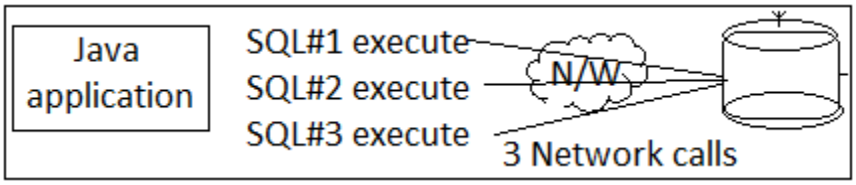
```
            st.executeUpdate("insert into emptab2 values(21,'WER',12.2)");
            //3.call commit()-if all are ok
            con.commit();
            System.out.println("inserted");
        } catch ( SQLException se) {
            //4.if exception-rollback()
            con.rollback();
            System.out.println("problem:"+se);
        }
        con.close();
        }
}
```

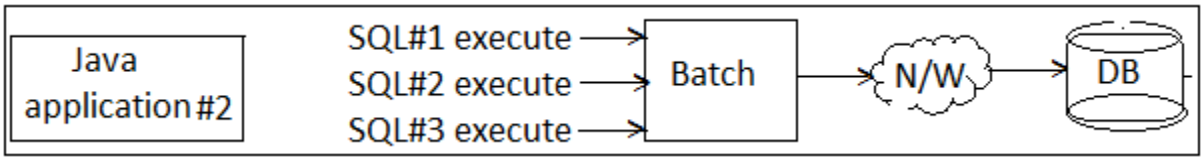
Output:-



BatchProcessing:-

=>This concept is used to execute multiple SQLs at a time, instead of sending them one by one. It saves only execution time by reducing network calls between Java Application and Database.



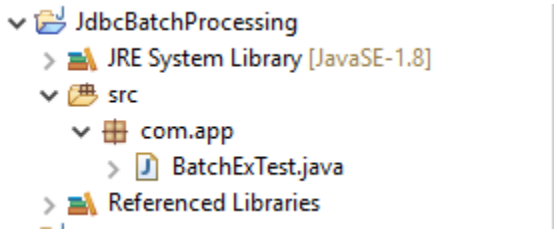


Step#1 Disable auto-commit

Step#2 create statement object

Step#3 use addBatch(SQL) to add multiple SQL queries to statement.
Step#4 call executeBatch():int[ ].Here Array indicates no.of.rows effected counts in order.
Step#5 commit/rollback

FolderSystem:-

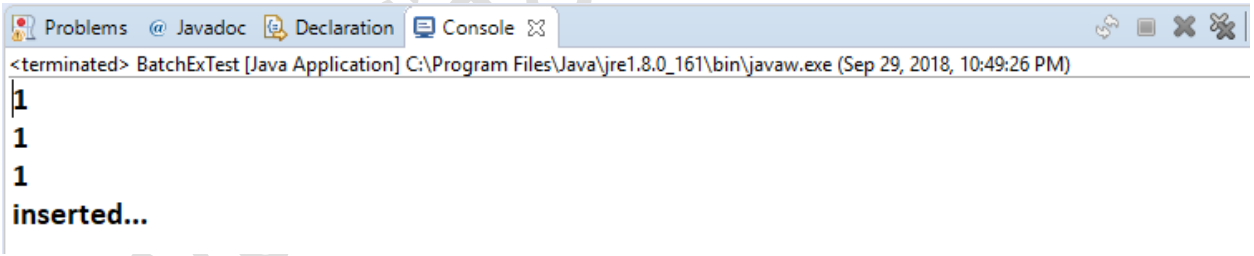


Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class BatchExTest {
    public static void main(String[] args) throws Exception
    {
        //jdbc properties
        String driverclass="oracle.jdbc.driver.OracleDriver";
        String url="jdbc:oracle:thin:@localhost:1521:xe";
        String username="system";
        String password="manager";
        //jdbc code
        Class.forName(driverclass);
        Connection con = DriverManager.getConnection(url,username,password);
        //1.disable auto commit
        con.setAutoCommit(false);
        Statement st=con.createStatement();
        try {   //3.use method addBatch(SQL)
```

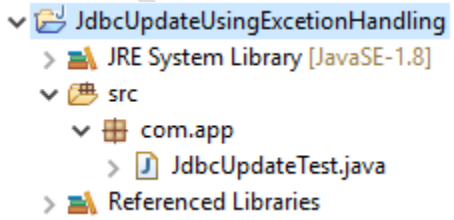
```
            st.addBatch("insert into emptab2 values(21,'WER',12.2)");
            st.addBatch("insert into emptab2 values(22,'BNML',22.2)");
            st.addBatch("insert into emptab2 values(23,'AASD',32.2)");
            int[] count=st.executeBatch();
            System.out.println(count[0]);
            System.out.println(count[1]);
            System.out.println(count[2]);
            //4.no exception then call commit()
            con.commit();
            System.out.println("inserted...");
           }
            catch (SQLException se)
            {
            con.rollback();
            System.out.println("problem:"+se);
            }
        con.close();
        }
}
```

Output:-



JdbcUpdateUsing ExceptionHandling:-

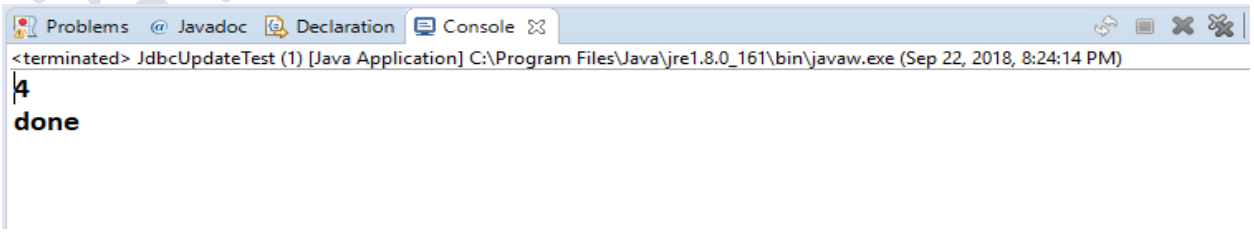
Folder System:--



Code:-

```
package com.app;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class JdbcUpdateTest
 {
        public static void main(String[] args) throws Exception
        {
                try {
                        Class.forName("oracle.jdbc.driver.OracleDriver");
                        Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","manage
r");
                        Statement st=con.createStatement();
                        int count=st.executeUpdate("update student set sname=4");
                        System.out.println(count);
                        con.close();
                        System.out.println("done");
                }catch(SQLException sqle) {
                        System.out.println("JDBC code Problem:"+sqle);
                }
        } }
```

Output:-



JDBC END

SERVLETS

Web Application:-- It is a collection of web pages (Resources) used to provide service to end user.

Ex:-- web applications:-www.bookmyshow, www.paytm.com, redbus, bigbasket, icici, net banking, espncricinfo, youtubes, gmai, facebook, timeindia newspaper, twitter, amazon.

=>web application provides better services, saves time and faster service.

Webpages:-- A webpage can be created using mainly HTML concept.

Types of web Pages:--

a>Static web page:-- A page has fixed content show same to all end users, everytime is known as static page.

Dynamic web page:-- A page has varying (changes) content is known as Dynamic web page.

Types of web Applications:--

a>static web application:-- Web application created using static pages is known as static web application.

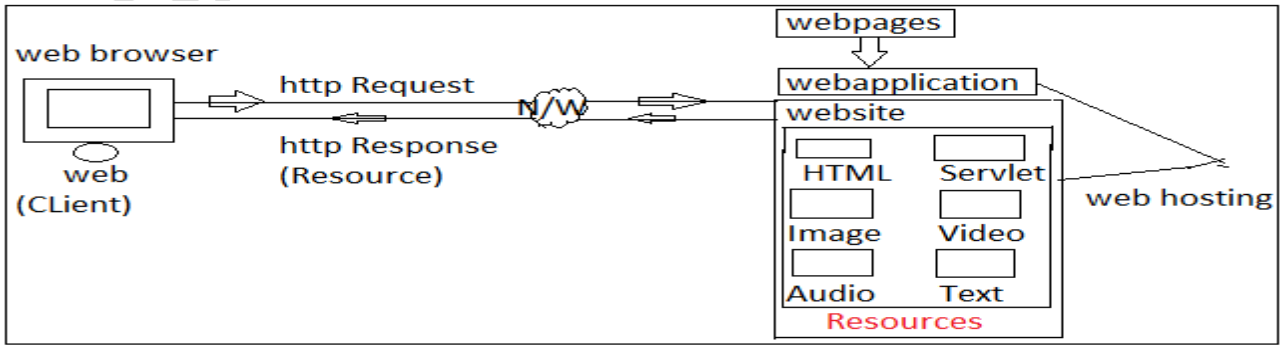
b>Dynamic web application:-- Web application created using dynamic pages (can also have static pages) is known as Dynamic web application.

=>Static web app is created using HTML mainly.

Ex:- Tutorial sites, Wikipedia sites, few blogs …etc.

Dynamic web app is created using programming languages Java (Servlet/JSP) .net (ASP.net), PHP, JSF…etc.

Architecture of web Applications:--



Web Server:-- A System (Computer) which holds web Applications hosted (web sites) and provides to web clients (web browser).

Web client:-- A System (Computer) which is used to read/access web application is using httpRequest (URL) and gets HttpResponse (Resource).

URL:-- Uniform (unique/unified) Resource location (it is a location of one source).

Resource:-- Any file (audio/video/text/image/document/html/servlet/jsp)in web site is known as Resource.

Network:-- A wireless communication between two(or) more computers is known as network.

Web browser:--A software installed on computer used to make HttpRequest(Acceess web sites).
Ex:-- Google chrome, Firefox, Opera etc..

Web Server:-- A System installed on computer to host web sites is known as web servers.
Ex:-- Apache Tomcat, Apache Http, IIS etc..

Web Client:-- A System has web browser Software is called as Web Client.
Ex:--Mobile phone has Google chrome, hap top, desktop etc…

End User:-- A Person who operates client to make request and gets response.
=>web programmer (Web Designer):- A Person who creates static/dynamic web pages.

Web hosting:-- Placing a web application inside web server and starts service.

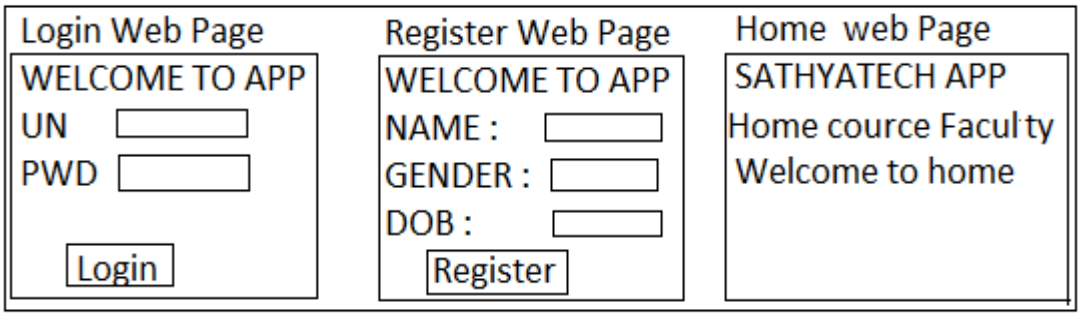
Domain name:-- A Unique name given to website.
Ex:-- https://sathyatech.com/
=>few examples website based on usage fecabook, twitter, LinkedIn-social networking.
=>Amazon, flipkart, snapdel, PayTm—online shopping [E-commerce]

=>PayTm, Freechar, Mobikwik => online websites
=>BookMyShow, TicketData =>Entertainment
=>RedBus, IRCTC =>Transport
=>TimeHindu,DC =>News
=>YouTube, Netflix =>Video service
=>Google, yahoo => Mail & search engine.

Web Pages Example based on application type:--

ex:-- Gmail web Application, It contains different pages like: Login web pages, Register web page, Inbox web page, send Email web page, Draft page web page etc..

Example UI designed for web pages:--



Servlets:-- These are used to develop Dynamic web application also supported static web applications.

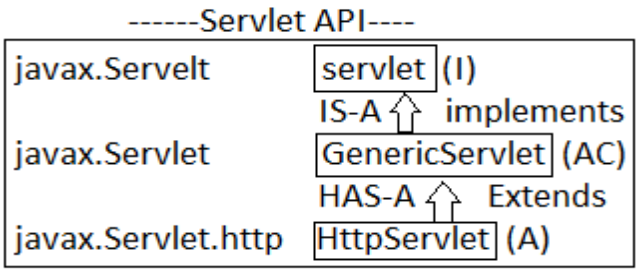
=>This is an API called as Servlet API [having classes and Interface] provided by sun Micro system.

=>This API is provided in 2 Package

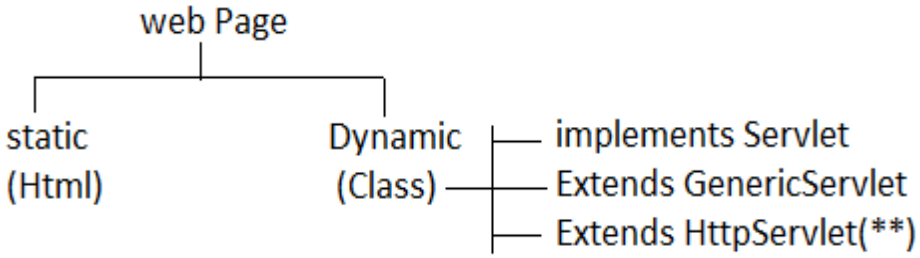
a>javax.Servlet [Package]

b>javax.Servlet.http [Package]

=>Servlet API is provided with basic Interface and Abstract classes along with relations and package.



=>By using above API we can create one class that provide Dynamic Web page as output.



Example:--

1>using Servlet Interface:--

```
class MyServlet implements Servlet
{
}
```

2>Using Generic Servlet:--

```
class MyServlet extends GenericServlet
{
}
```

3>Using HttpServlet abstract class:--

```
class MyServlet extends HttpServlet
{
 }
```

Work flow with Servlets:--

1>Programmer has to create web application by writing Servlet classes code with methods.

2>Every Servlet class must have 3 life cycle methods (A methods called by server in between object creation to destroy process/). Those are init(), service() and destroy().

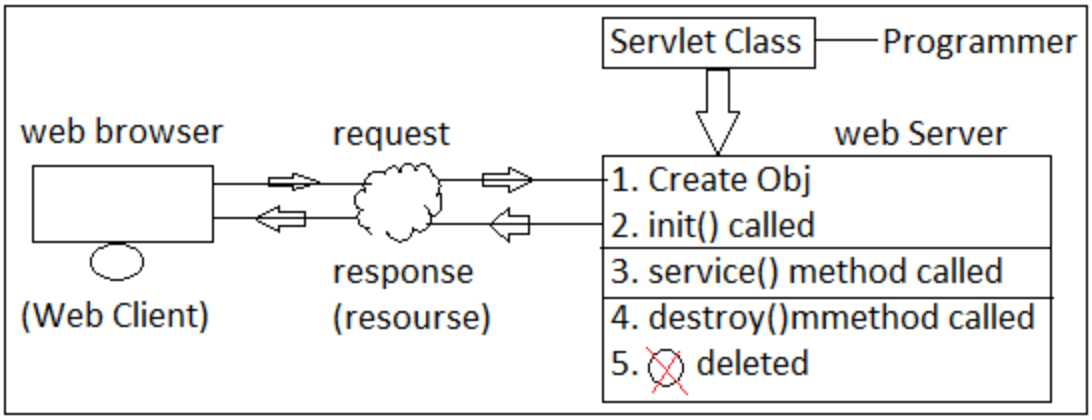
3>web server creates object to Servlet classes and calls 3 methods

i>init():--It will be called only once after creating object by server.

ii>service():-- It will be called for every request made by browser.

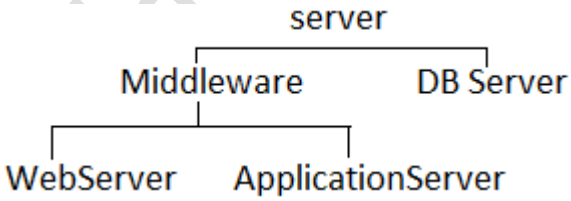
iii>destroy():-- It will be called only once before deleting (destroying) object by server.

=>End user should use web client and make Request get Response back.



Server:-- A component (program) which provides service when request is made.

Types of Server:--



1>Web Server:-- In java, web Servers are use to run Servlets/JSP based web applications(web sites).
=>It supports to run both static and dynamic web application.

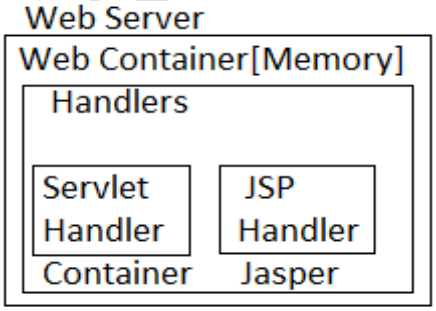
2>Application Server:-- In java, App servers are used to run EJBS [Enterprise Java Bean]. App servers also supports servlets/JSP based application.

3>Database Servers:-- These are used to store and read data from External services. It supports CURD operations. An External source can be end user, Java App, .net App, or Angular) etc…

Web Server Execution Steps:--
1>It will take Servlet class(.class) file as input and creates Object to it using default constructor.
=>If programmer has written no constructor in servlet then Java compiler provides default constructor.
2>After object created successfully then Server calls init() method. Programmer can provide input to this method.[Using web.xml code].
3>For every request made by browser, server call service() method defined in class[Like servletClassObj.service()] It returns output as response to client.
4>Before stopping server destroy() method will be called as server.
5>finally servlet class object is removed [destroyed] from server.

Server Components:--

Web Server Components:--



=>Here web Server is main program which contains web Container works as Sub rogram.

a>Web Container creates and holds Servlet class objects.

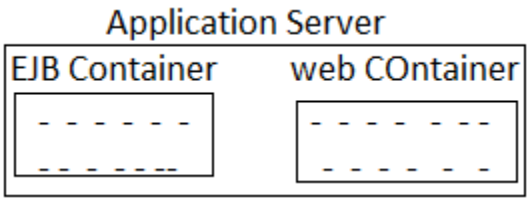
b>To create objects and call methods and to known their status containers uses Handlers [Task Executions].

c>To work on Servlet classes “Servlet Handler” is provided and named as “Container”.

d>To work on JSP [Java Server Pages] “JSP Handler” is provided and named as Jasper.

e>These are also called as Cotelina Engine and Jasper Engine.

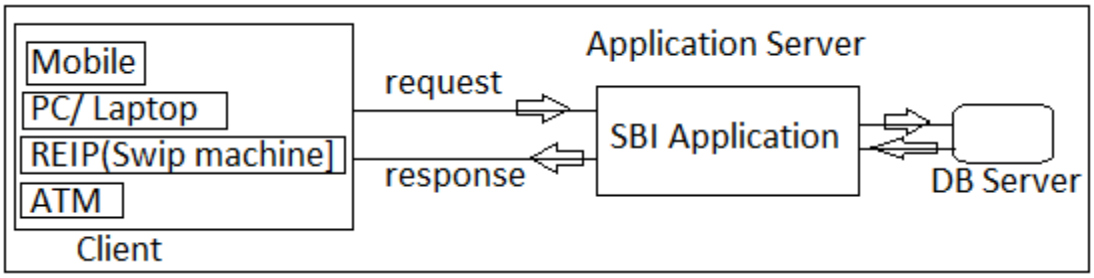
Application Server Components:-- EJB =Enterprise Java Bean



=>EJB are used to develop applications once and used to many times.

=>To create objects to EJBs and to holds those, EJB container is used.

Example EJB is SBI Application:--



Folder System for web applications:--

=>Every web application developed in java must follow folder System rules given by Sun MicroSystem.

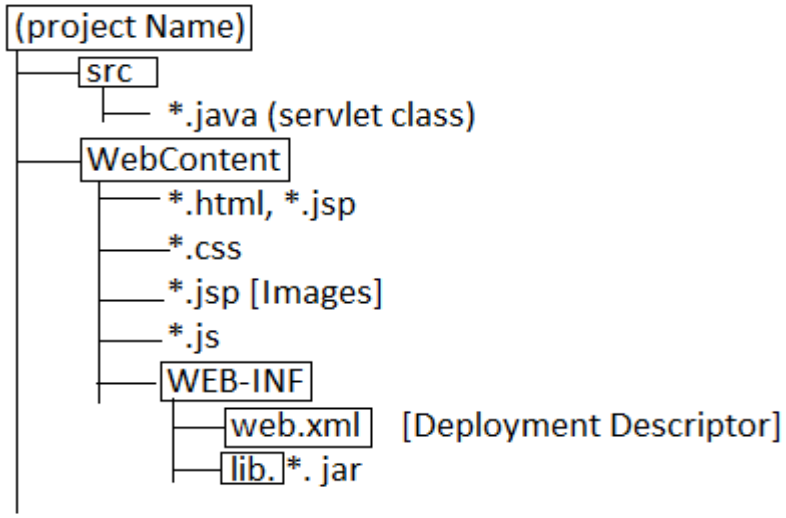
=>It should have two main files

a>Servlet class.

b>web.xml [Deployment Descriptor].

=>All jars must be placed under folder lib[*.jar]
Please do not use build path for adding jars to web application. [It will not work].
=>All static resources must be provided in root folder is “WebContent”[indicates usng /symbol]
Static resource means: HTML, CSS, Script Files, Images etc…

Eclipse web application folder System:--



Setup for Servlets:--
--->JDK 32/64 bit [8/9/10]
--->Tomcat 32/64 [7/8/9]
--->Eclipse Java EE (Photon] 32/64 bit
Step#1:-- Download Tomcat 7 and install it
Step#2:--Open Eclipse/STS and configure Apache Tomcat Server
-->window=>perspective=>open perspective=>other=>choose java EE=>Finish

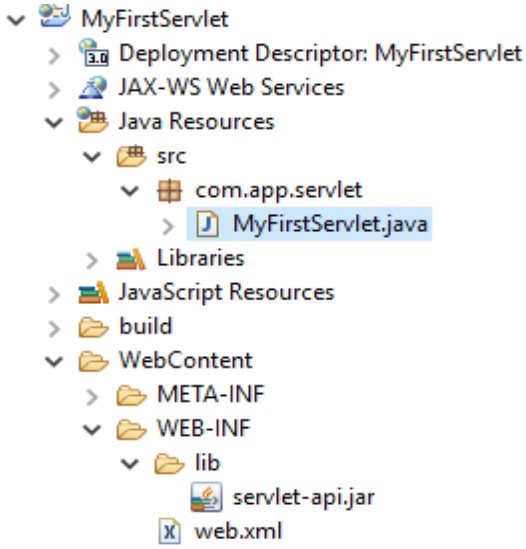
=>Click on server tab=>right click =>new => server => choose Apache Tomcat [any version] =>next => browse => choose Tomcat installation location.
Ex: c:/Program Files/Apache Software foundation/tomcat 9.0
=>next => finish

Step#3:--Create one Dynamic web Project in eclipse
File => new => Dynamic web project
Enter Details like:
Project Name: ServletFirstApp
Target Runtime: Apache Tomcat
Dynamic wb module version 3.1
=>next => next =>choose checkbox.

[v] Genertae web.xml deployment descriptor => finish.

Step#4:-- Copy servlet.api.jar from Tomcat installation folder to project lib folder.
Copy from: c:/Program Files/Apache software Foundation/Tomcat 9.0/lib
Paste in: Project => lib folder under [WEB-INF]

Folder Structure:--



Code:--
Step#1:-- Create one class to generate Dynamic web page
---->right click on src folder => new => class => Enter package and name
Ex:-- package com.app
Name : MyFirstServlet
=>finish

```
package com.app.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class MyFirstServlet extends GenericServlet
{  @Override
public void service(ServletRequest req, ServletResponse res) throws ServletException,
IOException
 {
   //#1 Create Writer Object
  PrintWriter out = res.getWriter();
  //#2 Print Data
  out.println("Hello");
 } }
```

Step#2:-- open web.xml file in source mode
=>double click on web.xml file
=>Click on source mode on button of file.
NOTE:-- Please do not delete top 2 lines in web.xml
Here write code for creating object by web container and line(map) object with URL, so that browser on access.

web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>MyFirstServlet</display-name>
```

```
 <servlet>
  <servlet-name>sample</servlet-name>
  <servlet-class>com.app.servlet.MyFirstServlet</servlet-class>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/welcome</url-pattern>
</servlet-mapping>
</web-app>
```

Step#3:-- Right click on project => Run as => Run on Server
=>Enter URL in browser: http://localhost:3030/MyFirstServlet/welcome
=>To write one Application in servlets we should provide two files.

Those are:--
1>Servlet class 2>web.xml ***
1>Servlet class:-- It is a class which will be written and compiled by programmer.
=>Above example class, wrote in below steps,
a>Write one public class.
b>extends with GenericServlet abstract class.
c>provide public service() method, with ServletRequest and ServletResponse.
d>service method must throw 2 Exceptions.
Those are : ServletException, IOException
e>To display data at UI page(web page) use PrintWriter object (Do not use System.out.println), get this object using code: response.getWriter();
f>To print data on web page code is out.println(….);

Q>What is the different between out.println() and System.out.println()?
a>out.println():-- is used to print data in web page, shown in browser.
b>System.out.println():-- is used to print the data only on console screen. Not on web page.

2>web.xml*:--** It is also called as Deployment Descriptor File.

=>It is input to web server.
=>If servlet class is written then web server will not creates object. If class configured in web.xml then only web container creates object.
=>web.xml root tag is <web-app>
=>For one servlet class, Format is:

```
<servlet>
   <servlet-name>   </servlet-name>
   <servlet-class>    </serrvlet-class>
</servlet>
```

=>meaning of above code is, web container creates object using above class and name.
Ex:--

```
<servlet>
   <servlet-name>one</servlet-name>
   <servlet-class>com.app.MyPage</serrvlet-class>
</servlet>
```

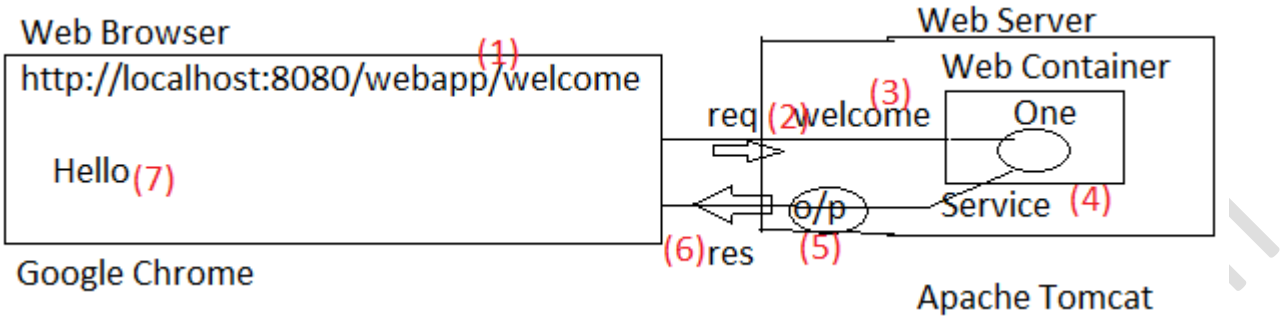
=>Meaning is:-- (by web container)
MyPage one= new MyPage();
=>Above Object cannot be accessed by web browser. Browser can understand only URLs. So, link (map) object with URL, using below format.

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

Example:--

```
<servlet-mapping>
   <servlet-name>one</servlet-name>
  <url-pattern>/welcome</url-pattern>
</servlet-mapping>
```

=>Now end user should enter URL ends with /welcome.
Example:-- http://localhost:8080/FirstApp/welcome in browser



*****Service() method parameters:--**

=>service() is a one of life cycle method defined in Servlet API. It will be called by server when browser makes request(URL).

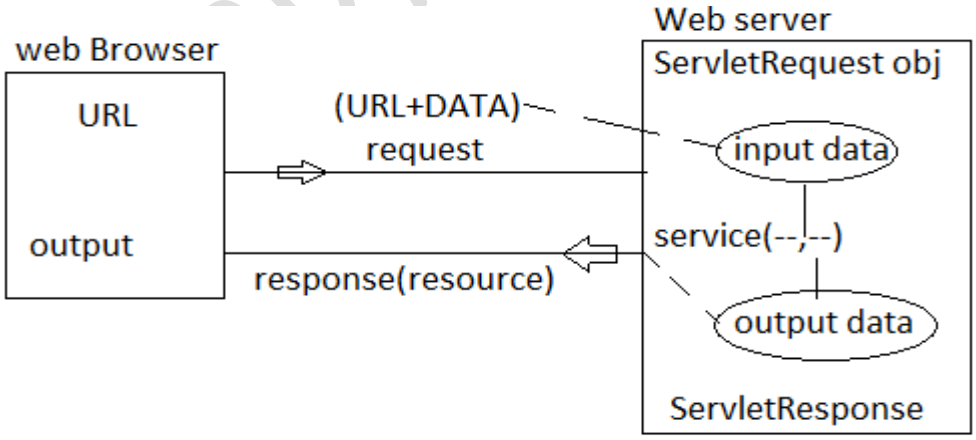
=>Along with request, browser can also send data in any format that will be copied to ServletRequest object, created by WebContainer object also called by Container.

=>service() method reads input data from ServletRequest and final output data will be copied to ServletResponse.

=>When browser makes request, then server creates one ServletRequest object and one ServletResponse Object.

=>Once Response is provided by server to browser, then Server destroy the ServletRequest and ServletResponse objs.

=>*** ServletRequest behaves like input to service() method and ServletResponse behaves like output of service() method.



Sending Data using URL request:--

a>Query Parameters:--

=>Here Parameters means input data to any method. [Here service() method]

=>Query Parameters means sending data along with Query (URL) to service() method. Format will be

URL?Key=Value&Key=Value…

=>Here data is send in key= value format.

=>Both key and value dataTypes are String only. After reading data, if required parse them into any DataType (ex:--Integer.parseInt….)

=>Symbols are ? and &. Form ? Symbol onwards it is special called as Query String.

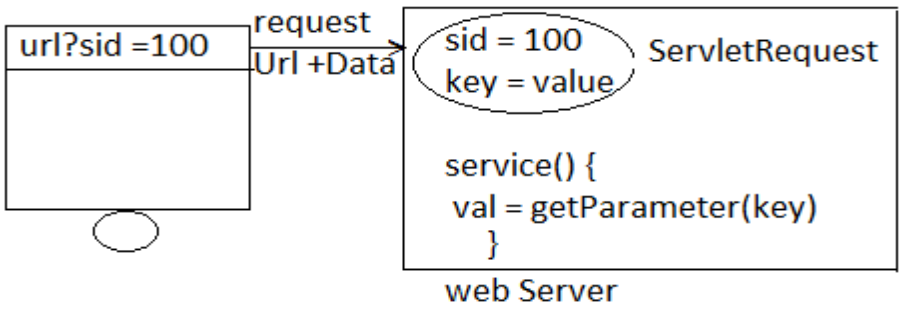
Ex:-- http://localhost:8080/App/welcome?sid=10&sname=AA&sfee=3.2

=>This data will be copied to ServletRequest object in web server.

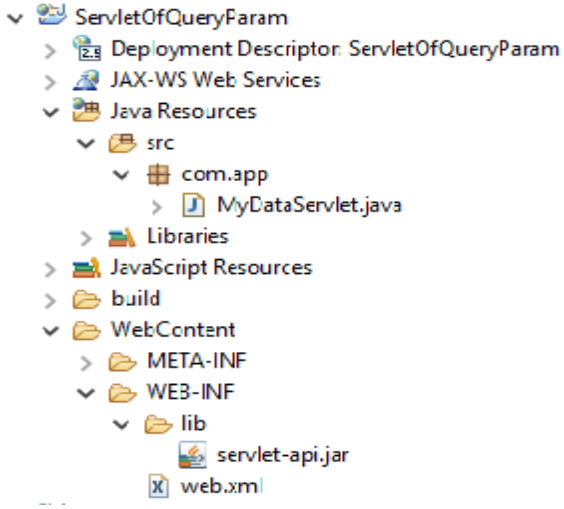
=>Here Query String is: ? sid = 10 & sname = Neha &sname = AAAA & sfee= 3.2

***To read this data in service() method use below format:

String value = request.getParameter("key");



Folder Structure:--



Servlet class code:--

com.app.servlet

```
public class MyDataServlet  extends GenericServlet
{
 public void service(ServletRequest req, ServletResponse res)throws ServletException,
IOException
 {       //reading Query Parameters
         String sid= req.getParameter("sid");
         //convert data Type from string to int
         int id = Integer.parseInt(sid);
         String sname = req.getParameter("sname");
         String fee = req.getParameter("sfee");

         //output data
         PrintWriter out =res.getWriter();
         out.println("Id is : "+id);
         out.println("/");
         out.println("Name is : "+sname);
         out.println("Fee is : "+fee);
         }       }
```

=>Run on server Enter URL like:

http://localhost:3030/ServletFirstApp/welcome?sid=10&sname=Uday&sfee=34.78

web.xml Code:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://java.sun.com/xml/ns/javaee"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 id="WebApp_ID" version="2.5">
 <display-name>ServletFirstApp</display-name>
 <servlet>
        <servlet-name>two</servlet-name>
        <servlet-class>com.app.MyDataServlet</servlet-class>
 </servlet>
 <servlet-mapping>
        <servlet-name>two</servlet-name>
        <url-pattern>/welcome</url-pattern>
 </servlet-mapping>
</web-app>
```

b>Sending Data using HTML Form:--

=>HTML form is mostly used concept in real-time Application. Few Examples are: Gmail/FB Registration Form. LoginForm FeedBack Form, Enquery Form, Commponent Form. etc…

=>To read form data in servlet use code request.getParameter(“key”) return String type value for all types of inputs (Text, TextArea, Password, Radio button, DropDown….).

=>For checkbox, it can have multiple values. So, to read checkbox values use code: request.getParameterValues(“key”) return String[] (String array).

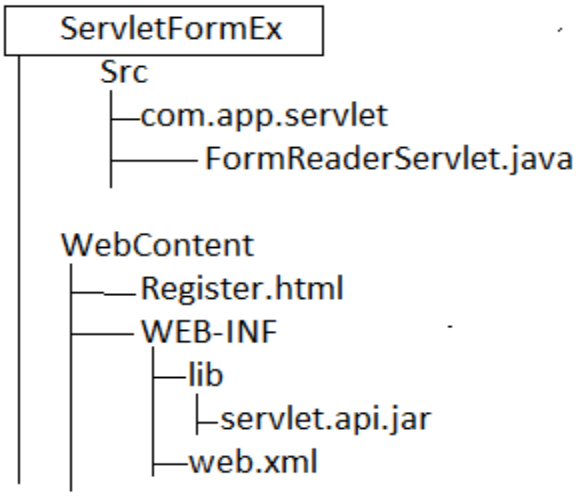
=>If we print String[] un-readable output is provided, so convert to List type and print using code:

Arrays.asList(arrayData) returns List.

=>Arrays is from java.util package and asList() is a static method.

=>On server startup to show one default page, without entering URL part, use <welcome-file> concept. Server only calls this page.

Folder System:--



Code:-- 1>Register.html:--

=>Right click on WebContent folder => new HTML File => Enter name => Finish'

```
<!DOCTYPE html>
<html><head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head><body>
<h2>Welcome to Register Page</h2>
<form action="insert" method="post">
<pre>
Name    : <input type="text" name="ename"/><br>
Email   : <input type="text" name="email"/>

Course  :<select name="course">
<option>-Select-</option>
<option>Core Java</option>
<option>Adv Java</option>
<option>Spring</option>
<option>Oracle</option>
</select>
```

```
Gender : <input type="radio" name="gender" value="Male">Male
<input type="radio" name="gender" value="Female">Female

Address : <textarea name="addr"></textarea>
Language:<input type="checkbox" name="languages">
<input type="checkbox" name="lang" value="ENG">ENG
<input type="checkbox" name="lang" value="HIN">HIN
<input type="checkbox" name="lang" value="TEL">TEL
<input type="checkbox" name="lang" value="KAN">KAN
<input type="submit" value="Register"/>
</pre></form></body></html>
```

2>Servlet code:--

```
package com.app.servlet;
public class FormReaderServlet  extends GenericServlet
 {
  public void service(ServletRequest req, ServletResponse res)throws ServletException,
IOException
{
        //read form input
        String ename = req.getParameter("ename");
        String email = req.getParameter("email");
        String course = req.getParameter("course");
        String gender = req.getParameter("gender");
        String addr = req.getParameter("addr");

        //convert array data to List Type
        List<String> languages = Arrays.asList("lang");
        //display data
        PrintWriter out =res.getWriter();
        out.print("<html> <body>");
        out.print("<h2><Your Data<h2>");
        out.print("<table border= 1> </tr>");
```

```
        out.print("<td>NAME</td><td>"+ename+"</td>");
        out.print("</tr><tr>");

        out.print("<td>EMAIL</td><td>"+email+"</td>");
        out.print("</tr><tr>");

        out.print("<td>COURSE</td><td>"+course+"</td>");
        out.print("</tr><tr>");

        out.print("<td>GENDER</td><td>"+gender+"</td>");
        out.print("</tr><tr>");

        out.print("<td>ADDRESS</td><td>"+addr+"</td>");
        out.print("</tr><tr>");

        out.print("<td>LANGS</td><td>"+languages+"</td>");
        out.print("</tr></table></body></html>");
 }
}
```

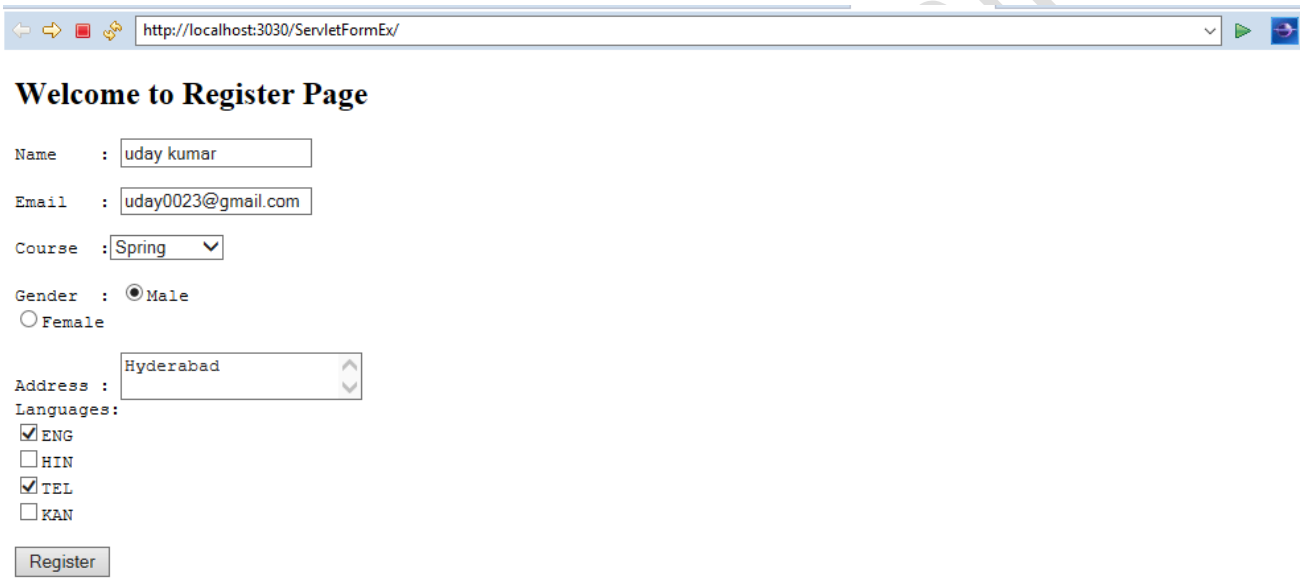
=>Run on server Enter URL like: http://localhost:3030/ServletFormEx/

3>web.xml file:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
  <display-name>ServletFormEx</display-name>
  <servlet>
    <servlet-name>FormReader</servlet-name>
 <servlet-class>com.app.servlet.FormReaderServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>FormReader</servlet-name>
```

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

<!-- To Show full Url Direct in Address bar after execution of project -->
  <welcome-file-list>
  <welcome-file>Register.html</welcome-file>
  </welcome-file-list>
</web-app>
```

Output:--



Working with Life cycle methods:--

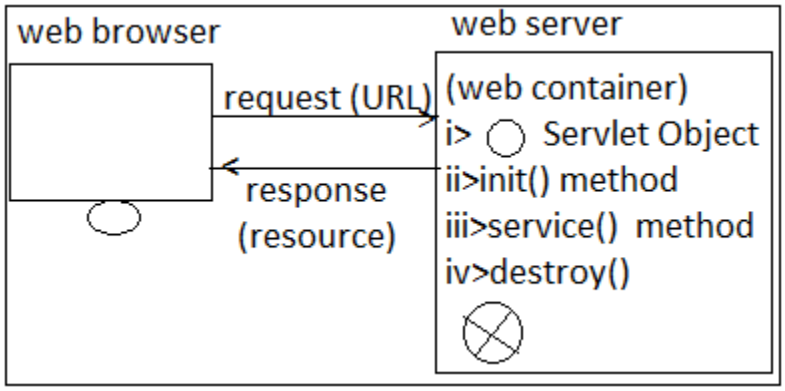
Servlet API has provided 3 abstract methods in javax.servlet Servlet (I) Interface. Those are simple called as init(), service() and destroy() known as life cycle methods.

Format of those methods:--

1>init() method:-- Public void init(ServletConfig config) throws ServletException.

2>service():-- public void service(ServletRequest req, ServletResponse res) throws ServletException IOException

3>destroy():-- public void destroy();



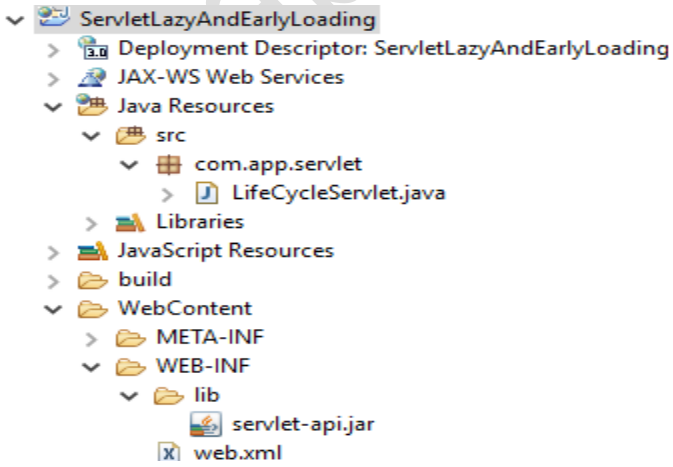
Q>What is the Lazy loading (or) when servlet object is created in server?

Ans>On First request made by web client, web server creates objects to servlet class and calls init() method at same time. This is also called as Lazy loading. So, First request will always slow (takes time to create object and call init() then service() method). 2nd Request onwards only service() will be called.

Eager Loading/Early loading:--

On server startup only [before making first request], server creates object to servlet class and calls init() method also at same time. This is known as Early/Eager loading of Servlet.
=>For this Programmer should provide <load-on-startup> tag with +ve number. Default value for
<load-on-startup> is "-1" (indicates lazy loading).

Folder Structure:--



Code:--

1>Servlet class:--

```
package com.app.servlet;
import java.io.IOException;
import javax.servlet.GenericServlet;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class LifeCycleServlet extends GenericServlet
{
 public void init(ServletConfig config) throws ServletException
        {
         System.out.println("Im from init method");
 }
 public void service(ServletRequest req, ServletResponse res) throws    ServletException,
IOException  {
         System.out.println("Im from service method");
 }
  public void destroy()
         {
                System.out.println("Im from destroy method");
  }
}
```

=>Run on server Enter URL like: http://localhost:3033/ServletLazyAndEarlyLoading/cycle

2>web.xml code:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
```

```
 <display-name>ServletRedirectEx</display-name>
  <servlet>
  <servlet-name>lifecycle</servlet-name>
  <servlet-class>com.app.servlet.LifeCycleServlet</servlet-class>
  <load-on-startup>10</load-on-startup>
 </servlet>
 <servlet-mapping>
  <servlet-name>lifecycle</servlet-name>
  <url-pattern>/cycle</url-pattern>
 </servlet-mapping>
</web-app>
```

=>Right click on Project => Run As => Run on Server
=>In console, see init() method output. (Executed before making first request)
=>Now goto browser Enter URL: http://localhost:3030/ServletLazyAndEarlyLoading/cycle
=>Now goto console see service() method output.
=>In case of multiple servlets needs to be configure <load-on-startup>then we can provide same number (for any order [random] object creation).
=>To maintain chain [one after another] Provide numbers in order

Consider below example:--

Servlets	Load-on-startup	object creation order
A	12	Iii
B	8	I
C	14	Iv
D	9	ii

Servlets Supportive Object Memories:--

As Like instance and static variables in class, in same way servlet has 2 special memories to store in key=value format.

1>ServletConfig:-- It is created for every Servlet class, one config object cannot be accessed by other servlet obj. [we can compare like instance variable].

2>ServletContext:-- It is a common memory created for all Servlets objects.

Config	Context
1>Ratio is n:n	1>Ratio is 1 : n
2>When servlet obj is created, then config is created	2>Created on server startup
3>If servlet is destroyed then config also destroyed	3>destroyed when server is stopped.

NOTE:--

=>Both are temporary Memories.

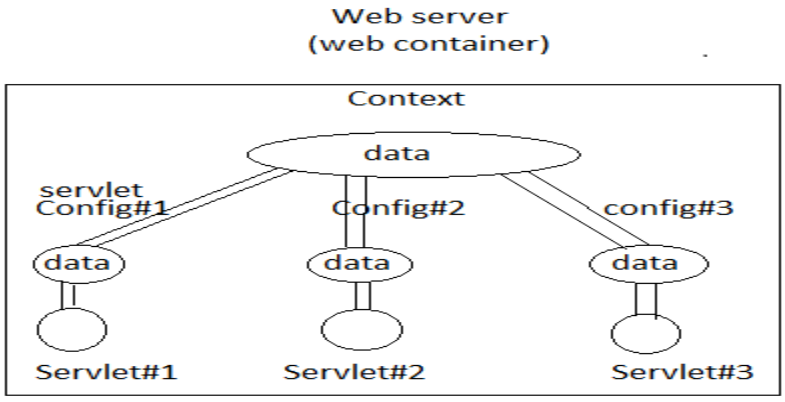
=>Both holds data in key =value format.

=>By default key, value is string types.

=>By using ServletConfig we can get ServletContext memory link (because for all config only one context).

=>But using ServletContext we cannot get ServletConfig memory (there are multiple config memories exist).

=>Config memory exist if servlet object is exist, context memory exist even server has zero servlet objects.



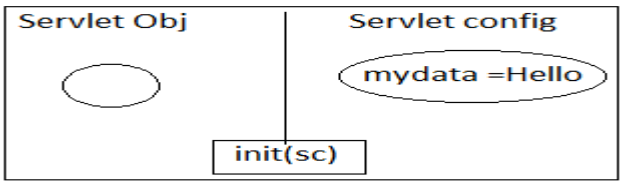
ServletConfig:--

=>It is a memory holds data in key=value format, It is used to pass input data to init-method mainly.

=>in web.xml file under <servlet> tag write code like,

```
  <init-param>
     <param-name>mydata</param-name>
     <param-value>Hello</param-value>
</init-param>
```

=>Then server created objects with data like.



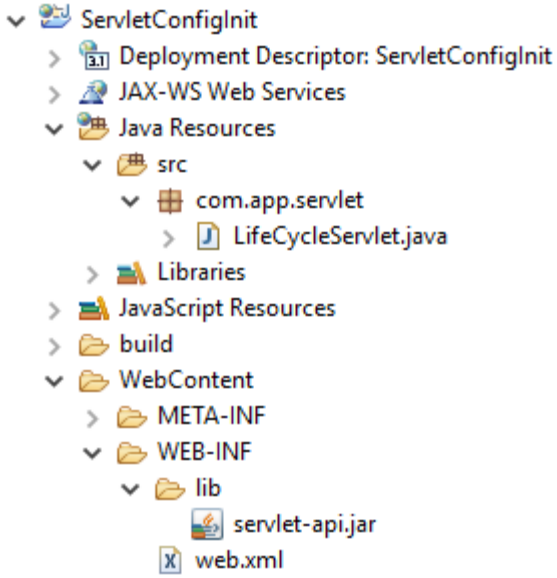
=>To read data in init() method code is:--

String val = config.getInitParameter("key");

Here <param-name> behaves like key

<param-class> behaves like value

Folder Structure:--



Code:--

#1>Servlet class:--

```
package com.app.servlet;
import java.io.IOException;
import javax.servlet.GenericServlet;
import javax.servlet.ServletConfig;
```

```
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class LifeCycleServlet extends GenericServlet
{
 public void init(ServletConfig config)throws ServletException
 {
 String val =config.getInitParameter("Mydata");
 System.out.println("From init method call : " +val);
   }
 public void service(ServletRequest req, ServletResponse res) throws ServletException,
IOException
 {
        System.out.println("From service method");
 }
}
```

=>Run on server Enter URL like: http://localhost:8080/ServletEx/cycle

2>web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
 <display-name>ServletConfigInit</display-name>
 <servlet>
  <servlet-name>Life</servlet-name>
  <servlet-class>com.app.servlet.LifeCycleServlet</servlet-class>
    <init-param>
    <param-name>mydata</param-name>
    <param-value>Hello</param-value>
  </init-param>
 </servlet>
```

```
<servlet-mapping>
  <servlet-name>Life</servlet-name>
  <url-pattern>/cycle</url-pattern>
</servlet-mapping>
</web-app>
```

ServletContext:-- It is also called as global memory for all Servlets.

=>It will store data in key = value format both are type String only.

=>To get ServletContext object code is:

--->In init() method :using config

```
ServletContext sc = config.getServletContext();
```

=>in service() method : using request

```
ServletContext sc =req.getServletContext();
```

****)To store data globally using web.xml

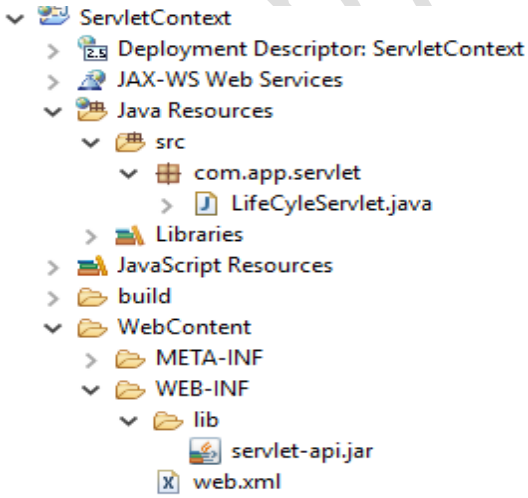
Syntax is:--

```
<contex-param>
   <param-name>key</param-name>
  <param-value>val</param-value>
</context-param>
```

***)To read this, syntax is:--

```
String val=sc.getInitParammeter("key");
```

Folder Structure:--



Example code:--

1>Servlet code:--

```
package com.app.servlet;
public class LifeCyleServlet extends GenericServlet
{
 public void init(ServletConfig config)throws ServletException    {
        //Way#1 to read Context param
        ServletContext sc =config.getServletContext();
        String val = sc.getInitParameter("mydata");
        System.out.println("from init"+val);
 }
 public void service(ServletRequest req, ServletResponse res)throws ServletException,
IOException
 {
        //way#2 to read Context param
        ServletContext sc = req.getServletContext();
        String val =sc.getInitParameter("mydata");
        System.out.println("from service :" +val);
 }
}
```

=>Run on server Enter URL like: http://localhost:2018/ServletInitParam/cycle

2>Web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
id="WebApp_ID" version="2.5">
  <display-name>ServletFirstAppOfQueryParam</display-name>
  <context-param>
      <param-name>mydata</param-name>
      <param-value>Hello Data</param-value>
```

```
 </context-param>
<servlet>
   <servlet-name>Life</servlet-name>
   <servlet-class>com.app.servlet. LifeCyleServlet</servlet-class>
 </servlet>
 <servlet-mapping>
   <servlet-name>Life</servlet-name>
   <url-pattern>/cycle</url-pattern>
 </servlet-mapping>
</web-app>
```

Q>What is default servlet? In Application?.

=>A servlet which gets

=>A servlet which is mapped to url-pattern "/"

Ex:-- public class MyServlet Extends GenericServlet { }

Web.xml:--

```
<servlet>…</servlet>
<servlet-mapping>
…..
<url-pattern>/</url-pattern>
</servlet-mapping>
```

=>When server is started then MyServlet service()method is automatically called by server.

***There can be only one Default servlet in one Application. Here rule is no two servlets should have same URL pattern.

	Servlet	**URL Pattern**	**?**
1	MyData	/	Valid
2	InfoServlet	/Info	Valid
3	GetData	/info	Invalid
4	ResultServ	/	Invalid
	GetModel	/info	Valid

=>URL is case-sensetive (/info, /INFO, /Info are different)

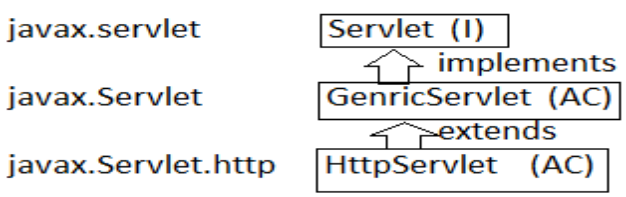
Protocals:--Set of rules (guidelines) prepared and followed between two systems for task execution.

Few Examples are:--

1>TCP/IP ==>Transmission Control Protocol / Internet Protocol
2>ARP ==>Address Resolution Protocal)
3>DHCP ==>Dynamic Host Configuration Protocal
4>FTP ==>File Transfer Protocal
5>HTTP ==>Hyper Text Transfer Protocal
6>ICMP ==>Internal Group Management Protocal
7>IGMP ==>Internate Group Management
8>IMAP4 ==>Internet MessageAccess Protocal version 4
9>NTP ==>Network Time Protocal
10>POP3 ==>Post office protocol version 3
11>RTP ==>Real-time Transfer Protocal
12>SMTP ==Simple Mail Transfer Protocal
13>TFTP ==>Trivial File Transfer Protocal
14>UDT ==>User Datagramprotocal

HttpServlet:--

=>GenericServlet (Abstract class) provided in javax.servlet is protocol independent i.e. works for any type of protocol implement i.e. works for any type of protocol.
Ex:-- HTTP, FTP, SMTP….
=>But in real-time mostly used Protocol is HTTP (and HTTPS), to handle this protocol specially with all its features Specially with all its features Sun Microsystem has provided HttpServlet (Abstract class) in package javax.servlet.http.
=>Here GenericServlet is having only one abstract mthod “service()” and HttpServlet is having zero abstract methods.
=>HttpServlet extends GenericServlet internally.



Q>Why HttpServlet (or any other) is declared as abstract even it is not having abstract methods?

Ans>HttpServlet has all method with no Customer specific logic, all method having simple/default logic which provide no service. So, it is not recomanded to create object this, finally declared as abstract.

=>GenericServlet (Abstract class) is having one abstract method, also called as life cycle method.

service(ServletRequest, ServletResponse) throws ServletException, IOException [Life Cycle method]

=>HttpServlet(Abstract class) is having zero abstract methods. Also service() method with http

Parameters are given as:

Service(HttpServletRequest, HttpServletResponse) throws ServletException, IOException ---->[No Lify cycle method]

=>HttpServlet has 7 HTTP methods, follows Syntax

doXXX(HttpServletRequest, HttpServletResponse)

Those are:-- doGet(GET), doPost(POST), doPut(PUT), doDelete(DELETE), doGHead(HEAD), doTrace(TRACE), doOptions(OPTIONS).

=>Here browser supports only GET and POST operations.

=>For every task one doXXX() method is used. For example every doXXX method is compared with basic operations, then

1>doGet()==>Get the data from Server.

2>doPort()==>Create (new) data in server

3>doPut()==>modify (alter) data in server

4>doDelete()==>remove data in server

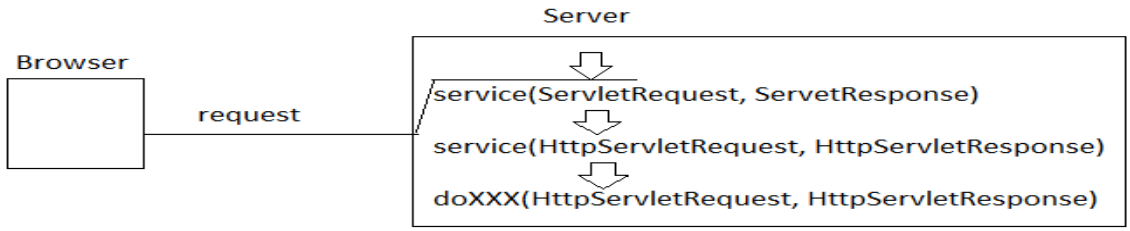
5>doHead()==>Execute work in server and return “No response Body”

6>doTrace()==>Find location of data in server.

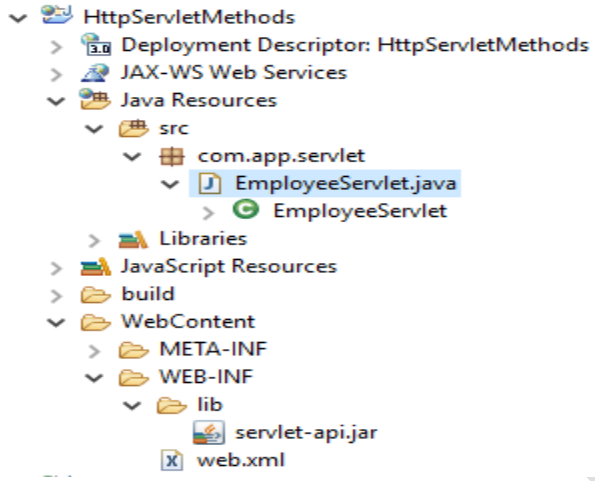
7>doOptions()==>Get multiple options of data in server.

=>When browser made request, then execution flow at server side is

---



Folder Structure:--



Example Code:--

1>Create Servlet class under src folder [Which re-direct application after some time (sec)]

```
package com.app.servlet;
public class EmployeeServelet extends HttpServlet
{
public void service(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException
{
PrintWriter out = res.getWriter();
res.addHeader("Refresh", "12; http://www.google.com/");
out.print(new java.util.Date());
}
}
```

=>Run on server Enter URL like: http://localhost:3030/HttpServletMethods/message

2>web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>HttpServletMethods</display-name>
 <servlet>
  <servlet-name>Sample</servlet-name>
  <servlet-class>com.app.servlet.EmployeeServlet</servlet-class>
  </servlet>
 <servlet-mapping>
  <servlet-name>Life</servlet-name>
  <url-pattern>/message</url-pattern>
 </servlet-mapping>
 <!-- To Show full Url Direct in Address bar after execution of project -->
</web-app>
```

URL format:--

Protocal:--//IP:PORT/ContextPath/ServletUrlPattern

URI: -- /ContextPath/ServletUrlPattern

=>URL is used to identify one unique location of a resource (file/data).

(Uniform Resource Locator)

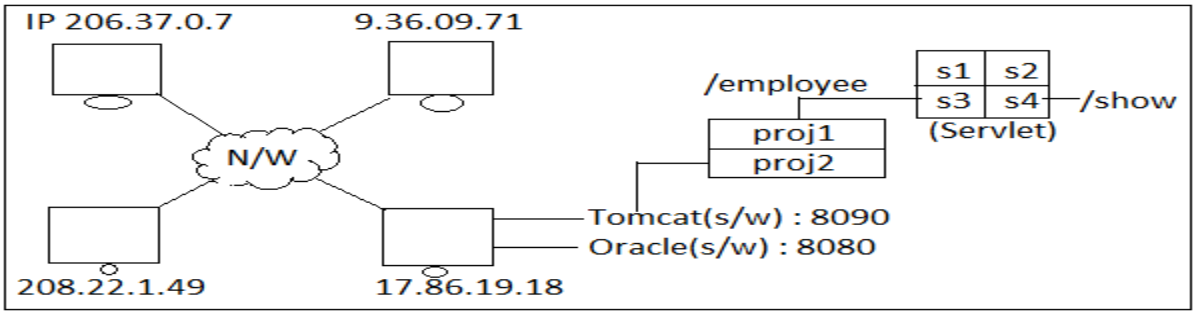
=>URI is a combination of ProjectName (ContextPath) and other part of URL patterns of servlets.

=>IP : Unique number given to one system (Computer) in a network.

PORT:-- A unique number given to every service(software) in a System, assigned by OS, Starts from zero (0) upto 65535.

=>Reserved port numbers are: 0 - 1024.

Example:--



Ex:-- http://17.86.19.18:8090/employee/show?sid=10

Here http ==>Protocal

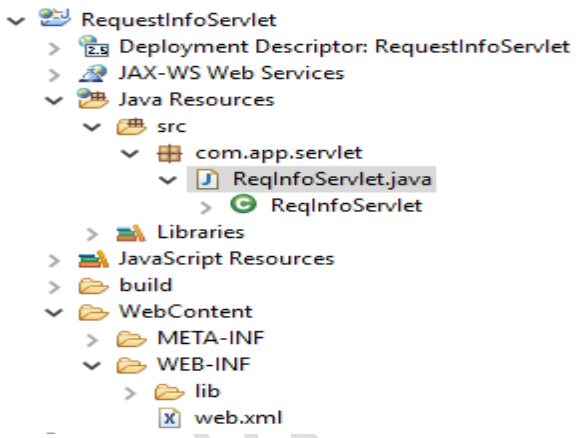
8090 ==>port number

Employee==>ContextPath (ProjectName)

?sid=10==>Query String

=>These details, we can get in HttpServlet using "HttpServletRequest" (I).

Folder Structure:--



Code:--

1>Servlet class:--

```
package com.app.servlet;

public class RegInfoServlet extends HttpServlet {
public void service(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
{
```

```
PrintWriter out = res.getWriter();
StringBuffer  url =req.getRequestURL();
String uri =req.getRequestURI();
String qs = req.getQueryString();
String projName = req.getContextPath();

out.println("URL is = " +url.toString());
out.println("URI is = " +uri);

out.println("Query String is = " +qs);
out.println("ContextPath is = " +projName);
} }
```

=>Run on server Enter URL like: http://localhost:3030/RequestInfoServlet/message

2>Web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">
 <display-name>RequestInfoServlet8</display-name>
 <servlet>
  <servlet-name>Reg</servlet-name>
  <servlet-class>com.app.servlet.ReqInfoServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>Reg</servlet-name>
  <url-pattern>/message</url-pattern>
 </servlet-mapping>
</web-app>
```

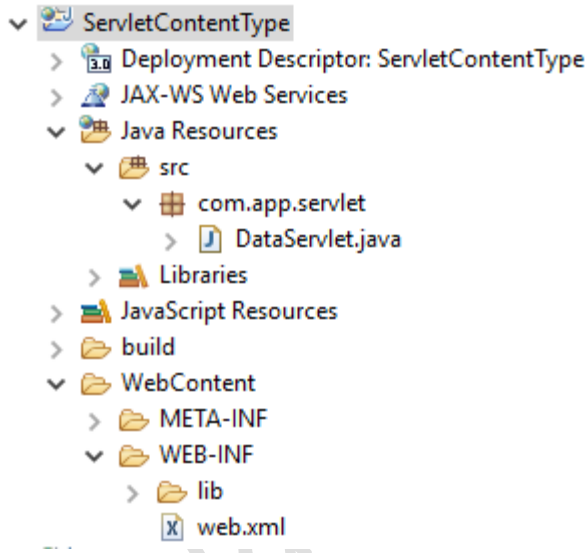
ContentType:-- Here Content means data this option is used to provide "What type of data is provided by a servlet as a Response".

=>In simple “Response has what data?” Text Data?, Image Data?, PDF Data?, Excel Data?, Audio Data?, etc…
=>Every Response should better have Content Type. It follows format: “type/subtype”

Few examples are:--
text/thml, text/xml, image/jpg, imagepng, image/gif, application/pdf (PDF), application/vnd.ms-excel (EXCEL), audio/mp3, audio/mpeg, audio/wav, video/webm, video/mp4, application/javascript, application/ etc…

Basic Example: contentType--
Use code response.setContentType(“---“) to set the type.

Folder Structure:--



1>Servlet class:--

```
package com.app.servlet;

public class DataServlet extends HttpServlet
 {
public void service(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
 {
```

```
res.setContentType("text/html");
PrintWriter out = res.getWriter();
out.println("<html><body>");
out.println("<h1>Welcome All </h1>");
out.println("</body> </html>");
}
}
```

=>Run on server Enter URL like: http://localhost:3033/ServletContentType2/content

2.>web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>ServletContentType</display-name>
 <servlet>
  <servlet-name>DataServlet</servlet-name>
  <servlet-class>com.app.servlet.DataServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>DataServlet</servlet-name>
  <url-pattern>/content</url-pattern>
 </servlet-mapping>
</web-app>
```

Generating PDF Output using Servlets:--

=>Use any 3rd party API to design PDF data and provide servlets code.

response.setContentType("application/pdf")

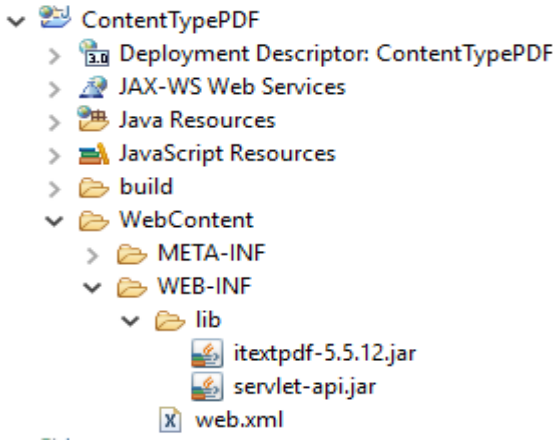
=>So that final output is shown as PDF data

**>Here 3rd party API is : iText-5.x Which is used to design and code for PDF Document.

=>Add this Jar in lib folder, then only coding is possible.

=>**http://central.maven.org/maven2/com/itextpdf/itextpdf/5.5.4/itextpdf-5.5.4.jar**

Folder Structure:--



Code:--

1>Servlet class:--

```
package com.app.servlet;
public class PdfDataServlet extends HttpServlet
{
 public void service(HttpServletRequest req, HttpServletResponse res)throws
ServletException, IOException
{
 res.setContentType("application/pdf");
 res.addHeader("Content-Disposition","attachment;filename=mydata.pdf");
        try  {
        //1. Create one Document
        Document doc = new Document();
        //2. Get Writer over Doc
        PdfWriter.getInstance(doc, res.getOutputStream());
        //3. Open Doc
        doc.open();
        //4. Write data
        doc.add(new Paragraph("Hello Pdf"));

        //table with 2 columns in a row
        PdfPTable t = new PdfPTable(2);
        t.addCell("S1No");   t.addCell("Name");
```

```
            t.addCell("101");    t.addCell("Ajay");
            t.addCell("102");    t.addCell("Vijay");
            doc.add(t);
            doc.add(new Paragraph(new Date().toString()));
            //5. Close doc
            doc.close();
            }
            catch (Exception e)
            {
            System.out.println(e);
            }
        }
}
```

=>Run on server Enter URL like: http://localhost:3033/ContentTypePDF/pdf

2>web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>ContentTypePDF</display-name>
 <servlet>
  <servlet-name>DataServlet</servlet-name>
  <servlet-class>com.app.servlet.PdfDataServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>DataServlet</servlet-name>
  <url-pattern>/pdf</url-pattern>
 </servlet-mapping>
</web-app>
```

Excel Output Generation Using Servlets:--

=>To Generate Excel Data output use

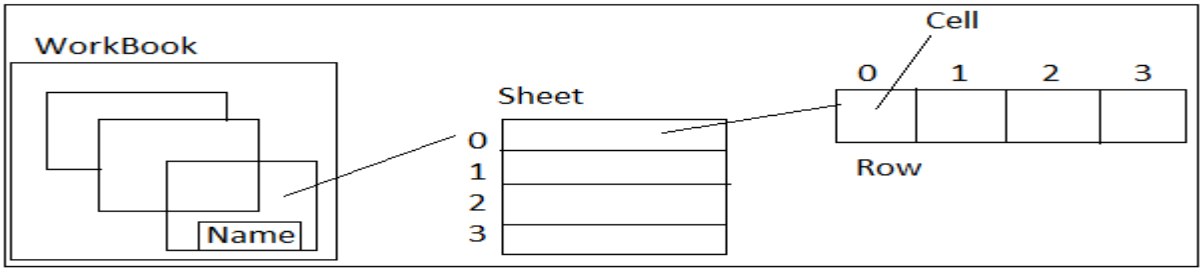
Content Type: application/vnd.ms-excel

=>Every Excel File is a Book (WorkBook).

It contains multiple Sheets (Pages). Every Sheet identified by using one name.

Ex:- Sheet1, Sheet2…(We can rename this)

=>Sheet contains Rows. Row number starts from zero (0). Every row contains Cells (one Box = one cell). Cell number starts from zero.



=>Apache POI is a 3Rd party API used to generate Excel Output using “HSSF API” classes are given as:

=> jars download Links:

http://www-eu.apache.org/dist/poi/release/bin/poi-bin-4.0.0-20180907.zip

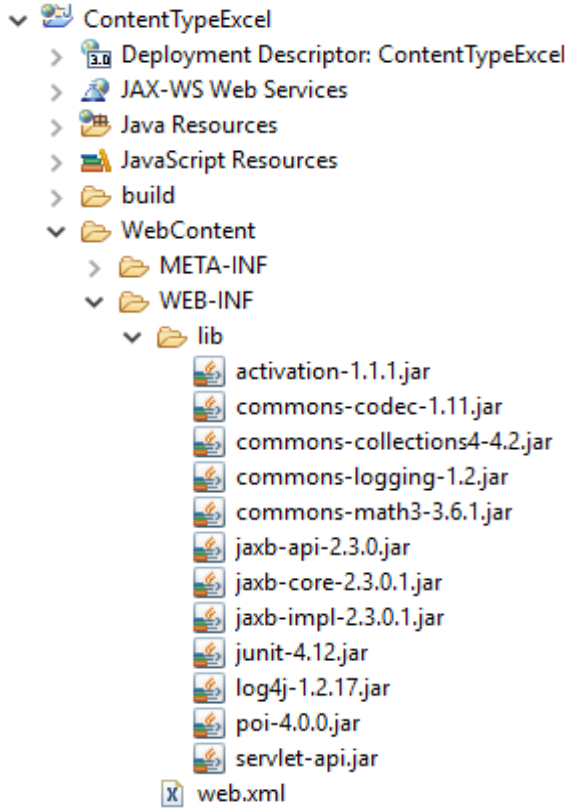
TYPE	CLASS	Const/Method
WorkBook	HSSFWorkBook	HSSFWorkBook()
Sheet	HSSFSheet	createSheet()
Row	HSSFRow	createRow()
Cell	HSSFCell	createCell()

=>After creating WorkBook with data write it to servelet output-stream.

Use method: response.getOutPutStream();

=>After downloading Apache POI .zip extract to one folder that looks like below (copy all 11 jars from here to Project lib folder).

Folder Structure:--



Code:--

1>Servlet class:--

```
package com.app.servlet;
//ctrl+shift+O(Imports)
public class ExcelDataServlet extends HttpServlet
{
 @Override
 protected void service(HttpServletRequest req, HttpServletResponse res)   throws
ServletException, IOException    {
      res.setContentType("application/vnd.ms-excel");
 res.addHeader("Content-Disposition","attachement:filename=students.xls");
      try {                    //1. Create WorkBook
            HSSFWorkbook book = new HSSFWorkbook();
            //2.Create Sheet in workbook
      HSSFSheet sheet = book.createSheet("My STUDENTS DATA");
            //3.Create Rows in sheet
```

```
                HSSFRow row0=sheet.createRow(0);
                //4.Create Cells and add data
                //>> 1st row with data
                row0.createCell(0).setCellValue("S1.No.");
                row0.createCell(1).setCellValue("Name");
                row0.createCell(2).setCellValue("Marks");

                //>>2nd row with data
                HSSFRow row1=sheet.createRow(1);
                row1.createCell(0).setCellValue("101");
                row1.createCell(1).setCellValue("Uday");
                row1.createCell(2).setCellValue("98.36");

                HSSFRow row2=sheet.createRow(1);
                row2.createCell(0).setCellValue("102");
                row2.createCell(1).setCellValue("Ramu");
                row2.createCell(2).setCellValue("95.36");

                //5. Write data to ServletOutPutSteram
                book.write(res.getOutputStream());
                //6. close book
                book.close();
        }
        catch(Exception e)
        {
                System.out.println(e);
        }
 }
}
```

=>Run on server Enter URL like: http://localhost:3030/ContentTypeExcel/excel

2.>web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>ContentTypeExcel</display-name>
  <servlet>
   <servlet-name>Sample</servlet-name>
   <servlet-class>com.app.servlet.ExcelDataServlet</servlet-class>
   </servlet>
 <servlet-mapping>
   <servlet-name>Sample</servlet-name>
   <url-pattern>/excel</url-pattern>
 </servlet-mapping>
</web-app>
```

Servlet Redirect:--

=>When ever request is made by browser it will goto one servlet here this servlet only provides response.

=>Instead of this if we want to execute another servlet or another application service then we should re direct from our servlet to another servlet or another application by using bellow code

response.sendRedirect("/url");

NOTE:--

1>This re-direction works between

=>Servlet to servlet with in Application

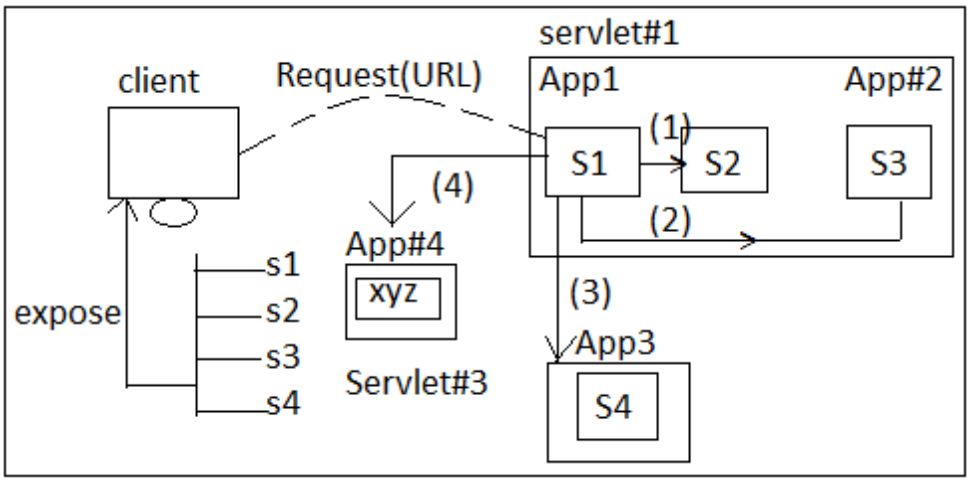
=>Servlet to servlet between 2 applications running in same server.

=>Servlet to Servlet between 2 applications running in different Server.

=>Servlet to Non-Servlet(Non-Java) between 2 Application running in same/different server.

=>First servlet Request, Response are ignored, finally last one is taken to browser.

==>It is mainly designed to work between two server/apps. Like our application to Gmail, Google, or facebook etc. It will not come back again, only forward direction.



S1 = Servlet #1 in App#1

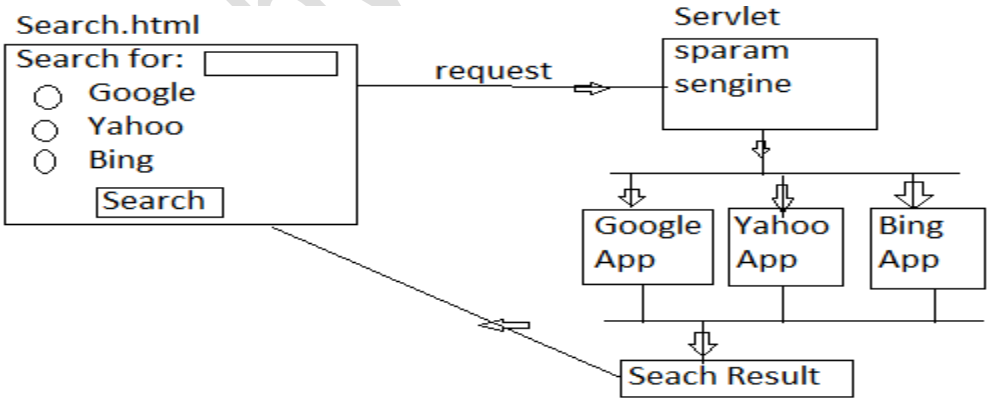
S2 = Servlet #2 in App#1

S3 = Servlet #3 in App#2

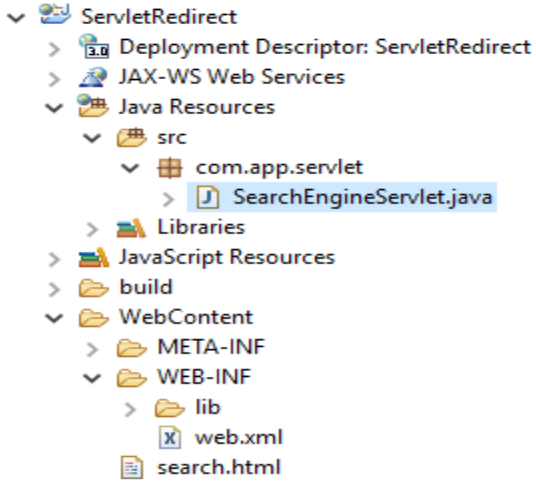
S4 = Servlet #4 in App#3

xyz = non Java (not a servlet) in App#4

Consider below example for redirection:--



Folder Structure:--



Code:--

1>Under WebContent create one HTML, file

```
<Html><body>
<h1>Search Engine Page</h1>
<body>
 <h1>Search Engine Page</h1>
 <form action="redirect" method="get">
     <pre>
 Search Here:
 <input type="text" name="sparam" placeholder="Enter Text Here" /><br><br>
 <input type="radio" name="sengine" value="google"  checked="checked">Google
 <input type="radio" name="sengine" value="yahoo">Yahoo
 <input type="radio" name="sengine" value="bing">Bing<br><br>
 <input type="submit" value="Search"/>
    </pre></form></body></html>
```

2>Create one Servlet class:--

```
package com.app.servlet;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;

public class SearchEngineServlet extends HttpServlet
{
public void service(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
{
 //1. read sparam, sengine
  String param = req.getParameter("sparam");
  String engine = req.getParameter("sengine");
  String url = null;
  if(engine.equals("google"))   {
   url ="https://www.google.com/search?q="+param;
  }
  else if(engine.equals("yahoo"))   {
   url ="https://www.yahoo.com/search?q="+param;
  }
    else
  {
   url ="https://www.bing.com/search?q="+param;
  }
  //goto URL (Search engine
  res.sendRedirect(url);
} }
```

=>Run on server Enter URL like: http://localhost:3030/Redirect

2>web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 id="WebApp_ID" version="3.0">
```

```
 <display-name>ServletRedirectEx9</display-name>
 <servlet>
  <servlet-name>search</servlet-name>
  <servlet-class>com.app.servlet.SearchEngineServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>search</servlet-name>
  <url-pattern>/redirect</url-pattern>
 </servlet-mapping>
 <welcome-file-list>
  <welcome-file>search.html</welcome-file>
 </welcome-file-list>
</web-app>
```

RequestDispatcher (Servlet Communication):--

=>RequestDispatcher is used to navigate from one servlet(or JSP) to another servlet(or JSP) within same application (and same server)

=>It will not work between two Apps or two different servers.

Types of Dispatching:--

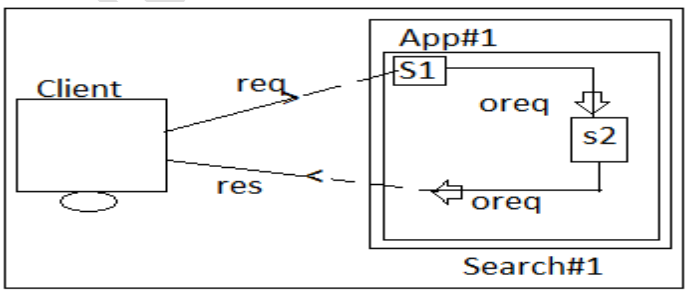
a>forward:-- If request made to Servlet#1 then it can send to Servlet#2 and Servlet#2 finally gives response.

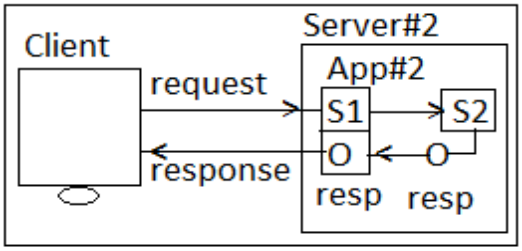
=>servlet#2 can read servlet#1 request data

=>It is one directional (forward) only not backward.

b>include:-- Current servlet can goto another servlet, execute that and get response back to our servlet.

=>Bi-Directional possible. It should also carry Data from one servlet to another servlet.



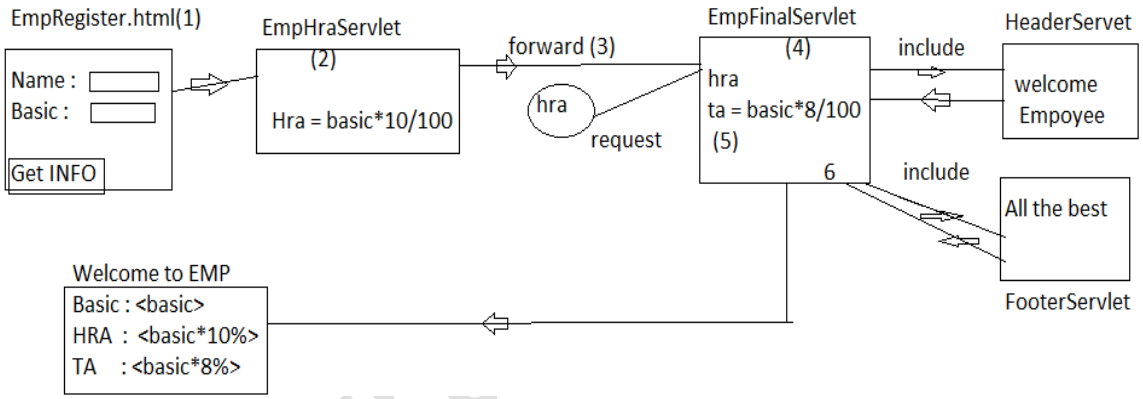


Code is:--

RequestDispatcher dispatcher = request.getReuestDispatcher("/url");

dispatcher.forward(request, response); Or dispatcher.include(request, response);

Consider below example for RequestDispatcher:--



Data Sharing between Servlets using Request Dispatcher:--

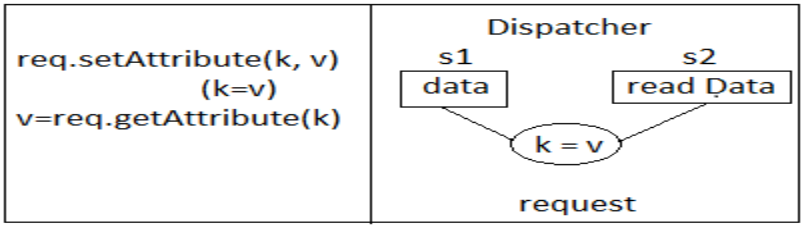
=>If request Dispatcher is used to move from one servlet to another servlet, in this case we send data from one servlet to another using "Request" object this is also called as "Attribute" Data.

Attribute:--Storing data in key=value in special server/browser memories using set/get methods known as Attributes (It is just like variable and value)

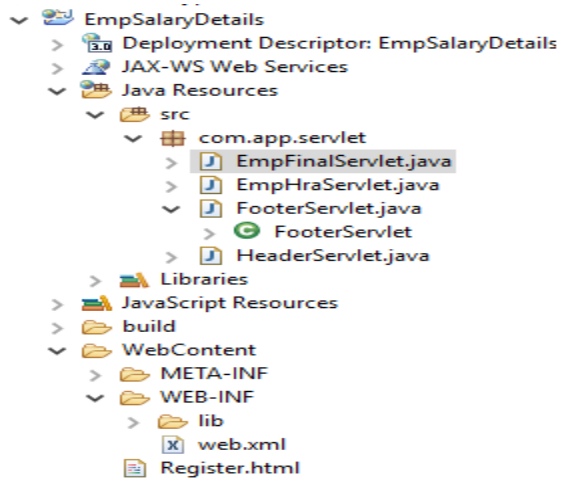
Method used are:--

setAttribute(key, value);

getAttribute(key) : value



Folder Structure:--



Code:--

1>EmpRegister.html:--

```
<!DOCTYPE html>
<html><head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
 <form action="calculate" method="post">
Enter Name : <input type="text" placeholder="ename" name="ename" /><br>
Enter BASIC Salary : <input type="text" placeholder="basic" name="basic"/><br>
        <input type="submit" value="get info" />
 </form></body></html>
```

2>Servlet code:--

a>EmpHraServlet:--

```
package com.app.servlet;
```

```
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class EmpHraServlet extends HttpServlet
{
 public void service(HttpServletRequest req, HttpServletResponse res) throws
ServletException,ArithmeticException , IOException
 {
        //1. read Basic data from FORM
        String str =req.getParameter("basic");
        //2.Parse data
        double basic = Double.parseDouble(str);
        //3. Calculate HRA
        double hra = basic*10/100;
        //4.sed Hra data using request
        req.setAttribute("hra", hra);
        //5.Move to EmpFInalServlet
        RequestDispatcher rd = req.getRequestDispatcher("/final");
        rd.forward(req,res);
 } }
```

b>EmpFinalServlet Code:--

```
 package com.app.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
public class EmpFinalServlet extends HttpServlet
{
 public void service(HttpServletRequest req, HttpServletResponse res) throws
ServletException, ArithmeticException,IOException
 {
        //1. read basic, name data
         String ename=req.getParameter("ename");
         String str =req.getParameter("basic");
         System.out.println(ename);
         System.out.println(str);
         //2.Parse data
         double basic = Double.parseDouble(str);
         //Calculate TA
         double ta =basic*8/100;
         //4.Read HRA Servlet data
         Object ob =req.getAttribute("hra");

         double hra = (Double)ob; //downcast
         //5. Include HeaderServlet Response
         RequestDispatcher rd1 = req.getRequestDispatcher("/header");
         rd1.include(req,res);

         //6. Print message
         PrintWriter out = res.getWriter();
         out.println("Name is :"+ename);
         out.println("BASIC is :"+basic);
         out.println("HRA is :"+hra);
         out.println("TA is :"+ta);
         //7.include FooterServlet Response
         RequestDispatcher rd2 =req.getRequestDispatcher("/footer");
         rd2.include(req, res);
 }
}
```

c>HeaderServlet Code:--

```
package com.app.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HeaderServlet extends HttpServlet
{
 public void service(HttpServletRequest req, HttpServletResponse res) throws
ServletException, ArithmeticException,IOException
 {
        PrintWriter out =res.getWriter();
        out.println("WELCOME TO EMPLOYEE");
 } }
```

d>FooterServlet Code:--

```
package com.app.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HeaderServlet extends HttpServlet  {
 public void service(HttpServletRequest req, HttpServletResponse res) throws
ServletException, ArithmeticException,IOException
 {
        PrintWriter out =res.getWriter();
        out.println("WELCOME TO EMPLOYEE");
 } }
```

3>web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>EmpSalaryDetails</display-name>
 <servlet>
  <servlet-name>Service1</servlet-name>
  <servlet-class>com.app.servlet.EmpHraServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>Service1</servlet-name>
  <url-pattern>/calculate</url-pattern>
 </servlet-mapping>

 <servlet>
  <servlet-name>Service2</servlet-name>
  <servlet-class>com.app.servlet.EmpFinalServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>Service2</servlet-name>
  <url-pattern>/final</url-pattern>
 </servlet-mapping>
 <servlet>
  <servlet-name>Service3</servlet-name>
  <servlet-class>com.app.servlet.HeaderServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>Service3</servlet-name>
  <url-pattern>/header</url-pattern>
 </servlet-mapping>
 <servlet>
```

```
  <servlet-name>Service4</servlet-name>
  <servlet-class>com.app.servlet.FooterServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>Service4</servlet-name>
  <url-pattern>/footer</url-pattern>
 </servlet-mapping>
 <welcome-file-list>
  <welcome-file>Register.html</welcome-file>
 </welcome-file-list>
</web-app>
```

HTTP (Hyper Text Transfer Protocol):--

=>It is a protocol used to send the data between Client-server.

=>Ex Clients are: web Browser, Cloud HttpClient etc…

Http supports two messages. Those are

1>HTTP Request

2>HTTP Response

1>HTTP Request:--

Head contains request/Response line and Parameters in key : value format

=>Between Head and Body it contains one blank link (CR+IF= new Line)

CR = Carrier Return =\r: It means move cursor to starting place in same line

LF = Line Feed = Move Vertically one Line

=>In java CR(\r)+LF=\n

=>Http Supports 7 methods to execute different tasks. Those are: GET, POST, PUT, DELETE, HEAD, TRACE and OPTIONS.

=>Servlet implemented them as doXXX() method in HttpServlet(AC)

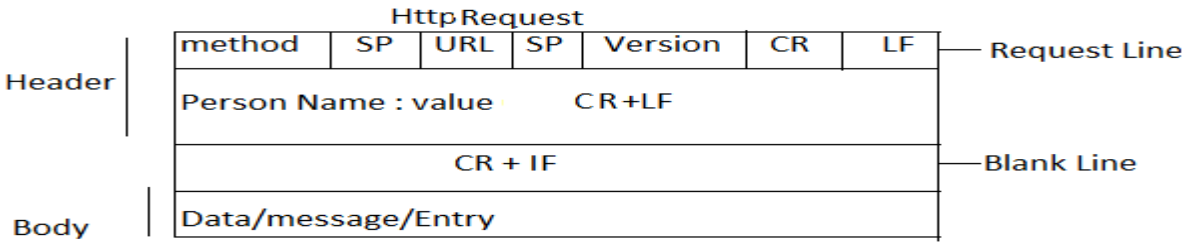
=>By using browser request can be made in 3 Ways. Those are:

a>Enter URL in Address bar(GET)

b>HTML From submit (**GET/POST)

c>Click on Hyperlink (<a>)(GET)

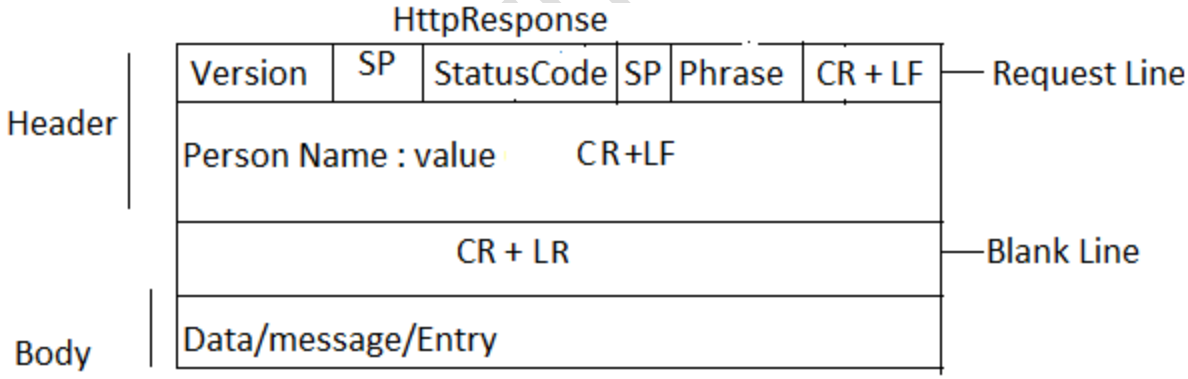
HTTP Request Format:--



Example:-

GET	www.facebook.com	Http 1.1
Client : Google Chrome Date : 12 feb 2018 6:58 AM Accept : text/html, image , png + Accept Lang : Eng		

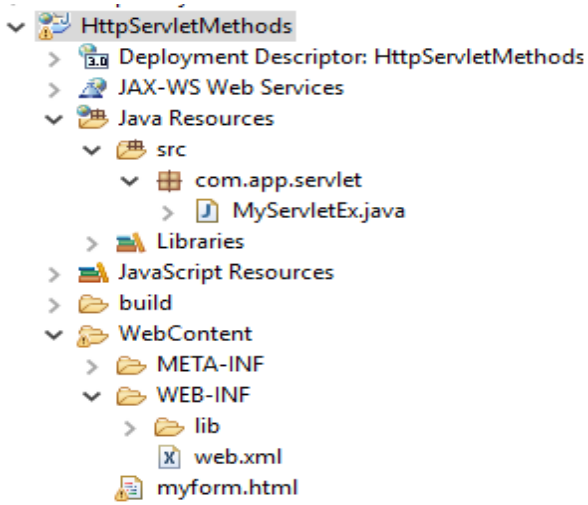
HTTP Response Format:--



Http 1.1	200 Ok
Content-Type : Text/Html Server :Apache Http 6.5 (MAC) Connection : Keep-alive	

Examples codes for Http Servlets methods:--

Folder Structure:--



1.>servlect class:--

```
package com.app.servlet;
public class MyServletEx extends HttpServlet
{@Override
 public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException    {
        PrintWriter out = res.getWriter();
        out.print("From GET Method");
 }
 @Override
 public void doPost(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
 {
        PrintWriter out = res.getWriter();
        out.print("From POST Method");
 }
}
```

2>web.xml file:--

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

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>ServletLifyCycleEx</display-name>
 <servlet>
 <servlet-name>s1</servlet-name>
 <servlet-class>com.app.servlet.MyServletEx</servlet-class>
 </servlet>
 <servlet-mapping>
 <servlet-name>s1</servlet-name>
<url-pattern>/data</url-pattern>
 </servlet-mapping>
</web-app>
```

Run on server and:

Case#1: Enter the URL in browser: http://localhost:3030/HttpServletMethodsEx/data

Case#2: Enter Below: URL:--http://localhost:3030/HttpServletMethodsEx/myform.html

=>And then=>click on “Register” HyperLink

Case#3: Enter URL:--http://localhost:3030/HttpServletMethodsEx/myform.html

=>And then=>click on “click Me” HyperLink

NOTE:--

1>URL is case sensitive. i.e /data and /Data, /DATA, /DaTa are different.

2>If Request URL is not matched with any <url-pattern> in web.xml, server returns HttpStatus 404 – not Found

3>If URL matched and Request Method Type is not matched with any Servlet doXXX method then server throws Http Status method then server throws Http Satus 405 Method not Allowed.

Ex:--Request is GET Type and sservlet has only doPost() and doPost() then 405 Error.

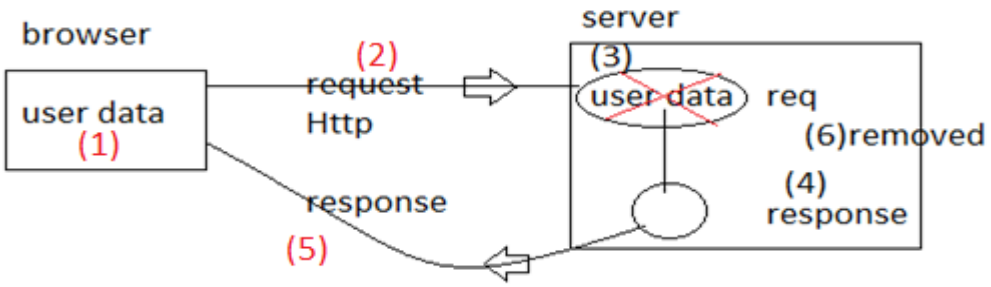
4>One servlet at max can have 7 doXXX() method (GET, POST….)

but Browser can call only GET, POST methods.

State Management/Session Management:--

Stateless Process:--Not storing end client/browser details at server side after providing response back.

=>By default HTTP is a Stateless protocol. It means "when browser made request, this data taken into HttpServletRequest object and maintained until providing HttpServletResponse commit. Once Response provided back to client, then requested data is no longer maintained".



Statefull/State management:--

Session Management:--

=>Storing user related data at server for long time (Login=Logout Time) is know as Session/State Management.

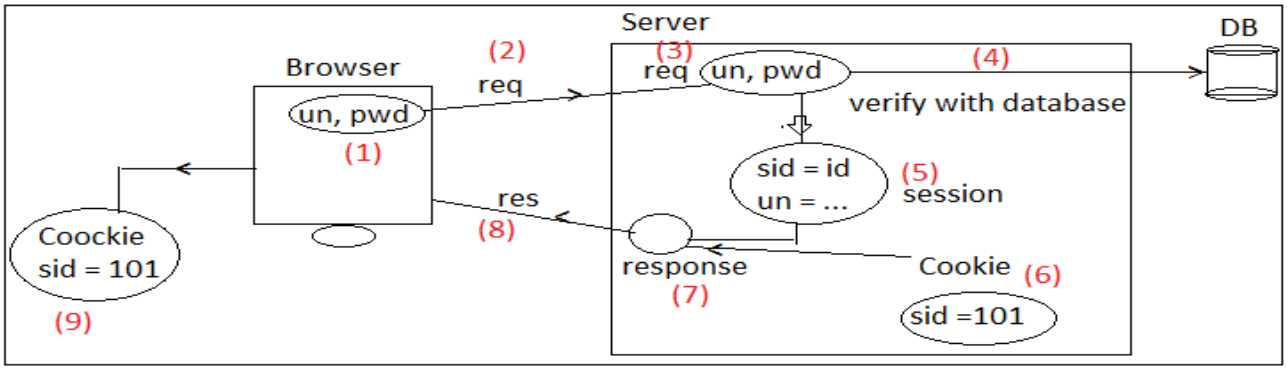
To do Session Management, concepts are:--

1>Hidden Form fields 2>URL Re-Writing

3>Cookie 4>Session

Are used

NOTE:-- In Real-time Session-with-Cookie concept is used.



Cookie:--

a>Cookie is a memory created by server and stored in browser.

b>Cookie stores data (at max 4KB) only text data.

c>A cookie created by server will be sent to browser using Http Response Head Area.

d>All cookies will be submitted along with your request using Http Request Head.

e>Cookie Stores data in key=value format. Those are also called as Attribute Name and Attribute Value.

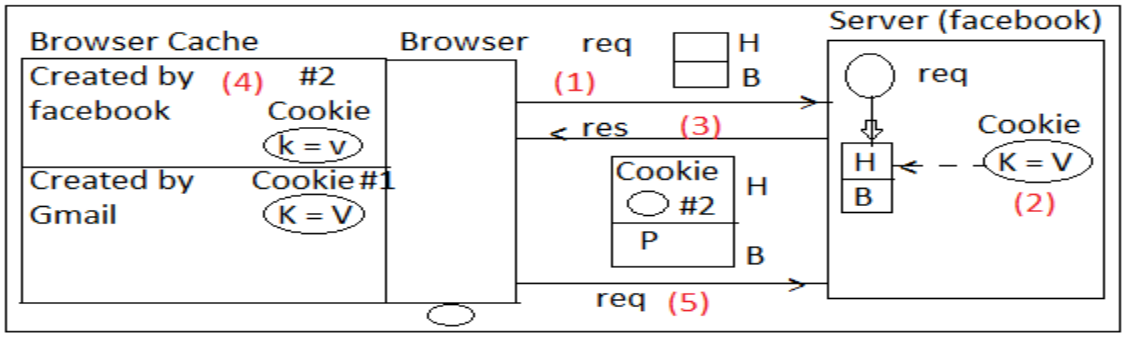
f>Every Cookie will be having max life time (expiry time). Default is -1 indicates on browser close all cookies are removed from browser cache.

g>We/Programmer can set life time of a cookie using method setMaxAge in sec(int value) values.

i.e 60 sec = 1 min

h>**A Cookie created by Server (S#1) can only read/modify/deleted by same Server(S#1) only.

Not by other servers by default.



Steps:--

1>Make New/First Request using browser

2>Server creates new Cookie and adds to Http Response (Head).

3>Response send back to the browser.

4>Browser creates all new cookies in Browser cache.

5>Browser submits all Cookies related to same Server using Http Request Head.

Limitations:--

=>In Browser we can disable Cookie concept, then application may not work properly.

=>Stores only Text data, not binary data like Images, Audio, Video, etc.
=>Secure data cannot be stored in Cookie.

Working on Cookie:--

1>To create one Cookie use class Cookie given by servlet HTTP API in package "javax.servlet.http" use parameterized constructor.

```
Cookie c = new Cookie (name, value);
```

=>name and both value are String type only.
=>set Lifetime of Cookie using method

```
setMaxAge(int); in seconds.
```

Default value is -1, indicates "remove cookie after closing browser.

2>Send Cookie From Server to browser using HTTP Response Head.

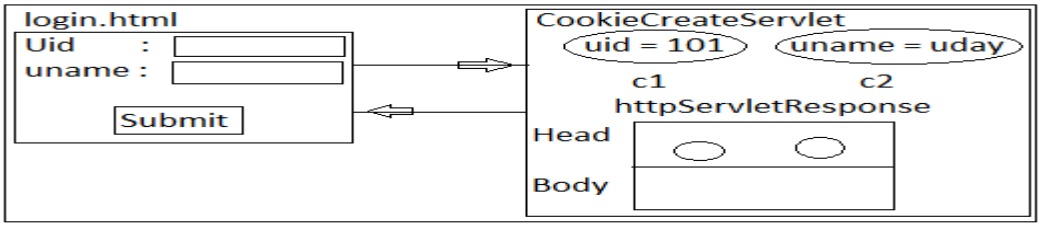
```
response.addCookie(c);
```

3>On request made, browser submits all Cookies to server using HTTP Request Head. To read those codes is:

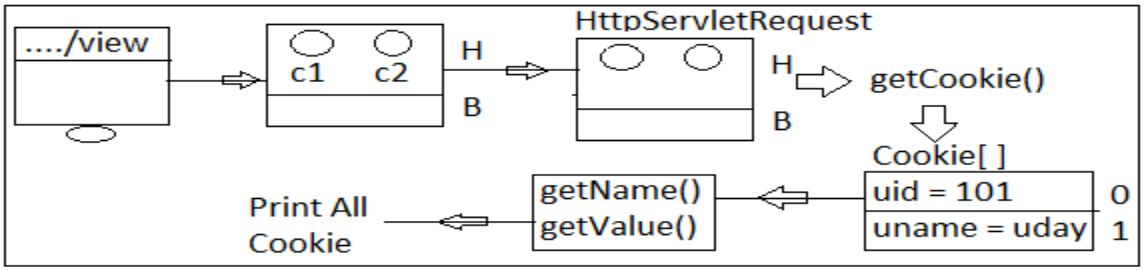
```
Cookie[] cks =
request.getCookies();
```

4>To read one Cookie data use index number and methods like getName(), cks[0].getName(); cks[0].getValue);

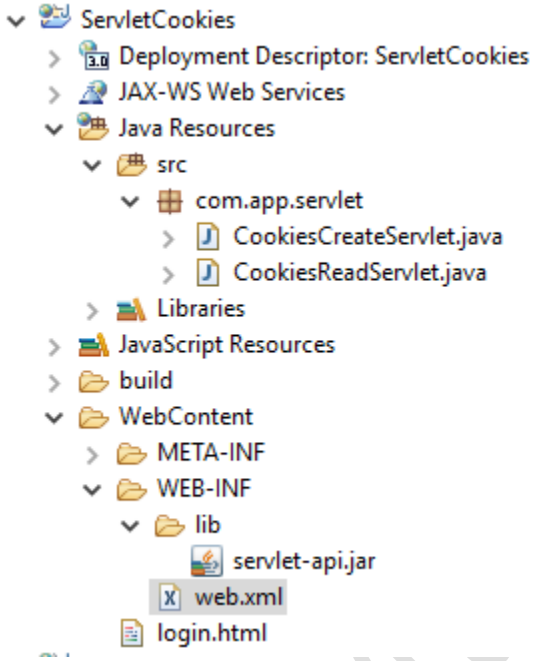
1>Create Cookie:--



2>Read Cookie:--



Folder Structure:--



Code:--

1>Login.html (Under WebContent):--

```
<!DOCTYPE html>
<html><head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h2>Welcome to Cookie Example</h2>
<form action ="create" method="POST">
<pre>
ID :<input type="text" name="uid"/><br>
```

```
Name :<input type="text" name ="uname"/><br>
<input type ="submit" value="submit"/>
</pre></form>
</body></html>
```

2>Create two servlet class one for Creating and another for Reading Cookies:

i>For Creating Cookies:--

```
package com.app.servlet;
public class CookiesCreateServlet extends HttpServlet
{
  public void doPost(HttpServletRequest req, HttpServletResponse  res) throws
ServletException, IOException  {
        //1.Read data From input
        String uid = req.getParameter("uid");
        String uname = req.getParameter("uname");
        //2.Create Cookies
        Cookie c1 = new Cookie("uid", uid);
        Cookie c2 = new Cookie("uname", uname);

        //set max life time in sec
        c2.setMaxAge(600); //in sec
        //3.Add to response
        res.addCookie(c1);
        res.addCookie(c2);
        //4.Print done message
        PrintWriter out = res.getWriter();
        out.print("Cookies Created");
  }
}
```

ii>For Reading Cookie:--

```
package com.app.servlet;
public class CookiesReadServlet extends HttpServlet
{
```

```
 public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
 {
    //1.Read all Cookies from request-head
         Cookie[] cks = req.getCookies();
         //2.Print cookie name, value
         PrintWriter out = res.getWriter();
         if(cks!=null && cks.length!=0)
         {
                 for(int i =0; i<cks.length; i++)
                 {
         out.print("Cookie name = "+cks[i].getName()+
                    ",value="+cks[i].getValue());
                 } }
         else   {
            out.print("No Cookies Found");
         }
 } }
```

3>web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>ServletCookies</display-name>
 <servlet>
 <servlet-name>c1</servlet-name>
 <servlet-class>com.app.servlet.CookiesCreateServlet</servlet-class>
 </servlet>
 <servlet-mapping>
 <servlet-name>c1</servlet-name>
 <url-pattern>/create</url-pattern>
 </servlet-mapping>
```

```
 <servlet>
 <servlet-name>c2</servlet-name>
 <servlet-class>com.app.servlet.CookiesReadServlet</servlet-class>
 </servlet>
 <servlet-mapping>
 <servlet-name>c2</servlet-name>
 <url-pattern>/read</url-pattern>
 </servlet-mapping>
 <welcome-file-list>
  <welcome-file>login.html</welcome-file>
  </welcome-file-list>
</web-app>
```

Execution Order:--

#1>Start the server and Enter URL in browser:
http://localhost:3030/ServletCookies/create

#2>Enter details and submit Form.

#3>Enter below URL to view Cookies: http://localhost:3030/ServletCookies/read

Types of Cookies:--

1>SessionCookies/Browsing Session Cookies:--

=>A Cookie which do not have expiration time As soon as the web browser is closed this cookies gets destroyed.

2>Persistent Cookies:--

=>A Cookie is created with life-time and Destroyed based on the expiry time.

Q.>How to remove a Cookie from browser before expiry time?

Ans:-- setMaxAge to zero and add to response.

Ex:-- c.setMaxAge(0);

HttpSession:-- It is given by Servlet API in package “javax.servlet.http”.

=>It is used to do session management state management at server side.

=>In simple Securing Application using Login (create session) and logout (invalid session-remove session) is done using HttpSession concept.
=>Session means String identity/data for a time period at server side of an user.
=>For every user one session will be created on successful login.
=>HttpSession holds data in key=value format. Key is String type and value is Object type.
=>In Servlets HttpSession auto validated and removed after 30 min if user is inactive (no request is made for 30 min)
=>It is also called as “MaxInactive Interval” value.

Coding Point:--

a>Create one new session.

```
HttpSession ses = request.getSession();
                Or
HttpSession ses = request.getSession(true);
```

b>Reading existing session

```
HttpSession  ses =request.getSession(false);
```

c>Store data in session

```
ses.setAttribute(String key, Object value);
```

d>Read data from Session

```
Object value = ses.getAttribute(key);
```

e>Remove data from session

```
ses.removeAttribute(key);
```

f>Remove session from server

```
ses.invalidate();
```

g>Modify Max Inactive time for one session

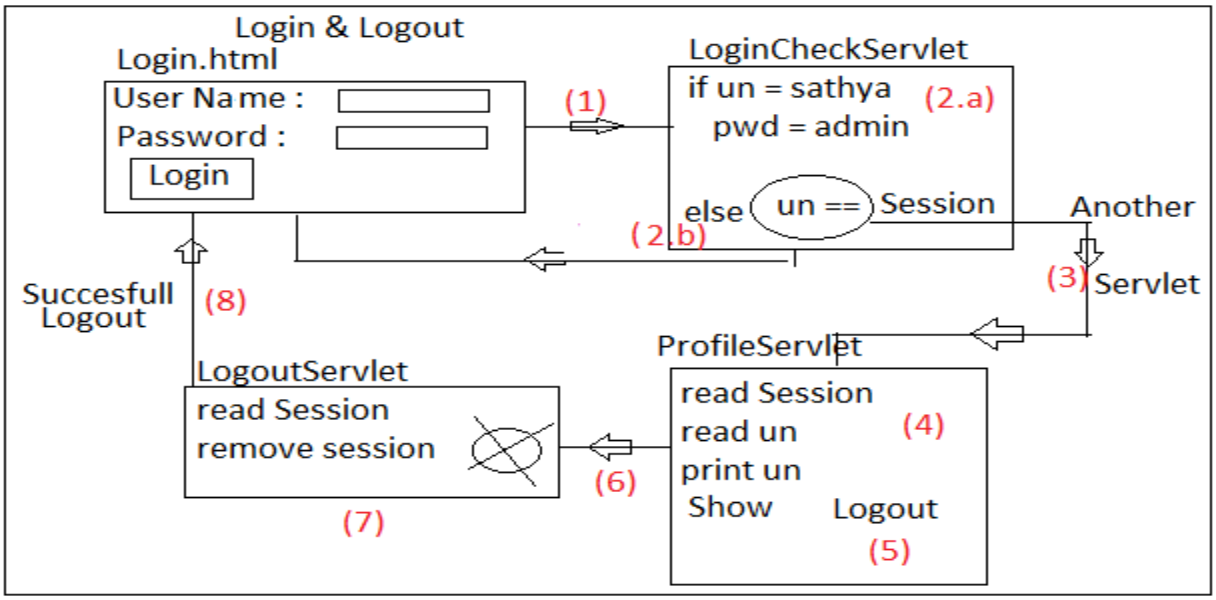
```
ses.setMaxInactiveInterva(int sec);
```

h>Modify Max Inactive time for all sessions (Write code in web.xml)

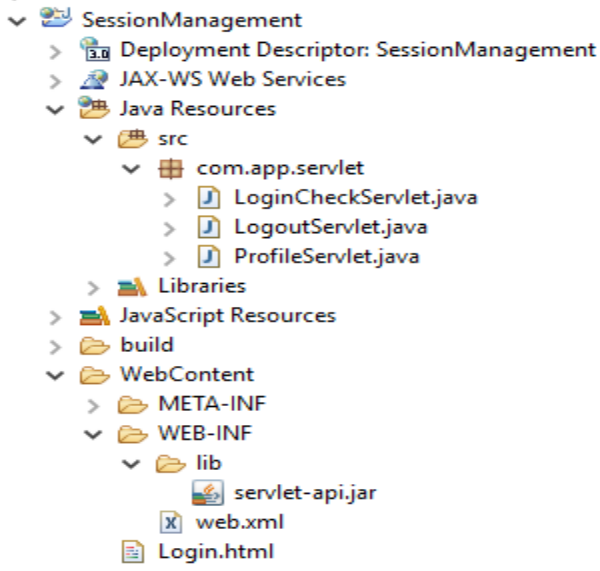
```
<session-config>
  <session-timeout>15</session-timeout>
</session-config>
```

NOTE:-- Session handling is supported by only javax.servlet.http package and HttpServlet, not by javax.servlet package and GenericServlet.

Example (Secure) Session Management App:--



Folder Structure:--



Code:--

1>Login.html:--

```
<!DOCTYPE html>
<html><head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head><body>
<h2>welcome to Login page!!</h2>
<form action="login" method="post">
<pre>
NAME          : <input type="text" name="username"/>
PASSWORD   : <input type="password" name="userpwd"/>
<input type="submit" value="Login"/>
</pre></form></body></html>
```

2>LoginCheckServlet:--

```
package com.app.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class LoginCheckServlet extends HttpServlet
{
 public void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        //1. read un,pwd
        String un=req.getParameter("username");
        String pwd=req.getParameter("userpwd");
```

```
        RequestDispatcher rd=null;
        resp.setContentType("text/html");

        //2. verify details
        if("sathya".equals(un) && "admin".equals(pwd)) {
                //create new session
                HttpSession ses=req.getSession();
                //store un in session
                ses.setAttribute("user", un);
                //goto profile servlet
                rd=req.getRequestDispatcher("profile");
                rd.forward(req, resp);
        }else {
                PrintWriter out=resp.getWriter();
                out.println("Please try again");
                out.println("<a href='login.html'>Login Again</a>");
        }
 } }
```

3>LogoutServlet:--

```
package com.app.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class LogoutServlet extends HttpServlet
{
 public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        //read current session
```

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

        resp.setContentType("text/html");
        PrintWriter out=resp.getWriter();
        if(ses!=null) {
                //remove user name
                ses.removeAttribute("user");
                //remove session
                ses.invalidate();
        }
        //show Success Message
        out.println("Successfully logged out");
        out.println("<a href='login.html'>Login Here</a>");
 }
}
```

4>ProfileServlet:--

```
package com.app.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
public class ProfileServlet extends HttpServlet
{
 public void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException
 {
        //read current session
        HttpSession ses=req.getSession(false);

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

```
        PrintWriter out=resp.getWriter();
        if(ses!=null)
        {
                //if session exist
                Object ob=ses.getAttribute("user");
                out.println("Welcome to User ::"+ob);
                out.print("<a href='logout'>Logout</a>");
        }else
        {
                //session not exist
                out.print("You are not logged in ");
                out.print("<a href='login.html'>Please Login</a>");
        }
} }
```

5>web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>SessionManagement</display-name>
 <welcome-file-list>
        <welcome-file>Login.html</welcome-file>
 </welcome-file-list>
 <servlet>
        <servlet-name>LoginCheck</servlet-name>
        <servlet-class>com.app.servlet.LoginCheckServlet</servlet-class>
 </servlet>
 <servlet-mapping>
        <servlet-name>LoginCheck</servlet-name>
        <url-pattern>/login</url-pattern>
 </servlet-mapping>
```

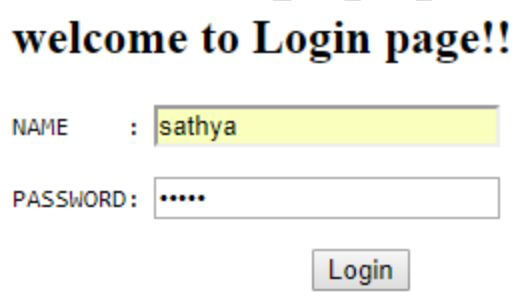
```
<servlet>
        <servlet-name>Profile</servlet-name>
        <servlet-class>com.app.servlet.ProfileServlet</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>Profile</servlet-name>
        <url-pattern>/profile</url-pattern>
</servlet-mapping>
<servlet>
        <servlet-name>Logout</servlet-name>
        <servlet-class>com.app.servlet.LogoutServlet</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>Logout</servlet-name>
        <url-pattern>/logout</url-pattern>
</servlet-mapping>
</web-app>
```

Execution Order:--

#1>Start the server and Enter URL in browser:

http://localhost:3037/SessionManagement/

#2>Enter user name as sathya and password as admin and submit Form.



#3>Click on logout Button for logout.

Servlet and JDBC Integration:--

=>Servlet class must be linked with JDBC classes using Factory with POJI-POJO design pattern, which creates communication link between two in loosely coupled way.

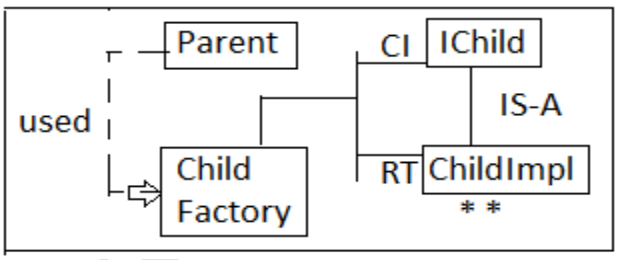
Coupling:-- Linking (Integration) of two types of code (Servlet, JDBC).

1>Tight coupling:-- If one code(class) gets changed then another code(class) gets effected. But they are connected final.

```
Ex:-- class Parent {
Void get() {
Child  c = new Child();
c.show();
} }
class Child  {
Void show() {  ---}
}
```

=>if we modify child class name (or moved) to another class) or method name (or changed to another method) then parent class code gets effected (need to modify).

2>Loosely Couplinng:--Link (Integrate) two codes (classes) if child gets modified then it's parent should not be, changed should come only at link.(use Factory with POJI-POJO)



POJI

```
Interface IChild {
void show ();
}
```

POJO

```
class childImpl implements
IChild {
void show () {---}
}
```

=>In JDBC, to get one Communication object for all DB operations use “Singleton” design pattern. Here use static block which will be executed only once while loading class.

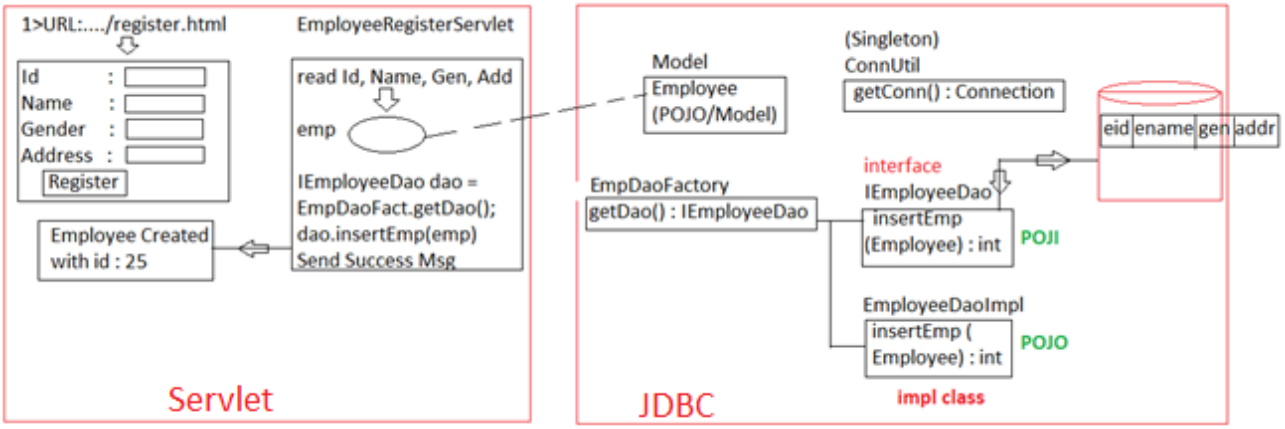
Format:

```
public class _____Util {
 private static Type var =null;
static {
 var = new (Object…) ;
}
public static Type getOb() {
return var;
}
```

POJO/Model:-- A class not connected to any special Technology (like Servlets, JSP, JDBC, Java Mail, JMS, JSF..) having simple variable with set/get methods. No logical/operations methods.

[POJO = Plane Old Java Object]

**These classes object are used to store/read only data.



Coding Order:--

1.>ConnUtil (Singleton) :--This class return one connection obj

pack : com.app.util

class : ConnUtil.java

2.>Employee (Model) :-- It is used to store only FORM data and same inserted in DB Table row.

pack : com.app.model

class : Employee.java

3.>IEmployeeDao(POJI) and EmployeeDaoImpl (POJO) used for loosely coupling process

pack : com.app.dao

cinterface : IEmployeeDao.java

pack : com.app.dao

class : EmployeeDaoImpl.java

4>EmployeeDaoFactory (Factory class) :-- It creates POJO object and return POJI format.

pack : com.app.factory

class : EmpDaoFactory.java

5>register.html (UI Page) :-- It will display HTML Form to register employee. It is set as welcome-file Location: Under “WebContent”

6>EmployeeRegisterServlet (HttpServlet) :-- This servlet will execute below steps.

i>raed form data on click on submit button.

ii>Parse data if required

iii>Convert data to mpdel class Object.

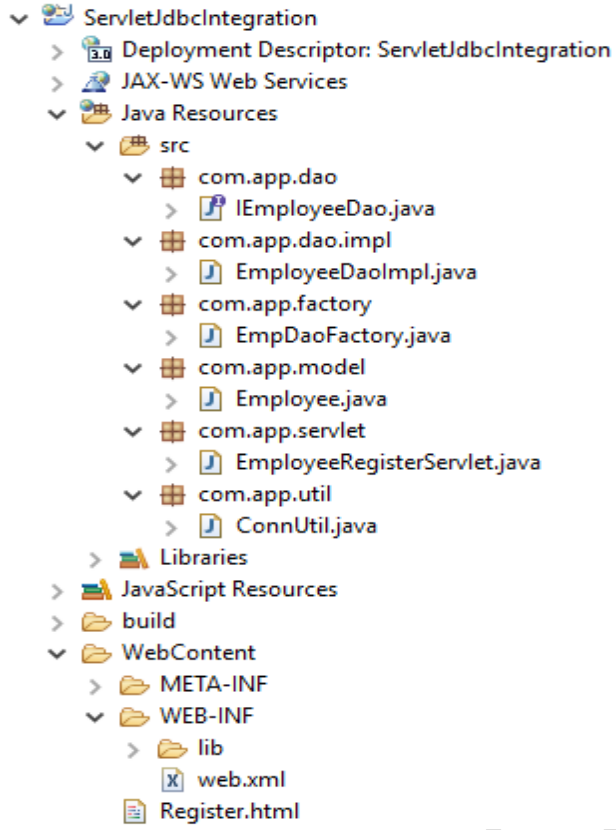
iv>Craete Dao object and call operation

v>print result and dispatch to register.html(rd.include())

NOTE:--Create below table in Oracle DB:

create table empTabs(eid number, ename varchar2(20), egen varchar2(10),eaddr varchar2(25);

Folder Structure:--



Code:--

1>Create a ConnUtil class:--

```
package com.app.util;
import java.sql.Connection;
import java.sql.DriverManager;

public class ConnUtil {
 private static Connection con=null;
 static  {
       try {
          Class.forName("oracle.jdbc.driver.OracleDriver");
  con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
                        "system","system");
          }
        catch (Exception e) {
               e.printStackTrace();
```

```
        }
 }
 //return conn object
 public static Connection getConn(){
        return con;
 }
}
```

2>Create an Employee Model class:--

```
package com.app.model;
public class Employee  {
 private int empId;
 private String empName;
 private String empGen;
 private String empAddr;
 //set/get methods
 //alt+shift+S,R(SelectAll>OK)
}
```

3> Create an IEmployeeDao interface:--

```
import com.app.model.Employee;
//POJI-Plane Old Java Interface
public interface IEmployeeDao
{
 public int insertEmp(Employee emp);
}
```

4>Create an EmployeeDaoImpl class:--

```
package com.app.dao.impl;
//POJO
public class EmployeeDaoImpl implements IEmployeeDao{
 public int insertEmp(Employee emp) {
        String sql="insert into emptabs values(?,?,?,?)";
        Connection con=null;
```

```
        PreparedStatement pstmt=null;
        int count=0;
        try {
                //read connection
                con=ConnUtil.getConn();
                //create statement
                pstmt=con.prepareStatement(sql);
                //set data to stmt
                pstmt.setInt(1, emp.getEmpId());
                pstmt.setString(2, emp.getEmpName());
                pstmt.setString(3, emp.getEmpGen());
                pstmt.setString(4, emp.getEmpAddr());
                //execute stmt
                count=pstmt.executeUpdate();
        } catch (Exception e) {
                e.printStackTrace();
        }finally {
                //release connection
                con=null;
        }
        return count;
 } }
```

5>Create an EmpDaoFactory class:--

```
package com.app.factory;
import com.app.dao.IEmployeeDao;
import com.app.dao.impl.EmployeeDaoImpl;
//factory class
public class EmpDaoFactory {
 //return POJO object in POJI casted
        public static IEmployeeDao getDao() {
                return new EmployeeDaoImpl();
        }
}
```

6>Create an EmployeeRegisterServlet class:--

```
package com.app.servlet;
public class EmployeeRegisterServlet extends HttpServlet{
 public void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException
{
        resp.setContentType("text/html");
        //1. read input form data
        String  eid=req.getParameter("eid");
        String  empName=req.getParameter("ename");
        String  empGen=req.getParameter("egen");
        String  empAddr=req.getParameter("eaddr");

        //2. parse data if required
        int empId=Integer.parseInt(eid);
        //3. create model class object
        Employee emp=new Employee();

        //4. store data in model class obj
        emp.setEmpId(empId);
        emp.setEmpName(empName);
        emp.setEmpGen(empGen);
        emp.setEmpAddr(empAddr);

        //5. create Dao Object using Factory
        IEmployeeDao dao=EmpDaoFactory.getDao();
        //6.call insert operation
        int count=dao.insertEmp(emp);

        //7. print final message
        PrintWriter out=resp.getWriter();
        if(count==1) {
                out.println("Successfully created record with Id : "+empId);
```

```
        }else {
                out.print("Check manully ... giving problem");
        }
        //8.
        RequestDispatcher rd=req.getRequestDispatcher("Register.html");
        rd.include(req, resp);
 }
}
```

7>Write a web.xml file:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>ServletJdbcIntegrationEx</display-name>
 <welcome-file-list>
  <welcome-file>Register.html</welcome-file>
 </welcome-file-list>
  <servlet>
        <servlet-name>EmpReg</servlet-name>
        <servlet-class>com.app.servlet.EmployeeRegisterServlet</servlet-class>
 </servlet>
 <servlet-mapping>
        <servlet-name>EmpReg</servlet-name>
        <url-pattern>/insert</url-pattern>
 </servlet-mapping>
</web-app>
```

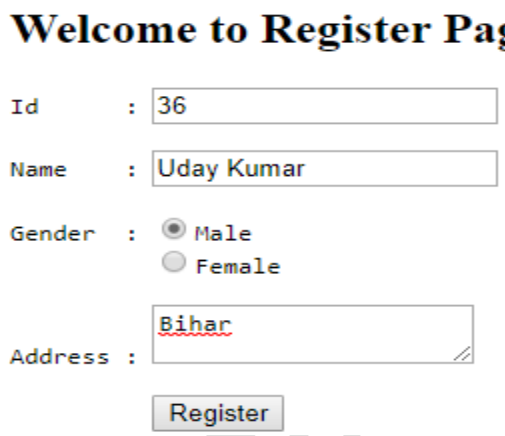
8>Write a Register.html file under WebContent:--

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

```
<title>Insert title here</title>
</head><body>
<h2>Welcome to Register Page!!</h2>
<form action="insert" method="post">
<pre>
Id     : <input type="text" name="eid"/><br>
Name   : <input type="text" name="ename"/><br>
Gender  : <input type="radio" name="egen" value="Male"/>Male
      <input type="radio" name="egen" value="Female"/>Female
Address : <textarea name="eaddr"></textarea><br>
      <input type="submit" value="Register"/>
</pre></form></body></html>
```

=>Enter the form details and click on Register button.

Output Screen:--



Types of URL Patterns:--

Q>Why URL Patterns?

Ans>Browser can't understand Java Objects and can't make servletObj.service() method call. So, programmer should write code in web.xml as.

```
<servlet>
  <servlet-name>s1</servlet-name>
 <servlet-class>com.app.Emp</servlet-class>
</servlet>
```

=>Above code means creating a servlet object (by using container).

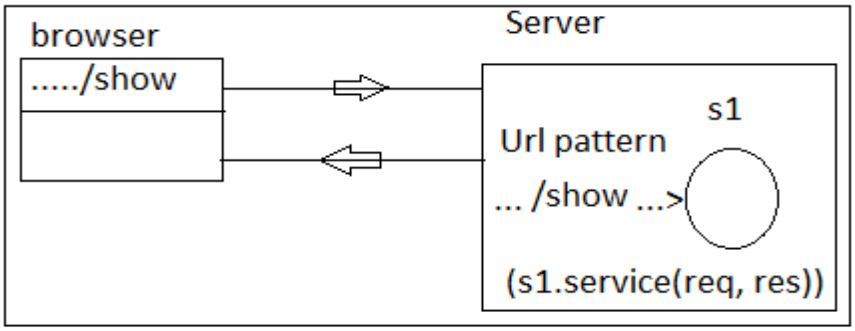
It means

Emp s1 = new Emp();

Next we should write code like:

```
<servlet-mapping>
  <servlet-name>s1</servlet-name>
  <url-pattern>/show</url-pattern>
</servlet-mapping>
```

=>It means link URL with above object, when browser enter above URL and then server calls s1.service(req, res) method.



Q>How many Types of URL Patterns supported by servlet? Why?

Ans>3 types Those are

1>Exact Match ex:/insert

2>Extension Match ex :*.do, *.sathya

3>Directory match Ex:/emp/*, /sathya/*, /abc/*

1>Exact match:-- One servlet will do one task by default, in this case use this URL, Pattern.

2>Extension Match:-- One servlet wants to do multiple operations based on request URL. But it will not support all types of Parameters [input data] concept.

Ex Path parameters is not supported.

>>example Pattern : *.do

Valid Request URLs:

http://...../a/b.do

http://...../a/.a.a.do.do

http://...../a/emp/save/55.do

Invalid request URLs:

http://...../a/b.do.one

http://...../a.form

http://...../emp/save/55.Do

3>Directory match URL Pattern:--

It is also used to work with one servlet which executes multiple operations based on Request URL. ** It also supports any kind of input parameters.

Ex:-- URL Pattern : /sathya/*

Valid Request URLs:

http://...../sathya/insert/10/20

http://...../sathya/delete

http://……/sathya

http://.....sathya/a.b.c

Invalid Request URL:--

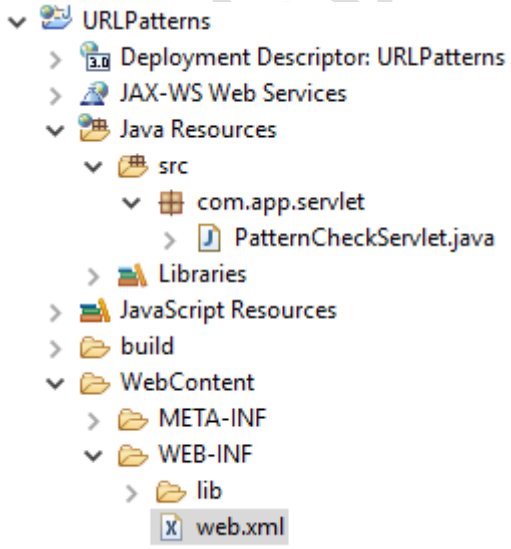
http://...../Sathya/hello

http://...../sathay/hello

http://...../emp/insert

Ex:--One servlet multiple tasks using Directory Match

Folder Structure:--



1>Servlet class (PatternCheckServlet):--

```
package com.app.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class PatternCheckServlet extends HttpServlet
{
public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
{
 //Read input data
 String p1 = req.getParameter("p1");
 String p2 = req.getParameter("p2");
 PrintWriter out = res.getWriter();
 if(p1==null || p2==null)
 {
        out.println("Provider P1, P2 inputs");
 }
 else   {       //parse data to double
        double d1 =Double.parseDouble(p1);
        double d2 =Double.parseDouble(p2);
        //read request URI
        String uri = req.getRequestURI();
        if(uri.contains("add"))
        {
                out.print("Result is : "+(d1+d2));
        }
        else if(uri.contains("sub"))
        {
```

```
                out.print("Result is : "+(d1-d2));
        }
        else if(uri.contains("mul"))
        {
                out.print("Result is : "+(d1*d2));
        }
        else if(uri.contains("div"))    {
                out.print("Result is : "+(d1/d2));
        }
        else
                out.println("Result is Not Found");
    }
} }
```

2>web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>URLPatterns19</display-name>
 <servlet>
  <servlet-name>URLS</servlet-name>
  <servlet-class>com.app.servlet.PatternCheckServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>URLS</servlet-name>
  <url-pattern>/pattern/*</url-pattern>
 </servlet-mapping>
</web-app>
```

Execution Steps:--

=>Run on server and type bellow URL in browser:

Case#1>http://localhost:3037/URLPatterns19/pattern/add?p1=10&p2=20

Case#2>http://localhost:3037/URLPatterns19/pattern/sub?p1=10&p2=20
Case#3>http://localhost:3037/URLPatterns19/pattern/mul?p1=10&p2=20
Case#4>http://localhost:3037/URLPatterns19/pattern/div?p1=10&p2=20

Filter:-- Filters are auto-executable codes for servlets. These are used to execute "Pre-processing logic on request" and "Post-processing logic on response".
1>To create one filter write one class and implement Filter (I) interface given by javax.servlet package.
2>It has 3 abstract methods also called as life cycle methods. Those are init(), doFilter(), destroy().
3>After writing this filter class, must configure in web.xml using below code.

```
<filter>
    <filter-name></filter-name>
   <filter-class> </filter-class>
</filter>
```

=>It is used to create Filter class obj by servlet container (web container)

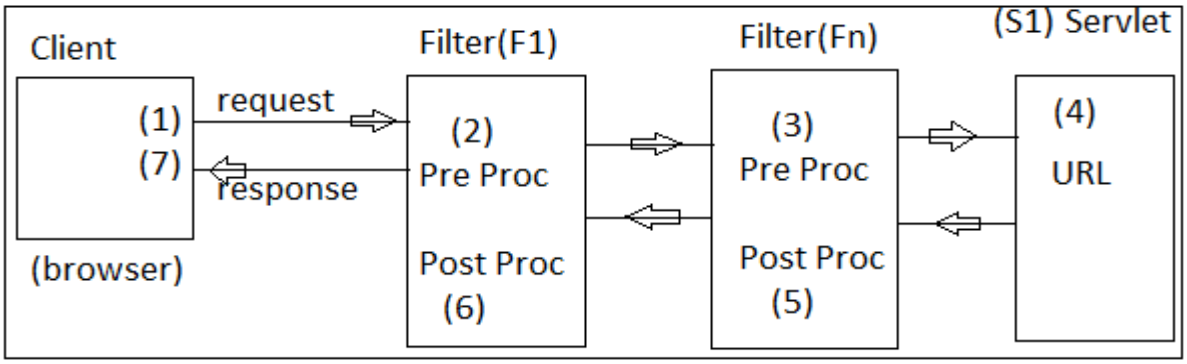
```
<filter-mapping>
   <filter-name> </filter-name>
   <url-pattern> </url-pattern>
</filter-mapping>
```

=>This code is used to link one filter with servlet, i.e ****Filter URL Pattern must match with servlet URL pattern.

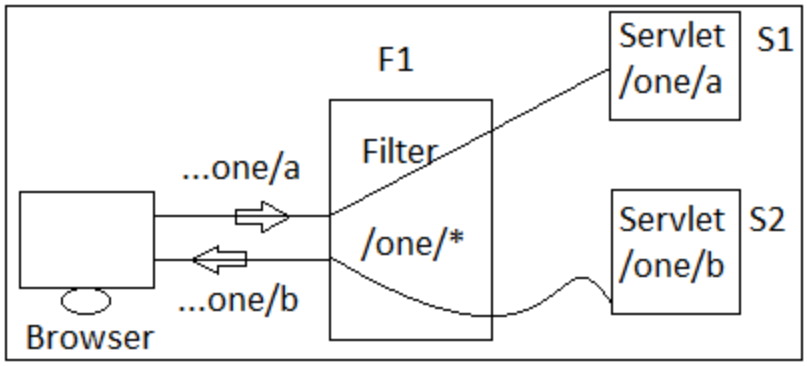
4>Filter class Objects are created by Servlet container on server startup (default: eager/early loading).
<load-on-startup> is not possible.
5>For one servlet we can write multiple Filter, execute in web.xml configuration order.
6>Filter Accept the init() parameters like servlet. Input to init method.
7>To unlink Filter with servlet, just change filter URL pattern .ie
**** servlet URL pattern != Filter URL pattern
=>It is also called as Filter are plugable. i.e Easy link/unlink.

8>Filter can be used only for either “pre-processing” or “Post-processing” or even both.
9>Browser makes request to servlet only but filter process it then goes to servlet.
10>***doFilter() method is available in Filter interface as life cycle method and in FilterChain to call next Filter or servlet.
11>Filter are used in security Programming Session check, encrypt and decrypt, identity verification, request data validation, response conversions etc..
12>One filter can be linked with multiple servlets also. But servlets URL pattern some part must be same, then Filter URL Pattern should follow either Extension match or directory match.

Design#1:-- Multiple filter for one Servlet:--
=>Request Execution order F1, F2, F3, ….Fn
=>Response Execution Order Fn, Fn-1…..F2, F1



Design#2:--One Filter linked with multiple servlets:--



Filter Coding Steps:--

Step#1:- Write on public class and implement Filder(I).

Step#2:-Override 3 abstract method init(), doFilder(), destroy().

Step#3:-Provide logic in three methods in Lifecycle doFilter() method call FilterChain doFilter() method, which calls next Filter in series or servlet service method (or any doXXX() method).

Filter(I):-- doFilter(servletRequest, servletResponse, FilterChain)

=>It is a LifeCycle method. It is a 3 parameter

FilterChain(I):-- doFilter(servletRequest, servletResponse)

=>Calls next Filter or servlet. It is a 2 parameter

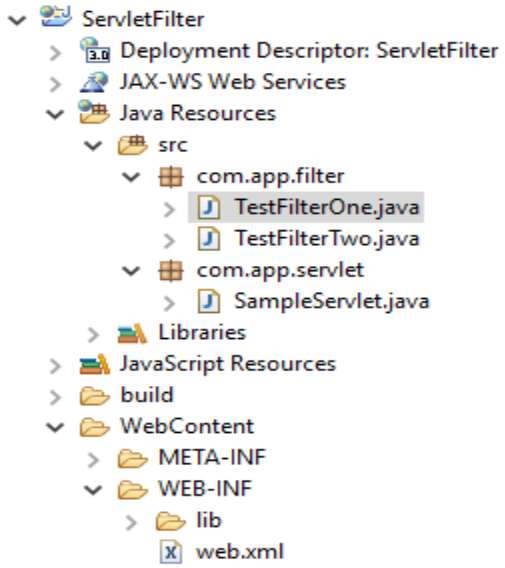
Step#4:-Configured filter in web.xml make sure Filter URL Pattern must match with servlet URL pattern.

Format should be:--

```
<filter>
  <filter-name>---</filter-name>
  <filter-class>---</filter-class>
</filter>
<filter-mapping>
  <filter-name>---</filter-name>
  <url-pattern>---</url-pattern>
</filter-mapping>
```

Step#5:-Run application in server and make request to servlet.

Folder Structure:--



Code:--

1>SampleServlet:--

```
package com.app.servlet;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SampleServlet extends HttpServlet
{
 public void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        System.out.println("im from servlet-service");
 }
}
```

2>TestFilterOne:--

```
package com.app.filter;
import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
```

```
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class TestFilterOne implements Filter
{
 public void init(FilterConfig fConfig) throws ServletException {
        System.out.println("Im from Filter-One- Init..");
 }
 public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
         throws IOException, ServletException {
        System.out.println("from doFilter-One-pre-processing");
        chain.doFilter(request, response);
        System.out.println("from doFilter-One-post-processing");
 }
 public void destroy() {
        System.out.println("from destory-One");
 }
}
```

3>TestFilterTwo:--

```
package com.app.filter;
import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class TestFilterTwo implements Filter
{
```

```
 public void init(FilterConfig config) throws ServletException
 {
        String data=config.getInitParameter("data");
        System.out.println("Im from Filter Two Init.."+data);
 }
 public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
         throws IOException, ServletException
 {
        System.out.println("from doFilter-Two-pre-processing");
        chain.doFilter(request, response);
        System.out.println("from doFilter-Two-post-processing");
 }
 public void destroy() {
        System.out.println("from -Two-destory");
 }
}
```

4>web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>ServletFilter20</display-name>
 <servlet>
  <servlet-name>SampleServlet</servlet-name>
  <servlet-class>com.app.servlet.SampleServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>SampleServlet</servlet-name>
  <url-pattern>/sample</url-pattern>
 </servlet-mapping>
```

```
 <filter>
  <filter-name>F1</filter-name>
  <filter-class>com.app.filter.TestFilterOne</filter-class>
 </filter>
 <filter-mapping>
  <filter-name>F1</filter-name>
  <url-pattern>/sample</url-pattern>
 </filter-mapping>
 <filter>
  <filter-name>F2</filter-name>
  <filter-class>com.app.filter.TestFilterTwo</filter-class>
  <init-param>
   <param-name>data</param-name>
   <param-value>Hello</param-value>
  </init-param>
 </filter>
 <filter-mapping>
  <filter-name>F2</filter-name>
  <url-pattern>/sample</url-pattern>
 </filter-mapping>
</web-app>
```

=>Enter bellow URL in browser:

Case#1: http://localhost:3037/ServletFilter/sample

Case#2: http://localhost:3037/ServletFilter20/sample?Hello

Servlet Listeners:--

A Listeners is a code (class-with-method) which will be executed by container automatically on a event is created(fired) for an action.

Action:-- Doing is a task (execute method).

Event:-- On Action “Changing State of an Object” (ex: created=>Destroyed)

Listener:-- On event creation do some extra work.

=>ServletListeners are auto executable codes for ServletEvents.

=>All Events are classes and all Listeners are Interfaces.

Scope:-- It indicates time period of data in memory.

*****Servlet Scopes:--**

1>Request Scope 2>Session scope 3>Context Scope

=>These scope are used to store the data in memory in key=value format also known as Attributes.

(Attribute name and Attribute value)

=>These scopes are used to store data and share data between Servlets

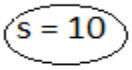
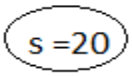
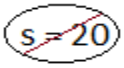
a>Request Scope:-- Used between two servlets

b>Session Scope:-- Used between multiple (not all) servlets between login to logout time.

c>Context Scope:-- Used between all servlets server start to stop.

Scope	**Interface**
Request Scope	ServletRequest
Session Scope	HttpSession
Context Scope	ServletContext

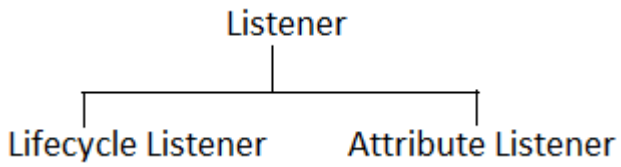
State of Object in servlet:--

Object	Action Event
(new) 	?Creating Obj Created*
setAttribute 	?Adding Attribute Attribute Added *
setAttribute 	?modifying Attribute Attribute Modifyed*
removedAttribute 	?removing Attribute Attribute removed
destroyed 	?destroying Obj Obj destroyed

NOTE:--Listeners are used to execute extra work when events are performed (fired).

=>Servlets Supports 2 Types of Listeners

1>Lifecycle Listener 2>Attribute Listener



A>Life Cycle Events/Listeners:--

Action	Event(C)	Listener(I)	Method
ServletRequest Created/destroyed	ServletRequestEvent	ServletRequestListener	requestInitialized() requestDestroyed()
HttpSession Created/destroyed	HttpSessionEvent	HttpSessionListener	sessionCreated() sessionDestroyed()
ServletContext Created/destroyed	ServletContextEvent	ServletContextListener	contextInitialized() contextDestroyed()

B>Attribute Events/Listeners:--

Action	Event	Listener	Method
Request Attribute Added/removed/modified	ServletRequest AttributeEvent	ServletRequest AttributeListener	attributeAdded() attributeRemoved() attributeReplaced
SessionAttribute added/removed/modified	HttpSession BindingEvent	HttpSession AttributeListener	attributeAdded() attributeRemoved() attributeReplaced
ContextAttribute added/removed/modified	ServletContext AttributeEvent	ServletContext AttributeListenre	attributeAdded() attributeRemoved() attributeReplaced()

C>Special Events : on HttpSession:--

Action	Event	Listener	Methods
HttpSession store/remove objects	HttpSession BindingEvent	HttpSession BindingListener	valueBound(), valueUnbound()
HttpSession Activated/deactivated	HttpSessionEvent	HttpSession ActivationListener	sessionDidActivate(), sessionWillPasivate()

Working with Listeners code:--

Step#1:--Write one public class with any name and any package under src folder

Step#2:--Incase of LifeCycle process uses below any one interface and implements 2 methods

a>ServletRequestListener

b>HttpSessionListener

c>ServletContextListener

or

=>Incase of Attribute process use below any one interface and implement 3 methods.

a>ServletRequestAttributeListener

b>HttpSessionAttributeListener

c>ServletContextAttributeListener

Step#3:--After writing class code, configure those in web.xml using below format.

```
<listener>
   <listener-class>……</listener-class>
</listener>
```

HttpSessionBindingListener:--

=>This Listeners are used to work on special type of classes, when these classes Object added to session or removed from session.

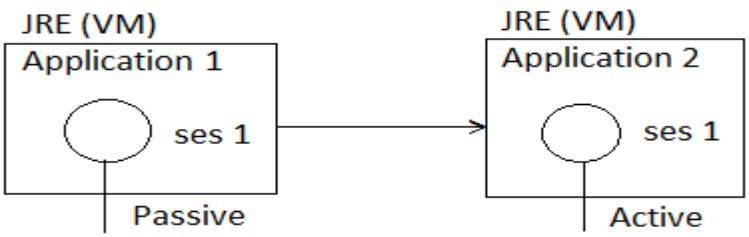
Ex:--Employee class is provided by programmer, for this create object and add to session or removed from session need extra automatic coding for this use above listener.

Q>Difference between HttpSessionAttributeListener and HttpSessionBindingListener?.

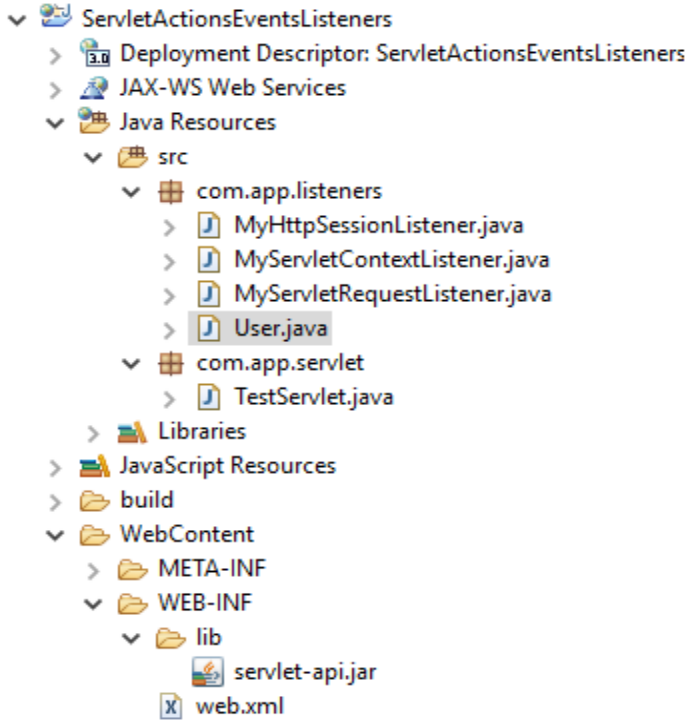
Ans:--AttributeListener:--works for any data(Primitive, collections, any type objects).

BindingListener:--Only for selected class (class should implement interface)

HttpSessionActivationListener:-- If one HttpSession is moved from one VM to another VM (Virtual Machine) then session will be passive in one VM and active in another machine.

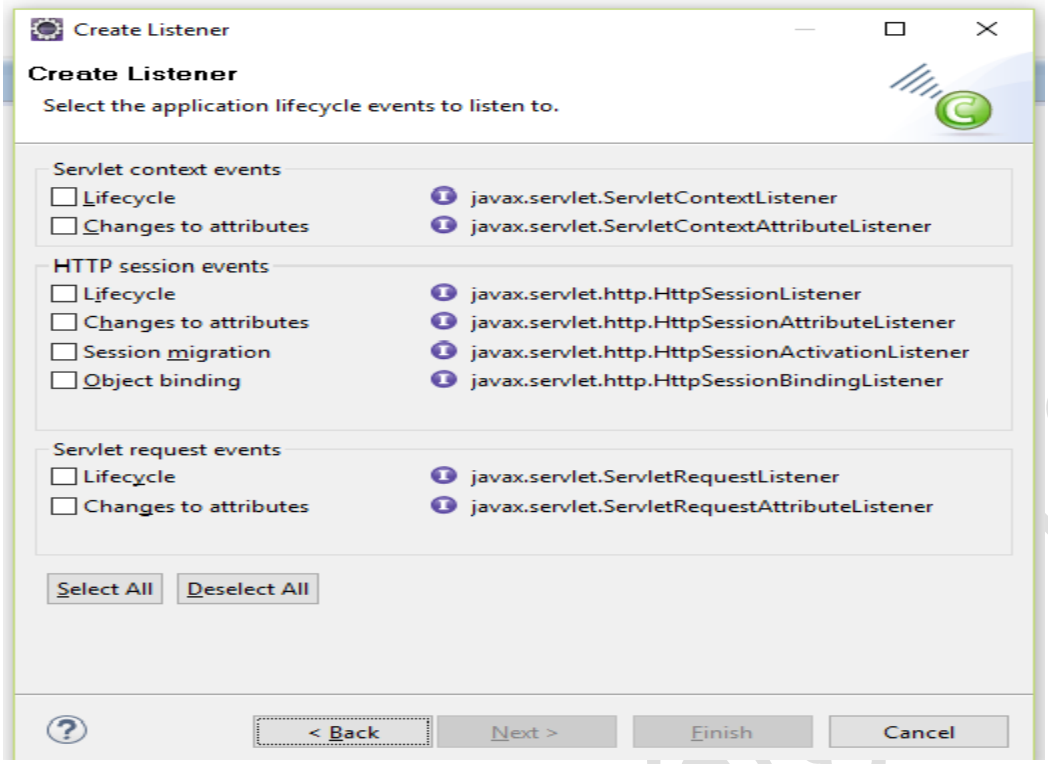


Folder Structure:--



Q>How to create a ServletLister?.

=>Click on File =>New=>Listener=>Fill package, class name=>select (click on any check box according to your requirement) =>Finish



Code:-

1>web.xml:--

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>ServletActionsEventsListeners</display-name>
 <servlet>
  <servlet-name>TestServlet</servlet-name>
  <servlet-class>com.app.servlet.TestServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>TestServlet</servlet-name>
  <url-pattern>/test</url-pattern>
 </servlet-mapping>
 <listener>
  <listener-class>com.app.listeners.MyHttpSessionListener</listener-class>
```

```
 </listener>
 <listener>
  <listener-class>com.app.listeners.MyContextListener</listener-class>
 </listener>
 <listener>
  <listener-class>com.app.listeners.MyServletRequestListener</listener-class>
 </listener>
</web-app>
```

2>Servlet Class:--

```
package com.app.servlet;
import java.io.IOException;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import com.app.listeners.User;
public class TestServlet extends HttpServlet
{
 protected void doGet(javax.servlet.http.HttpServletRequest req, HttpServletResponse
res) throws ServletException, IOException {
        System.out.println("--Request Attribute Related events--");
        req.setAttribute("user", "ajay");
        req.setAttribute("user", "vijay");
        req.removeAttribute("user");

        System.out.println("====================================");
        System.out.println("--Session Attribute Related events--");
        HttpSession session=req.getSession();
        session.setAttribute("admin", "uday");
        session.setAttribute("admin", "varun");
        session.removeAttribute("admin");
```

```
        System.out.println("--Session Bind Related Events--");
        session.setAttribute("myobj", new User("SATHYA"));
        session.removeAttribute("myobj");
        session.invalidate();
        System.out.println("====================================");

        System.out.println("--Context Attribute Related events--");
        ServletContext context=req.getServletContext();
        context.setAttribute("master", "sam");
        context.setAttribute("master", "ram");
        context.removeAttribute("master");
        System.out.println("====================================");
 }
}
```

3>Request Listener:--

```
package com.app.listeners;
import javax.servlet.ServletRequestAttributeEvent;
import javax.servlet.ServletRequestAttributeListener;
import javax.servlet.ServletRequestEvent;
import javax.servlet.ServletRequestListener;
public class MyServletRequestListener implements
ServletRequestAttributeListener,ServletRequestListener
{
//attribute specific
   public void attributeRemoved(ServletRequestAttributeEvent event)  {
        System.out.println("Request Attribute Removed
:name="+event.getName()+",value="+event.getValue());
   }

   public void attributeAdded(ServletRequestAttributeEvent event)  {
        System.out.println("Request Attribute created
:name="+event.getName()+",value="+event.getValue());
```

```
  }
  public void attributeReplaced(ServletRequestAttributeEvent event)  {
        System.out.println("Request Attribute Replaced
:name="+event.getName()+",value="+event.getValue());
  }
  //life cycle

 public void requestDestroyed(ServletRequestEvent event) {
        System.out.println("Request Destroyed");
 }

 public void requestInitialized(ServletRequestEvent event) {
        System.out.println("Request Initialized");
 }
}
```

4>Session Listener:--

```
package com.app.listeners;
import javax.servlet.http.HttpSessionAttributeListener;
import javax.servlet.http.HttpSessionBindingEvent;
import javax.servlet.http.HttpSessionEvent;
import javax.servlet.http.HttpSessionListener;

public class MyHttpSessionListener implements
HttpSessionAttributeListener,HttpSessionListener
{
 public void attributeAdded(HttpSessionBindingEvent event)  {
        System.out.println("Session Attribute Added
:name="+event.getName()+",value="+event.getValue());
  }

  public void attributeRemoved(HttpSessionBindingEvent event)  {
        System.out.println("Session Attribute Removed
:name="+event.getName()+",value="+event.getValue());
```

```
  }
  public void attributeReplaced(HttpSessionBindingEvent event)  {
        System.out.println("Session Attribute Replaced
:name="+event.getName()+",value="+event.getValue());
  }
 public void sessionCreated(HttpSessionEvent event) {
        System.out.println("Session Created");
 }

 public void sessionDestroyed(HttpSessionEvent event) {
        System.out.println("Session Destroyed");
 }
}
```

5>Context Listener:--

```
package com.app.listeners;
import javax.servlet.ServletContextAttributeEvent;
import javax.servlet.ServletContextAttributeListener;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.annotation.WebListener;

/**
 * Application Lifecycle Listener implementation class MyServletContextTestListener
 *
 */
@WebListener
public class MyServletContextListener implements ServletContextAttributeListener,
ServletContextListener {

 public void attributeAdded(ServletContextAttributeEvent event)  {
```

```
        System.out.println("Context Attribute Added
:name="+event.getName()+",value="+event.getValue());
   }

   public void attributeRemoved(ServletContextAttributeEvent event)  {
        System.out.println("Context Attribute Removed
:name="+event.getName()+",value="+event.getValue());
   }
   public void attributeReplaced(ServletContextAttributeEvent event)  {
        System.out.println("Context Attribute Replaced
:name="+event.getName()+",value="+event.getValue());
   }
   //life cycle
 public void contextDestroyed(ServletContextEvent event) {
        System.out.println("Context Destroyed");
 }
 public void contextInitialized(ServletContextEvent event) {
        System.out.println("Context Initialized");
 } }
```

6>Session Binding Listener:--

```
package com.app.listeners;
import javax.servlet.annotation.WebListener;
import javax.servlet.http.HttpSessionBindingEvent;
import javax.servlet.http.HttpSessionBindingListener;

/**
 * Application Lifecycle Listener implementation class User
 *
 */
@WebListener
public class User implements HttpSessionBindingListener
{
private String userData;
```

```
  public User(String userData) {
     this.userData=userData;
  }   public void valueBound(HttpSessionBindingEvent event) {
        System.out.println("Session Bound:"+event.getName());
 }
 public void valueUnbound(HttpSessionBindingEvent event) {
        System.out.println("Session Unbound:"+event.getName());
 }
 public String getUserData() {
        return userData;
 }
 public void setUserData(String userData) {
        this.userData = userData;
 }
 }
```

=>Run as run on server and pass below URL in browser:

Servlets Annotations (Servlet 3.x):--

=>Servlet Annotations are introduced in Servlet API 3.x version (Tomcat version >=7.x).

=>These all annotations are provided in package javax.servlet.annotation.

=>In servlets annotations are provided to reduce XML coding (web.xml).

#1:--@WebServlet:-- This is used for servlet configuration supports URL Pattern, load on startup, init-param etc.

=>Writing this annotation is equals to:--

```
<servlet>
   <servlet-name>….</servlet-name>
   <servlet-class>…..</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>….</servlet-name>
 <url-pattern>…. </url-pattern>
</servlet-mapping>
```

Ex:- Code:--

```
package com.app.servlet;
//ctrl+shift+O (Imports)
@WebServlet(value="/test", loadOnStartup=1, initParams
={@WebInitParam(name="usr", value="uday")})
public class SampleTest extends HttpServlet {….}
```

#2>WebFilter:-- It is used to avoid XML configuration for Filter

Ex:- Code:--

```
@WebFilter("/test")
public class MyTestFilter implements Filter {…}
```

#3>@WebListener:-- It is used to avoid XML Configuration for Listener

Ex:- code:-- @WebListener

```
public class Test implements ServletContextListener{ …}
```

NOTE:--

1>Compared to XML (web.xml) Annotations are less in code, faster in executation, Error Rates is less.

2>Annotation mistakes are show at compile time only. But XML mistakes are shown at runtime.

3>Minimum JDK version is : 6.X

Tomcat is: 7.X, Servlets is : 3.X

FB: https://www.facebook.com/groups/thejavatemple/

SERVLETS END

JSP

JSP (Java Server Pages):--

=>To develop dynamic web pages using Java “Servlets” are used. But servlets programming is more complicated because.

1.>Need Good Core Java skilled.

2>Coding is lengthy.

3>Write HTML code in Java file.

4>Programmer has to compile and configure in web.xml manually.

5>Applying CSS/Java Script or any UI technologies is complicated.

=>In This case .net-ASP and PHP are become most popular.

=>So, Sun Microsystem has provided an Easy Programming to write Dynamic Web Pages, known as “JSP”.

=>Using JSP provides flexibilities like

1>Easy to Java (More HTML code and less Java code).

2>Easy to Apply UI Technologies, like CSS and Java Script.

3>Easy to maintain (No manual compilation or web.xml configuration required).

4>***Using JSP with JSTL has become Zero Java coding = Easy Web page Design.

5>Compared to ASP and PHP, JSP is very fast in coding and execution.

JSP Phases and Lifecycle methods:--

1>Easy to write JSP file by programmer, It is a servlet internally.

2>Programmer should write JSP file as Tag based (like HTML file) and save with “/jsp” extension.

Ex:--Home.jsp

3>This JSP file handover to JSP Engine which is part of servlets.

4>Here .jsp file converted to .java format as a servlet by JSP Engine. It is called as Translation.

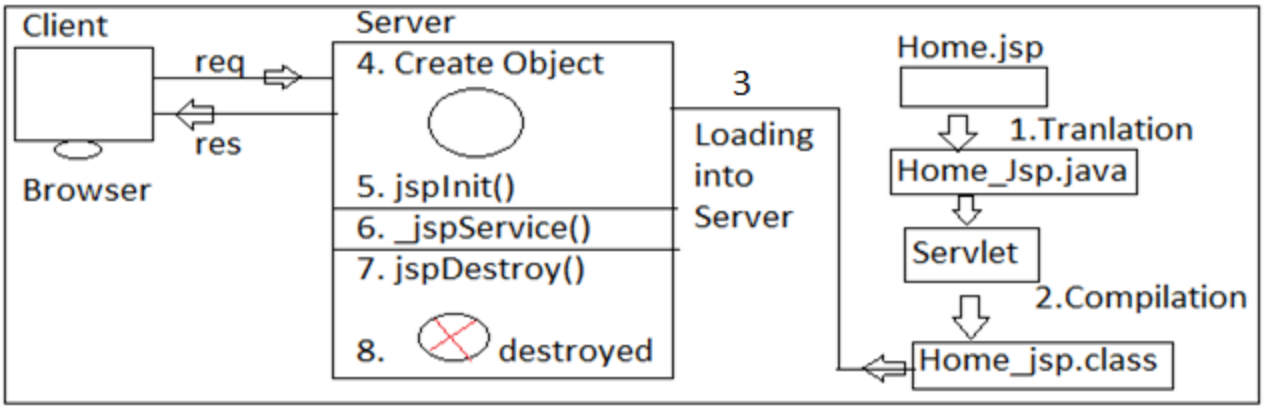
Ex:-- Home_jsp.java

5.>JSP Engine converts .java(servlet) to .class file.

Ex:--Home_jsp.class

6>”.class” is loaded into server by JSP Engine.

7>Servlet Container creates Object to Generated class and calls init method once. Named as “jspInit()”
8>When browser/Client made request, then service method is executed Named as “_jspService()”.
=>This method gives response back.
9>Finally destroy method is called before destroying object, named as “jspDestroy()”.
10>Servlet container destroy the object.



=>In JSP File we should write everything as Tag format only. We can write HTML Tags and JSP Tags.
=>JSP Tags are again divided into 4 types.

1>Scripting Tags:--
=>It is used to write Java code, like variables, methods, block, inner classes, printing statements etc.

2>Directive Tags:--
=>It is used to Link other sources with this file. Like import statements, link with other JSP, Session linking, error pages etc.

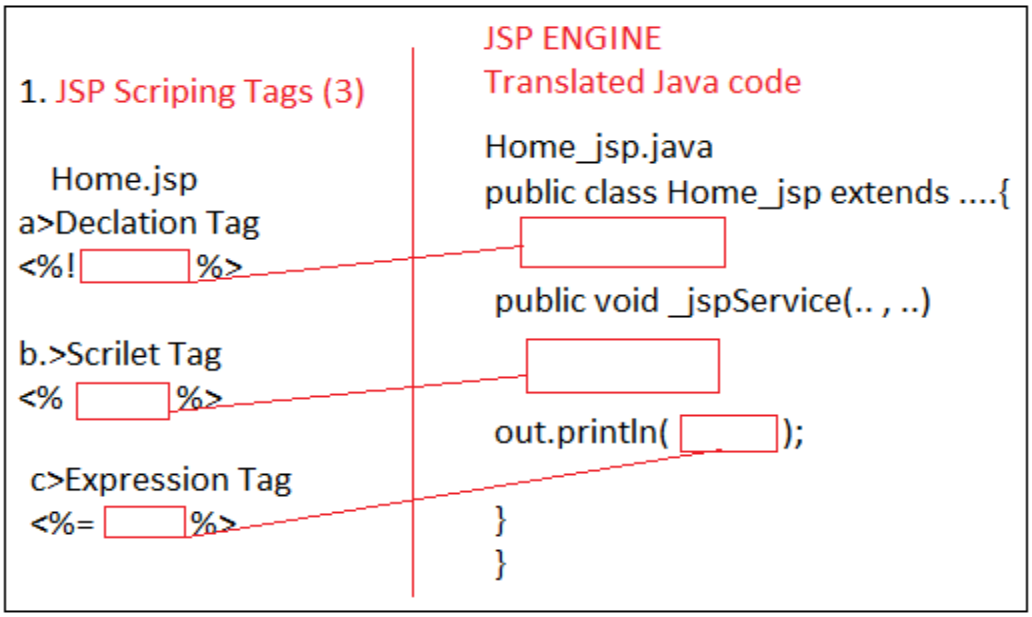
3>Action Tags:--It is used to perform operations over data and flow.
4>Custom Tags:--It is used to create your own tags, few special tags are provided by JSP only called as JSTL (JSP Standard Tag Library)

1.>SCRIPTING TAGS:--These are used to insert java code in JSP file as Tags given in 3 types.

a>Declration : <%! %>:--To insert code inside class and outside service method.

b>Scriplet <% %>:--To insert code inside service method and outside println().

c>Expression <%= %>:--To insert code inside println().



---

a>Declaration : <%! %>:--

=>This tag is used to insert java code inside class and outside all methods.

Valid code:--

1.>instance or static variables

2>instance or static methods

3>instance or static blocks

4> Constructors [no use]

5>member (instance) or static inner class

Invalid Code:--

1.>Only writing Statements

Ex:- System.out.println()

2>only imports

E:--import javax.servlet;

3>only writing Literals
Ex:-- “Hello”, 33

Ex:--
<!%
int sid=0;
void show(){…}
%>
Result : Valid
2. <%!
public static final String a=”A”;
%>
Result :-
3><%!
Int sid=show();
public int show() { return 10;}
%>
Result : Valid
4><%!
System.out.println(“HI”);
%>
Result: Invalid
5><%!
class Hello {
void show{ }
%>
Result : Valid
6><%!
  {
     System.out.println(“Hello”);
  }
%>
Result : Valid

7><%!
static {
String s = Hello";
System.out.prinln(s);
}
%>
Result: Valid [s is local Variable]
8><%!
String s ="Hello";
int a =s.length();
%>
Result : Valid
9><%!
A a = new A();
class A{ }
%>
Result :-Valid
10><%! %>
Result : Valid

b>Scriplet : <% %>:--

This is code is inserted in _jspService() method. [Like code inserted inside method and outside println()]

valid Codes:--
a.>local variables
b.>Statements
c.>block [but not static]
d.>writing loops, conditional statements
e.>calling methods
f.>writing class [local class]

Not Valid:--
a.>nesting of methods [writing one method in another method]
b>static variables or blocks.

Examples:--
1.><%
System.out.println("Hl");
%>
Result: Valid
2.><%
for(int i=0; i<=10; i++) {….}
%>
Result: Valid
3><%
Date d = new Date();
System.out.println(d);
%>
Result: Valid
4>><%
public static final int a=10
System.out.println(a);
%>
Result: InValid
5>><%
Int a =10;
System.out.println(a);
%>
Result: Valid

6><%
A a = new A();
a.show();
%>

Result: Valid

7><%

Class Sample{ }

%>

Result: Valid

8><%

Interface A { }

A a = new A() { }

%>

Result: Valid

c.>Expression Tag <%= %>:--

=>Code written inside this is printable on browser screen it is like: out.println(-)

Valid code:--

a>Any literal (without semi-column).

Ex:-- 10, “Hello”, ‘A’, -8.9

b>calling method returns non-valid

ex:--int show() { } is exist

ob.show(); [Valid]

c>printing variables, object references Collection variable.

InValid code:--

a>Writing statements.

b>declaring variables.

c>Calling void methods.

d>***Writing semicolons.

Examples:--

1><%=”Hello” %>==>[Valid]

2><%=10; %>==>[InValid]

3>assume show() returns int type

<%=ob.show() %>[valid]

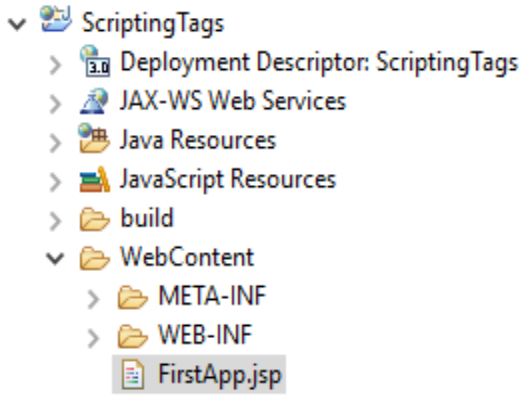
4>assume list is references variable of java.util.ArrayList.
<%=list%> [Valid]
<%=new java.util.Date()%> [Valid]]

5>assume show() returns void type
<%=ob.show()%> ====>[Invalid]

First Example:--
#1>Create one new dynamic web project
=>file =>new=>Dynamic Web Project
=>Enter name: FirstApp
=>choose also
Target Runtime : Apache Tomcat
Dynamic web module version : 2.5
=>next=>next=>finish

#2>Create JSP file under Root folder
Name: WebContent
=>Right click on WebContent=>new=>JSP File
=>Enter name ex: FirstApp.jsp
=>Finish.

#3>Add “servlet-api.jar” in lib folder.
#4>Write code inside<body> </body> tag
Folder Structure:--



FirstPage.jsp

```
<html><body>
<%! int count=0; %>
<%count++; %>
<%= "No of views: "+count%>
</body></html>
```

#4>Run Application in server

=>Right click on Project=>Run as => Run on server and

Enter URL like: http://localhost:2018/ScriptingTags/FirstApp.jsp

Implicite Objects in JSP(9):--

=>In case of Servlets objects code must be written by programmer, but coming to jSP, it is providing as pre-define named as Implicit (Created by container not by programmer). Those are.

1.	out	JspWriter
2.	request	HttpServletRequest
3.	response	HttpServletResponse
4.	session	HttpSession
5.	config	ServletConfig
6.	application	ServletContext
7.	page	Object(java.lang)
8.	pageContext	PageContext
9.	exception	Throwable

PageContext :--

It is a global object created by WebContainer which holds other objects (implicite) behaves like holder. It provides auto set/get methods to use them in code. Directly cannot be used by Programmer.

1>out:--This object is used to print the data on UI Page. It is type of JspWriter we can use this object in script.

Ex:--

```
<%
    out.println("Hello");
%>
```

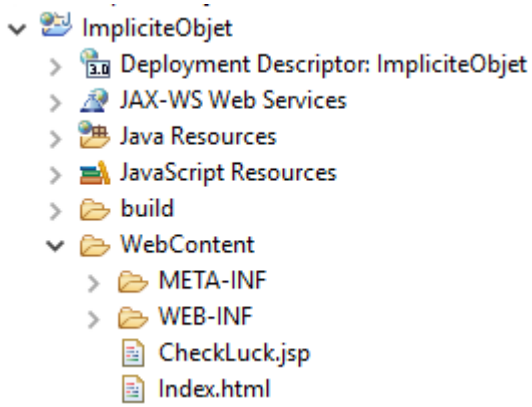
Which is equal to:--

```
<%="Hello"  %>
```

Ex#2:-- Using Implicite Objects----

Create all html, JSP files under WebContent folder.

Folder Structure:--



Index.html (under WebContent)

```
<html>
<body>
<form action="CheckLuck.jsp" method="post">
<pre>
Enter Number (Between 0-10):
<input type="text" name="num"/><br>
<input type="submit" value="Find Luck"/>
</pre></form>
</body>
</html>
```

CheckLuck.jsp (under WebContent)

```
<html><body>
<%
	String ob =request.getParameter("num");
	int num= Integer.parseInt(ob);
	java.util.Random r = new java.util.Random();
	int gnum=r.nextInt(10);
	if(num==gnum)
{
	out.print("You won 10L Rupes");
}
else
{
	out.print("Sorry!");
}
%>
</body></html>
```

=>Run on server Enter URL like: http://localhost:3030/ImpliciteObjet/Index.html

Ex#3:-- Define non void methods and call--

Calculate.jsp (Under WebContent)

```
<body>
<%!
	int add(int a, int b) {
	return a+b;
  }
%>
<%=add(10, 20) %>
</body></html>
```

2.>Directive Tags:--These tags are used to link other resources (like import, include etc).

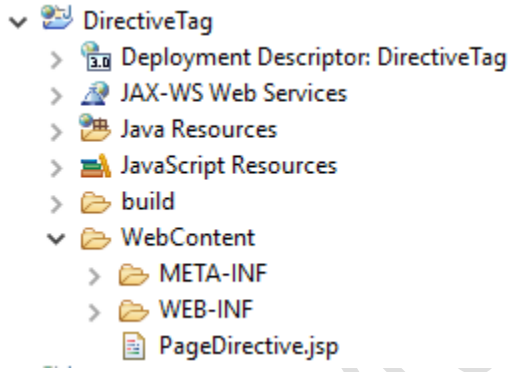
Syntax:--<%@ %>

These are given as 3 types.

a>page Directive
b>include Directive
c>taglib Directive

a.>page Directive:-- It is used to provide JSP Page level links, few are given as
=>contentType
=>session
=>import
=>isELIgnored
=>isErrorPage
=>pageEncoding
Etc…

Folder Structure:--



Example #1: PageDirective.jsp (Under WebContent):--

```
<%@page import="java.util.Date"%>
<html><body>
<%
Date d = new Date();
out.println(d);
response.addHeader("Referesh", "1");
%>
</body></html>
```

=>Run on server & Enter URL : http://localhost:3030/DirectiveTag/PageDirective.jsp

Example#2:--JSP with JDBC:--

Step#1: Create one table in Database.

SQL:-- create table userdata(un varchar2(25), pwd varchar2(25));

Step#2:--Insert few records in DB and commit

INSERT INTO userdata VALUES ('uday', 'kumar');

INSERT INTO userdata VALUES ('ram', 'ram');

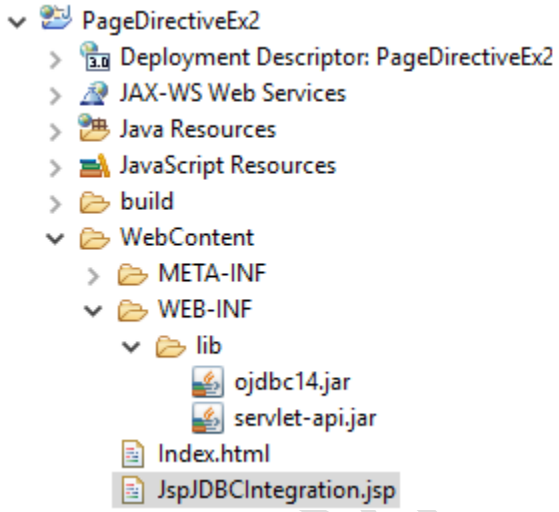
commit;

Step#3:- Copy DB jar in project lib folder

Ex:--ojdbc.jar into /WebContent/WEB-INF/lib

Step#4:-Create index.html under WebContent

Folder Structure:--



Code:-Index.html

```
<html><body>
<form action="JspJDBCIntegration.jsp" method="post">
<pre>
User Name: <input type="text" name="user"/><br>
Password : <input type="password" name="pass"/>
<input type="submit" value="Login"/>
</pre></form>
</body></html>
```

Step#5:-- Create JspJDBCIntegration.jsp under WebContent

Code: Create JspJDBCIntegration.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>

<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>JSP with JDBC Integration</title>
<body>
<%!
   //Writing methods with logic
public Boolean checkUser(String un, String pwd)
{
boolean isExist=false;
try {
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE",
"system","system");
PreparedStatement pstmt = con.prepareStatement("select * from userdata where un=?
and pwd=?");
pstmt.setString(1,un);
pstmt.setString(2,pwd);

ResultSet rs = pstmt.executeQuery();
if(rs.next())
{ /*  user exist */
isExist=true;
}
Else {
//not Exist
isExist=false;
```

```
}}
catch(Exception e)
{
System.out.println(e);
}
return isExist;
}
%>

<%   //read data from HTML Form
String   un= request.getParameter("user");
String   pwd = request.getParameter("pass");
//calling method
boolean isExist = checkUser(un, pwd);
//using if-else
if(isExist)
{
out.println("Welcome to user: "+un);
}
else
{
out.print("Invalid Login Details!!");
}
%>
</body></html>
```

=>Run on server=>Enter URL like: http://localhost:3030/PageDirectiveEx2/Index.html

b.>include directive tag:--

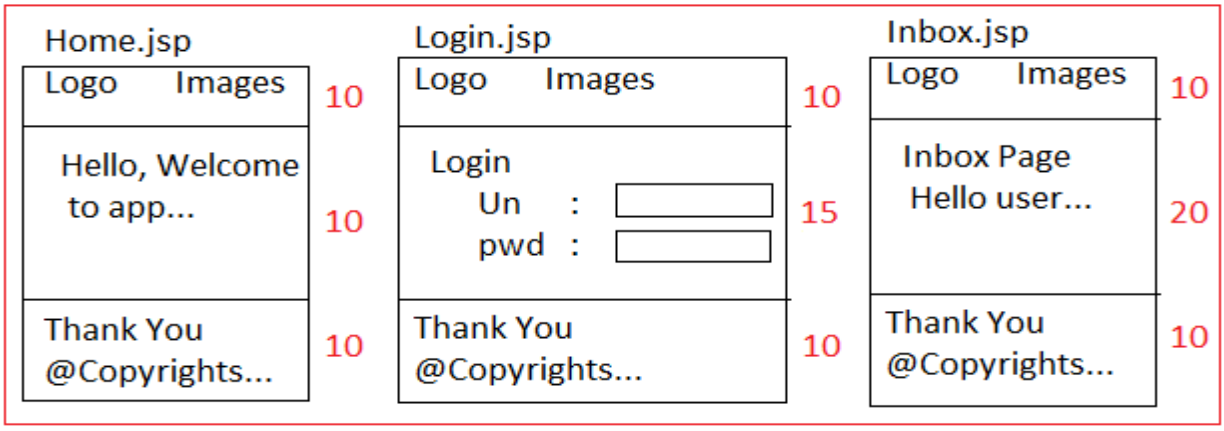
[Format : <%@include file="---"%>]

=>It is used to include a resource to current JSP file. It can be HTML, JSP or any text file.

=>It is also called s static include. It means same content is coped to JSP first then translated into .java file.

=>Finally one .java (.class).

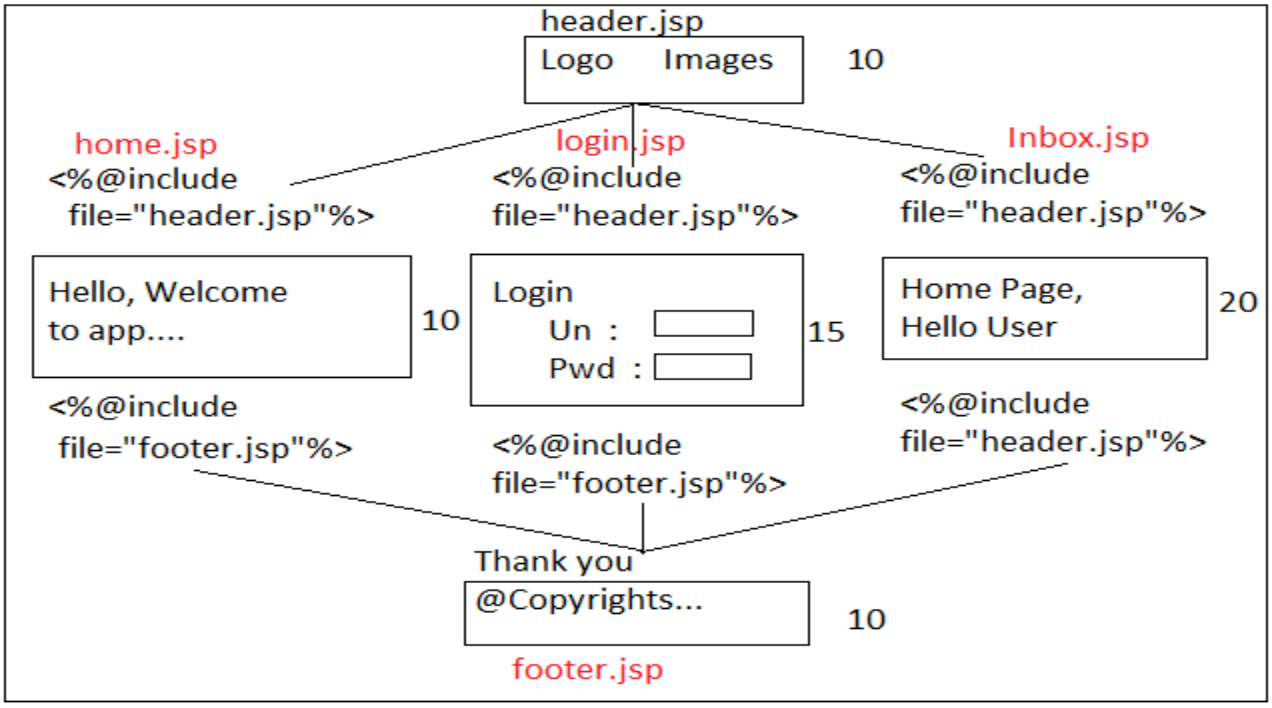
=>It is used mainly in common lines of code re-using concept. Example considers below different JSP pages having common lines at top and bottom of JSP.



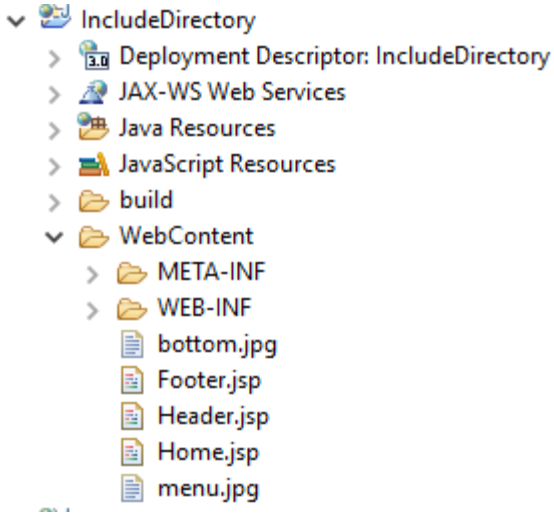
=>In above JSP pages Top 10 lines of code is common in every page, so move to Header.jsp and link this in every JSP using code.

< %@include file="header.jsp"%>

Now design will be:--



Folder Structure:--



Code:--

#1>Header.jsp:--

```
<img src ="menu.jpg"/>
<h3>This is A Header Page</h3>
```

#2>Footer.jsp:--

```
<img src="bottom.jpg"/>
<h3>This is Footer Page</h3>
```

#3>Home.jsp:--

```
</head><body>
<h1>Welcome to Home page to all </h1>
<%@include file="Header.jsp"%>
<%@include file="Footer.jsp"%>
</body></html>
```

=>Run on server=>Enter URL like: http://localhost:3030/IncludeDirectory/Home.jsp

pageContext:--

=>It is one of implicit object in JSP which holds all page related data and other objects internally. Provides them to them to page based on requirement.

=>It also store data using different scopes in memory as attributes.

SCOPE:-- It indicates time period of data in memory. This concept is used to specify life time of data and where it can be accessed (shared).

=>Servlet has 3 scopes. Those are:

a>Request scope:--To share data between two servlets.

b>Session scope:--To share data between multiple servlets from login to logout.

c>Context scope:--To share data to all servlets, from server start to stop.

=>JSP supports 4 scopes. Those are

a>Page scope:--To share data within same JSP. It is like a local variable.

b>Request scope:--To share data between two JSP pages.

c>Session scope:--It is used to share data between multiple JSP files, from login to logout time.

d>Application scope:--It is used to share to all JSP, also called as Global scope.

NOTE:--

1>Data will be stored in key = value format.

2>It is also called as Attribute methods used here : setAttribute() and getAttribute().

3>scope order is

Page < Request < Session < Application

4>Highest scope is : Application and

Lowest scope is : Page

5>pageContext is a class holds all scope levels (given below) App scope variables are public static final.

Type	ID	CONSTANT
1>Page	1	pageContext.PAGE_SCOPE
2>Request	2	pageContext.REQUEST_SCOPE
3>Session	3	pageContext.SESSION_SCOPE
4>Application	4	pageContext.APPLICATION_SCOPE

6>pageContex also supports RequestDispatcher operations: forward (URL) and include (URL).

pageContext purpose/Advantages:--

1>It holds all remaining object (implicit object).

2>pageContext supports scope (4).

3>pageContext supports Request Dispatcher .forward and .include.

=>example attribute data setting and getting using different syntaxes.

1>PAGE SCOPE:--Add data to memory in 3 ways, using syntaxes.

a>pageContext.setAttribute(“user”, “UDAY”);

b>pageCOntext.setAttribute(“user”, “UDAY”, 1)

c>pageCOntext.setAttribute(“user”, “UDAY”, PageContext.PAGE_SCOPE);

=>To read data from memory using syntaxes:-

a>pageContext.getAttribute(“user”);

b>pageContext.getAttribute(“user”,1);

c>pageContext.getAttribute(“user”,PageContext.PAGE_SCOPE);

2>REQUEST SCOPE:--Add data to memory in 3 ways, using syntaxes.

a>request.setAttribute(“uid”,”U123”);

b>pageCOntext.setAttribute(“uid”, “U123”,2);

c>pageContext.setAttibute(“uid”,”U123”, pageContext.REQUEST_SCOPE);

=>To read data from memory using syntaxes:-

a>pageContext.getAttribute(“uid”);

b>pageContext.getAttribute(“uid”,2);

c>pageContext.getAttribute(“uid”,PageContext.REQUEST_SCOPE);

3>SESSION SCOPE:-- Add data to memory in 3 ways, using syntaxes.

a>request.setAttribute(“pwd”,”P123”);

b>pageCOntext.setAttribute(“pwd”, “P123”,3);

c>pageContext.setAttibute(“pwd”,”P123”, pageContext.SESSION_SCOPE);

=>To read data from memory using syntaxes:-

a>pageContext.getAttribute(“pwd”);

b>pageContext.getAttribute("pwd",3);
c>pageContext.getAttribute("pwd",PageContext.SESSION_SCOPE);

4>APPLICATION SCOPE:-- Add data to memory in 3 ways, using syntaxes.
a>request.setAttribute("dob","1878");
b>pageCOntext.setAttribute("dob", "1878",4);
c>pageContext.setAttibute("dob"," 1878", pageContext.SESSION_SCOPE);

=>To read data from memory using syntaxes:-
a>pageContext.getAttribute("dob");
b>pageContext.getAttribute("dob",4);
c>pageContext.getAttribute("dob",PageContext.SESSION_SCOPE);

Expression:--It is a combination of operators and operands which returns finally one value.
Example:-- a+b-c, Her e consider a=10, b=2, c=1
=>a+b-c
=>10+2-1
=>12-1
=>11

EL in JSP:--EL stands for Expression Language. In JSP an expression is a combination of
a>variables (primitives, references, collection/array Types)
b>operators (relations, arithmetic, logical…)
c>Reserve words:-- (key words/ pre-defined words)
d>implicit objects [not all, related to memories-pageContext. request, session application)

=>Combination of these finally should return one value [not a type]
Like:--Primitive value (int, double, Boolean, String, char) or any object, array, collection data also.

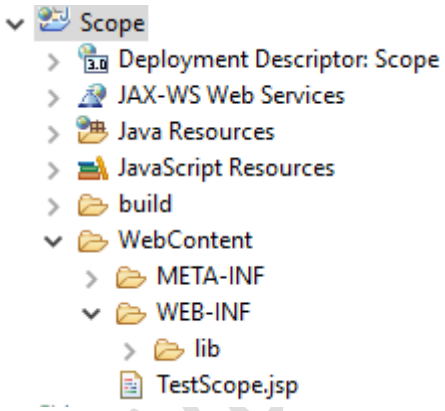
Syntax:-- ${expression}

Example#1:-- ${variable}
=>It will try to read the data from scope memory. To indicate them EL has provided 4 implicit objects.
Given as:

a>pageScope b>sessionScope
c>requestScope d>applicationScope

=>If we write ${sid} as example then Search in page scope if found ok, else (not found) then goto request, next session finally application.
=>If variable is not found in any scope then final value is null, printed as Scope on browser.
=>default checking order is:
page-next=>request-next=>session-next=>application.
=>To avoid this default order and read from one direct scope use EL implicit objects.

Folder Structure:--



JSP#1 TestScope.jsp (under WebContent)—

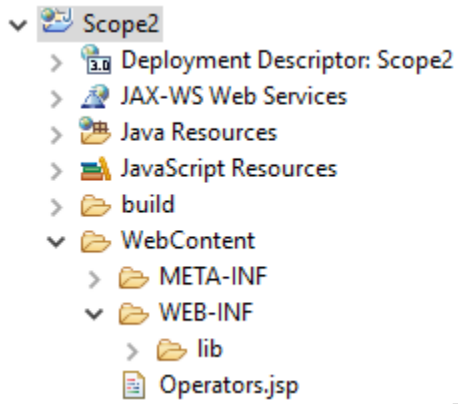
```
<html><body>
<%
  pageContext.setAttribute("myid", 10);
  request.setAttribute("myid", 20);
  session.setAttribute("myid", 30);
  application.setAttribute("myid", 40);
%>
```

```
<pre>
Data is                 : ${myid} or ${pageScope.myid}
Data from Request       : ${requestScope.myid}
Data from Session       : ${sessionScope.myid}
Data from Application   : ${applicationScope.myid}
</pre>
</body> </html>
```

=>Run on server=>Enter URL like: http://localhost:3030/Scope/TestScope.jsp

Example#2 : operators.jsp

Folder Structure:--



Code: Operators.jsp:--

```
</head><body>
<%
pageContext.setAttribute("myid", 500);
%>
<pre>
Data is:  ${myid}
Exp1:    ${myid <= 600} or  ${myid le 600}
Exp2:    ${myid != 600} or  ${myid ne 600}
Exp3:    ${myid ne 400 ? (myid mod 7): (myid div 8)}
Exp4:    ${!empty myid} -- ${empty myid}
Exp5:   ${myid + 20 - myid*2 - myid/3}
</pre> </body></html>
```

=>Run on server=>Enter URL like: http://localhost:3030/Scope2/Operators.jsp

NOTE:--

ne =>not equal to (!=)

eq = equal to (==)

Mod = % (reminder)

Div = /

1)SYNTAXES:

${expression}

2) OPERATORS

Arithmetic:-- +, - (binary), *, / and div, % and mod, - (unary)

Logical:-- and, &&, or, ||, not, !

Relational:-- ==, eq, !=, ne, <, lt, >, gt, <=, ge, >=, le. Comparisons can be made against other values or against Boolean, string, integer, or floating-point literals.

empty:-- The empty operator is a prefix operation that can be used to determine whether a value is null or empty.

Conditional:-- A ? B : C. Evaluate B or C, depending on the result of the evaluation of A.

=>The precedence of operators highest to lowest, left to right is as follows:

[] .

() (used to change the precedence of operators)

- (unary) not ! empty

* / div % mod

+ - (binary)

< > <= >= lt gt le ge

== != eq ne

&& and

|| or

? :

3) EL RESERVED WORDS:--

=>The following words are reserved for the EL and should not be used as identifiers:

and	or	not	eq
ne	lt	gt	le
ge	true	false	null
instanceof	empty	div	mod

4) JSP EL IMPLICIT OBJECTS:--

pageScope	=>Contains attributes with page scope.
requestScope	=>Contains attribute value with request scope.
sessionScope	=>Contains attribute value with session scope.
applicationScope	=>Contains attributes value from application scope.
param	=>To get request parameter value, returns a single value
paramValues	=>To get request param values in an array, useful when request parameter contain multiple values.
header	=>To get request header information.
headerValues	=>To get header values in an array.
cookie	=>Used to get the cookie value in the JSP.
initParam	=>Used to get the context init params, we can't use it for servlet init params.
pageContext	=>Same as JSP implicit pageContext object, used to get the request, session references etc. example usage is getting request HTTP Method name.

5) EXAMPLES OF EL EXPRESSIONS:--

EL Expression	Result
${1 > (4/2)}	false
${4.0 >= 3}	true
${100.0 == 100}	true
${(10*10) ne 100}	false
${'a' < 'b'}	true
${'hip' gt 'hit'}	false
${4 > 3}	true

${1.2E4 + 1.4}	12001.4
${3 div 4}	0.75
${10 mod 4}	2
${!empty param.Add}	False if the request parameter named Add is null or an empty string.
${pageContext.request.contextPath}	The context path.
${sessionScope.cart.numberOfItems}	The value of the numberOfItems property of the session-scoped attribute named cart.
${header["host"]}	The host.
${departments[deptName]}	The value of the entry named deptName in the departments map.

Exaple#3:- Input Query/Request Parameters:--

=> To read request/query parameters in servlets/JSP code is given as:

Syntax:--

String value =request.getParameter(“key”);

This code must be written in scriptlets in case of JSP. To avoid this java code EL has provided 2 implicit objects.

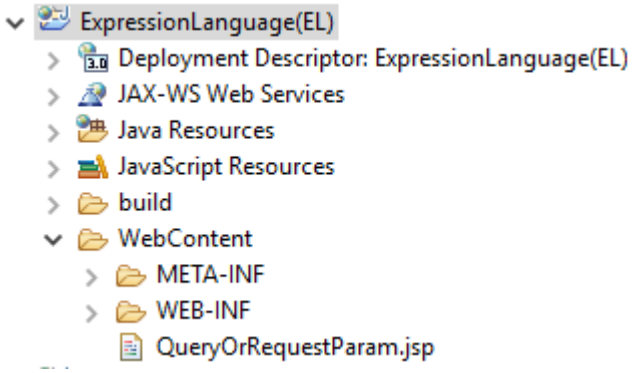
a>param : one key value

b>paramValues : one key multiple values which stores like array.

=>Param data read syntax is:

${param.key} or ${param[‘key’]}

Folder Structure:--



Example: QueryOrRequestParam.jsp (under WebContent)

```
<html>
<body>
<pre>
ID    : ${param.eid} or ${param['eid']}
NAME  : ${param.ename}
SAL   : ${param.esal}
Projects : ${paramValues.proj[0]}, ${paramValues.proj[1]}, ${paramValues.proj[2]}
</pre>
</body>
</html>
```

=>Run in server and Enter URL like:

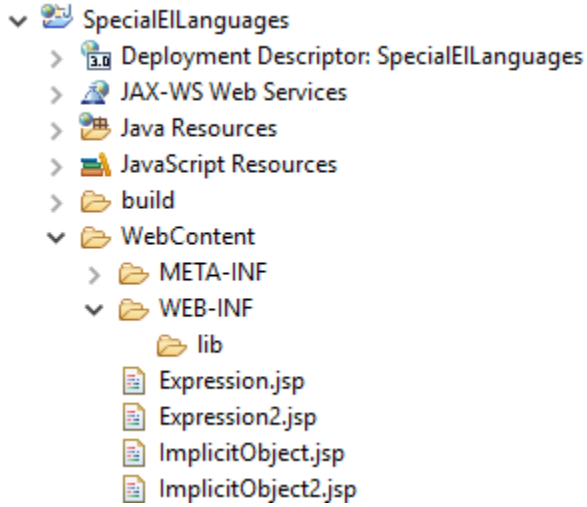
http://localhost:3030/ExpressionLanguage(EL)/QueryOrRequestParam.jsp?eid=5&ename=UDAY&esal=8.4&proj=JSP&proj=Spring&proj=Hibernate

=>Here ${param.eid} is equa to

```
<% String eid = request.getParameters{“eid”);
out.println(eid);
%>
```

Special EL Objects:--

Folder Structure:--



Code:--

1>Expression.jsp:--

```
<html> <body>
        <% request.setAttribute("data", "abc"); %>
<pre>
        Data            : ${data}
        EMPTY?          : ${empty data}
        NOT EMPTY?      : ${!empty data }
        EQUAL TO ABC    : ${data eq 'abc'}   :  ${'Hello' eq 'abc'}
        TERNARI CHECK   : ${data!=null?'YES':'NO'}
        LOGICAL CHECK   : ${data ne null and data eq 'abc'}
</pre> </body> </html>
```

2. Expression2.jsp:--

```
 <html> <body>
     <% request.setAttribute("data", 400); %>
<pre>
        Data            : ${data}
        MOD?            : ${data mod 8}
        DIV?            : ${data div 6}
        EQUAL TO ABC    : ${data eq 300}  : ${300 ge 400}
        LOGICAL CHECK   : ${data ne 80 || data lt 200}
   </pre></body> </html>
```

3.> ImplicitObject.jsp:--

```
<html><body>
        ${param.name}
        ${paramValues.courses[0]}
        ${paramValues.courses[1]}
<hr/>
        ${header.host } or ${header['host'] } all ${header}
        ${headerValues.accept[0]}
        ${pageContext.request.contextPath}
        ${pageContext.request.cookies[0].name}
</body> </html>
```

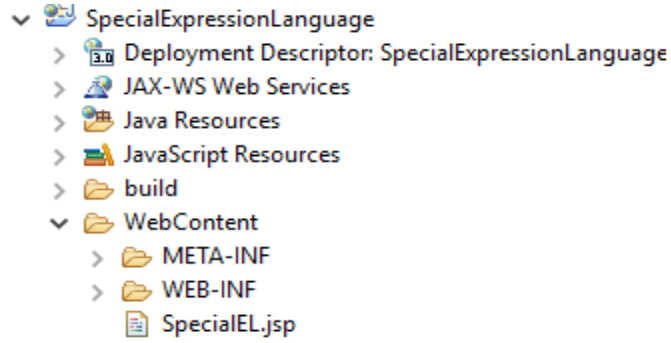
4. >ImplicitObject2.jsp:--

```
<html> <body>
  <%
        pageContext.setAttribute("user", "AJAY-A");
        request.setAttribute("user", "AJAY-B");
        session.setAttribute("user", "AJAY-C");
        application.setAttribute("user", "AJAY-D"); %>
   <pre>
       ${user} or  ${pageScope.user}
       ${requestScope.user}
       ${pageScope.user}
       ${pageScope.user}
   </pre>
</body></html>
```

=>Run on Server and Enter URLs one by one:

1>http://localhost:3033/SpecialElLanguages/Expression.jsp
2>http://localhost:3033/SpecialElLanguages/Expression2.jsp
3>http://localhost:3033/SpecialElLanguages/ImplicitObject.jsp
4>http://localhost:3033/SpecialElLanguages/ImplicitObject2.jsp

Folder Structure:--



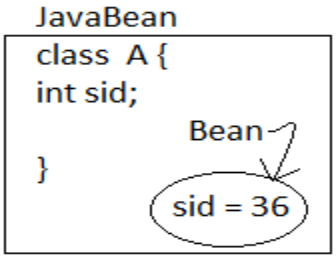
Code: SpecialEL.jsp

```
<html><body><pre>
${header.cookie} or ${header['cookie']}
${header.host} or ${header['host']}
${header['accept_encoding']}
${cookie.JSESSIONID.name}
${cookie.JSESSIONID.value}
${pageContext.request.contextPath}
</pre></body></html>
```

=>Run on Server and Enter URL in browser:
http://localhost:3033/SpecialExpressionLanguage/SpecialEL.jsp

3>JSP Action Tags:--

=>These tags are used to avoid “writing of JAVA code in JSP”. We can also called as pre-defined tags in JSP.
=>Every tag performs one operation (Action).
=>format looks like <jsp:tagName> here prefix “jsp” indicates to servlet container, as “It is not HTML, it is JSP”.
Bean = object-Data.



1.	Action Tag	Usages
2.	<jsp:forward>	=>It works like Request Dispatcher Forward.
3.	<jsp:include>	=>It works like Request Dispatcher Include.
4.	<jsp:param>	=>To provide some parameter while using forward or include.
5.	<jsp:useBean>	=>To use one object holds only one data, to pass object data between two JSPs.
6.	<jsp:setProperty>	=>To add data to bean.
7.	<jsp:getProperty>	=>To read data from bean.
8.	<jsp:plugin>	=>To link with additional services, ex: Applet.
9.	<jsp:fallback>	=>To show error message if service not found.

1><jsp:forward>:-- It is used to transfer execution flow from current JSP (Servlet) to next JSP (Servlet) and it will not come back again.

Syntax:--

```
<jsp : forward page="-----.jsp"/>
```

Ex:--<jsp : forward page="Data.jsp"/>

=>It can also be written in below format: (without using action tag)'

Format #2:-

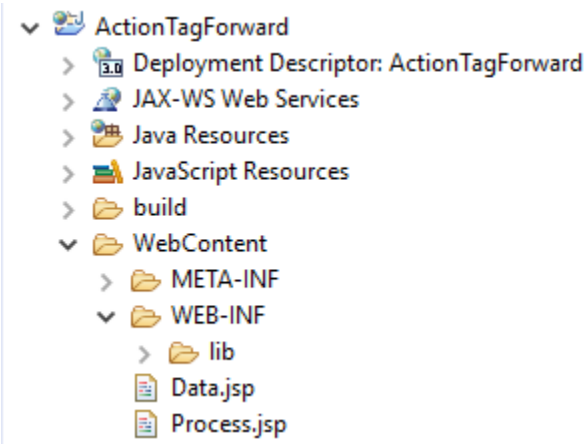
```
<%
    RequestDispatcher rd = request.getRequestDispatcher("Data.jsp");
    rd.forward(request, response);
%>
```

Format#3:--

```
<%
pageContext.forward("Data.jsp");
%>
```

Example:--Application

Folder Structure:--



a.>Process.jsp:--

```
<html><body>
<jsp:forward page ="Data.jsp">
      <jsp:param name="user" value="UDAY"/>
</jsp:forward>
</body></html>
```

b.>Data.jsp:--

```
<html><body>
<h2>Welcome to Data Page</h2>
      <% Object ob=request.getParameter("user");
      out.println(ob); %>
</body></html>
```

=>Run on Server and Enter URL : http://localhost:3033/ActionTagForward/Process.jsp

2.><jsp:include> :-- Use this action tag for RequestDispatcher include operation. We can also send parameters using this.

```
<jsp:include page="Data.jsp":>
      <jsp:param name="user" value="ABC"/>
</jsp:include>
```

Java Bean:--Java Bean is a class. It is used to create object which holds data which can be shared between project multiple classes.

Rules to write Java Bean (Required Rules):--

1>Class must be public type, then container or framework can access this.
2>Must have public default constructor (also called as zero parameter or no-arg constructor), because containers or frameworks creates objects.
3>Variable must type private.
4>Every variable should contain two methods. set, get methods, also called as mutators.

Optional Rules:--

5>Writing package statement,
6>Writing logics (method).
7>class can implements java.io.Serializable(I) Interface.

Java Bean in JSP:--

=>To pass data in object format between multiple JSP file, use Java Bean concept. Here, code (class) is written by programmer by,

>Creating object
>providing data
>reading data
>passing object
>destroy object

=>It is done by web container only.

=>To share this object between JSPs, It must be under one of given scopes.
1>page scope (default scope)
2>request scope
3>session scope
4>Application scope

=>To Work on Java Beans in JSP, we can use 3 action tags.
a. <jsp:useBean>

b. <jsp:setProperty>
c. <jsp:getProperty>

a.<jsp:useBean>:-- This action tag is used to read object from given scope. If object not exist then it will create new Object and placed in same scope.

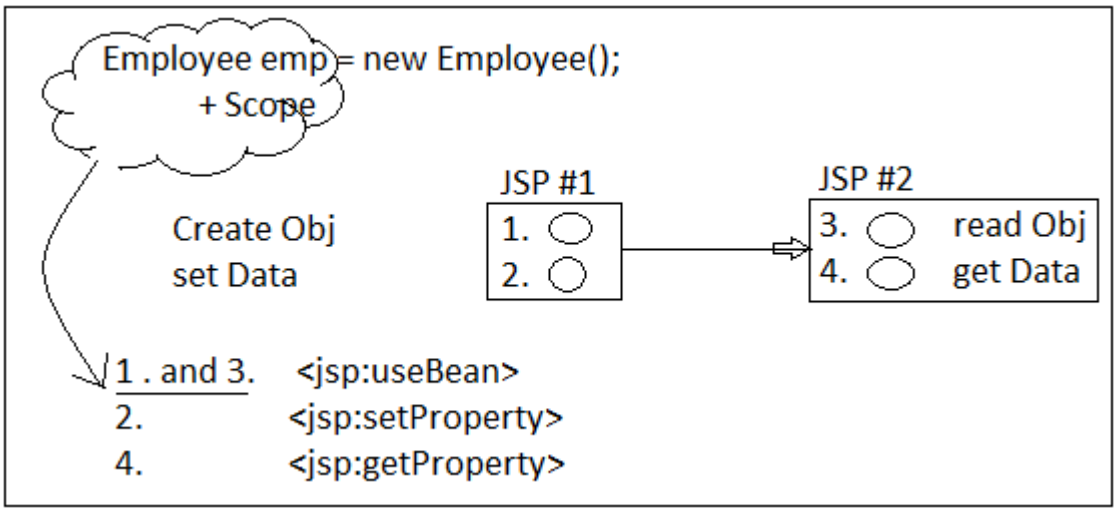
Syntax:--

```
<jsp:useBean   id=”objName” class=”fullyQualifiedClassName”
               Scope=”page/request/session/application”/>
```

Ex:-- <jsp:useBean id =”emp” class=”com.app.Employee” scope=”request”/>

=>Equal internal meaning is:

```
<%
   Employee emp = null;
   emp = request.getAttribute(“emp”);
   If (emp==null)  {
   emp = new Employee();
   request.setAttribute(“emp”, emp);
 }
%>
```



b><jsp:setProperty>:-- This line is used to provide the data to one variable in Bean.

Syntax:--

```
<jsp:setProperty property=”variableName” name=”objetName” value”data”/>
```

Ex:-- <jsp:setProperty property=”empId” name=”emp” value=”100”/>

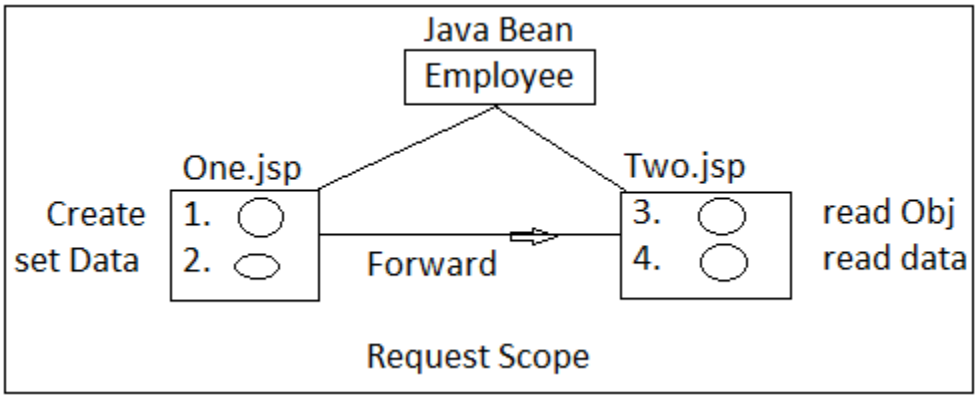
c><jsp:getProperty>:-- This line is used to read the data from given variable from Bean(object).

Syntax:--

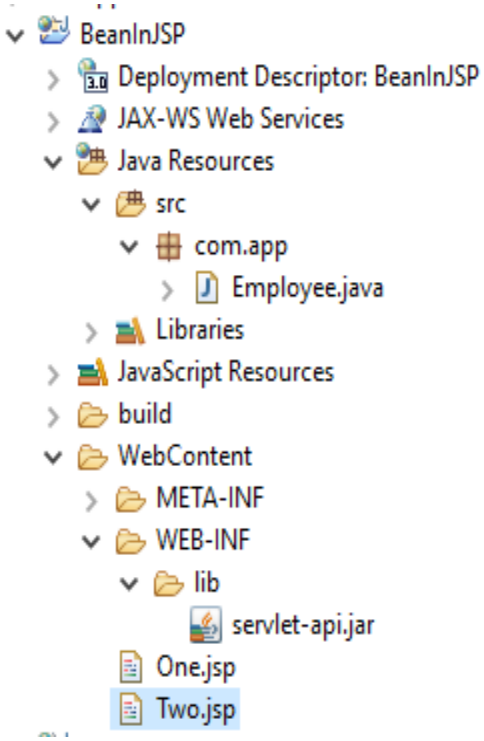
```
<jsp:getProperty property=”variableName” name=”objName”/>
```

Ex:-- <jsp:getProperty property=”empId” value=”emp”/>

=>It is equal meaning is:-- emp.getEmpId();



Folder Structure:--



Code:--

1>Create Employee Java Bean class under src folder:--

```
package com.app;

public class Employee
{
private int empId;
private String empName;
private String empSal;
//defaut constructor
//set,get method
//toString method
}
```

One.jsp:--

```
</head><body>
<jsp:useBean id ="emp" class="com.app.Employee" scope="request"/>
<jsp:setProperty property="empId" name="emp" value="102"/>
<jsp:setProperty property="empName" name="emp" value="Uday"/>
<jsp:setProperty property="empSal" name="emp" value="198.87"/>
<jsp:forward page="Two.jsp"/>
</body></html>
```

Two.jsp:--

```
</head><body>
Data from One jsp is :
<jsp:useBean id="emp" class ="com.app.Employee" scope="request"/>
<jsp:getProperty property="empId" name="emp"/>
<jsp:getProperty property="empName" name="emp"/>
<jsp:getProperty property="empSal" name="emp"/>
</body></html>
```

=>Run on server and Enter URL in browser: http://localhost:3033/BeanIn_JSP/One.jsp

4>CUSTOM TAG:--

TAG:--A Tag is component (code) written in HTML, XML, XHTML,… (UI Technology) which will do some task (Execute one work).

Types of Tags:--

Tags are defined as 2 types.

a>Pre-Defined Tags

b>Custom Tags**

a.>Pre-Defined Tags:-- A tag already created by language which executes one work.

Ex:-- HTML Tags, JSP Tags, JSF Tags…

Pre –Defined tags in HTML:

Tag <b> = It makes text bold in HTML.

Tag <u> = It makes text underline in HTML.

b.>Custom Tags*:--** A Tag defined by programmer to execute one task is known as Custom tag.

Format of Custom tag:--

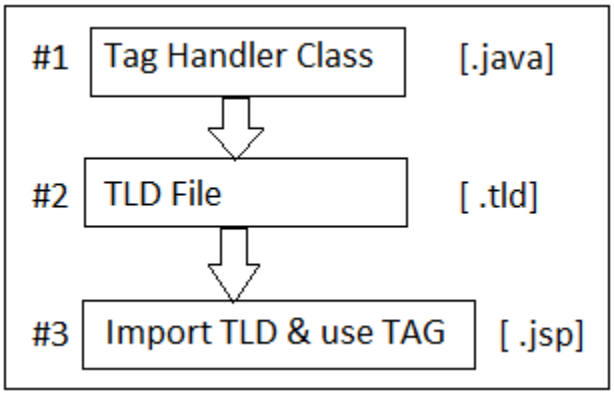
<prefix:tagName attribute='value'….>

=>To create a new tag in JSP, steps are given as.

#1>Write a class for one tag. Also called as Tag Handler class, it will execute logic when we use tag.

#2>Provide details of tags like: tag name, attribute, types of attribute (required, optional), type of tag = empty, body, having attribute etc. It is also called as TLD [Tag Library Descriptor].

#3>Finally Import tag library into JSP using <%@taglib....%> and use Tag in JSP.



Coding Steps:--

#1>To write tag handler class provide one public class and extends SimpleTagSupport class which is given in package: javax.servlet.jsp.target

=>Also override methods doTag() {….} and define logic inside this method.

=>Format looks like:--

```
package com.app;
public class ---------- extends SimpleTagSupport
{
public void doTag() throws JspException, IOException
{
```

…logic…
}
}
#2>Provide tag handler descriptor using .tld file format, looks like below

```
<taglib>
<tlib-version>1.3</tlib-version>
<jsp-version>2.1</jsp-version>
<short-name>……..</short-name>
<uri>……..</uri>

<tag>
   <name> TAG NAME </name>
   <tag-class> FULLY CLASS NAME </tag-class>
   <body-content> empty/scriptless/jsp </body-content>

<attribute>
  <name>attribute name</name>
  <required>true/false</required>
</attribute>
</tag>
</taglib>
```

#3>To import tag in JSP code is:
<%@taglib prefix=”--“ uri=”----“/>
Use tag: <prefix:tagname attribute=”data”….>

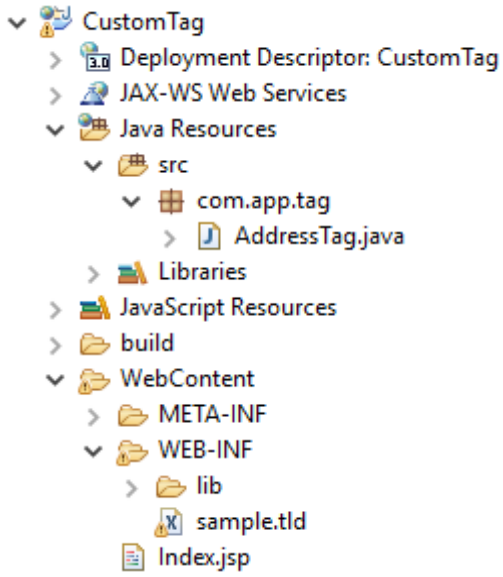
Example Custom Tag using JSP API [Tag without Body]:--

Here, define one tag with any name which print message on JSP Page (in browser at runtime).

Step#1:-- Create one Dynamic web project (web.xml is optional).

Step#2:-- Write one public class with any name and extends SimpleTagSupport also override doTag(){…} and define logic there.

Folder Structure:--



Code: AddressTag.java:--

```
package com.app.tag;
import java.io.IOException;
import java.util.Date;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.SimpleTagSupport;

public class AddressTag extends SimpleTagSupport
{
	public void doTag () throws JspException, IOException
	{
		JspWriter out= getJspContext().getOut();
		out.print("Welcome to Sathya technology");
		out.print("<br/> Date :"+new Date());
	}
}
```

Step#3:-- Create TLD file under WEB-INF folder

Code: **sample.tld (file name must be small latter):--**

```
<taglib>
      <tlib-version>1.3</tlib-version>
     <jsp-version>2.1</jsp-version>
     <uri>http://sathyatech.com/mytag-add</uri>

   <tag>
    <name>sathyaAddr</name>
    <tag-class>com.app.tag.AddressTag</tag-class>
     <body-content>empty</body-content>
  </tag>
</taglib>
```

#3 Import tag into JSP using <%@taglib %> and use tag in JSP.

Code:-Index.jsp:--

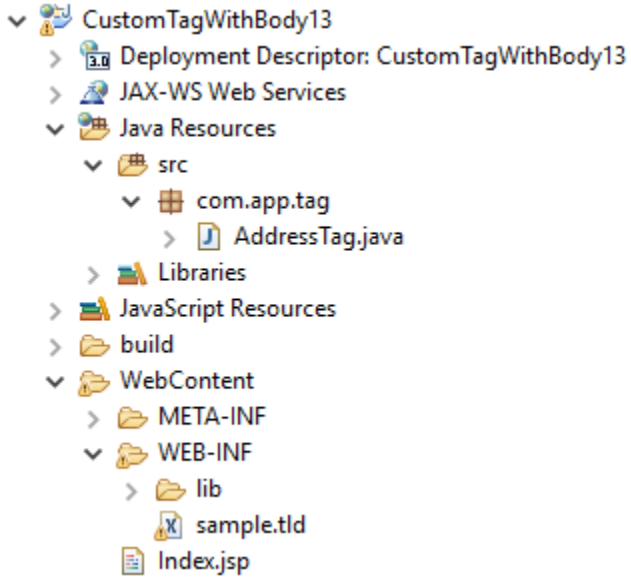
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
   <%@taglib prefix="my" uri="http://sathyatech.com/mytag-add"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h2>Hello Custom tag!!</h2>
<my:sathyaAddr/>
</body></html>
```

=>Run in server and Enter URL in browser:

http://localhost:3033/CustomTag/Index.jsp

Example Custom Tag#2:--

Folder Structure:--



b>Tag with body:--

Step#1:- Tag Handler class

```
package com.app.tag;
import java.io.IOException;
import java.io.StringWriter;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.SimpleTagSupport;

public class AddressTag extends SimpleTagSupport
{
	public void doTag() throws JspException, IOException
	{
		JspWriter out =getJspContext().getOut();
		out.print("<font colour=\"red\">");
		out.print("<b>Sathya Technology<b>");
		out.print("</front>");
		//to Store body data
		StringWriter str = new StringWriter();
```

```
            //read tag body and load to str Object
            getJspBody().invoke(str);
            out.print("</marquee>");
        } }
```

Step#2:-- Copy above tld file and modify body-content from “empty” to scriptless” like (in sample.tld)

```
<body-content>scriptless</body-content>
```

sample.tld (file name must be small latter):--

```
<taglib>
<tlib-version>1.3</tlib-version>
<jsp-version>2.1</jsp-version>
<uri>http://sathyatech.com/mytag-add</uri>

<tag>
<name>sathyaAddr</name>
<tag-class>com.app.tag.AddressTag</tag-class>
<body-content>scriptless</body-content>
</tag></taglib>
```

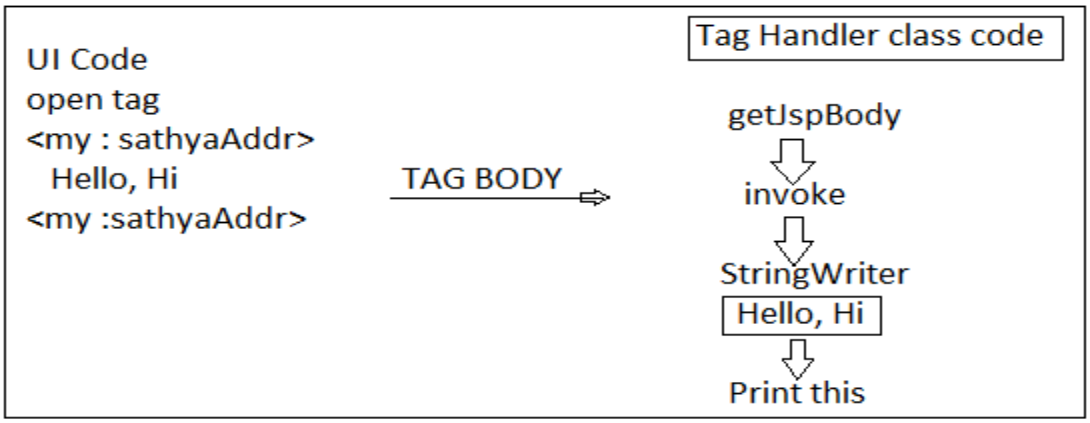
Step#3:-- In JSP file (index.jsp)

Write code like:--

```
<%@taglib prefix="my" uri="http://sathyatech.com/mytag-add"%>
<html>
<body>
    <my:sathyaAddr>
      Contact: 040-123456
    </my:sathyaAddr>
</body>
</html>
```

=>Run in server and Enter URL: http://localhost:3033/CustomTagWithBody13/Index.jsp

Flow Design:--



Attributes in Custom tags:--

A tag can have zero to multiple attributes which provides extra information related to tag. Format looks like:

<prefix:tagName attribute1=”value1” attribute2=”value2”……/>

=>Attributes are variables in Tag handler class.

=>These are two types in custom tags.

a>required attribute

b>optional attribute

=>TLD code to create one attribute is:

```
<attribute>
   <name>Attribute Name</name>
   <required>true/false</required>
</attribute>
```

=>Attributes can be created and used in any order. Order is not followed.

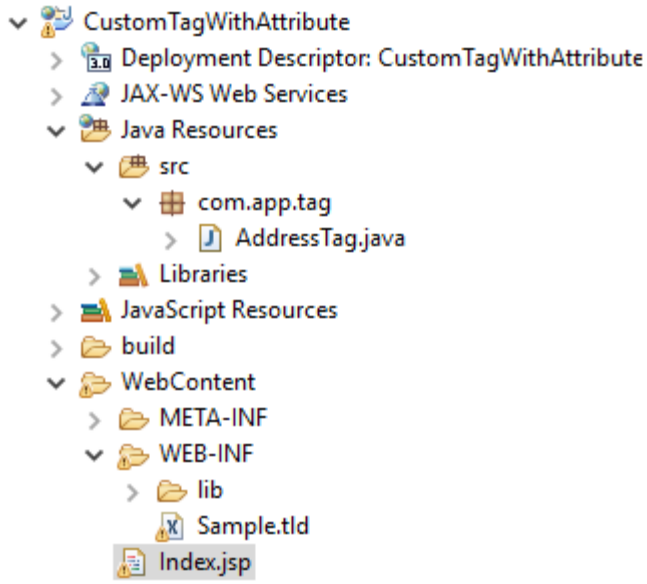
=>For every attribute write one variable in Tag handler class and provide set/get methods.

c.>Custom Tag with attribute Example:

Format:-- <my:sathyaTech contact=” “ branch=” “/>

=>Here contact is required and branch is optional.

Folder Structure:--



Tag Handler class:--

```
package com.app.tag;
import java.io.IOException;
import java.io.StringWriter;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.SimpleTagSupport;

public class AddressTag extends SimpleTagSupport
{
	//No. of Attributes = No. of Variable
	private String contact;
	private String branch="NONE";

	public String getContact() {
		return contact;
	}
	public void setContact(String contact) {
		this.contact = contact;
```

```
        }
        public String getBranch() {
                return branch;
        }
        public void setBranch(String branch) {
                this.branch = branch;
        }

        public void doTag()throws JspException, IOException
        {
        try     {
        JspWriter out =getJspContext().getOut();
        StringWriter str = new StringWriter();
        getJspBody().invoke(str);
        out.print("<h2>+str+</h2>");
        out.print("Contact is :"+contact);
        out.print("<br/>Branch is :" +branch);
        }
        catch(Exception e)
        {
          e.printStackTrace();
        } }
}
```

sample.tld (Under WEB-INF file name must be small latter):--

```
<taglib>
<tlib-version>1.3</tlib-version>
<jsp-version>2.1</jsp-version>
<short-name>Hello Tag</short-name>
<uri>http://sathyatech.com/mytag</uri>

<tag>
<name>sathya</name>
<tag-class>com.app.tag.AddressTag</tag-class>
```

```
<body-content>scriptless</body-content>
<attribute>
 <name>contact<name>
 <required>true</required>
</attribute>
<attribute>
 <name>branch<name>
 <required>false</required>
</attribute>
</tag></taglib>
```

Index.jsp (Under WebContent)--

```
<%@taglib prefix="my" uri="http://sathyatech.com/mytag" %>
<html><body>
<my:sathyaTech contect="9092576623" branch="/AMEERPET">
WELCOM TO SATHYA TECHNOLOGY
</my:sathyaTech>
</body></html>
```

=>Run in server and Enter URL:http://localhost:3033/CustomTagWithAttribute/Index.jsp

=>google=>jstl jars=>https://mvnrepository.com/...=>jar (404) KB=>download jstl 1.2 jar=>download jstl 1.2 jar.

JSTL (JSP Standard Tag Library):--

It is also custom tag set provided by JSP Team which are used to avoid <% %> (Script tags) and makes Easy UI coding.

=>To use these tags, we must add jstl.jar in project lib folder.

=>JSTL Tags are provided as 5 types. Those are

1>core Tags (prefix : c)***

2>SQL Tags (prefix : sql)

3>Formatting Tags (prefix= fmt)

4>Function Tags (prefix : fn)

5>XML (xml) Tags (prefix : x)

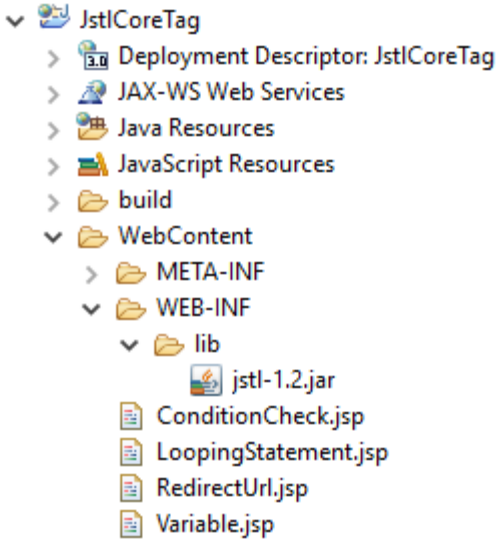
=>We can write our own prefix but these are the standard.

1>Core Tags:--These are mostly used JSTL tags in JSP design, which are mainly used for
i>Creating, removing variables.
ii>Conditional coding.
iii>url creation and redirect.
iv>catch exceptions
v>for each and for token for looping.

List of Core Tags:--

Tags	Descriptions
1>c:out	1>It prints given expression on browser, It is similar to <%= ….%> Expression tag.
2>c:set 3>c:remove	2>It creates variable in given 'scope' Default is page. 3>It delete variable from a particular scope.
4>c:if	4>To Write if condition statement
5>c:choose 6>c:when 7>c:otherwise	5-7>It is like switch case
8>c:forEach 9>c:forTokens	8>It is like for each loop 9>It works on String over given delimiter
10>c:param 11>c:redirect 12>c:url	10>It creates a parameter for URL. 11>It redirects to a new URL. 12>It creates a new URL.
13>c:import	13>It imports HTML content into current JSP
14>c:catch	14>It is used to store exception.

Folder Structure:--



Ex#1:-- Creating variables in scope, read, remove operations.

1>Variable.jsp (Under WebContent):-

```
<%@ 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" %>
<html>
<body><pre>
<c:out value="Hello"/>
<c:out value="${6+36}"/>
<c:out value="${6>36}"/>
<c:set var="counter" value="86"/>
<c:set var ="counter" value="90" scope="request"/>
<c:out value ="${counter+10}"/>
<c:out value ="${requestScope.counter}"/>
<c:set value="SAM" var="uname"/>
Before: <c:out value="${uname}"/>
<c:remove var="uname"/>
After: <c:out value="${uname}"/>
</pre></body></html>
```

=>Run on Server and Enter URL : http://localhost:3033/JstlCoreTag/Variable.jsp

Ex#2: Conditions check:--

=>To test one condition (Boolean) using Core Tag and execute a task use if or choose-when-otherwise.

=>There is no else block for “if”.

=>Choose-when-otherwise behaves like switch.

2>ConditionCheck.jsp (under WebContent):--

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
</head>
<body><pre>
<c:set var="counter" value="36"/>
<c:if test="${counter>0}">
Hi It is  a+positive value
</c:if>
<c:choose>
<c:when test="${counter > 0}">
It is a +positive number
</c:when>
<c:when test="${counter < 0 }">
It is –positive  value
</c:when>
<c:otherwise>
It might be +positive value
</c:otherwise>
</c:choose>
</pre>
</body></html>
```

=>Run on Server and Enter URL: http://localhost:3033/JstlCoreTag/ConditionCheck.jsp

3>Looping statements in Core JSTL use forEach or forTokens to execute code in loop.

=>forEach behaves like for loop and also forEach over collection.

=>forTokens makes one string into multiple parts based on delimiter ex:- space or ,

3>LoopingStatement.jsp (under WebContent):--

```
<%@page import="java.util.Arrays"%>
<%@page import="java.util.List"%>
<%@ 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"%>
<html>
<body><pre>
<c:set var="max" value="10"/>
<c:forEach begin="1" end="${max}" var="i"><c:out value="${i} Hello"/>
</c:forEach>
<% List<String> a1= Arrays.asList("A","B","C");
        request.setAttribute("list", a1);%>
<c:forEach items="${list}" var="ob">
<c:out value="${ob}"/>
</c:forEach>
<c:set var="str" value="Hello I am Uday Kumar"/>
<c:forTokens items ="${str}" delims=" " var="s">
<c:out value="${s}"/><br/>
</c:forTokens>
</pre></body></html>
```

=>Run on Server and Enter URL:
http://localhost:3033/JstCorelTag/LoopingStatement.jsp

Ex#4:-- Working with URLs

Create URL :-- <c:url…>
Provide parameters <c:param….>
Invoke URL <c:redirect…>

4>RedirectUrl.jsp (under WebContent):--

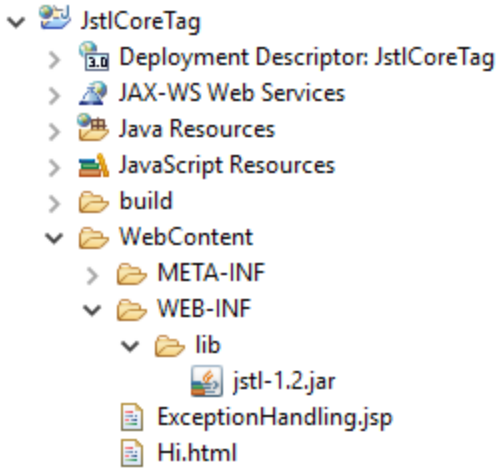
```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
</head>
```

```
<body><pre>
<c:url var="search" value="https://www.google.com/search">
<c:param name="q" value="Hyderabad"/>
</c:url>
<c:redirect url="${search}"/>
</pre></body></html>
```

=>Run on Server and Enter URL: http://localhost:3033/JstlCoreTag/RedirectUrl.jsp

Example#5—Import HTML data from other resource and Exception handling.

Folder Structure:--



1>Hi.html (under WebContent):--

```
<h2>Welcome to Exception Handling</h2>
```

2>ExceptionHandling.jsp (Under WebContent):--

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<html><body
<pre>
<c:import url="Hi.html"/>
<c:catch var="ae">
<%
    int a=5/0;
%>
</c:catch>
```

```
<c:out value="${ae}"/>
</pre>
</body></html>
```

=>Run on Server and Enter URL:
http://localhost:3033/JstlTagCore/ExceptionHandling.jsp

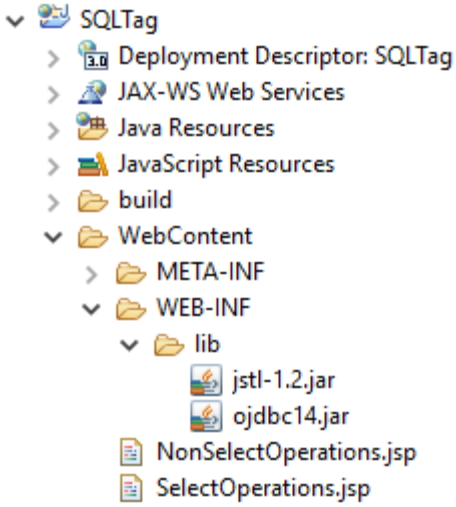
NOTE:-- Choose when otherwise is most important for Interview.

2>SQL Tags:-- These are used to perform SQL operation like select and non-select (insert, update, delete).
=>Prefix used is 'sql'.

Tags	Descriptions
1>sql : setDataSource	=>Use it for creating connection with DB.
2>sql:query	=>Use for select operation.
3>sql:update	=>Use for non-select operation.
4>sql:param	=>Use this to pass parameter to SQL.
5>sql:transaction	=>Use this to execute multiple non-select operations.

Ex#1:-- Create Connection and execute select statement.
Step#1:- open Oracle cmd.
Step#2:-Create table and insert records.
SQL>create table student (sid number, sname varchar2(20), sfee number(12,3));
Sql>insert into student values(10,'A', 2.5);
Sql>insert into student values(12,'B', 2.7);
Sql>insert into student values(14,'C', 2.8);

Folder Structure:--



Code: SelectOperations.jsp (under WebContent):--

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
 <%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%><body>
<pre>
<sql:setDataSource var="con"
driver="oracle.jdbc.driver.OracleDriver"
url="jdbc:oracle:thin:@localhost:1521:XE"
user="system"
password="system"
/>

<sql:query var="rs" dataSource="${con}"
sql="select * from student where sid=?">
<sql:param value="11"/>
</sql:query>

Data is
<c:forEach items="${rs.rows}"  var="std">
Student ID :  <c:out value="${std.sid}"/>
```

```
Student Name : <c:out value="${std.sname}"/>
Student Fee :  <c:out value="${std.sfee}"/>
</c:forEach>
</pre>
</body></html>
```

=>Run on Server and Enter URL: http://localhost:3033/SQLTag/SelectOperations.jsp

Example#2: Non select-operations:--

Code: NonSelectOperations.jsp (Under WebContent):--

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
<html>
<body><pre>
<sql:setDataSource  var="con" driver="oracle.jdbc.driver.OracleDriver"
url="jdbc:oracle:thin:@localhost:1521:XE" user="system”password="system" />

<sql:update var="c" dataSource="${con}"
                    sql="insert into student values(24,'Neha', 4.4)">
</sql:update>
No. of Rows Inserted : ${c}

<sql:update var="c1" dataSource="${con}" sql="delete from student where sid=?">
<sql:param value="16" />
</sql:update>
No. of Record deleted : ${c1}

<sql:update var="c2" dataSource="${con}"
                sql="update student8 set sname='Raj', sfee=223.87 where  sid=?">
<sql:param value="11" />
</sql:update>
No. of Record updated : ${c2}
<sql:transaction dataSource="${con}">
<sql:update sql="insert into student8 values(21,'Raja', 7.5)" />
```

```
<sql:update sql="insert into student8 values(22,'Rani', 8.5)" />
<sql:update sql="insert into student8 values(23,'Ramu', 9.5)" />
</sql:transaction>
</pre></body></html>
```

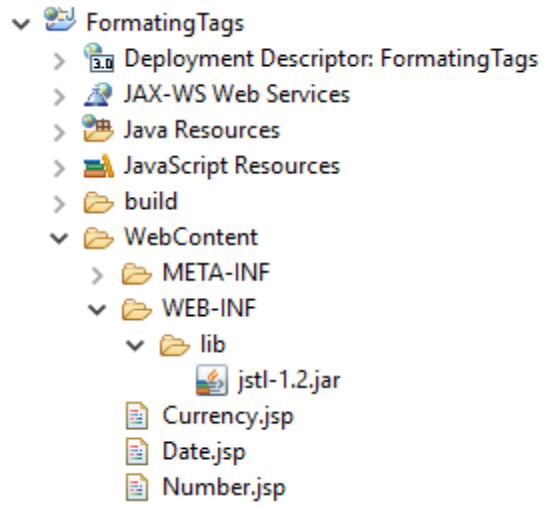
=>Run on Server and Enter URL : http://localhost:333/SQLTag/NonSelectOperations.jsp

3>formatting Tags:--

These are used to format number or data input in JSP page. i.e Change pattern of data and data type of double/int.

Tags	Descriptions
1>fmt:parseNumber	=>It is used to Parse the string to double or number.
2>fmt:formatNumber	=>It is used to format the numerical value with specific format or precision.
3>fmt:formatData	=>It formats the time date using the pattern and styles.

Folder Structure:--



Ex#1: Number Conversion:--

1>Number.jsp (Under WebContent):--

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<html>
<body><pre>
```

```
<c:set var="count" value="306.97"/>
<fmt:parseNumber   var="d" type="number"  integerOnly="true" value="${count}"/>
Final cost   : ${d*2}
</pre>
</body></html>
```

=>Run on Server and Enter URL: http://localhost:3033/FormatingTags/Number.jsp

Ex#2: Currency Formatting:--

=>Display currency code, provide comma symbol, remove fraction digits etc…

2>Currency.jsp (under WebContent):--

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head><body><pre>
<c:set var="cost" value="98764.8764"/>
<fmt:formatNumber   type="currency"  value="${cost}" currencyCode="INR"
maxIntegerDigits="3" groupingUsed="false"/>
</pre></body></html>
```

=>Run on Server and Enter URL : http://localhost:3033/FormatingTags/Currency.jsp

Ex#3: Date Formatting: We can show data in different styles date/time/both.
In different styles short/medium/long etc…

3>Date.jsp (Under WebContent):--

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
</head><body><pre>
<c:set value="<%=new java.util.Date()%>"  var="s"/>
```

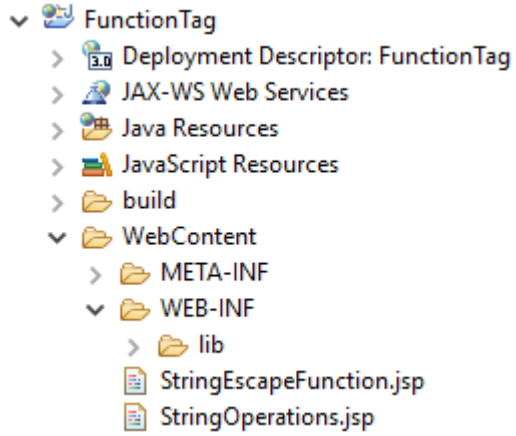
```
${s}
<fmt:formatDate value="${s}"/>
<fmt:formatDate value="${s}" type="date"/>
<fmt:formatDate value="${s}" type="time"/>
<fmt:formatDate value="${s}" type="both"/>
<fmt:formatDate value="${s}" type="both" dateStyle="short"/>
<fmt:formatDate value="${s}" type="both" dateStyle="medium"/>
<fmt:formatDate value="${s}" type="both" dateStyle="long"/>
</pre></body></html>
```

=>Run on Server and Enter URL: http://localhost:3033/FormatingTags/Date.jsp

4>Function Tags:--These tags are used to perform String operation over given input String data or any variable.

Tags	Description
1>fn:contains()	=>Given input string containing the specified substring.
2>fn:containsIgnoreCase()	=>Given input string containing the specified substring.
3>fn:endsWith()	=>given input string ends with the specified suffix or not?
4>fn:escapeXml()	=>show XML tags as it is in.
5>fn:indexOf()	=>Find index of a sub-string.
6>fn:trim()	=>Removes the blank spaces from both the ends.
7>fn:startsWith()	=>Used for check, given string is started with format or substring.
8>fn:split()	=>splits the string into an array of substrings.
9>fn:toLowerCase()	=>converts all chars to lower case.
10>fn:toUpperCase()	=>converts all chars to Upper case.
11>fn:substring()	=>Find subset of a string.
12>fn:substringAfter()	=>Find subset of string after specific substring.
13> fn:substringBefore()	=>Find subset of string before specific substring.
14>fn:length()	=>Find length of String.
15>fn:replace()	=>Replace all the occurrence of a String with another String.

Folder Structure:--



Ex#1: String operations:--

1>StringOperations.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"%>
<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<html>
<body>
        <pre>
      <c:set value="Hello from Sathya Technologies" var="str" />
         STR     : ${str}
      SUB STR : ${fn:substring(str,2,7)}
      SUB STR : ${fn:substringAfter(str,'from')}
      SUB STR : ${fn:substringBefore(str,'Sathya')}

<c:if test="${fn:contains(str,'Sathya')}">
 Hello Sathya Tech#1
</c:if>
 <c:if test="${fn:containsIgnoreCase(str,'sathya')}">
 Hello Sathya Tech#2
 </c:if>
<c:if test="${fn:startsWith(str,'Hel')}">
 Hello Sathya Tech#3
 </c:if>
<c:if test="${fn:endsWith(str,'ies')}"> Hello Sathya Tech#4 </c:if>
```

```
        Index : ${fn:indexOf(str,'from') }
<c:set var="str2" value="  Hello RAM  "/>
        LEN : ${fn:length(str2) }
        <c:set var="str3" value="${fn:trim(str2) }"/>
        LEN : ${fn:length(str3) }
   LOWER : ${fn:toLowerCase(str3) }
    UPPER : ${fn:toUpperCase(str3) }
 REPLACE : ${fn:replace(str3,'Hello','hi') }

<c:set var="str4" value="${fn:split(str3,' ') }">
</c:set> ${str4[0]} ${str4[1]}

<c:set var="str5" value="Hi Employee <ename> UDAY </ename>"/>
${str5} ${fn:escapeXml(str5) }
</pre>
<body></html>
```

=>Run on server and Enter URL :
http://localhost:3033/FunctionTag/StringOperations.jsp

Ex#2: String operations:--

2>StringEscapeFunction.jsp:--

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
<html><body><pre>
<c:set var="s" value="Hello from Sathya Technology"/>
<c:if test="${fn.contains(s,'Sathya')}">
Hello
</c:if>
<c:if test="${fn:containsIgnoreCase(s, 'sat')}">
Yes Exist
</c:if>
${fn:indexOf(s,'Sathya')}
<c:set value="Hi <ename>Uday</ename>" var="s3"/>
${s3}
```

```
${fn:escapeXml(s3)}
</pre>
</body>
</html>
```

=>Run on server and Enter URL :
http://localhost:3033/FunctionTag/StringEscapeFunction.jsp

XML Tags:-- These are used to read(parse) data from XML Content given in JSP file.

No.	Tags	Description
1	x:out	=>Read and print data using Xpath expressions.
2	x:parse	=>Parse the XML data.
3	x:set	=>Sets XML data to variable
4	x:choose x:when x:otherwise	=>It is a switch case over Xpath of XML Data
5	x:if	=>If condition over Xpath

XPath:-- It is path of data in XML which indicates tags flow.

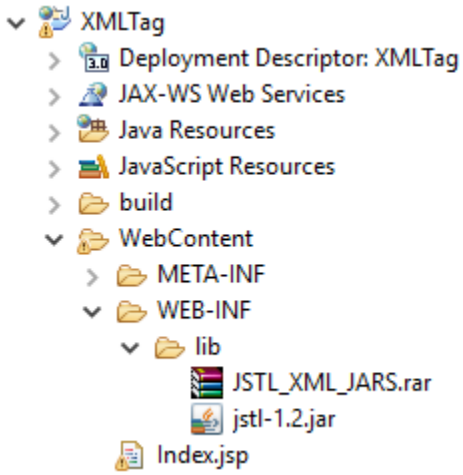
Ex:--

```
<employee>
   <emp>
        <eid>....</eid>
          ......
```

Here XPath for eid is:

$varName/employee.emp[1]/eid

Folder Structure:--



Index.jsp (Under WebContent):--

```
<%@ 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"%>
<%@taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml"%>
<html><body><pre>
<c:set var="emps">
<employee>
<emp>
<empId>10</empId>
<empName>Uday</empName>
<empSal>3334.78</empSal>
</emp>
 <emp>
  <empId>13</empId>
  <empName>Vijay</empName>
  <empSal>3338.78</empSal>
 </emp>
 <emp>
  <empId>12</empId>
  <empName>Neha</empName>
```

```
  <empSal>333464.78</empSal>
 </emp>
</employee>
</c:set>

<x:parse xml="${emps}" var="ob" />
<x:set var="ename" select="$ob/employee/emp[3]/empName" />
<x:out select="$ename" />
<x:out select="$ob/employee/emp[1]/empId" />
<x:choose>
<x:when select="$ob/employee/emp[3]/empSal>1000">
        IT IS GOOD
</x:when>
<x:otherwise>IT IS OK </x:otherwise>
</x:choose>
<x:if select="$ob/employee/emp[1]/empName='Uday'">
Your are Uday only!!
</x:if>
</pre></body></html>
```

=>Run on server and Enter URL : http://localhost:3033/XMLTag/Index.jsp